HW5

Zhenmin Li

2020/10/24

Q1

In the model trained using dataset in the previous homework (PRE), weight and height has a similar importance on the prediction. In the model trained using dataset in the current homework (CURR), the prediction result almost fully depends on the weight.

The result from the model based on PRE mostly cumulated on around 0.5, which means the model is not quiet sure about the results. The one from CURR mostly cumulated on about 0 and 1, which means the model is quite sure for most predictions.

I think this is because CURR is not normalized. According to our common sense, most males have higher weights than females, and the model would predict by weight.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(tidyverse)
```

```
## -- Attaching packages -------------------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v stringr 1.4.0
## v tidyr   1.1.2     v forcats 0.5.0
## v readr   1.4.0

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

```
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

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

```r
library(ggfortify)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
nice_names <- function(df){
  names(df) <- names(df) %>% str_replace_all("[^a-zA-Z0-9]+","_") %>%
    str_replace_all("[_]+$","") %>%
    str_replace_all("^[_]+","") %>%
    tolower();
  df
};

info <- timetk::tk_tbl(data.table::fread("500_Person_Gender_Height_Weight_Index.csv", header=T, strings
  drop_na() %>%
  nice_names() %>%
  mutate(female=gender=='Female',train=runif(nrow(.))<0.75) %>%
  filter(height > 0 & weight > 0);
```

```
## Warning in
## tk_tbl.data.frame(data.table::fread("500_Person_Gender_Height_Weight_Index.csv", :
## Warning: No index to preserve. Object otherwise converted to tibble
## successfully.
```

```r
form <- female ~ height +
  weight +
  I(height^2) +
  I(weight^2) +
  height:weight;

model.gbm <- gbm(form,
                 distribution="bernoulli",
                 info %>% filter(train),
                 n.trees = 200,
                 interaction.depth = 5,
                 shrinkage=0.1);
summary(model.gbm,plot=FALSE)
```

```
##                          var  rel.inf
## weight                weight 53.56935
## height                height 46.43065
## I(height^2)      I(height^2)  0.00000
## I(weight^2)      I(weight^2)  0.00000
## height:weight height:weight  0.00000
```

```r
test <- info %>% filter(!train);
test$female.p.gbm <- predict(model.gbm, test, type="response");
```

```
## Using 200 trees...
```

```r
p1 <- ggplot(test, aes(female.p.gbm)) + geom_density()

info1 <- timetk::tk_tbl(data.table::fread("datasets_26073_33239_weight-height.csv", header=T, stringsAs
```

```r
  drop_na() %>%
  nice_names() %>%
  mutate(female=gender=='Female',train=runif(nrow(.))<0.75) %>%
  filter(height > 0 & weight > 0);
```

```
## Warning in tk_tbl.data.frame(data.table::fread("datasets_26073_33239_weight-
## height.csv", : Warning: No index to preserve. Object otherwise converted to
## tibble successfully.
```

```r
model.gbm1 <- gbm(female ~ height +
                   weight +
                   I(height^2) +
                   I(weight^2) +
                   height:weight,
                 distribution="bernoulli",
                 info1 %>% filter(train),
                 n.trees = 200,
                 interaction.depth = 5,
                 shrinkage=0.1);
summary(model.gbm1,plot=FALSE)
```

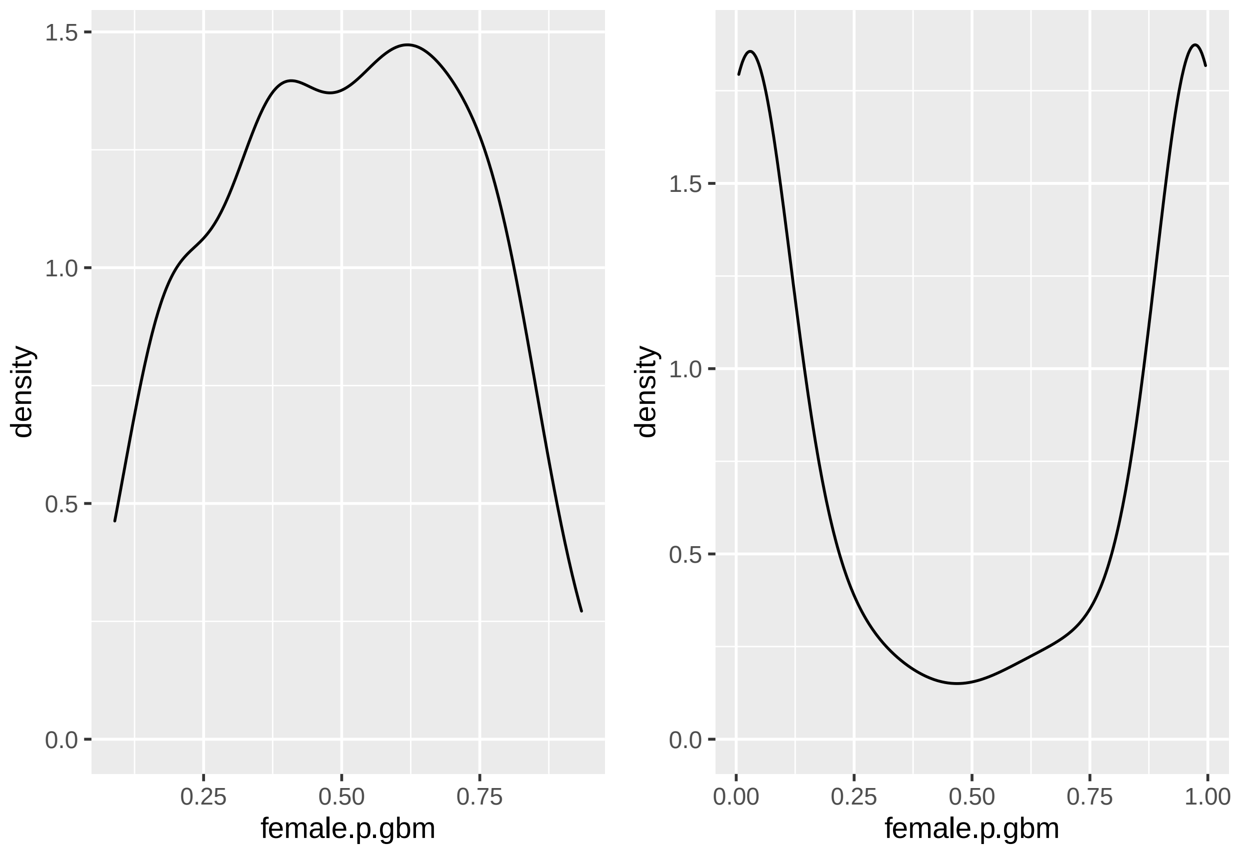
```
##                         var   rel.inf
## weight               weight 91.704968
## height               height  8.295032
## I(height^2)     I(height^2)  0.000000
## I(weight^2)     I(weight^2)  0.000000
## height:weight height:weight  0.000000
```

```r
test1 <- info1 %>% filter(!train);
test1$female.p.gbm <- predict(model.gbm1, test1, type="response");
```

```
## Using 200 trees...
```

```r
p2 <- ggplot(test1, aes(female.p.gbm)) + geom_density()

grid.arrange(p1, p2, ncol=2)
```




```
c(sum((test$female.p.gbm>0.5)==test$female)/nrow(test),
  sum(FALSE==test$female)/nrow(test),
  sum((test1$female.p.gbm>0.5)==test1$female)/nrow(test1),
  sum(FALSE==test1$female)/nrow(test1));
```

```
## [1] 0.5333333 0.4740741 0.9110567 0.4957160
```

Q2

1. we can see that there are some irregularities with a total of 5, and we remove them

```
info2 <- timetk::tk_tbl(data.table::fread("datasets_38396_60978_charcters_stats.csv", header=T, strings
  drop_na() %>%
  nice_names() %>%
  mutate(train=runif(nrow(.))<0.75)
```

```
## Warning in
## tk_tbl.data.frame(data.table::fread("datasets_38396_60978_charcters_stats.csv", :
## Warning: No index to preserve. Object otherwise converted to tibble
## successfully.
```

```
info2
```

```
## # A tibble: 611 x 10
##    name  alignment intelligence strength speed durability power combat total
##    <fct> <fct>            <int>    <int> <int>      <int> <int>  <int> <int>
##  1 3-D ~ good                50       31    43         32    25     52   233
##  2 A-Bo~ good                38      100    17         80    17     64   316
```

```
##   3 Abe ~ good                  88        14    35          42    35      85   299
##   4 Abin~ good                  50        90    53          64    84      65   406
##   5 Abom~ bad                   63        80    53          90    55      95   436
##   6 Abra~ bad                   88       100    83          99   100      56   526
##   7 Adam~ good                  63        10    12         100    71      64   320
##   8 Adam~ good                   1         1     1           1     0       1     5
##   9 Agen~ good                   1         1     1           1     0       1     5
##  10 Agen~ good                  10         8    13           5     5      20    61
## # ... with 601 more rows, and 1 more variable: train <lgl>
```

2. seems we only needs pc1

```r
info2 <- info2 %>% filter(total > 5)

pcs <- prcomp(info2[,3:9]);
summary(pcs)
```

```
## Importance of components:
##                             PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation     115.6597 24.68437 23.19102 19.01658 17.74925 17.02995
## Proportion of Variance   0.8635  0.03933  0.03472  0.02334  0.02034  0.01872
## Cumulative Proportion    0.8635  0.90288  0.93760  0.96094  0.98128  1.00000
##                              PC7
## Standard deviation     2.551e-14
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

3. I think not. The features are already in Percentile. Some of them are out of limitation but I think it is still in the scale. Here I use durability as an example

```r
data.frame(min=sapply(info2[,3:10],min),max=sapply(info2[,3:10],max))
```

```
##              min max
## intelligence   1 113
## strength       1 100
## speed          1 100
## durability     1 120
## power          0 100
## combat         1 101
## total         36 581
## train          0   1
```

```r
top5 <- info2 %>% top_n(5,durability)
top5[with(top5, order(-durability)), ]
```

```
## # A tibble: 55 x 10
##    name  alignment intelligence strength speed durability power combat total
##    <fct> <fct>            <int>    <int> <int>      <int> <int>  <int> <int>
##  1 Doom~ bad                 88       80    67        120   100     90   545
##  2 Star~ good                88       85   100        110   100     85   568
##  3 Nova  good               100       85    67        101   100     85   538
##  4 Silv~ good                63      100    84        101   100     32   480
##  5 Adam~ good                63       10    12        100    71     64   320
##  6 Amazo bad                 75      100   100        100   100    100   575
##  7 Apoc~ bad                100      100    33        100   100     60   493
##  8 Beyo~ good                88      100    23        100   100     56   467
##  9 Biza~ neutral             75       95   100        100    95     85   550
```

```
## 10 Blac~ bad                   88       100     92         100     89      56    525
## # ... with 45 more rows, and 1 more variable: train <lgl>
```

4. I create a new column like total and write the result of comparison in compare Then show the top 5 of compare and it seems they are equal (no value 1 which means they are not equal)

```
sum <- transform(info2, sum=rowSums(info2[,c(7,3,4,5,6,8)]))
sum$compare <- ifelse(sum$total == sum$sum, 0, 1)
comp <- sum %>% top_n(5,compare)
comp[with(comp, order(-compare)), ]$compare
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [149] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [186] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [223] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [260] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [297] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [334] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [371] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [408] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

5. No If we check the pca of the data, We can see that PC1 take accounts for most of PC1 which has most importance which means "total" is the principle component of the data, and make other features useless

```
pcs1 <- prcomp(info2[,3:8]);
summary(pcs1)
```

```
## Importance of components:
##                            PC1     PC2     PC3      PC4      PC5      PC6
## Standard deviation     46.664 23.6134 22.8884 18.88294 17.74412 17.02230
## Proportion of Variance  0.516  0.1321  0.1241  0.08449  0.07461  0.06866
## Cumulative Proportion   0.516  0.6481  0.7722  0.85673  0.93134  1.00000
```

```
pcs
```

```
## Standard deviations (1, .., p=7):
## [1] 1.156597e+02 2.468437e+01 2.319102e+01 1.901658e+01 1.774925e+01
## [6] 1.702995e+01 2.550823e-14
##
## Rotation (n x k) = (7 x 7):
##                     PC1        PC2        PC3         PC4          PC5
## intelligence 0.08724079 -0.4889535  0.0954139 -0.11144941  0.644722478
## strength     0.22815661  0.4035600 -0.4802491 -0.12337467  0.411166202
## speed        0.12927447  0.1308655  0.1693489  0.88977302 -0.036741381
## durability   0.21580237  0.3804719 -0.1866100 -0.25719501 -0.392768526
## power        0.16116389  0.1036374  0.7651091 -0.33537677 -0.124826239
## combat       0.09590209 -0.6424216 -0.3322937 -0.02450351 -0.493983148
## total        0.91754021 -0.1128403  0.0307192  0.03787365  0.007569386
##                      PC6        PC7
## intelligence  0.416250531 -0.3779645
## strength     -0.476768922 -0.3779645
## speed         0.039751187 -0.3779645
## durability    0.640760245 -0.3779645
## power        -0.327073790 -0.3779645
```

```
## combat       -0.283198306 -0.3779645
## total         0.009720944  0.3779645
```

```
pcs1
```

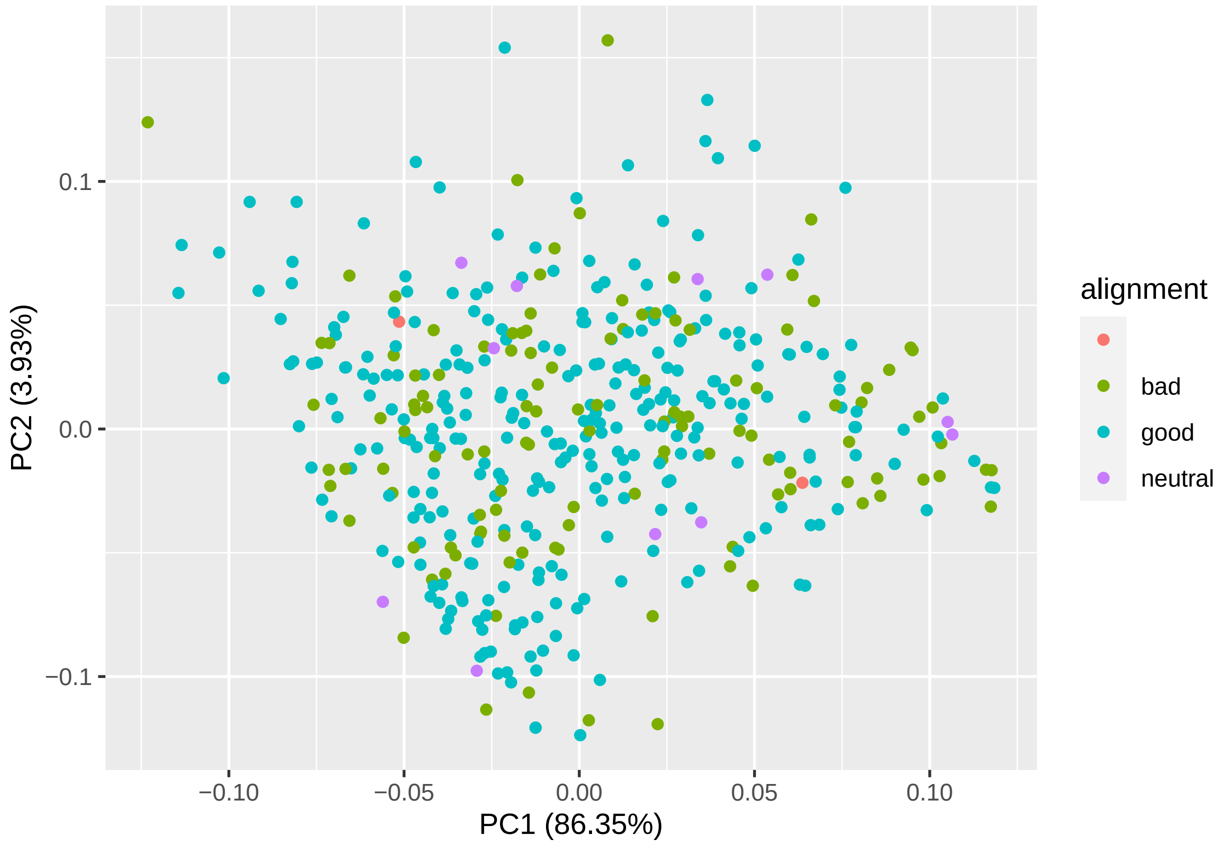
```
## Standard deviations (1, .., p=6):
## [1] 46.66412 23.61344 22.88843 18.88294 17.74412 17.02230
##
## Rotation (n x k) = (6 x 6):
##                    PC1         PC2        PC3         PC4          PC5
## intelligence 0.1610814 -0.42123376  0.4180341 -0.04892612  0.654052139
## strength     0.6072148 -0.06014799 -0.4733042 -0.10205240  0.418940908
## speed        0.3083501  0.16436338  0.1023077  0.92906380 -0.009315058
## durability   0.5691860  0.08291554 -0.2156489 -0.22860415 -0.396377363
## power        0.3918877  0.38491920  0.7199693 -0.25647714 -0.115438967
## combat       0.1808714 -0.79805466  0.1609454  0.07740552 -0.475580041
##                      PC6
## intelligence -0.43781780
## strength      0.46661477
## speed        -0.06474356
## durability   -0.64286309
## power         0.31748807
## combat        0.26892912
```

6. I don't know why I can't use the code in lecture 10 so I use another

```
pca.plot <- autoplot(pcs, data = info2, colour = 'alignment')
```

```
## Warning: `select_()` is deprecated as of dplyr 0.7.0.
## Please use `select()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
pca.plot
```



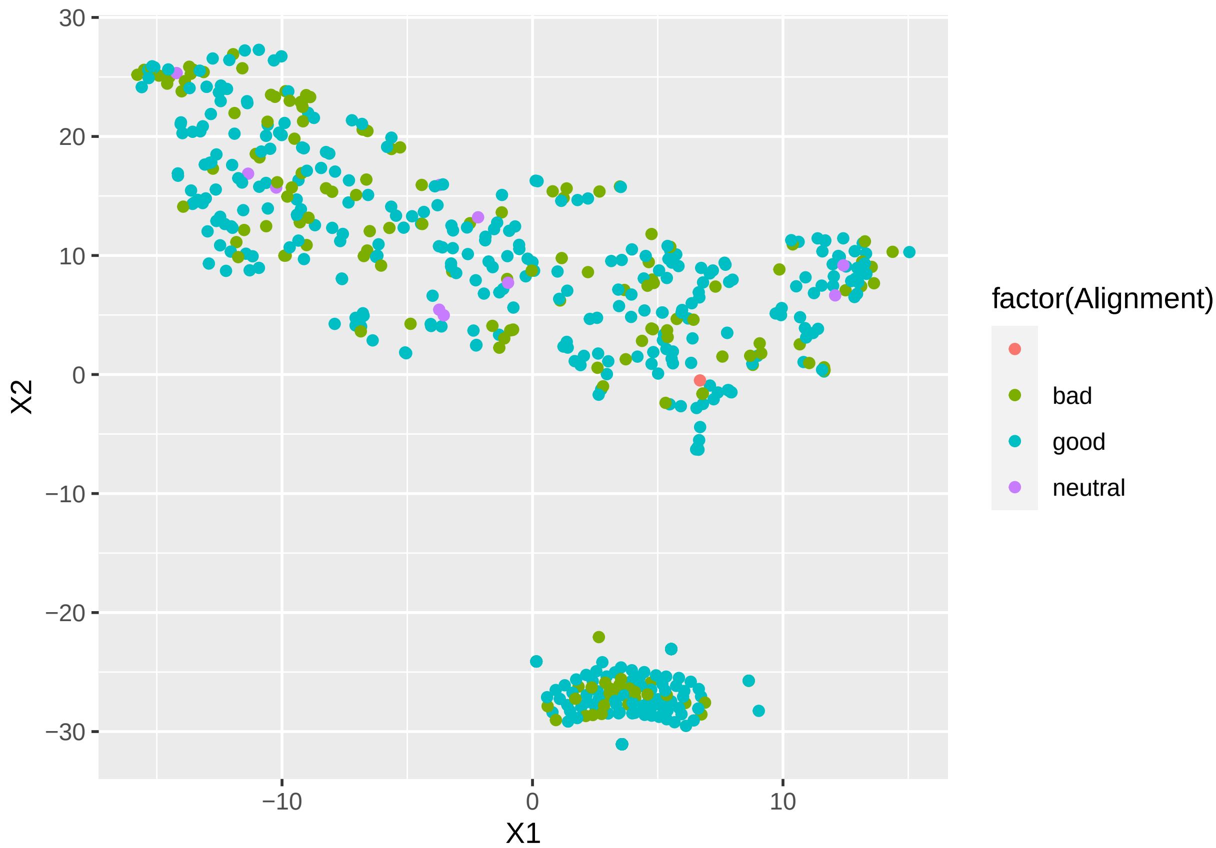
Q3

Please see the ipynb file attached. The following is the r code use to plot. I have no findings to the plot.

```
lowd <- timetk::tk_tbl(data.table::fread("lowd.csv"))
```

```
## Warning in tk_tbl.data.frame(data.table::fread("lowd.csv")): Warning: No index
## to preserve. Object otherwise converted to tibble successfully.
```

```
(ggplot(lowd,aes(X1,X2)) + geom_point(aes(color=factor(Alignment))))
```



Q4

It is in the last part of ipynb attached.

Q5

Theresult is in the picture, in which the final parameters are "height", "weight" and "height:weight".

```
library(caret)
```

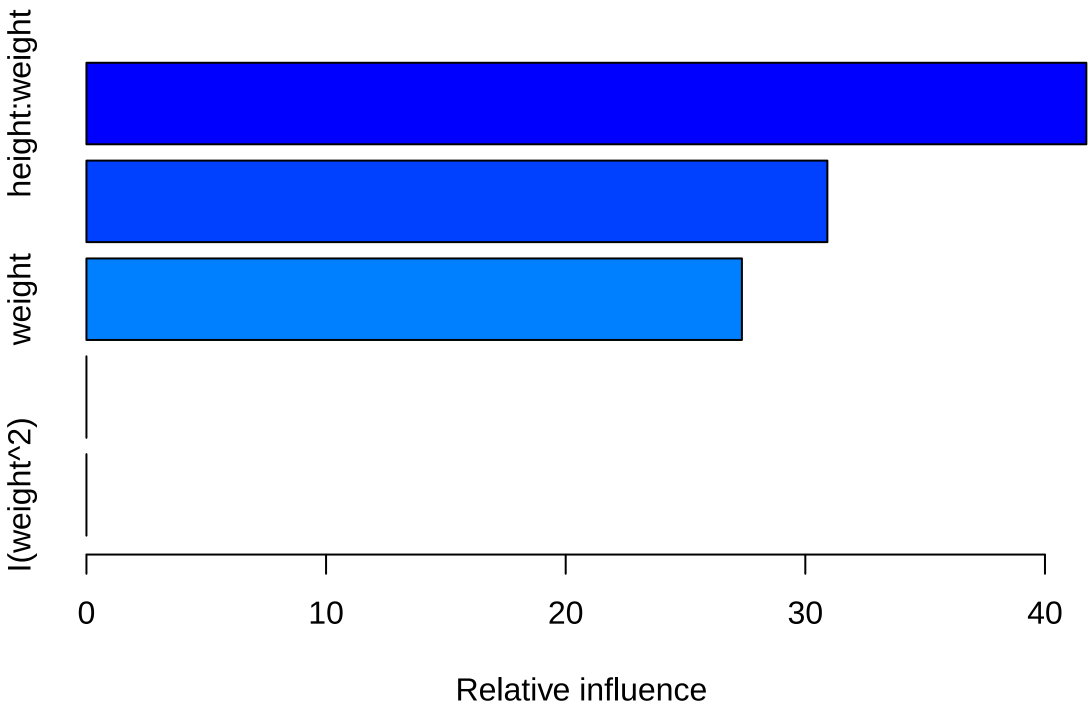
```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

```
trainIndex <- createDataPartition(info$female, p = .8,
                                  list = FALSE,
                                  times = 1)
info$female <- factor(info$female);
train_ctrl <- trainControl(method = "repeatedcv",number=50,repeats=5);
gbmFit1 <- train(form, data = info %>% slice(trainIndex),
                 method = "gbm",
                 trControl = train_ctrl,
```

```r
                  verbose = FALSE)
summary(gbmFit1)
```




```
##                           var  rel.inf
## height:weight height:weight 41.72747
## height                 height 30.91978
## weight                 weight 27.35275
## I(height^2)     I(height^2)  0.00000
## I(weight^2)     I(weight^2)  0.00000
```

Q6

If the size of dataset is unlimited, we can simply divide the dataset into train set and test set for machine learning. But in practical situation the dataset is often limited, which means the train set is likely to bias, because it can only reflect certain parts of the pattern of the dataset. Because we train the model using the train set, the model will also likely to bias. This phenomenon will be more obvious if the size of dataset is more small. K-fold, and other cross validation, divide the dataset into k group and use 1 for test and others for training. This method allows to use the whole dataset, instead of some parts of it, to train the model. And the model will be unbiased based on the dataset.

Because accuracy cannot fully reflect the performance of model, through it is an important criterion. For example, if I want to train a model to classify a dataset which has 990 males and 10 females, a model which classify all the data as male will have an accuracy of 99%, but is useless. We need more approach (for this example, TPR and FPR) to evaluate the model.

#Q7

1. Assign a weight to each feature, and then use the predictive model to train on these original features.
2. After obtaining the weight/ranking of the feature, take the absolute value of these values and remove the minimum one.
3. Iterate the steps mentioned above until the number of remaining features reaches the required limitations.