STATS 790 - Statistical Learning
Homework 4

Tommy Flynn

04 April, 2023

Contents

Question 1:

(I) Data Description

The data I have selected is called the *Body Fat Data Set* which is publicly available on the Data and Story Library (https://dasl.datadescription.com/datafile/bodyfat/). This is where I found the data, however, it originated at the Brigham Young University Human Performance Research Center. It contains 16 continuous body measurements of 250 adult human males. Of interest is accurately predicting body fat percentage using convenient measurements instead of more tedious and expensive methods like underwater weighing or the DEXA scan. This is significant in the context of consistently monitoring diet and exercise interventions.

```
# libraries
library(tidymodels)
library(ggcorrplot)
library(vip)
library(pdp)

# seed
set.seed(13)

# load data
bfat <- read.delim('bodyfat.txt')
```

(II) Exploratory Analysis

First, I ensured that there were no missing values. Next, I ran a summary of the data set. Table 1 shows a summary of the first 5 variables which provides a rough picture of the subjects in this study. The 250 males range from age 22 to 81, 118lbs to 262lbs, 5'3" to 6'5", 0.995g/cm^3 to 1.11g/cm^3 body density, and 0 to 48 percent body fat. Figure 1 displays the box plots of the remaining variables which are 11 circumferences of the major body parts. The summary and box plots indicate that the study includes a wide variety of male morphologies and most of the values agree with what is expected for the average male without digging too deeply. After this, I looked for outliers and

noticed four suspect data points. Two of these have values of 0 and 0.7 percent body fat. Only a subset of elite body builders achieve 2% body fat which is made up of essential fat stored in the brain, nerves, and surrounding organs. It is possible to be near zero but this requires a genetic anomaly such as marfanoid–progeroid–lipodystrophy syndrome which occurs in less than 100 people worldwide. The other two suspect data points have 47.5 and 40.1 percent body fat which is well over the obesity range of 25-30. Looking further, these two are also the 1st and 4th shortest subjects in the data set. Next, I examined the correlation of the variables. We see positive correlation between the circumferences and body fat percentage as expected. Abdomen has the highest correlation which is where males store the majority of their fat. Unexpectedly, the abdomen and waist have a correlation of 1 which warrants further investigation. Upon diving the abdomen column by the waist column, we get a vector of 2.54. This indicates that the abdomen column is equal to the waist column except the former is measured in centimeters and the latter is measured in inches. Note that abdomen and waist measurements are not supposed to be the same. The abdomen is measured at the widest location of the torso (belly button) and the waist is measured at the narrowest (above the abdomen half way to the chest). Density is highly negatively correlated with body fat percentage as it is body mass over body volume. Note that body density is not a convenient measurement and usually requires water weighing. In fact, there are equations such as the Siri equation which estimate body fat percentage using only body density: $\text{Siri}_{\%} = (495/\text{Body Density}) - 450$. Lastly, height has almost no correlation with body fat percentage.

Table 1: Demographic information.

	Density (g/cm^3)	Pct.BF (%)	Age	Weight (lbs)	Height (in)
Min:	0.995	0	22	118.5	64
1st Qu:	1.042	12.43	35.25	158.5	68.25
Median:	1.055	19.20	43	176.1	70
Mean:	1.056	19.03	44.88	178.1	70.30
3rd Qu:	1.070	25.20	54	196.8	72.25
Max:	1.109	47.50	81	262.8	77.75
Count:	250				

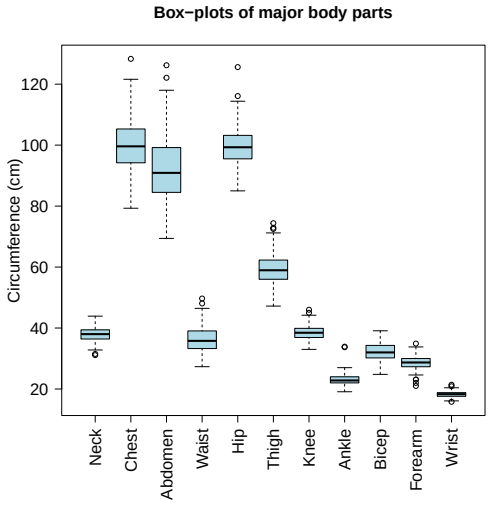

Figure 1: Box plots of circumference predictor variables.

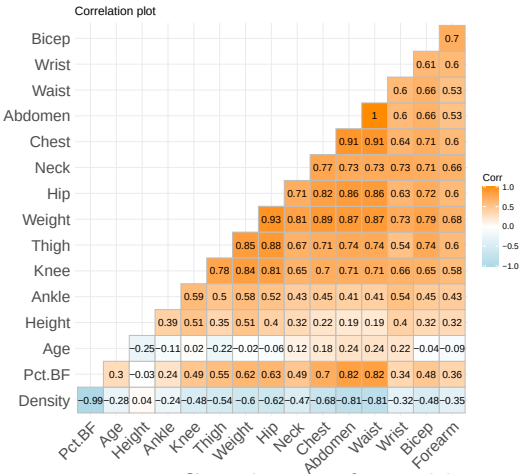

Figure 2: Correlation of variables.

```
# missing values
any(is.na(bfat))


# summary of data
summary(bfat)


# box plots
par(mar = c(7, 5, 5, 3))
boxplot(bfat[6:16], col='lightblue', las=2, ylab='Circumference (cm)',
        main='Box-plots of major body parts', cex.axis=1.5, cex.main=1.5,
```

```
        cex.lab=1.5)

# correlation
ggcorrplot(cor(bfat), type='lower', hc.order=TRUE, lab=TRUE, lab_size=4,
           colors=c('lightblue', 'white', 'darkorange'), tl.cex=17,
           title='Correlation plot')

# check abdomen vs waist
bfat$Abdomen / bfat$Waist
```

(III) Preprocessing

Informed by the exploratory analysis, we will discard the four outlier data points. We will also drop the waist column as it is a scalar multiple of the abdomen column. Furthermore, there is no point in keeping body density in the model since it defeats the purpose of using simple convenient predictors. All other variables will be used as features since correlation and domain knowledge suggest they are useful. We will not perform any scaling since the scale is inconsequential for random forest. Next, we split the data into 75% training and 25% testing so as to keep most of the data for learning. Lastly, we will make some cross validation folds for parameter tuning. Since the data set is quite small, we can use 10-fold cross-validation without computational concerns.

```
# drop waist and density columns
bfat <- bfat[, -c(1, 9)]

# drop suspect points
bfat <- bfat[-c(170, 180, 36, 214), ]

# train/test split
data.split <- initial_split(bfat, prop=0.75)
train <- training(data.split)
test <- testing(data.split)
```

```r
# 10-fold cross-validation
cv <- vfold_cv(train, v=10)
```

(IV) Model Choice

First, I set the recipe stating the response is body fat percentage and the predictors are the 13 remaining variables. Then I decided between using xgboost and random forest. Ultimately, I chose random forest since the data is quite small ($n = 246$). I was concerned that xgboost would be too complicated a model with too many hyperparameters resulting in drastic overfitting. Random forest should be all we need.

```r
# model recipe
model.rec <- recipe(Pct.BF ~ ., data=train)


# random forest model
model.rf <- rand_forest(
  mtry = tune(),
  trees = tune(),
  min_n = tune()) %>%
  set_engine('ranger', importance = "permutation") %>%
  set_mode('regression')


wf <- workflow() %>%
  add_recipe(model.rec) %>%
  add_model(model.rf)
```

(V) Model Tuning

There are 3 hyperparmeters to tune: predictors per split (mtry), number of trees (trees), and min data per node split (min_n). To tune them, we use grid search with 10-fold cross-validation. The grid produced 100 hyperparameter combinations. The loss function selected is RMSE for many reasons: we are performing regression, it is smooth and differentiable, it is easily interpretable, and there are few outliers. Table 2 shows the top 5 combinations based on the RMSE loss.

Table 2: Top 5 hyperparamter combinations.

	mtry	trees	min_n
1	9	816	5
2	7	1549	2
3	10	1140	6
4	8	1896	14
5	9	1595	6

```
# hyperparameters
hp.param <- parameters(
  finalize(mtry(), train),
  trees(),
  min_n())

# hyperparameter space
hp.grid <- grid_max_entropy(
  hp.param,
  size = 100)

# tune using hp space and cv folds
hp.tune <- tune_grid(
  object = wf,
  resamples = cv,
  grid = hp.grid,
  metrics = metric_set(rmse),
  control = control_grid(verbose=TRUE))

# display top 5
hp.tune %>% show_best(metric='rmse')
```

(VI) Best Model

From Table 2, we select the hyperparameters from row 1 which produces the smallest RMSE. Then we update the model and train it on the entire training set.

```r
# select best hyperparameters
hp.best <- hp.tune %>% select_best("rmse")

# create best model
model.best <- model.rf %>% finalize_model(hp.best)

wf.best <- workflow() %>%
  add_recipe(model.rec) %>%
  add_model(model.best) %>%
  fit(train)
```

(VII) Model Evaluation

Next, we apply the fitted best model to the test data set to get our predictions. The final RMSE is calculated to be 4.428791 i.e. the average distance of the predictions to the truth is around 4.42% body fat. Figure 3 shows the variable importance plot indicating that the model is heavily relying on the abdomen and not much else. Figure 4 shows the abdomen partial dependence plot which resembles a step function that increases by 5% when circumference increases by 10cm. Overall, the model is not very accurate since the measurements are so crude. However, this is the price paid for being able to do the measurements from home in seconds. If one is measuring the body fat percentage week to week, it might not matter since the relative body fat percentage should be fairly consistent. The variable importance suggests that one might even discard the other measurements and just take the average abdominal circumference every week until an official reading is needed. Note: I was curious at how significant body density would be in the model. Upon running the model with body density, we see that it dominates the other variables and we get a RMSE ≈ 0.85. Furthermore, if I was running the study I would've liked to see skinfold measurements as they are convenient and superior to circumferences and female measurements.

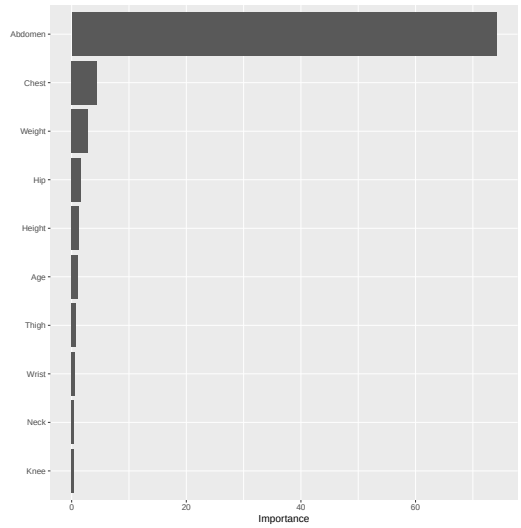

Figure 3: Variable importnace plot.

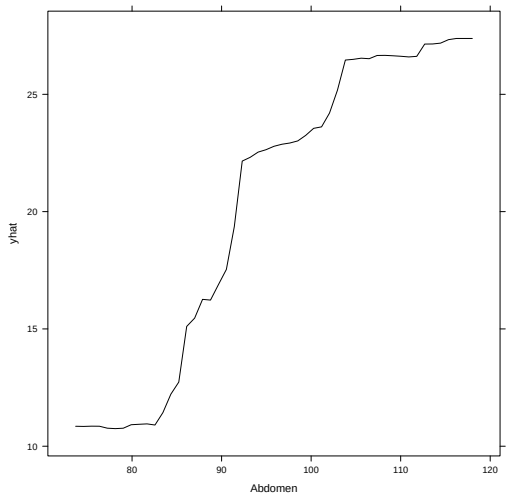

Figure 4: Partial dependence plot.

```
# RMSE on test set
predictions <- as.vector(unlist(predict(wf.best, test)))
residuals <- test$Pct.BF - predictions
RMSE <- sqrt(mean(residuals^2))
RMSE


# variable importance
wf.best %>%
  extract_fit_parsnip() %>%
  vip()
```

```r
# partial dependence
wf.best %>%
  extract_fit_parsnip() %>%
  partial(pred.var = "Abdomen", plot = TRUE, train = train)
```

(VIII) References

Below are the references I used to create my pipeline.

[1] Body fat data set from DASL (Data and Story Library). https://dasl.datadescription.com/datafile/bodyfat/

[2] Tidymodels documentation. https://www.tidymodels.org

I read both these tutorials on tidymodels for xgboost which were useful for setting up the structure of my random forest pipeline.

[3] "Using XGBoost with Tidymodels" https://www.r-bloggers.com/2020/05/using-xgboost-with-tidymodels/

[4] "Tune XGBoost with tidymodels" https://juliasilge.com/blog/xgboost-tune-volleyball/

Question 2:

a) First, suppose we have the MSE loss function $L(y, \hat{y}) = (y_i - \hat{y})^2$. The pseudo-residuals are calculated as

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}} = 2(y_i - f_{m-1}(x_i))$$

Fitting a regression tree to the pseudo-residuals results in terminal regions R_{jm}. Then the weights are calculated as

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

Taking the derivative with respect to γ and setting it to zero we have

$$0 = -2 \sum_{x_i \in R_{jm}} (y_i - f_{m-1}(x_i) - \gamma)$$

$$\iff \sum_{x_i \in R_{jm}} \gamma = \sum_{x_i \in R_{jm}} (y_i - f_{m-1}(x_i))$$

$$\iff \gamma_{jm} = \frac{\sum_{x_i \in R_{jm}} (y_i - f_{m-1}(x_i))}{|R_{jm}|}$$

Therefore, for MSE loss γ_{jm} is the mean of the residuals in leaf j at tree iteration m.

Now suppose we have the binomial deviance loss function. ESL page 346 shows that if we take $y \in \{0, 1\}$ we have

$$L(y, \hat{y}) = \log(1 + e^{\hat{y}}) - y\hat{y}$$

The pseudo-residuals are calculated as

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}} = y_i - \frac{e^{f_{m-1}(x_i)}}{1 + e^{f_{m-1}(x_i)}} = y_i - p_{m-1}(x_i)$$

Now we calculate

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

by taking the derivative with respect to γ and setting it to zero.

$$0 = \sum_{x_i \in R_{jm}} \frac{1}{1 + e^{-(f_{m-1}(x_i)+\gamma)}} - y_i$$

Since γ is in the denominator, there is no closed form solution. We obtain γ_{jm} using the second order approximation in part b).

b) If we use second-order approximations for the weights we have

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

$$\approx \arg\min_{\gamma} \sum_{x_i \in R_{jm}} [L(y_i, f_{m-1}(x_i)) + \frac{\partial}{\partial f_{m-1}} L(y_i, f_{m-1}(x_i))\gamma + \frac{1}{2}\frac{\partial^2}{\partial f_{m-1}^2} L(y_i, f_{m-1}(x_i))\gamma^2]$$

Using the MSE loss and setting the derivative wrt γ equal to zero gives us

$$0 = 0 + \sum_{x_i \in R_{jm}} -2(y_i - f_{m-1}(x_i)) + \sum_{x_i \in R_{jm}} 2\gamma$$

$$\iff \gamma_{jm} = \frac{\sum_{x_i \in R_{jm}} (y_i - f_{m-1}(x_i))}{|R_{jm}|}$$

Thus, the second order approximation for MSE is identical to the result derived in part a).

For the binomial deviance loss function we have

$$0 = 0 + \sum_{x_i \in R_{jm}} \frac{e^{f_{m-1}(x_i)}}{1 + e^{f_{m-1}(x_i)}} - y_i + \sum_{x_i \in R_{jm}} \frac{e^{f_{m-1}(x_i)}}{(1 + e^{f_{m-1}(x_i)})^2}\gamma$$

$$\iff \gamma_{jm} = \frac{\sum_{x_i \in R_{jm}} y_i - p_{m-1}(x_i)}{\sum_{x_i \in R_{jm}} p_{m-1}(x_i)(1 - p_{m-1}(x_i))}$$

Thus, the weights for the binomial deviance loss function are the sum of the residuals over the sum of the predicted probabilities for the leaf j at iteration m.

References

The first two are the given references.

[1] Bujokas, Eligijus. 2022. "Gradient Boosting in Python from Scratch." Medium. https://towardsdatascience.com/gradient-boosting-in-python-from-scratch-788d1cf1ca7.

[2] Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–94. https://doi.org/10.1145/2939672.2939785.

[3] ESL page 346 provides many details for classification loss functions.

[4] YouTube video "Gradient Boost Part 4 (of 4): Classification Details" Goes through some of the details in Algorithm 10.3 for classifcation. https://www.youtube.com/watch?v=StWY5QWMXCw