College Students and Weight

Sydney Nwakanma

April 25, 2021

Table of Contents

1. Abstract

Since moving away from their parent's nest, college students are tasked with paving their own way and making independent decisions in their new environments. Food culture is prevalent in many college campuses as students often rely on fast food and processed cafeteria food to get by. However, some do prioritize healthy balanced diets. This project will focus on food culture and the eating habits of the college students at Mercyhurst University in Erie, Pennsylvania in 2017. The objective is to predict weight using different factors associated with food behaviors among college students.

2. Data Introduction

2.1. Data cleaning

The data set is initially extracted from Kaggle.com online. However, the same dataset can be found at various websites online. It includes 126 observations and 61 variables with 12 of them consisting of categorical data and the rest with numerical values. Some variables of significance include, but are not limited to exercise, GPA, gender, employment status, if they live on or off campus, and how often the student cooks. The data is downloaded from an Excel Google Spreadsheet and imported into R as a CSV file.

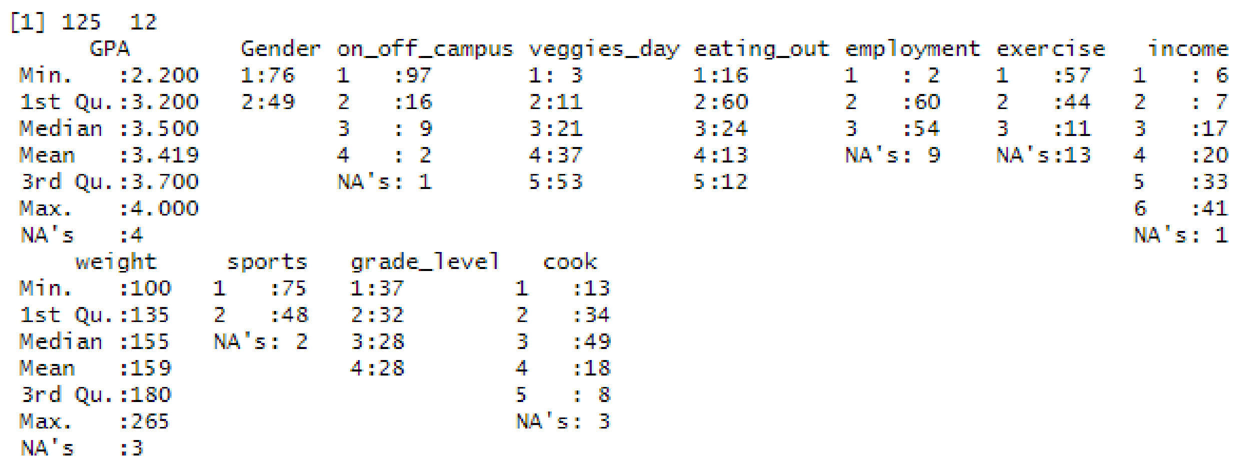
As part of the data cleaning process, the string values are replaced as factors. There are about 15 variables like father's profession and what type of sports they play that contain open ended responses and commentary. These are removed due to lack of significance. In the end, only 12 variables are chosen to continue with.

2.2. Variables

`Weight` is chosen as the numerical response variable. The predictors include `employment`, `GPA`, `Gender`, if the student lives on or off campus (`on_off_campus`), if the student plays sports(`sports`), if the student exercises (`exercise`), if the student cooks (`cook`), `income`, how likely they are to eat vegetables per day (`veggies_day`), and how often they eat out (`eating_out`). The variable `Gender` is binary with female set as 1 and male as 2. `GPA` records the actual GPA of students on a 4.0 scale. `Sports` is a categorical variable with 1 representing if the student plays sports and 2 if the student doesn't. `Employment` takes the values of 1 as full time, 2 as part time, 3 as no employment, and 4 as other. `Exercise` takes on a scale from 1 to 5 with 1 being everyday and 5 being never. `Veggies_day` also takes on a scale of 1 to 5 with 1 being for very unlikely and 5 being very likely. Observations for `income` are on a scale from 1 to 5 with 1 being anything less than $15,000 and a 5 being any amount higher than $100,000. The variable, `eating_out`, takes on a scale from 1 to 5 with 1 being never and 5 being everyday. Last but not least, observations for `cook` are on a scale from 1 to 5 with 1 being everyday and 5 being never.

2.3. Summary

To summarize the data, the `summary()` function is used, which produces an output of the chosen 12 variables and the quantity of observations that correspond to each level. For instance, `Gender` reports 76 female participants, which is represented with a 1, and 49 male participants, which is a 2. Figure 1 shows the summary output of the data.



```
[1] 125  12
      GPA          Gender  on_off_campus  veggies_day  eating_out  employment  exercise    income
 Min.   :2.200   1:76   1   :97       1: 3        1:16       1   : 2    1   :57    1   : 6
 1st Qu.:3.200   2:49   2   :16       2:11        2:60       2   :60    2   :44    2   : 7
 Median :3.500          3   : 9       3:21        3:24       3   :54    3   :11    3   :17
 Mean   :3.419          4   : 2       4:37        4:13       NA's: 9    NA's:13    4   :20
 3rd Qu.:3.700          NA's: 1       5:53        5:12                             5   :33
 Max.   :4.000                                                                     6   :41
 NA's   :4                                                                         NA's: 1
     weight        sports     grade_level    cook
 Min.   :100   1   :75    1:37        1   :13
 1st Qu.:135   2   :48    2:32        2   :34
 Median :155   NA's: 2    3:28        3   :49
 Mean   :159              4:28        4   :18
 3rd Qu.:180                          5   : 8
 Max.   :265                          NA's: 3
 NA's   :3
```

Figure 1: Dimensions and summary of data with 12 variables

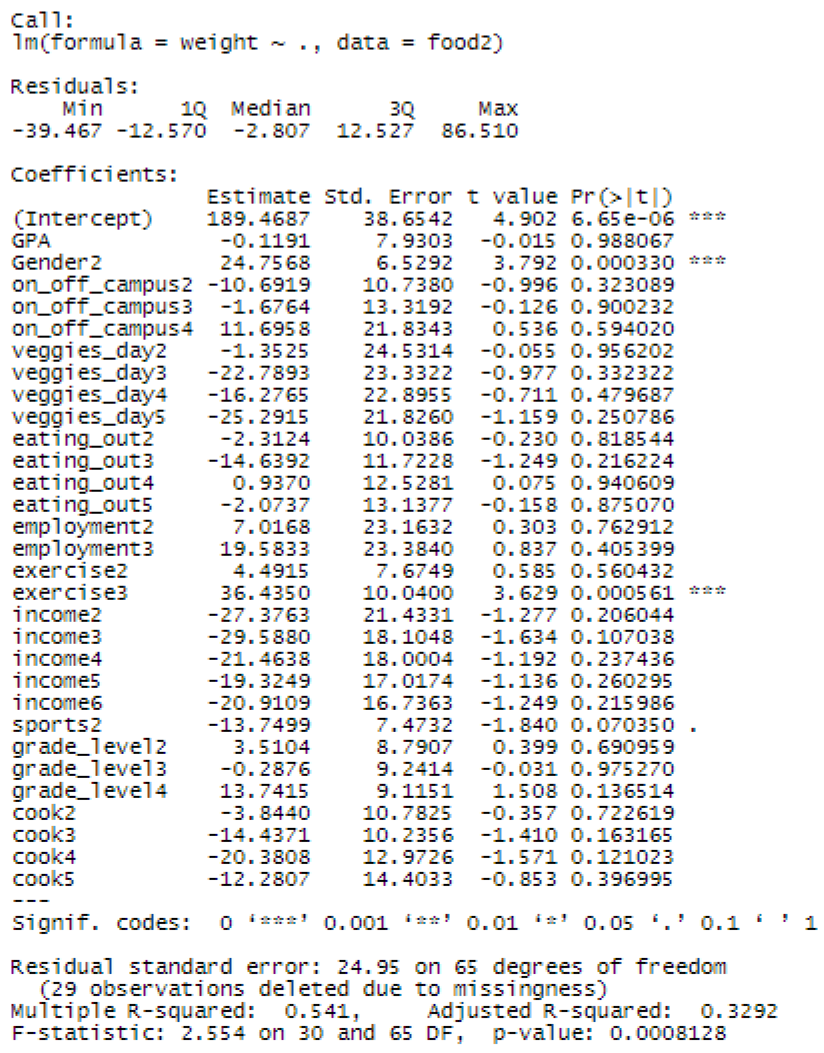
2.3.1. Descriptive Statistics

In order to summarize the basic features of the data and relate them to the response variable, `weight`, six variables are chosen. These variables include `Gender`, `veggies_day`, `employment`, `sports`, `GPA`, `and` `grade_level`. Using the mean function to find the average weight of female and male participants, it is shown that men have an average weight of 162.15 while women have an average weight of 151.8. The mean function is continuously used for the rest of the five variables. The variable, `veggies_day`, reported an average weight of 162.5 for people who are very unlikely to eat vegetables per day, a 175 for people who are just unlikely, 165.4 for people who are moderate, 168.9 for those just likely, and 149.6 for those who are very likely to eat vegetables everyday. The relationship between sports and weight shows that people who play sports had an average weight of 163.9, and people who don't play sports had an average weight of 153.6. For the grade_level variable, we have seniors reporting the highest average weight with 173.8, and freshmen having the second highest with an average weight of 160.3. Juniors and sophomores had average weights of 153.9 and 155.3, respectively. Participants who reported exercising everyday had an average weight of 159.62.

Those who reported exercising two or three times a week had an average weight of 155.7, and participants that reported once a week had an average weight of 179.7. For the correlation between employment status and weight, participants who worked part-time had an average weight of 155.1, and those who had no employment had an average weight of 167.7.

3. Linear Regression

A simple linear regression model is fitted to model the relationship between weight and the predictors, using the lm() function in R. The figure below illustrates the variables that had some level of significance. The categories in each variable are split up and treated as their own variable. For instance, `Gender2` represents male, and it shows in Figure 2 that it is statistically significant with a p-value of 0.000330. To find detailed information about the model, the summary(lm.fit) is used, which produces the R^2, the F-statistic, the p-values, and the standard error for the coefficients. Figure 2 displays the linear model output, which shows a residual standard error of 24.95 and an F-statistic of 2.544.



```
Call:
lm(formula = weight ~ ., data = food2)

Residuals:
    Min      1Q  Median      3Q     Max 
-39.467 -12.570  -2.807  12.527  86.510 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)    189.4687    38.6542   4.902 6.65e-06 ***
GPA             -0.1191     7.9303  -0.015 0.988067    
Gender2         24.7568     6.5292   3.792 0.000330 ***
on_off_campus2 -10.6919    10.7380  -0.996 0.323089    
on_off_campus3  -1.6764    13.3192  -0.126 0.900232    
on_off_campus4  11.6958    21.8343   0.536 0.594020    
veggies_day2    -1.3525    24.5314  -0.055 0.956202    
veggies_day3   -22.7893    23.3322  -0.977 0.332322    
veggies_day4   -16.2765    22.8955  -0.711 0.479687    
veggies_day5   -25.2915    21.8260  -1.159 0.250786    
eating_out2     -2.3124    10.0386  -0.230 0.818544    
eating_out3    -14.6392    11.7228  -1.249 0.216224    
eating_out4      0.9370    12.5281   0.075 0.940609    
eating_out5     -2.0737    13.1377  -0.158 0.875070    
employment2      7.0168    23.1632   0.303 0.762912    
employment3     19.5833    23.3840   0.837 0.405399    
exercise2        4.4915     7.6749   0.585 0.560432    
exercise3       36.4350    10.0400   3.629 0.000561 ***
income2        -27.3763    21.4331  -1.277 0.206044    
income3        -29.5880    18.1048  -1.634 0.107038    
income4        -21.4638    18.0004  -1.192 0.237436    
income5        -19.3249    17.0174  -1.136 0.260295    
income6        -20.9109    16.7363  -1.249 0.215986    
sports2        -13.7499     7.4732  -1.840 0.070350 .  
grade_level2     3.5104     8.7907   0.399 0.690959    
grade_level3    -0.2876     9.2414  -0.031 0.975270    
grade_level4    13.7415     9.1151   1.508 0.136514    
cook2           -3.8440    10.7825  -0.357 0.722619    
cook3          -14.4371    10.2356  -1.410 0.163165    
cook4          -20.3808    12.9726  -1.571 0.121023    
cook5          -12.2807    14.4033  -0.853 0.396995    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 24.95 on 65 degrees of freedom
  (29 observations deleted due to missingness)
Multiple R-squared:  0.541,	Adjusted R-squared:  0.3292 
F-statistic: 2.554 on 30 and 65 DF,  p-value: 0.0008128
```

Figure 2: Summary of linear regression model.

4. Best Subset Selection

In an attempt to find the subset of my independent variables that best predict weight, the regsubsets() function is used. The main idea is to perform best subset selection by identifying the best model that contains a given number of predictors, of which the best model is quantified using RSS. The summary() function is then used to produce the best set of variables for each model size. By default, the regsubsets () function only outputs the best eight-variable model. RSS model had an output of 1 variable, adjusted R^2 squared had an output of 8 variables, C*p* model had an output of 6 variables, and Bayesian Information Criterion (BIC) model had an output of 2 variables.

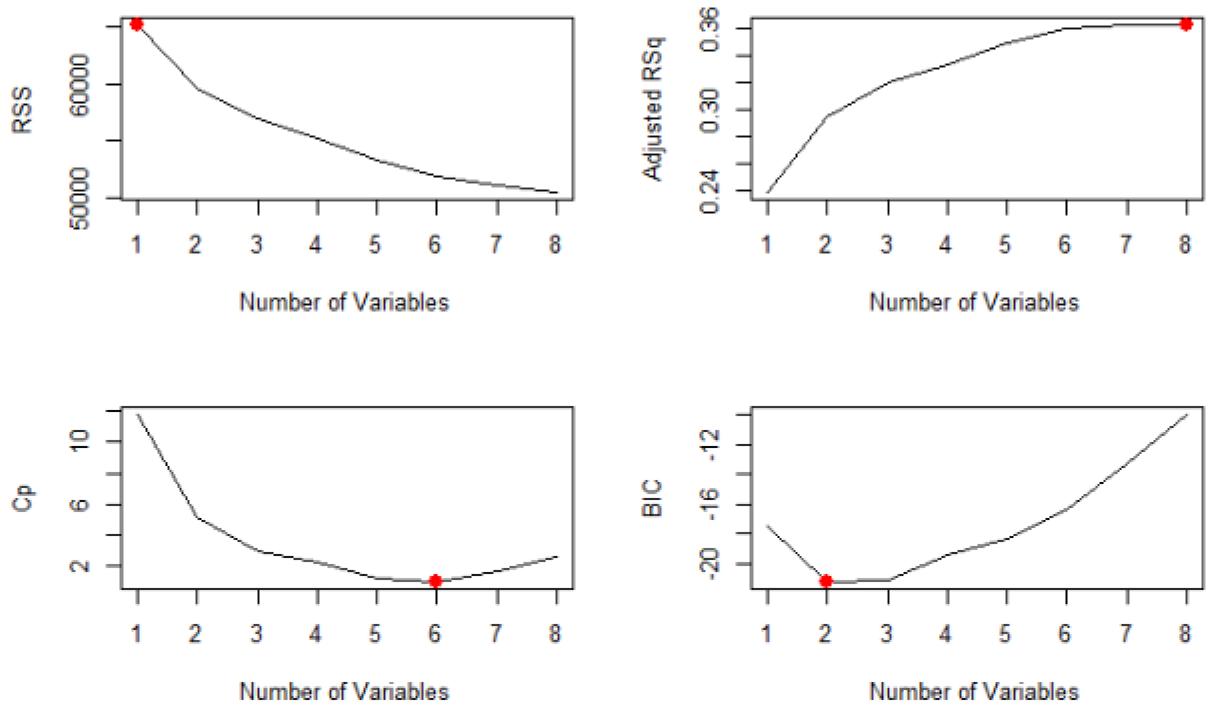


Figure 3: RSS, adjusts R², C*p,* and BIC are shown for the best models of each size for the data.

Next is the plotting of RSS, adjusted R², C*p*, and BIC for all the models to show which model to select. The plot() function creates a line plot in Figure 3, and the which.max() function is used to identify the location of the maximum point of the vector, which is indicated by a red dot. To display the coefficient estimates associated with each model, the coef() function is used. For instance, the C*p*, the six-variable model, contains variables `Gender2,veggies_day2, eating_out3,employment3,exercise3, grade_level4`. As stated previously, the model splits each category in each variable, and presents them as their own variable. Here `Gender2` stands for male, `employment3` represents those that are unemployed, `exercise3` denotes participant that exercise once a week, `veggies_day2` are those that are unlikely to eat vegetables everyday, `eating_out3` represents people who eat out two or three times a week, and `grade_level4` represents college seniors.

4.1. Forward and Backward Selection

The regsubsets() function is also used to perform forward and backward stepwise selection. Forward selection begins with an empty model, and predictors are added one by one, beginning with the predictor that has the highest correlation with `weight`. Backward selection, on the other hand, begins with a full model and predictors are eliminated one at a time in order to find the model that best explains the data. Using the argument, method=”forward”, for forward selection and method=”backward” for backward selection, it shows that the best one-variable model contains `Gender2`, and the best two-variable model adds `grade_level4`. The best six-variable models produced different outputs between best subset selection and forward and backward selection.

5. Regression Tree

A regression tree-based method is used to further analyze the data. The goal is to predict a student’s weight based on the 12 factors provided. The seed value is set to 1 first, and then missing values are removed from the data. After installing the tree package in R, 80 percent of the data is used to create a training set with the rest as the test set. The regression tree consists of a set of splitting rules that start at the top of the tree. `Gender1`, which is interpreted as the most important factor in predicting weight, sits at the top split, and is assigned to the left side of the branch. Figure 4 illustrates that the predicted weight for women that are juniors is 132.8.

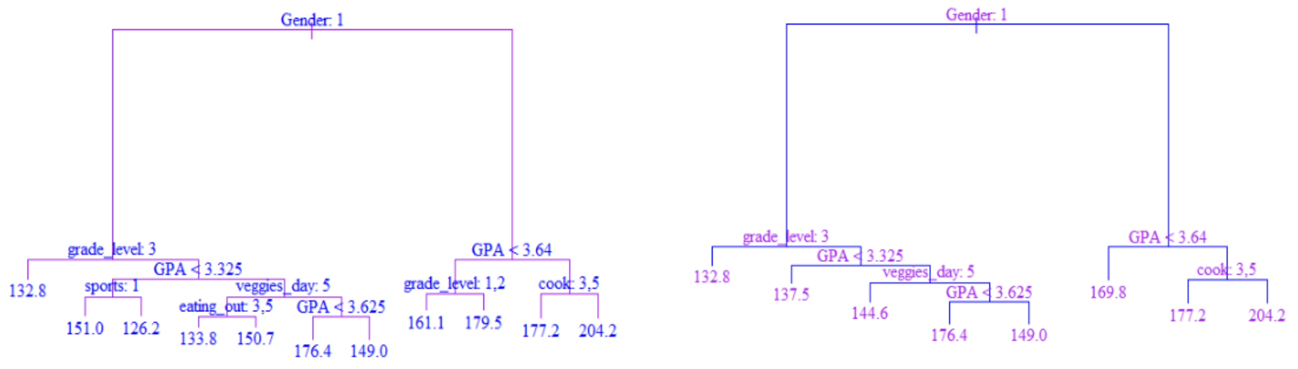


Figure 4: Unpruned (left) and pruned (right) regression tree for predicting weight

The summary output shows that only seven variables are used in constructing the tree with 11 terminal nodes and a residual mean deviance 351.9, which is the sum of squared errors for the tree. In plotting the tree, the plot() function is used, and the cv.tree() function is used to test whether pruning the tree will improve performance. To prune the tree, the prune.tree() function is used. However, the unpruned tree is necessary to make predictions on the test set. In conclusion, the test MSE for this regression tree is 1870.4, and the square root of the MSE is 43.2.

6. Bagging and Random Forests

Next, bagging and random forests are applied to the data, using the randomForests package in R. Similar methods are used for both bagging and random forests. Bagging is used to avoid overfitting and reduce variance in the data, and random forests are also used to reduce variance in the data. The difference between bagging and random forests is the choice of *m*, which is the predictor size. A small value of *m* is typically used in random forests, while bagging takes a larger value. In bagging the argument mtry=11 suggests that all 11 variables are used for each split of the tree, while the argument mtry=6 is chosen for random forests. Setting the seed value to 1 again and still using 80 percent of my data as the training set, the test set MSE is calculated. For bagging the test MSE is 1436.8 with the root MSE as 37.9. For random forests the test MSE is 1416.2 with a root MSE as 37.6. Therefore, it can be concluded that random forests performed better than regression trees and slightly better than bagging.

7. Conclusion

The best prediction method for this kind of data that consists of both numerical and categorical variables proved to be decision trees. While the data set is small with only 125 observations and initially, 61 predictors, it is still possible to find significant correlations between the predictors and the response variable, `weight`. Decision trees provide a picture representation of students' weight given different factors. Whereas, best subset selection introduced the best subset of the independent variables that best predicted weight. The goal with the MSE was to achieve a good balance between overfit and underfit, and as expected, random forests performed the best out of the three decision trees by having the lowest MSE. In the future, a much larger sample size will be considered in order to avoid more errors and inaccurate mean values. In addition, a larger sample size in a topic such as college students and weight can provide more insight into food culture on college campuses that could support research into related problems and solutions to these problems.

8. References

[1] James, Gareth, et al. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2021.

9. Code

```{r}

library("readxl")

library("car")

library(ISLR)

install.packages("tree")

library(tree)

install.packages("glmnt")

install.packages("randomForest")

library( randomForest)

```

Summary of data. Change categorical variables to factors.

```{r}

food = data.frame(read.csv("food_original.csv",stringsAsFactors = T))

#summary(food)

food1 = subset(food, select = -c(comfort_food_reasons_coded, calories_day,

cuisine,comfort_food,comfort_food_reasons,food_childhood, father_profession,

mother_profession,fav_cuisine,type_sports,diet_current,eating_changes,healthy_meal,ideal_diet,

meals_dinner_friend) )
```

~~~
#summary(food_remove_variables)
```
~~~

Reduce data to just 12 variables. remove characters and strings.

~~~
```{r}
food2=subset(food1,select=c(GPA,Gender,on_off_campus,veggies_day,
eating_out,employment,exercise,income,weight,sports,grade_level,cook))
#dim(food_remove_variables)
#food2 = na.omit(food1)
for (i in c(2:8,10:12))
{
  food2[,i]=as.factor(food2[,i]) #convert categorical variables to factors
}

dim(food2)
summary(food2)

```
~~~

~~~
```{r}
#count gender
#count female = 1
~~~

~~~
female = length(subset(food2$Gender, food2$Gender == 1))
female


#count male = 2
male = length(subset(food2$Gender, food2$Gender == 2))
male
```
```{r}
#average of female and male weight
#average of female weight
food3 <- na.omit(food2)
female_weight = mean(subset(food3$weight, food3$Gender == 1))
female_weight


#average of male weight
male_weight = mean(subset(food3$weight, food3$Gender == 2))
male_weight


#find max of weight
weight_max = max(food3$weight)
weight_max
```
~~~

~~~
```{r}

#GPA and Weight

food3 <- na.omit(food2)

female_weight = mean(subset(food3$weight, food3$GPA > 3.0))

female_weight


#average of male weight

male_weight = mean(subset(food3$weight, food3$GPA < 3.0))

male_weight

```

```{r}

#veggies and weight

food3 <- na.omit(food2)

veggies_weight = mean(subset(food3$weight, food3$veggies_day == 1))

veggies_weight


veggies_weight = mean(subset(food3$weight, food3$veggies_day == 2))

veggies_weight


veggies_weight = mean(subset(food3$weight, food3$veggies_day == 3))

veggies_weight


veggies_weight = mean(subset(food3$weight, food3$veggies_day == 4))
~~~

~~~
veggies_weight


veggies_weight = mean(subset(food3$weight, food3$veggies_day == 5))

veggies_weight


#Average weight of people who have the highest likelihood of eating veggies everyday


```

```{r}

#sports and weight

#1 for yes and 2 for no

food3 <- na.omit(food2)

sports_weight = mean(subset(food3$weight, food3$sports == 1))

sports_weight


sports_weight = mean(subset(food3$weight, food3$sports == 2))

sports_weight

```




```{r}

#grade level and weight

#1 for fresh, 2 for soph, 3 for jun, 4 for sen
~~~

~~~~
food3 <- na.omit(food2)
grade_level_weight = mean(subset(food3$weight, food3$grade_level == 1))
grade_level_weight

grade_level_weight = mean(subset(food3$weight, food3$grade_level == 2))
grade_level_weight

grade_level_weight = mean(subset(food3$weight, food3$grade_level == 3))
grade_level_weight

grade_level_weight = mean(subset(food3$weight, food3$grade_level == 4))
grade_level_weight


```

```{r}
#exercise and weight
#1 for everyday, 2 for 2/3 times a week, 3 for once a week
food3 <- na.omit(food2)
exercise_weight = mean(subset(food3$weight, food3$exercise == 1))
exercise_weight
~~~~

~~~
exercise_weight = mean(subset(food3$weight, food3$exercise == 2))
exercise_weight


exercise_weight = mean(subset(food3$weight, food3$exercise == 3))
exercise_weight
```
```{r}
#employment and weight
# 1 for full time, 2 for part time, 3 for no
food3 <- na.omit(food2)
exercise_weight = mean(subset(food3$weight, food3$employment == 2))
exercise_weight


exercise_weight = mean(subset(food3$weight, food3$employment == 3))
exercise_weight
```


```{r}
food3 <- na.omit(food2)
exercise_weight = mean(subset(food3$weight, food3$cook == 1))
exercise_weight
```
~~~

Best Subset selection: for the best predictors. Use food2 with the 12 variables from here on out.

```{r}
library(ISLR)
#objective: predict weight.
#View(food2) #note missing values
#names(food2)
#dim(food2)
#sum(is.na(food2$weight))


#dim(food2)
#sum(is.na(food))


lm.fit <- lm(weight~.,food2)
summary(lm.fit)
```

```{r}
library(leaps)#one library for subset selection
```

~~~
#regfit.full=regsubsets(weight~.,food2, really.big = T) #best subset selection

#food_no_na$weight.1=NULL

#food_no_na$weight.2=NULL

regfit.full=regsubsets(weight~.,food2,really.big = T) #best subset selectio

#summary(regfit.full)

#default output only reports results up to the best 8 variable model

#by you can specify the number using option nvmax

#regfit.full=regsubsets(weight~.,data=food2,really.big = T)

reg.summary=summary(regfit.full)

```


```{r}

names(reg.summary)

reg.summary$rsq

```

#R squared increases from almost 30 % to 65.5 % as more variables are added


```{r}

par(mfrow=c(2,2))




plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")

#which.max(reg.summary$rss)
~~~

~~~
points(1,reg.summary$rss[1], col="red",cex=2,pch=20)


plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
#which.max(reg.summary$adjr2)
points(8,reg.summary$adjr2[8], col="red",cex=2,pch=20)


plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
#which.min(reg.summary$cp)
points(6,reg.summary$cp[6],col="red",cex=2,pch=20)


#which.min(reg.summary$bic)
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
points(2,reg.summary$bic[2],col="red",cex=2,pch=20)
```
~~~

Graphs shows how many variables are best from each model.

```{r}
plot(regfit.full,scale="r2")
plot(regfit.full,scale="adjr2")
plot(regfit.full,scale="Cp")
plot(regfit.full,scale="bic")
```

```{r}
```

~~~
#see the coefficient estimates estimated with the model

coef(regfit.full,6) #6 variables

```


Will forward and backward stepwise selection yield the same results as best selection?


```{r}

regfit.fwd=regsubsets(weight~.,data=food2,method="forward")

summary(regfit.fwd)

regfit.bwd=regsubsets(weight~.,data=food2,method="backward")

summary(regfit.bwd)

#coef(regfit.full,6)

#coef(regfit.fwd,6)

#coef(regfit.bwd,6)

```

#No, it didn't give the same results.




Linear Regression


```{r}
~~~

~~~
#make weight numeric instead of character
#weight_num <- as.numeric(food_remove_variables$weight)
#w= na.omit(weight_num)
lm.fit=lm(weight ~.,food_no_na)
summary(lm.fit)


train=sample(nrow(food),0.8*nrow(food)) #seperate data with 80 percent training set and the rest at the test set
train.data=food2[train,]
test.data=food2[-train,]

```
~~~

Regression trees

~~~
```{r}
###############################
# Fitting Regression Trees###
###############################
~~~

~~~
library(tree)
set.seed(1)
food2=food2[!is.na(food2$weight),]
train = sample(1:nrow(food2), nrow(food2)*0.8) #use 80 percent of the data as the training set
tree.food2=tree(weight~.,food2,subset=train)
summary(tree.food2)
#in regression, deviance is the RSS
```


```{r}
windowsFonts(A = windowsFont("Times New Roman"))
plot(tree.food2, col = "purple")
text(tree.food2,pretty=0, col = "blue", family = "A")



cv.food2=cv.tree(tree.food2)
plot(cv.food2$size,cv.food2$dev,type='b', col = "purple", family = "A")


#pruning the tree
prune.food2=prune.tree(tree.food2,best=6)
plot(prune.food2, col = "blue")
text(prune.food2,pretty=0, col = "purple", family = "A")
~~~

~~~
yhat=predict(tree.food2,newdata=food2[-train,])

food2.test=food2[-train,"weight"]

plot(yhat,food2.test, col = "purple", family = "A")

abline(0,1, col = "blue")




mean((yhat-food2.test)^2)#test MSE

sqrt(mean((yhat-food2.test)^2)) #RMSE is 40.8
```
~~~

Bagging

~~~
```{r}
library(randomForest)

set.seed(1)

food3 <- na.omit(food2)

train3 <- sample(1:nrow(food3),0.8*nrow(food3))

bag.food2= randomForest(weight~.,data=food3, subset=train3,mtry=11,importance =TRUE)

#The argument mtry=13 indicates that all 13 predictors should be considered for each split of the
tree—in other words, that bagging should be done
~~~

~~~
bag.food2
```
~~~

~~~
```{r}
yhat.bag = predict (bag.food2 , newdata=food3[-train3 ,])
food2.test=food3$weight[-train3]
plot(yhat.bag ,food2.test, col = "blue", family = "A")
abline (0,1, col = "purple")
mean((yhat.bag -food2.test)^2)#MSE
sqrt(mean((yhat.bag -food2.test)^2))#RMSE decreased to 37.9, so it's better
```
~~~

Random Trees

~~~
```{r}
set.seed(1)
rf.food2= randomForest(weight~.,data=food3, subset=train3 , mtry=6, importance =TRUE)
yhat.rf = predict(rf.food2 ,newdata=food3[-train3 ,])
mean((yhat.rf-food2.test)^2)
sqrt(mean((yhat.rf -food2.test)^2))
```
~~~