Machine Learning Using R Final Project

Utilizing ML Algorithms to Predict BMI Status

Christopher Johnson and Laurence Giglio

UCLA Extension, Summer 2018

Acknowledgments

We wish to thank Michael Chang for providing his feedback, insights, advice and consultation time.

Table of Contents

1 Introduction

In the United States, a substantial number of people have devastating health issues. Tragically, many of those issues are preventable. The Centers for Disease Control and Prevention [1] has some alarming statistics on current health issues in the US:

1. **Body Health:** 70%+ of adults over 20 are overweight or obese
2. **Sleep:** 30%+ of adults do not get the recommended amount of sleep
3. **Mental Health:** 10% of physician office visits indicated depression on the medical record.

It is important to understand who these issues affect and the risk factors involved so we can help treat/prevent those health issues. There has been a significant amount of work done in this space, but we have a long way to go given our nation's health crisis, especially with obesity. The prevalence of obesity has been growing for several decades and now nearly 1 out of every 3 American adults is obese. Individuals suffering from obesity are at a higher risk of diabetes, cardiovascular disease, cancer and many other health care problems [2, 3], limiting life expectancy. It is vital that we continue to search for predictors and at-risk populations so that we can effectively combat obesity.

2 Machine Learning Problem Description

We will be using machine learning algorithms to predict if a given patient is overweight or obese, based on numerous variables such as age, cholesterol levels, and sleep habits. Overweight/obese will be determined by a BMI value that is greater than or equal to 25. Predicting the BMI status of patients will be a classification problem, so we will be converting BMI values into a 2-factor variable ("Yes", "No").

We will be evaluating the models, built using training data, with confusion matrices based on test data. We are most interested in overall prediction accuracy for the models. This metric is appropriate given Overweight/Obese patients are 56% of the total dataset, which means we are not as concerned about Type I and Type II errors as we would be for more unbalanced datasets. We will also be evaluating other aspects of the models, including total number of variables used and computational efficiency.

3 Data Source, Data Cleaning and Exploratory Analysis

3.1 Data Source and Variables Added

We used the CDC's NHANES dataset, which is survey data collected from 10,000 patients from 2009-2012. The division of the CDC that performed the survey is the US National Centre for Health Statistics (NCHS), which has been doing surveys on health since the 1960s. Key information in the NHANES dataset is:

1. 10,000 patient records sampled to represent the US population
2. 75 variables for each patient (Plus 1 added, discussed below)
3. Format is an R library (NHANES) [14]

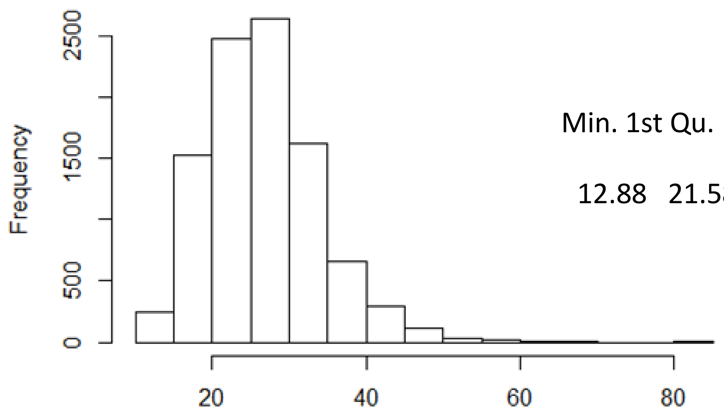
3.2 Data Cleaning and Variables Added

After exploring the dataset and deciding on our goal of predicting if a patient's BMI value indicates Overweight/Obese (i.e. BMI is greater than or equal to 25), we removed all records with a BMI = "NA." This process removed 366 of 10,000 records, meaning 96%+ of the original data remained. We decided to keep outlier BMI values because we are building classification algorithms. If we were building regression models we would have removed the outlier BMIs because they would have significantly distorted MSE outputs (we are using overall accuracy from a confusion matrix to measure error).

We added a new variable called "BMIOver25", which has possible values of "Yes" or "No". It is based on the numerical value of the original variable "BMI". "BMIOver25" is what will be our Y value. We then narrowed down the dataset to include only numerical and integer values for our X variables (eliminating factor variables). We chose 18 variables of interest to include as inputs, bringing total variable count of 19 inclusive of "BMIOver25". For input records = "NA", we replaced the "NA" with the median value for the column (vs. omitting the record entirely). This adjustment was made so that we would not delete the majority of the records in the dataset. Support code is included in Appendix 1 for reference.

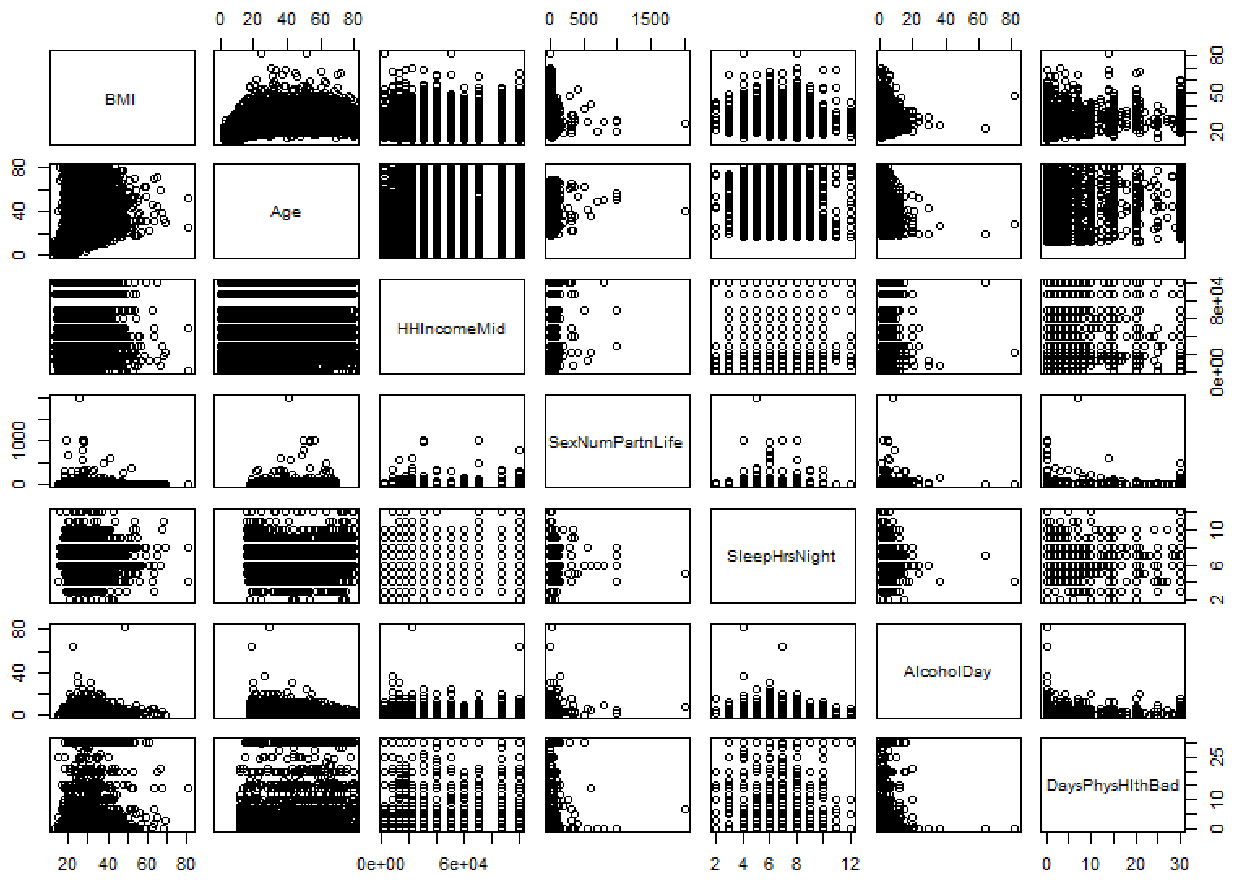
3.3 Exploratory Data Analysis

3.3.1 Histogram and Summary Statistics of BMI in NHANES



```
Min. 1st Qu.  Median   Mean 3rd Qu.   Max.   NA's
 12.88   21.58   25.98   26.66   30.89   81.25    366
```

3.3.2 EDA- Matrix Scatterplot for Pairs of Variables of Interest



4 Feature Engineering and Selection

4.1 Feature Engineering

Feature engineering is the process of transforming the gathered data into features that better represent the problem that we are trying to solve to the model in order to improve its performance and accuracy. We converted the numerical variable “BMI” into a 2-factor variable called “BMIOver25” to allow us to perform classification. For our KNN and SVM models, scaling the X inputs was vital to ensure the models were working correctly. We used the scale() function on the dataset for the KNN model. For the SVM model, we were able to use the built-in “scale” argument in the svm() function. We did not focus on interaction terms in this project scope primarily due to capacity constraints and the large number of initial variables but given additional time we would analyze opportunities to improve performance by doing so.

4.2 Feature Selection

4.2.1 All Models

The initial NHANES dataset has nearly 80 input variables to choose from. We focused on using numerical variables (vs. factors) for our inputs. Doing so allowed us to more easily clean/replace “NA” values where appropriate as well as avoid using dummy variables when working with complex algorithms such as the SVM. Using this approach, we narrowed our initial dataset down to 18 input variables. We’ve included a brief summary of features selection by model is below, with more detail in the appendix.

4.2.2 Logistic Regression Model

Our initial model included all 18 input variables. Then, all input variables with p-values >.05 were removed for statistical significance reasons, leaving 8 input variables in the final model.

4.2.3 Classification Tree Model

Classification tree algorithms automatically select variables to use in tree construction, so we started with all 18 input variables. In this case, the algorithm kept only 2 input variables with 5 terminal nodes. We then performed cross-validation to find the optimal number of nodes, which turned out to be 5.

4.2.4 KNN Model

All 18 input variables in this model, but as mentioned these input variables were scaled to improve model performance.

4.2.5 SVM Model

For the first SVM model, we used all 18 input variables and scaled them. In order to perform computationally intensive parameter tuning, we then reduced the number of input variables to 4. Th variables selected had the lowest p-values on the logistic regression model. Using this smaller dataset allowed us to tune the model’s cost function.

5 Modelling

The four machine learning algorithms chosen were:

1. Logistic Regression [8]
2. Classification Tree [9]
3. K Nearest Neighbors (KNN) [10]
4. Support Vector Machine (SVM) [11]

5.1 Model 1 – Logistic Regression

The logistic regression model [8] was fitted with the full data set. The model was then trained with training data and evaluated with test data giving an accuracy of 74.6%. A new model built with the 8 variables that had p-values < 0.05 on the prior model. The new model yielded a test accuracy of 75.1%, but used less than half the variables in the previous model. The model was finally tested with 3 different probability thresholds. The best logistic regression model contained 8 variables, had a 50% probability threshold setting, and scored a test accuracy of 75.1%.

# Logistic Regression- Best Model

```
Coefficients:
                   Estimate Std. Error z value Pr(>|z|)    
(Intercept)       0.2727874  0.2965338   0.920 0.357614    
Age               0.0461866  0.0015448  29.897  < 2e-16 ***
DirectChol       -2.3616900  0.0922705 -25.595  < 2e-16 ***
TotChol           0.4213295  0.0346654  12.154  < 2e-16 ***
DaysMentHlthBad   0.0210524  0.0043989   4.786  1.7e-06 ***
SleepHrsNight    -0.0530973  0.0246823  -2.151 0.031458 *  
AlcoholDay        0.0508076  0.0158672   3.202 0.001364 ** 
SexAge           -0.0285362  0.0100702  -2.834 0.004601 ** 
UrineVol1         0.0012641  0.0003336   3.789 0.000151 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 9222.3  on 6719  degrees of freedom
Residual deviance: 7085.9  on 6711  degrees of freedom
AIC: 7103.9

Number of Fisher Scoring iterations: 4
```

# Test accuracy is 75.1%

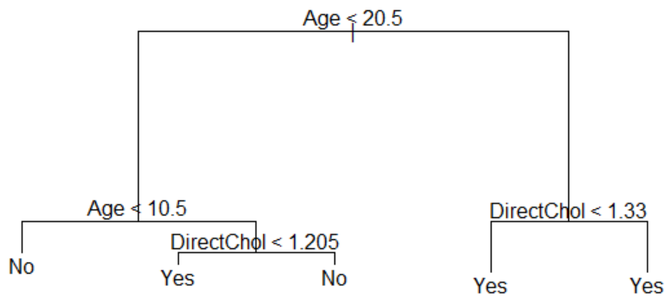
```
       BMITest
glmPred   No  Yes
    No   890  369
    Yes  358 1297
```

5.2 Model 2 – Classification Tree

The classification tree model [9] was fitted with the full data set and plotted. Cross-validation indicated a best tree size of 5. The unpruned tree model was trained using the training data and evaluated with the test data giving a training set accuracy of 73.4% and a test set accuracy of 72.8%. A pruned tree of size 3 was tested for comparison using the test set giving an accuracy of 72%, slightly lower than the tree of size 5 as the cross-validation results predicted.

# Classification Tree- Best Model

```
Variables actually used in tree construction:
[1] "Age"        "DirectChol"
Number of terminal nodes:  5 
Residual mean deviance:  1.035 = 6951 / 6715 
Misclassification error rate: 0.2659 = 1787 / 6720
```



# Test set accuracy of 72.8%

```
BMITest
treePred   No  Yes
     No   536   82
     Yes  712 1584
```

5.3 Model 3 – K Nearest Neighbors (KNN)

The KNN classification model [10] was fitted with the full data set and with k value set at 3 nearest neighbors, giving an accuracy score of 75.3%. The model was then tested with k values of 5 and 7, resulting in test accuracies of 74.9% and 75.0% respectively.

Overall, the KNN model with a k value of 3 was the best model, with a test accuracy result of 75.3%.

# KNN- Best Model

# Test accuracy is 75.3%

```
BMITest
knnPred   No  Yes
    No   911  382
    Yes  337 1284
```

5.4 Model 4 – Support Vector Machine (SVM)

The SVM model [11] was fitted with the full dataset, used a linear kernel, had a cost value setting of 10, and had 5602 support vectors. For faster processing on a notebook, the nhanesSub dataset was created containing BMIOver25 as well as the following 4 variables with the lowest p-values from the logistic regression model: Age, DirectChol, TotChol, DaysMenHlthBad. The new model using the nhanesSub dataset resulted in 5662 support vectors.

Tuning was performed on the model to compare cost values of 0.1, 10 and 100, resulting in the cost value of 0.1 and gamma of 0.25 providing the lowest error. Finally, the model was trained and tested on the split dataset with the optimal cost value of 0.1 resulting in a test accuracy of 74.4%.

# SVM- Best Model

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1
      gamma:  0.25
Number of Support Vectors:  3932
 ( 1967 1965 )
```

```
Number of Classes:  2
Levels:
 No Yes
```

# Test accuracy is 74.4%

```
  truth
predict   No  Yes
    No   876  374
    Yes  372 1292
```

6 Conclusion

6.1 Model Recommendations

After reviewing the model results, we recommend using the logistic regression model to estimate if a patient is overweight/obese. The logistic regression model offers several advantages: it calculates relatively quickly, is easier to interpret and explain than the more complicated algorithms and had the second highest test accuracy scores out of all the models. It also used 8 variables, all of which were selected based on significant p-values.

The KNN model had the highest test accuracy, but we had a few concerns: it used all 18 variables to reach that score, there was a high drop off of accuracy from training to test set, it is less straightforward to explain, and the variables need to be scaled. The SVM model's main issue is that runtime increases significantly as variables are added, which limits the number of variables that can be analyzed in practice. Finally, the classification tree was easy to interpret, but it had the lowest accuracy scores and only used 2 variables.

The table below summarizes accuracy for each algorithm, in order of best to worst.

Algorithm	Best Model Parameters	Training Accuracy	Test Accuracy
KNN	• K = 3 • All 18 Variables Included, Scaled	86.1%	75.3%
Logistic Regression	• Probability Threshold = 50% • Selected 8 Variable Subset (P-Values <.05)	76.3%	75.1%
SVM	• Cost = 0.1, Gamma = 0.25 • 3,932 Support Vectors • 4 Variable Subset Selected to Expedite Runtime for Tuning	76.0%	74.4%
Classification Tree	• Nodes = 5 • Tree Selected 2 Variables to Use in Model	73.4%	72.8%

6.2 Lessons Learned

This project emphasized the importance of having a large, relatively clean dataset. We saw firsthand how the availability of data is can impact model outputs and limit one's ability to use certain algorithms. We also learned to be careful when comparing test scores across models, as it is important to understand the variables included in each and if there are any differences. Finally, we saw how divergent approaches in cleaning datasets can vastly affect results.

6.3 Potential Next Steps

Our first step will be to merge in as much additional NHANES data as possible from other years the data is available. We will also evaluate different ways to clean the data and look more closely into correlation between input variables. Finally, we will place emphasis on studying input variables describing dietary/nutritional and lifestyle habits (smoking, drug use, exercise).

7 References

[1] Centers for Disease Control and Prevention (CDC)

[2] Kannel WB, (1961) Factors of risk in the development of coronary heart disease six-year follow-up experience. The Framingham Study. Ann Intern Med 55:33–50.

[3] L. N. Borrell and L. Samuel, “Body Mass Index Categories and Mortality Risk in US Adults: The Effect of Overweight and Obesity on Advancing Death,” Am. J. Public Health, vol. 104, no. 3, pp. 512–519, Mar. 2014.

[4] P. Scarborough, P. Bhatnagar, “The economic burden of ill health due to diet, physical inactivity, smoking, alcohol and obesity in the UK: an update to 2006-07 NHS costs.,” J. Public Health (Oxf)., vol. 33, no. 4, pp. 527–35, Dec. 2011.

[5] Inadequate Sleep as a Risk Factor for Obesity: Analyses of the NHANES I

[6] CDC Overweight & Obesity Data & Statistics

[7] US Obesity Rates and Trends

[8] Intro. To Statistical Learning with Applications in R, Gareth James. 4.3, pp. 120-128.

[9] Intro. To Statistical Learning with Applications in R, Gareth James. 8.1.2, pp. 311-314.

[10] Intro. To Statistical Learning with Applications in R, Gareth James. 4.6.5, pp. 163-165.

[11] Intro. To Statistical Learning with Applications in R, Gareth James. 9.3, pp. 349-355.

[12] Saeys, Y., Inza, I, (2007). A review of feature selection techniques in bioinformatics. Bioinformatics, 23(19), 2507–2517. doi:10.1093/bioinformatics/btm344

[13] NHANES dataset

[14] NHANES R Package

[15] Google’s R Style Guide

[16] ggplot2 for visualization

8 Appendices

The appendices contain the R code and any other information supporting our report that

was stated above but not included in the body of this report.

8.1 Appendix 1 – R Code

We used the Google R Style Guide [15] for R programming and code reviews.
ggplot2 [16] for visualizations.

8.1.1 Appendix 1 – R Code – Cleaned NHANES Dataset

# Contains dataset used for models

# Summary of Major steps

```
        # Remove BMI values = "NA"

        # Choose subset composed of num/int values only

        # Create BMIOver25 variable

        # Replace NAs with column medians for num/int values
```

# Cleaned Dataset (nhanesMajor)

```
        # Step 1 - Load NHANES dataset and libraries used

        library(NHANES)

        library(dplyr)

        library(imputeTS)

        nhanesData <- NHANES

        # Step 2 - Remove variables where BMI = "NA"

        nhanesMajor <- nhanesData %>%

          filter(BMI != "NA")
```

```
# Step 3 - Create BMIOver25 variable to use for classification
attach(nhanesMajor)
BMIOver25 = ifelse(BMI>=25,"Yes","No")
nhanesMajor = data.frame(nhanesMajor, BMIOver25)
dim(nhanesMajor)
# Step 4 - Reduce dataset to smaller number of variables
# All are num/int, with the exception of “BMIOver25”
inputVars = c("BMIOver25", "Age", "HHIncomeMid", "Poverty", "HomeRooms",
"Pulse","Testosterone", "DirectChol", "TotChol","DaysPhysHlthBad",
"DaysMentHlthBad", "PhysActiveDays", "SleepHrsNight", "AlcoholDay",
"AlcoholYear", "SexAge", "SexNumPartnLife", "UrineVol1", "UrineFlow1")
nhanesMajor  = nhanesMajor[inputVars]
dim(nhanesMajor) # dimension check- should have 19 columns
names(nhanesMajor) # look at columns
# Step 5 - Set “NA” records = median for each column
install.packages(‘imputeTS’) # Used to replace missing values with mean value
nhanesMajor <- na.mean(nhanesMajor[inputVars], option = "median")
#inputVars excludes BMIOver25
summary(nhanesMajor)
# Check, NAs should now = 0
colSums(is.na(nhanesMajor)|nhanesMajor == '')
```

8.1.2 Appendix 2 – R Code – NA Counts By Variable

The R code for the missing values represented by the symbol **NA** (not available) is:

colSums(is.na(nhanesData)|nhanesData == '')

```
            ID        SurveyYr          Gender             Age       AgeDecade
             0               0               0               0             333
     AgeMonths           Race1           Race3       Education   MaritalStatus
          5038               0            5000            2779            2769
      HHIncome     HHIncomeMid         Poverty       HomeRooms         HomeOwn
           811             811             726              69              63
          Work          Weight          Length        HeadCirc          Height
          2229              78            9457            9912             353
           BMI BMICatUnder20yrs         BMI_WHO           Pulse        BPSysAve
           366            8726             397            1437            1449
      BPDiaAve          BPSys1          BPDia1          BPSys2          BPDia2
          1449            1763            1763            1647            1647
        BPSys3          BPDia3    Testosterone      DirectChol         TotChol
          1635            1635            5874            1526            1526
      UrineVol1      UrineFlow1       UrineVol2      UrineFlow2        Diabetes
           987            1603            8522            8524             142
   DiabetesAge       HealthGen DaysPhysHlthBad DaysMentHlthBad  LittleInterest
          9371            2461            2468            2466            3333
     Depressed    nPregnancies         nBabies      Age1stBaby   SleepHrsNight
          3327            7396            7584            8116            2245
  SleepTrouble     PhysActive PhysActiveDays       TVHrsDay      CompHrsDay
          2228            1674            5337            5141            5137
 TVHrsDayChild CompHrsDayChild Alcohol12PlusYr      AlcoholDay     AlcoholYear
          9347            9347            3420            5086            4078
      SmokeNow        Smoke100       Smoke100n        SmokeAge       Marijuana
          6789            2765            2765            6920            5059
  AgeFirstMarij    RegularMarij     AgeRegMarij       HardDrugs         SexEver
          7109            5059            8634            4235            4233
        SexAge SexNumPartnLife  SexNumPartYear         SameSex  SexOrientation
          4460            4275            5072            4232            5158
   PregnantNow
          8304
```

8.1.3 Appendix 3 – R Code – Models

8.1.3.1 R Code – Model 1 - Logistic Regression

# Logistic Regression Model

# Fit initial model on full dataset

```
glmFits=glm(BMIOver25~.-BMIOver25,data=nhanesMajor, family=binomial)
summary(glmFits)

Call:
glm(formula = BMIOver25 ~ . - BMIOver25, family = binomial, data = nhanesMajor)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8871  -0.8078   0.4201   0.8217   2.4365

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      2.499e-01  3.158e-01   0.791  0.42878
Age              4.629e-02  1.436e-03  32.234  < 2e-16 ***
HHIncomeMid      7.468e-07  1.795e-06   0.416  0.67733
Poverty          9.522e-04  3.480e-02   0.027  0.97817
HomeRooms       -1.656e-02  1.186e-02  -1.397  0.16243
Pulse            3.769e-03  2.212e-03   1.704  0.08845 .
Testosterone    -2.440e-04  1.520e-04  -1.605  0.10850
DirectChol      -2.377e+00  7.874e-02 -30.191  < 2e-16 ***
TotChol          3.756e-01  2.859e-02  13.139  < 2e-16 ***
DaysPhysHlthBad  6.605e-03  4.248e-03   1.555  0.12002
DaysMentHlthBad  1.816e-02  3.789e-03   4.794 1.64e-06 ***
PhysActiveDays  -2.640e-02  1.832e-02  -1.441  0.14956
SleepHrsNight   -4.228e-02  2.094e-02  -2.019  0.04352 *
AlcoholDay       3.886e-02  1.265e-02   3.072  0.00213 **
AlcoholYear      3.783e-04  3.156e-04   1.199  0.23067
SexAge          -2.311e-02  8.812e-03  -2.623  0.00872 **
SexNumPartnLife -8.907e-05  5.647e-04  -0.158  0.87467
UrineVol1        1.620e-03  3.368e-04   4.810 1.51e-06 ***
UrineFlow1      -3.227e-02  3.318e-02  -0.972  0.33083
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 13203  on 9633  degrees of freedom
Residual deviance: 10190  on 9615  degrees of freedom
AIC: 10228

Number of Fisher Scoring iterations: 4
```

# Split data into training and test sets

```
# Split of 70% train / 30% test
set.seed(1)
train=sample(1:nrow(nhanesMajor), 6720)
majorTest=nhanesMajor[-train,2:19]
BMITest=nhanesMajor[-train,"BMIOver25"]

# Look at "Yes" %s in Training Set
```

```
summary(nhanesMajor[train,"BMIOver25"])
No  Yes
2964 3756
> 3756/6720
[1] 0.5589286 # Yes 56% of the time


# Look at "Yes" %s in Test Set
summary(BMITest)
  No  Yes
1248 1666
> 1666/2914
[1] 0.5717227 # Yes 57% of the time (similar)
```

# Calculate accuracy on full dataset

```
# Predict  probabilities
glmProbs=predict(glmFits, type="response")

 # create vector of class predictions for "Yes" or "No"
glmPred=rep("No", 9634)
glmPred[glmProbs>.5]="Yes"

# Use table function to create a confusion matrix to determine correct classifications
table(glmPred,nhanesMajor$BMIOver25)
glm.pred   No  Yes
     No  2974 1073
     Yes 1238 4349
> (2974+4349)/9634
[1] 0.7601204
```

# Accuracy of 76.0%

# Train model on training data then evaluate with test data

```
# Train model
glmFits=glm(BMIOver25~.-BMIOver25,data=nhanesMajor, subset=train, family=binomial)
glmProbs=predict(glmFits, majorTest, type="response")

# Test model
glmPred=rep("No", 2914)
glmPred[glmProbs>.5]="Yes"
table(glmPred, BMITest)
      BMITest
glmPred   No  Yes
    No   886  379
    Yes  362 1287
> (886+1287)/2914
[1] 0.7457104
```

# Accuracy of 74.6%

```
# Model information
summary(glmFits)
Call:
glm(formula = BMIOver25 ~ . - BMIOver25, family = binomial, data = nhanesMajor,
    subset = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9374  -0.7936   0.4110   0.8214   2.4329
```

```
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      1.142e-01  3.756e-01   0.304 0.761082
Age              4.595e-02  1.702e-03  26.994  < 2e-16 ***
HHIncomeMid     -7.956e-07  2.148e-06  -0.370 0.711149
Poverty          4.168e-02  4.173e-02   0.999 0.317968
HomeRooms       -1.708e-02  1.423e-02  -1.200 0.230062
Pulse            3.892e-03  2.646e-03   1.471 0.141309
Testosterone    -2.147e-04  1.810e-04  -1.186 0.235562
DirectChol      -2.378e+00  9.509e-02 -25.004  < 2e-16 ***
TotChol          4.153e-01  3.494e-02  11.886  < 2e-16 ***
DaysPhysHlthBad  6.387e-03  5.030e-03   1.270 0.204149
DaysMentHlthBad  1.912e-02  4.522e-03   4.228 2.35e-05 ***
PhysActiveDays  -9.282e-03  2.177e-02  -0.426 0.669811
SleepHrsNight   -5.484e-02  2.491e-02  -2.202 0.027701 *
AlcoholDay       5.222e-02  1.644e-02   3.175 0.001496 **
AlcoholYear      3.007e-04  3.799e-04   0.791 0.428727
SexAge          -2.811e-02  1.024e-02  -2.745 0.006044 **
SexNumPartnLife  6.278e-05  6.722e-04   0.093 0.925583
UrineVol1        1.429e-03  4.034e-04   3.541 0.000398 ***
UrineFlow1      -2.439e-02  3.913e-02  -0.623 0.533078
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 9222.3  on 6719  degrees of freedom
Residual deviance: 7076.7  on 6701  degrees of freedom
AIC: 7114.7

Number of Fisher Scoring iterations: 4
```

# Subset Selection

```
# Built new model with the 8 variables that had p-Value < .05 in previous model
# Age + DirectChol + TotChol + DaysMentHlthBad + SleepHrsNight + AlcoholDay + SexAge + UrineVol1
glmFits=glm(BMIOver25~ Age + DirectChol + TotChol + DaysMentHlthBad + SleepHrsNight +
AlcoholDay + SexAge + UrineVol1, data=nhanesMajor, subset=train, family=binomial)
glmProbs=predict(glmFits, majorTest, type="response")

# Test model
glmPred=rep("No", 2914)
glmPred[glmProbs>.5]="Yes"
table(glmPred, BMITest)
       BMITest
glmPred   No  Yes
    No   890  369
    Yes  358 1297
> (890+1297)/2914
[1] 0.7505148
```

# Test accuracy of 75.1% with less than half the variables in the prior model
# BEST MODEL

```
# Model information
summary(glmFits)
Call:
```

```
glm(formula = BMIOver25 ~ Age + DirectChol + TotChol + DaysMentHlthBad +
    SleepHrsNight + AlcoholDay + SexAge + UrineVol1, family = binomial,
    data = nhanesMajor, subset = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9151  -0.7960   0.4137   0.8227   2.4368

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)      0.2727874  0.2965338   0.920 0.357614
Age              0.0461866  0.0015448  29.897  < 2e-16 ***
DirectChol      -2.3616900  0.0922705 -25.595  < 2e-16 ***
TotChol          0.4213295  0.0346654  12.154  < 2e-16 ***
DaysMentHlthBad  0.0210524  0.0043989   4.786  1.7e-06 ***
SleepHrsNight   -0.0530973  0.0246823  -2.151 0.031458 *
AlcoholDay       0.0508076  0.0158672   3.202 0.001364 **
SexAge          -0.0285362  0.0100702  -2.834 0.004601 **
UrineVol1        0.0012641  0.0003336   3.789 0.000151 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 9222.3  on 6719  degrees of freedom
Residual deviance: 7085.9  on 6711  degrees of freedom
AIC: 7103.9

Number of Fisher Scoring iterations: 4
# AIC score slightly lower than prior model

# Try model with different probability thresholds
# 40%
glmPred=rep("No", 2914)
glmPred[glmProbs>.4]="Yes"
table(glmPred, BMITest)
       BMITest
glmPred   No  Yes
    No   758  250
    Yes  490 1416

> (758+1416)/2914
[1] 0.7460535 # 74.6% accurate, slightly lower than 50% threshold model accuracy

# 60%
glmPred=rep("No", 2914)
glmPred[glmProbs>.6]="Yes"
table(glmPred, BMITest)
       BMITest
glmPred   No  Yes
    No   973  520
    Yes  275 1146
> (973+1146)/2914
[1] 0.7271791 # 72.8% accurate, lower than 50% threshold model accuracy
```

# Best logistic regression model has 8 variables and set to 50% threshold, test accuracy of 75.1%

8.1.3.2 R Code – Model 2 - Classification Tree

# Classification Tree Model

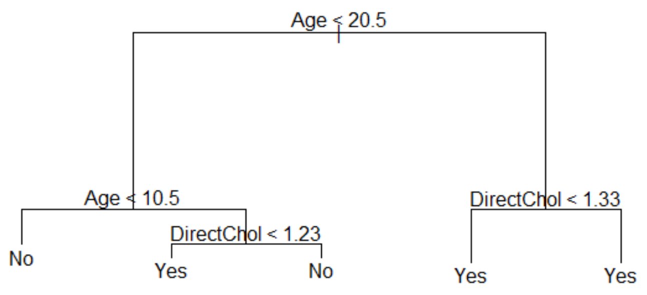
# Tree on Full Dataset

# Fit initial regression tree using selected variables

```
library(tree)
treeBMI=tree(BMIOver25~., data=nhanesMajor)
summary(treeBMI)
Classification tree:
tree(formula = BMIOver25 ~ . - BMIOver25, data = nhanesMajor)
Variables actually used in tree construction:
[1] "Age"       "DirectChol"
Number of terminal nodes:  5
Residual mean deviance:  1.037 = 9990 / 9629
Misclassification error rate: 0.2694 = 2595 / 9634
```

# Plot tree

```
plot(treeBMI)
text(treeBMI, pretty=0)
```



# Look at cross-validation stats

```
set.seed(1)
cvBMI=cv.tree(treeBMI, FUN=prune.tree)
names(cvBMI)
cvBMI
```

# best CV result was at size of 5

# Split data

```
# 70% train / 30% test
set.seed(1)
train=sample(1:nrow(nhanesMajor), 6720)
majorTest=nhanesMajor[-train,2:19]
BMITest=nhanesMajor[-train,"BMIOver25"]
```

# Unpruned Tree

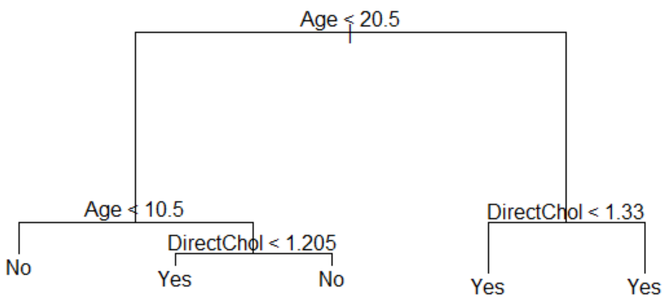
# Fit model to training set

```
treeBMI=tree(BMIOver25~., data=nhanesMajor,subset=train)
Classification tree:
tree(formula = BMIOver25 ~ ., data = nhanesMajor, subset = train)
Variables actually used in tree construction:
[1] "Age"       "DirectChol"
Number of terminal nodes:  5
Residual mean deviance:  1.035 = 6951 / 6715
Misclassification error rate: 0.2659 = 1787 / 6720
> 1-.2659
[1] 0.7341
```

# Training set accuracy of 73.4%

# Unpruned Tree

```
        # Test accuracy before pruning
        treePred=predict(treeBMI, majorTest, type="class")
        table(treePred, BMITest)
         BMITest
        treePred   No  Yes
            No   536   82
            Yes  712 1584
        > (536+1584)/2914
        [1] 0.7275223
        # Test set accuracy of 72.8%
```



# Pruned Tree

```
    # Pruned Tree Test Performance with Best = 3 for Comparison
    pruneBMI=prune.tree(treeBMI, best=3)
    prunePred=predict(pruneBMI, majorTest, type="class")
    table(prunePred,BMITest)
           BMITest
    prunePred   No  Yes
         No   614  181
         Yes  634 1485
    > (614+1485)/2914
    [1] 0.720315
    # Accuracy on test set 72.0% is slightly worse than Best = 5, as CV predicted
```

8.1.3.3 R Code – Model 3 - KNN

# KNN is from class library

```
library(class)
```

# Split data into training and test sets, scale X inputs for KNN

```
    # Split of 70% train / 30% test (assumes nhanesMajor is already defined)
    set.seed(1)
    train=sample(1:nrow(nhanesMajor), 6720)
    majorTest=scale(nhanesMajor[-train,2:19]) # SCALED for KNN
    BMITest=nhanesMajor[-train,"BMIOver25"]
    majorTrain=scale(nhanesMajor[train,2:19]) # SCALED for KNN
    BMITrain=nhanesMajor[train,"BMIOver25"]
```

# Build KNN Model, Test Performance, Experiment with K Values

```
    # 4 inputs required for KNN
        # Input 1- matrix of training data Xs, Input 2- matrix of test data Xs
        # Input 3- vector of training data Ys (class labels), Input 4- value for K (# of nearest neighbors)
```

```
# Predict BMIOver25, k=3
set.seed(1)
knnPred=knn(majorTrain, majorTest, BMITrain, k=3)
table(knnPred, BMITest)
  BMITest
knnPred   No  Yes
    No   911  382
    Yes  337 1284
>
> (911+1284)/2914
[1] 0.7532601 # 75.3% test accuracy, best KNN model
```

```
# Training Error
        set.seed(1)
        knnPred=knn(majorTrain, majorTrain, BMITrain, k=3)
        table(knnPred, BMITrain)
        BMITrain
        knnPred   No  Yes
            No  2521  491
            Yes  443 3265
        > (2521+3265)/6720
        [1] 0.8610119

        # 86.1% training accuracy
```

```
# Predict BMIOver25, k=5
set.seed(1)
knnPred=knn(majorTrain, majorTest, BMITrain, k=5)
table(knnPred, BMITest)
     BMITest
knnPred  No  Yes
   No  900  380
   Yes  348 1286

> (900+1286)/2914
[1] 0.7501716 # 75.0% test accuracy
```

```
# Predict BMIOver25, k=7
set.seed(1)
knnPred=knn(majorTrain, majorTest, BMITrain, k=7)
table(knnPred, BMITest)

BMITest
knnPred   No  Yes
    No   870  351
    Yes  378 1315
>
> (870+1315)/2914
[1] 0.7498284  # 74.9% test accuracy
```

8.1.3.4 R Code – Model 4 – SVM

```
# SVN Classification Model

# SVM uses e1071 library
library(e1071)

# Fit SVM model to full nhanesMajor dataset
        svmFit=svm(BMIOver25~.-BMIOver25, data=nhanesMajor, kernel="linear", cost=10, scale=TRUE)
        summary(svmFit)
```

```
Call:
svm(formula = BMIOver25 ~ . - BMIOver25, data = nhanesMajor, kernel = "linear", cost = 10,
    scale = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.05555556

Number of Support Vectors:  5602

 ( 2803 2799 )


Number of Classes:  2

Levels:
 No Yes
```

# Select fewer variables to create nhanesSub dataset for runtime purposes

```
# Used the 4 variables with lowest P-Values from logistic regression model
# Age + DirectChol + TotChol + DaysMenHlthBad
# Also need BMIOver25
inputSubVars =c("BMIOver25","Age","DirectChol","TotChol","DaysMentHlthBad")
nhanesSub  = nhanesMajor[inputSubVars]

# Checks- should have 5 columns and 0 NAs
dim(nhanesSub)
colSums(is.na(nhanesSub)|nhanesSub == '')
```

# Fit SVM model to nhanesSub dataset

```
svmFit=svm(BMIOver25~.-BMIOver25, data=nhanesSub, kernel="linear", cost=10, scale=TRUE)
summary(svmFit)
            Call:
svm(formula = BMIOver25 ~ . - BMIOver25, data = nhanesSub, kernel = "linear",
    cost = 10, scale = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.25

Number of Support Vectors:  5662

 ( 2825 2837 )


Number of Classes:  2

Levels:
 No Yes
```

# Perform cross-validation on cost setting using tune

```
set.seed(1)
tuneOut=tune(svm,BMIOver25~.-BMIOver25, data=nhanesSub, kernel="linear", scale=TRUE, ranges=
list(cost=c(0.1,10,100)))
summary(tuneOut)
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.2434065

- Detailed performance results:
   cost     error dispersion
1   0.1 0.2434065 0.01833796
2  10.0 0.2436141 0.01827217
3 100.0 0.2434066 0.01850804
```

```
# Cost should be set at 0.1
```

# Look at optimal model for nhanesSub dataset

```
bestMod=tuneOut$best.model
summary(bestMod)
Call:
best.tune(method = svm, train.x = BMIOver25 ~ . - BMIOver25, data = nhanesSub,
    ranges = list(cost = c(0.1, 10, 100)), kernel = "linear", scale = TRUE)

Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1
      gamma:  0.25

Number of Support Vectors:  5666

 ( 2832 2834 )


Number of Classes:  2

Levels:
 No Yes
```

# Split data into training and test sets

# Split of 70% train / 30% test

```
set.seed(1)
train=sample(1:nrow(nhanesMajor), 6720)
```

# Train and test model with cost = 0.1

```
svmFit=svm(BMIOver25~.-BMIOver25, data=nhanesSub[train,], kernel="linear", cost=0.1, scale=TRUE)
```

# predict class label on test data set

```
yPred=predict(svmFit, nhanesSub[-train,])
```

table(predict=yPred, truth= nhanesSub[-train,"BMIOver25"])

```
  truth
predict   No  Yes
    No   876  374
    Yes  372 1292
>
> (876+1292)/2914
[1] 0.7439945
```
# Test accuracy is 74.4%, best model for SVM

# Training Accuracy

svmFit=svm(BMIOver25~.-BMIOver25, data=nhanesSub[train,], kernel="linear", cost=0.1, scale=TRUE)

yPred=predict(svmFit, nhanesSub[train,])
table(predict=yPred, truth= nhanesSub[train,"BMIOver25"])

```
      truth
predict   No  Yes
    No  2098  747
    Yes  866 3009
>
> (2098+3009)/6720
[1] 0.7599702
```
# Test accuracy is 76.0%