Data 2010 Project

Lucas Berzuk, Ethan Robson, Hugo Ng, Rodell Salonga

April 5, 2024

Introduction

Abstract

Obesity is a chronic disease that is defined by having excessive amount of body fat. It increases the risk of other diseases and health problems including heart disease, high blood pressure, diabetes and certain cancers. First world countries are experiencing an epidemic of obesity in recent years. This includes Canada, which reported a 30% obesity rate out of all Canadian adults, an increase from the 21% reported in 2003 [2]. There must be a change and this includes diagnosing obesity early on for people and providing the preventative therapies to decrease body fat percentage. This involves measuring levels of obesity in people.

There are many different tools for measuring obesity levels, these include the body mass index, waist-to-hip ratio, skin fold thickness, and other more costly and intense body fat measuring procedures. The most common measure is the body mass index, BMI for short, which is calculated by using an individuals weight and height, with the following formula: $BMI = \frac{weight}{height^2}$. However in many cases the height and weight of individuals are not available in the diagnosing of obesity level.

In this paper we investigate different models that can be to predict obesity levels based on other variables other than height and weight. The models will be based on the following variables.

We build various regression models with the target variable being BMI and using the rest of the variables in the dataset to see if we are able to predict obesity with the lifestyle of a person. We end up using Linear, Lasso, and Ridge regression to build our models and calculate the root mean squared error (RMSE).

//**TODO** We then build classification models including...

Data Analysis

This data comes from a study performed in Mexico, Peru and Colombia. It has 17 attributes and 2111 data points. The 17 attributes comprise 4 different variable types: 6 continuous, 2 ordinal, 5 categorical, and 4 binary. This data consists of an individual's behavioural patterns such as eating habits and physical condition, as well as their level of obesity. It is important to note that up to 77% of this data has been synthetically generated because of a greatly unbalanced number of samples between the different obesity categories in the sample data. The balancing was performed using the SMOTE filter with the Weka tool. The purpose was to create data points within the obesity categories with lower counts. This lead to an overall data set with uniform counts for the different obesity levels. The data was collected through a voluntary online survey that had 16 questions with a variety of responses. It was accessible for 30 days for users to complete. The researchers of this study calculated Body Mass Index using the equation $BMI = \frac{weight}{height^2}$ to determine the BMI of every respondent. The BMI values were then placed into different obesity level categories based on the World Health Organization and Mexican Normativity. The categories can be observed in Table 1.

Table 1: Categorizing Obesity Levels

Obesity Level	Insufficient Weight	Normal Weight	Overweight Level I	Overweight Level II	Obesity Type I	Obesity Type II	Obesity Type III
BMI	<18.5	18.5-22.9	23.0-25.9	26.0-29.9	30.0-34.9	35.0-39.9	>40

//TODO - Talk about our preprocessing stuff (do we need this????)

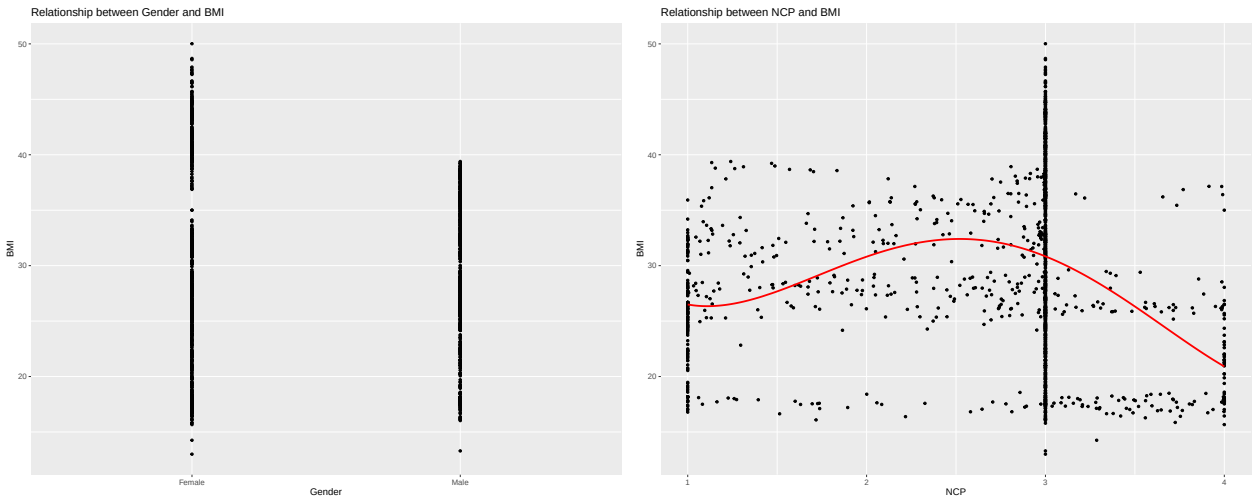
Methods

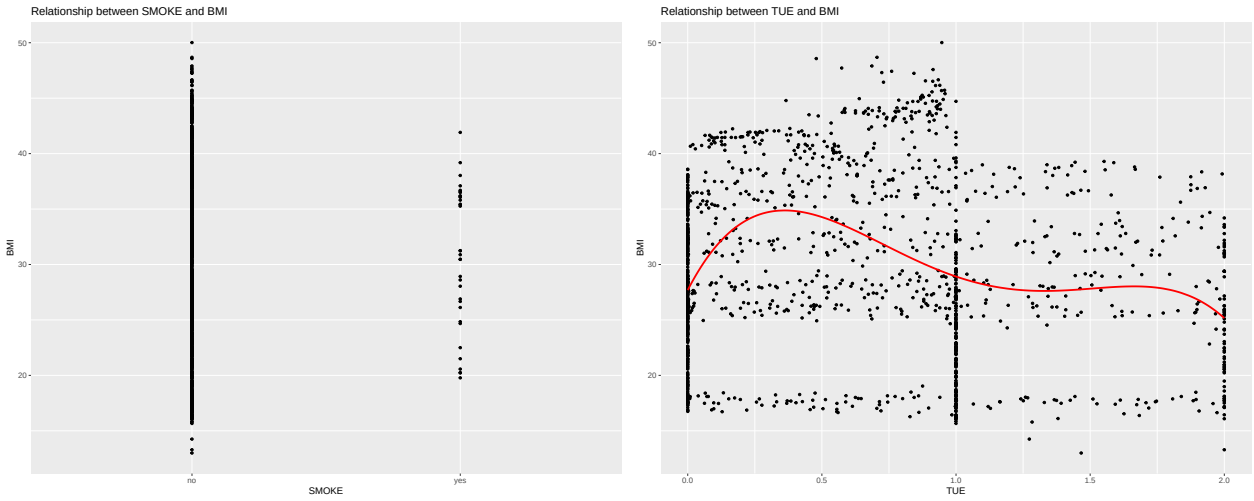
Regression

We decide to model many forms of regression on our data to calculate RMSE and see how far off predicted values would be from observed values for this obesity dataset. We first started off by performing standard Linear regression with BMI where $BMI = \frac{weight}{height^2}$, on every variable that was in the dataset (aside from height and weight because that is what is used in BMI). For the categorical variables we had to factor them before performing regression because otherwise we would not be able to test them. These are the RMSE values that we found for every variable in the dataset.

```
## Gender = 7.939414
## Age = 8.178174
## History of overweight = 9.08678
## Freq consumption of high caloric food = 8.134233
## Freq consumption of vegetables = 8.167971
## Number of main meals = 7.932018
## Consumption of food between meals = 8.807099
## Smoke = 7.929959
## Consumption of water = 8.065398
## Calorie consumption monitoring = 8.106003
## Freq of physical activity = 8.109581
## Time using electronics = 7.973657
## Consumption of alcohol = 8.130043
## Transportation used = 8.07459
```

Noticing that *Gender*, *Number of main meals*, *Smoke* and *Time using electronics* were the variables that lead to the lowest RMSE values (under 8), we decided to graph these variables with for visual input. Here are those graphs.





After some consideration, we decided to try spline regression on the 2 numerical variables out of these 4 to see if that would improve the RMSE values any further. These were the results:

```
## Number of main meals = 8.540398
## Time using electronics = 8.488051
```

It turns out that spline regression did not improve the error in our case so we had to try something else.

We moved to attempt Lasso regression to yet again try and improve our values. When performing Lasso regression on the 4 original variables that all had RMSE under 8, this is what we got with a lambda of 1.2.

```
## RMSE = 7.968656
```

This is a good RMSE value but it is not lower than any 1 of the 4 original RMSE values gained from using Linear regression. Because we still can not seem to get it lower that the original 4, we decided to do Lasso regression with all of the variables and this was the result with a lambda of 4:

```
## RMSE = 7.968656
```

This turned out to give us the exact same RMSE. Maybe we could try and use less variables? This is with just the Gender and SMOKE variable at the lambda of 0.27.

```
## RMSE = 7.96817
```

The result is just slightly lower than the previous 2 Lasso regression attempts so it seems like we may be at a dead end here. But we tried one more just to see what happened. Ridge regression gave us similar RMSE values to Lasso but just slightly higher no matter what variables we used in the model.

The range of BMIs that we acquired from the dataset are from:

```
## Minimum = 12.99868
```

```
## Maximum = 50.01402
```

So the RMSE floating around 8 is not the end of the world but we definitely wish that we could improve it more.

Next we wanted to try to see if there was any evidence of multicollinearity. So we created a correlation matrix and scanned the values to see if there was anything close enough to 1 or -1 to be considered evidence of multicollinearity.

```
##                Age     Height     Weight       FCVC        NCP        CH2O
## Age    1.00000000 0.02080862 0.21793935 0.02225751 0.04393122 0.034840229
## Height 0.02080862 1.00000000 0.47117782 0.04608912 0.25248988 0.204148356
## Weight 0.21793935 0.47117782 1.00000000 0.19534515 0.11347087 0.190483184
## FCVC   0.02225751 0.04608912 0.19534515 1.00000000 0.05008597 0.052498525
## NCP    0.04393122 0.25248988 0.11347087 0.05008597 1.00000000 0.042227619
## CH2O   0.03484023 0.20414836 0.19048318 0.05249853 0.04222762 1.000000000
## FAF    0.16415271 0.30640582 0.05650405 0.01100181 0.11328858 0.161417486
## TUE    0.30123390 0.05600413 0.06577166 0.10525079 0.04894118 0.009913777
##                FAF         TUE
## Age    0.16415271 0.301233897
## Height 0.30640582 0.056004127
## Weight 0.05650405 0.065771664
## FCVC   0.01100181 0.105250787
## NCP    0.11328858 0.048941181
## CH2O   0.16141749 0.009913777
## FAF    1.00000000 0.068454890
## TUE    0.06845489 1.000000000
```

As we can see, there is nothing close enough to prove this so we state that there is no multicollinearity in this dataset.

Results

// **TODO** - Restate results from the methods section, talk about what they mean and why they happened

After performing all of our regression testing, we saw that the lowest RMSE values we were able to obtain through various methods of regression was always around 8. There was no method that we could find that would allow us to narrow the model even further in order to get a more accurate prediction of BMI.

Linear regression put our RMSE values in the range of *7.929959* to *9.08678*. Spline regression was worse and it left us with two values, those being: *8.540398* and *8.488051*. Lasso regression seemed to be the most consistent in getting us RMSE values under 8, no matter what variables were used to build the model. We resulted with RMSE of *7.968656* and *7.96817* accross 3 Lasso regression tests. And finally with the ridge regression (whose results are not included in this paper for brevity) left us with RMSE values of *8.034446* and *8.793966*.

This could be due to various reasons, one that came to mind is because of the synthetically generated data in this dataset. That generated data could have lead to inaccurate inputs which would hold us back from being able to accurately predict BMI of a person. Although the data generated was useful for filling in noticeable gaps, it has most likely lead to some sort of skew and/or bias in the dataset which prevents us from being able to get a more accurate prediction.

Classification

In addition to predicting BMI levels, determining whether or not an individual is obese would be a powerful tool in health care to diagnose indiviuals early and prevent further disease. We now focus on developing

classification models to determine whether or not an individual is obese based on various variables of our dataset.

Pre-processing

The BMI values have been calculated for each individual and we will use those values to determine the true class if an indivual is obese. Based on the CDC's suggestions, an individual with BMI level greater than 30 is considered obese. We require these true class values on our dataset for the developing and testing of our models. Therefore we add the following variable to the dataset:

$$Y = \begin{cases} 0, & BMI \leq 30 \\ 1, & BMI > 30 \end{cases}$$

We then remove the following variables: BMI, Height, Weight, and NObeyesdad from our training data. This essential to the purpose behind building our models our values of the target variable Y is directly calculated from BMI, which is a result of Height and Weight and gives the resulting NObeyesdad categories. Our objective to to discover relationships between obesity and the other variables present in the dataset so that an obesity prediction can be provided when height and weight are not available.

Y	proportion
0	0.5457007
1	0.4542993

When developing a model it is important to initially create the model assuming that the training data comprises our population, therefore, we analyze only the training data in our feature selection. For our feature selection in the classification models we will perform filtering methods. Filtering methods involve looking at the each feature of the dataset individually without considering its relationship with other covariates. Although this may overlook relationships between covariates such as multicollinearity, it is an effective appraoch. For each feature we perform the appropriate statistical test to determine its relationship with the target variable Y.

For the categorical variables, we will use a **Pearson's Chi-Squared Test of Independence**. The Chi-Squared test is designed to analyze the relationship between categorical variables. It is called a "goodness of fit" test because it measures how likely the observed distribution of values fits the distribution under the null hypothesis, which assumes the two variables are independent.

For the remaining variables, which are numerical, we will use a **Two-sided T-test**. The Two-sided T-test is a statistic test used for evaluating if the means of two populations differ or not. Here the two populations are $Y = 0$, Non-obese individuals, and $Y = 1$, Obese individuals. For each feature we will test whether their mean values within the two groups are significantly different.

Table 3: P-values for Chi-Squared Test

Variable	Gender	fhwo	FAVC	CAEC	SMOKE	SCC	CALC	MTRANS
p-value	0.875	4.03e-56	6.13e-27	2.1e-45	0.215	5.35e-12	2.14e-09	1.56e-06

Table 4: P-values for T-test

Variable	Age	FCVC	NCP	CH2O	FAF	TUE
p-value	6.65e-20	8.33e-11	0.0249	0.000231	6.54e-10	0.0011

From the tables we see that two categorical variables have a resulting p-value less than 0.05 from the Chi-Squared test. Therefore, at a 95% significance level we conclude that Gender and also

Now that we have our selected features to build our model, we will use them to predict our target class, the binary variable Y indicating if an individual is obese. There are many different classification models, in this report, we focus on logisitic regression, nearest neighbours, decision trees and random forests.

Logistic Regression

Logistic regression is a regression model that can be turned into a classification model. This is done through the logit function which takes probabilities between 0 and 1 and transforms them to any real number. The result of this is a model that when given values of covariates it produces a probability that can be used as a threshold to determine what binary class that observation is to be labelled. This is particularly useful in our desired model as our target variable Y is binary. Therefore, a logistic model could be used to predict the value of Y for an individual given value sof the covariates, that is, predict whether or not that individual is obese. However, Logistic regression uses the regression equation and therefore requires assumptions about the data to be satisfied. These assumptions include : Linearity of the logit, Independence of errors, Absence of multicollinearity, Lack of overdispersion, Adequacy of the model. For simplicity we will assume these properties are satisfied for our dataset.

perform Lasso? regularization regression is not likely to to improve our logistic model much since the number of features is not close to the 2111 observations and we do not see signs of multicollinearity.

```
##                        (Intercept)                                Age
##                      -19.14673615                         0.07686856
## family_history_with_overweightyes                             FAVCyes
##                        3.33607016                         2.17814917
##                               FCVC                    CAECFrequently
##                        0.82771148                        -2.74728184
##                            CAECno                     CAECSometimes
##                       -1.29436040                         1.13438656
##                               CH2O                           SCCyes
##                        0.16849378                        -1.90843201
##                               FAF                               TUE
##                       -0.28076508                        -0.22501541
##                     CALCFrequently                            CALCno
##                        7.87713736                         8.25902544
##                      CALCSometimes                       MTRANSBike
##                        8.51740486                       -10.59411529
##                    MTRANSMotorbike       MTRANSPublic_Transportation
##                        2.28097387                         1.19615925
##                     MTRANSWalking
##                       -1.06212338
```

Nearest Neighbors

While some classification models require assumptions, other models such as Nearest Neighbors do not make any assumptions about the data. Nearest Neighbors is a simple model that involves determining the class of

an observation based on the class of the other observations closest to it. Typically, the Euclidean distance is used to calculate the distance from the given observation to the observations of the given data base on values of the features. Since distance is calculated between the covariate values, it becomes clear that the covariates must be numeric. In our training data we have 8 categorical variables that we will exclude from this model. Therefore we will considering the 5 numerical features that were selected: . However, there are techniques to convert categorical variables to numeric while keeping there relationship consistent and this should be explored in future analysis if wanting to improve our models. The class of a given observation is then determined by considering what the majority class of its nearest neighbors are. K is a hyper-parameter of the model that determines how many nearest neighbours to consider for any given observation.

Determining the value of K is specific to the data at hand and there is not a value of K that will work all of the time. Hence, we will calculate multiple models and evaluate them to determine the best value of K. We proceded by created models with the following values of K, $K = \{1, 3, ..., 101\}$. It is important to note that we are only considering odd values for K. This is becasue we have a a binary classification, if K were allowed ot be even then the majority class between its K nearest neighbors could be a tie and we would be unable to directly choose a class for that observation. In comparing the models numerically for each of the different K values we evaluate each one by calculating the models accuracy. Accuracy is defined as ratio of correct predictions over total predictions. Since our training data is approximately balanced accuracy is a valid measure of the model. We also consider Area Under the Curve (AUC)

Decision Tree

Nearest Neighbors is flexible as there are not assumptions but it is not the most interpreditable as it is difficult to tell which features are the most important in determining the class. Like Nearest Neighbors, Descision trees are another classification model that are flexible and do not make assumptions about the data. The main advantage of decesion trees is its easy interpretation. Decesion trees are known as universal approximators as it is easy to use them to create a model with perfect classification. However, it is important to evaluate the bias-variance tradeoff when consider the size and depth of the tree to avoid perfect classification model that results in overfitting.

We use all of the selected categorical and numerical variables to in the development of the decision tree. From this tree see that the root node consists of the variable family_history_with_overweight. We will assume that the `ctree()` function from the 'party' package selects the predicate at each node that maximizes the drop in entropy. This drop in entropy cant be quantified by the *Information Gain* of a predicate. This tells use that the predicate for if an individual has family history of overweightness maximizes the information gain our of all possible predicates. Since this variable is binary (yes/no), it tells use that this feature does the best job at dividing the individuals into the the correct class of obese and non-obese.

Large trees can lead to overfitting and small trees can lead to underfitting so it is sometimes difficult to manually determine the best depth of the tree to minimize these risk while balancing the bias and variance. One strategy that addresses this issue is bagging. Bagging on decision trees involves creating bootstrap samples, that is sampling from the training data with replacement, and then building trees for each bootstrap sample. *Random Forests* are a classification model that utilizes bagging combined with random subspaces to improve the accuracy of decision trees. For each iteration a random selection of the features is performed so that each tree is not highly correlated.

We performed Random Forests on our dataset in attempt to improve our original decision tree performance. Below are the resulting confusion matrices when we applied these two models to our training data

```
## [1] 0.8328076
```

```
## [1] 0.8817035
```

	0	1
0	273	58
1	48	255

	0	1
0	295	36
1	39	264

Results

There are various numerical measures of classification models including accurracy, precision, and recall. We deciding on the mosrt effective measure to use it is important to consider if our dataset is balanced or unblanced. Since the classes of obese and non-obese people is close to half of the total number of observations, we can say that the binary classification is balanced.

There are also other visual measures to evaluate the performance of classification models such as the Receiver Operating Characteristic (ROC) curve and the Precision-Recall curve. Since our data is balanced we will plot the ROC curve for our various models to compare them. The ROC curve plots the true positive rate against the false positive rate for values of t between 0 and 1, where t is the probability threshold that determines the outcome class for that individual. When looking at he resulting line for a model, the closer the line goes towards the top left corner of the unit square, the better the model. As a result of this we can quantify the models performance by the Area Under the Curve (AUC) which directly calculates the area under the line within the unit square. An AUC value of 1 would be the result of a perfect classification model, while values from 0.8-0.9 represent an excellent model. We want to see the AUC value of a model greater than 0.5, otherwise that suggests that we could get a better performance by swapping the binary classes out of the model. The following is the ROC curve of the logistic model, the nearest neighbors model with the optimal k value, and the decision tree improved by the random forest method.

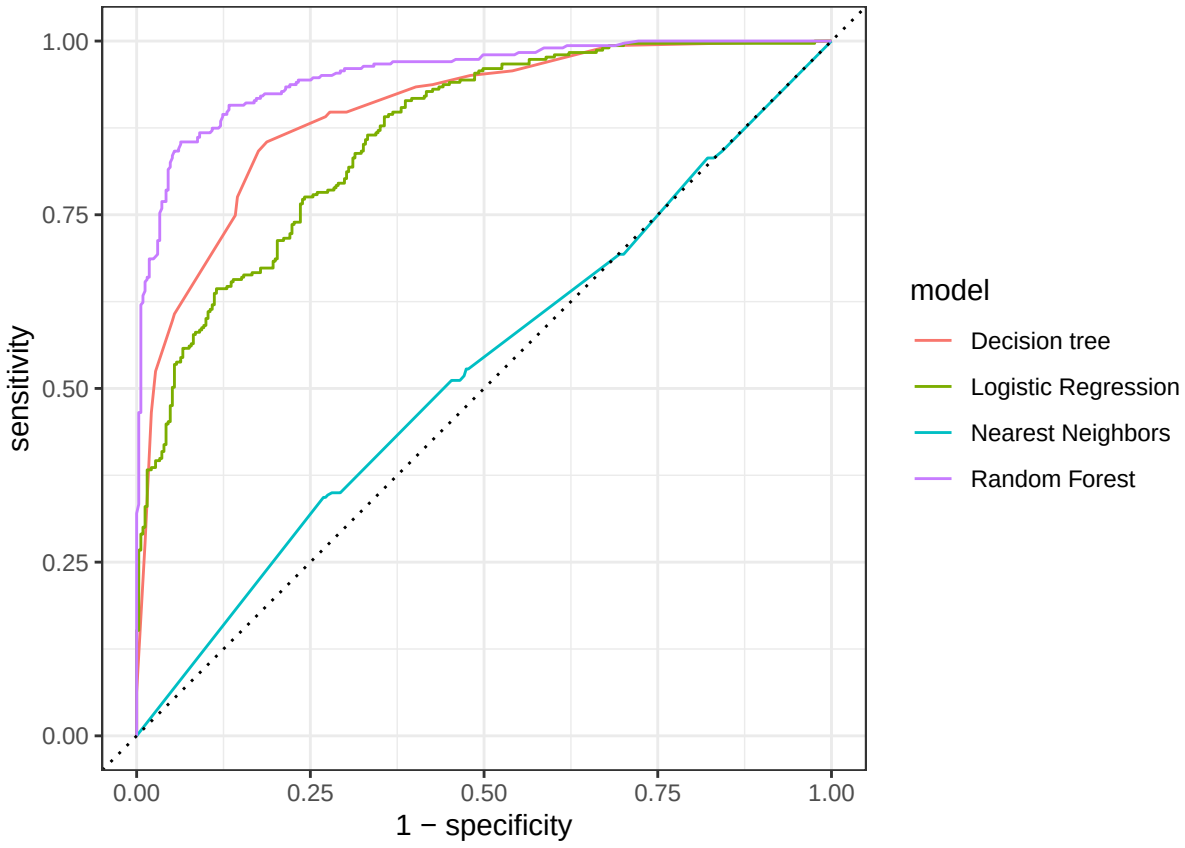


Table 5: Area Under the Curve for ROC

Model	Decision tree	Logistic Regression	Nearest Neighbors	Random Forest
AUC	0.8986918	0.8630213	0.5274197	0.952549

From the plot, we see that the nearest neighbor model had the worst performance with an AUC of *0.5*. This is insightful as the nearest neighbor classification was trained only using the numerical features, suggesting that the categorical variables play a strong role in classifying an individual as obese. The logistic regression model involving all features has an AUC of *0.8*, performing much better than nearest neighbors but not nearly as strong as the decision tree model and the random forest model which had AUC value of *0.8* and *0.9*, respectively. The weaker performance of the logistic model may be a result of certain assumptions that were not satisfied. For example, the relationship between the features and obesity may not result in linearity fo the logit.

Conclusions

//*__TODO__ - Essay type write up for the conclusion

To conclude, blah blah blah i dont think i can conclude yet because were not done.

References

1. Estimation of Obesity Levels Based On Eating Habits and Physical Condition
2. An overview of weight and height measurements on World Obesity Day

Appendix

```
knitr::opts_chunk$set(echo = FALSE, warning = FALSE, message = FALSE, tidy.opts = list(width.cutoff = 9
    tidy = TRUE)
library(kableExtra)
# df = data.frame('Obesity Level' = c('Insufficient Weight', 'Normal Weight', 'Overweight
# Level I', 'Overweight Level II', 'Obesity Type I', 'Obesity Type II', 'Obesity Type
# III'), 'BMI' = c('<18.5', '18.5-22.9', '23.0-25.9', '26.0-29.9', '30.0-34.9',
# '35.0-39.9', '>40')) kable(df, caption = '\\label{tab1}Categorizing Obesity Levels')
# %>% column_spec(2:2,border_left = T)

mat = matrix(c("BMI", "<18.5", "18.5-22.9", "23.0-25.9", "26.0-29.9", "30.0-34.9", "35.0-39.9",
    ">40"), nrow = 1, ncol = 8, byrow = TRUE)

colnames(mat) = c("Obesity Level", "Insufficient Weight", "Normal Weight", "Overweight Level I",
    "Overweight Level II", "Obesity Type I", "Obesity Type II", "Obesity Type III")
kable(mat, align = "c", caption = "\\label{tab1}Categorizing Obesity Levels")

# Setup Code Chunk Add any libraries etc used here.
library(ggplot2)
library(tidyverse)
library(splines)
library(glmnet)
```

```r
# Data importing
dataset = read.csv("ObesityDataSet_raw_and_data_sinthetic.csv")

# factor all categorical variables
dataset[sapply(dataset, is.character)] = lapply(dataset[sapply(dataset, is.character)], as.factor)

set.seed(1)
row.number = sample(1:nrow(dataset), 0.7 * nrow(dataset))
trainData = dataset[row.number, ]
testData = dataset[-row.number, ]

# BMI to use
BMI = trainData$Weight/(trainData$Height^2)
# Performing regression on every variable with BMI

# Gender
m = lm(BMI ~ Gender, data = trainData)
p = predict(m, newdata = testData)
rmse1 = sqrt(mean((BMI - p)^2))

# Age
m = lm(BMI ~ Age, data = trainData)
p = predict(m, newdata = testData)
rmse2 = sqrt(mean((BMI - p)^2))

# family_history_with_overweight
m = lm(BMI ~ family_history_with_overweight, data = trainData)
p = predict(m, newdata = testData)
rmse3 = sqrt(mean((BMI - p)^2))

# FAVC - Frequent consumption of high caloric food
m = lm(BMI ~ FAVC, data = trainData)
p = predict(m, newdata = testData)
rmse4 = sqrt(mean((BMI - p)^2))

# FCVC - Frequent consumption of vegetables
m = lm(BMI ~ FCVC, data = trainData)
p = predict(m, newdata = testData)
rmse5 = sqrt(mean((BMI - p)^2))

# NCP - Number of main meals
m = lm(BMI ~ NCP, data = trainData)
p = predict(m, newdata = testData)
rmse6 = sqrt(mean((BMI - p)^2))

# CAEC - Consumption of food between meals
m = lm(BMI ~ CAEC, data = trainData)
p = predict(m, newdata = testData)
rmse7 = sqrt(mean((BMI - p)^2))

# SMOKE
m = lm(BMI ~ SMOKE, data = trainData)
p = predict(m, newdata = testData)
```

```r
rmse8 = sqrt(mean((BMI - p)^2))

# CH2O - Consumption of water
m = lm(BMI ~ CH2O, data = trainData)
p = predict(m, newdata = testData)
rmse9 = sqrt(mean((BMI - p)^2))

# SCC - Calories consumption monitoring
m = lm(BMI ~ SCC, data = trainData)
p = predict(m, newdata = testData)
rmse10 = sqrt(mean((BMI - p)^2))

# FAF - Frequency of physical activity
m = lm(BMI ~ FAF, data = trainData)
p = predict(m, newdata = testData)
rmse11 = sqrt(mean((BMI - p)^2))

# TUE - Time using electronics
m = lm(BMI ~ TUE, data = trainData)
p = predict(m, newdata = testData)
rmse12 = sqrt(mean((BMI - p)^2))

# CALC - Consumption of alcohol
m = lm(BMI ~ CALC, data = trainData)
p = predict(m, newdata = testData)
rmse13 = sqrt(mean((BMI - p)^2))

# MTRANS - Transportation used
m = lm(BMI ~ MTRANS, data = trainData)
p = predict(m, newdata = testData)
rmse14 = sqrt(mean((BMI - p)^2))

cat("Gender =", rmse1, "\nAge =", rmse2, "\nHistory of overweight =", rmse3, "\nFreq consumption of high
    rmse4, "\nFreq consumption of vegetables =", rmse5, "\nNumber of main meals =", rmse6,
    "\nConsumption of food between meals =", rmse7, "\nSmoke =", rmse8, "\nConsumption of water =",
    rmse9, "\nCalorie consumption monitoring =", rmse10, "\nFreq of physical activity =", rmse11,
    "\nTime using electronics =", rmse12, "\nConsumption of alcohol =", rmse13, "\nTransportation used =
    rmse14)
# Lets graph the regression models

# Gender
m = lm(BMI ~ Gender, data = trainData)
trainData |>
    mutate(fitted = fitted(m)) |>
    ggplot(aes(x = Gender)) + geom_point(aes(y = BMI), size = 1) + ggtitle("Relationship between Gender

# NCP
m = lm(BMI ~ poly(NCP, 4), data = trainData)
trainData |>
    mutate(fitted = fitted(m)) |>
    ggplot(aes(x = NCP)) + geom_point(aes(y = BMI), size = 1) + geom_line(aes(y = fitted),
    colour = "red", linewidth = 1) + ggtitle("Relationship between NCP and BMI")
```

```r
# SMOKE
m = lm(BMI ~ SMOKE, data = trainData)
trainData |>
    mutate(fitted = fitted(m)) |>
    ggplot(aes(x = SMOKE)) + geom_point(aes(y = BMI), size = 1) + ggtitle("Relationship between SMOKE ar

# TUE
m = lm(BMI ~ poly(TUE, 4), data = trainData)
trainData |>
    mutate(fitted = fitted(m)) |>
    ggplot(aes(x = TUE)) + geom_point(aes(y = BMI), size = 1) + geom_line(aes(y = fitted),
    colour = "red", linewidth = 1) + ggtitle("Relationship between TUE and BMI")
# Spline regression on the numerical variables to see if its a better approach

# NCP
m = lm(BMI ~ ns(NCP, df = 5), data = trainData)
p = predict(m, newdata = testData)
rmse1 = sqrt(mean((BMI - p)^2))

# TUE
m = lm(BMI ~ ns(TUE, df = 5), data = trainData)
p = predict(m, newdata = testData)
rmse2 = sqrt(mean((BMI - p)^2))

cat("Number of main meals =", rmse1, "\nTime using electronics =", rmse2)
# Lasso regression with all the lower error variables

# Create model
fit = lm(BMI ~ Gender + NCP + SMOKE + TUE, data = trainData)
X = model.matrix(fit)
y = BMI

# Remove intercept
X = X[, -1]
fit_lasso = glmnet(X, y)
X_test = model.matrix(~Gender + NCP + SMOKE + TUE, data = testData)
X_test = as.matrix(X_test)
# Remove intercept
X_test = X_test[, -1]
y_test = BMI[1:634]
predLasso = predict(fit_lasso, newx = X_test, s = 1.2)
rmseLasso = sqrt(mean((y_test - predLasso)^2))
cat("RMSE =", rmseLasso)
# Lasso regression all variables (no weight and height)

# Create model
fit = lm(BMI ~ Gender + Age + family_history_with_overweight + FAVC + FCVC + NCP + CAEC + SMOKE +
    CH2O + SCC + FAF + TUE + CALC + MTRANS, data = trainData)
X = model.matrix(fit)
y = BMI

# Remove intercept
X = X[, -1]
```

```r
fit_lasso = glmnet(X, y)
X_test = model.matrix(~Gender + Age + family_history_with_overweight + FAVC + FCVC + NCP +
    CAEC + SMOKE + CH2O + SCC + FAF + TUE + CALC + MTRANS, data = testData)
X_test = as.matrix(X_test)
# Remove intercept
X_test = X_test[, -1]
y_test = BMI[1:634]
predLasso = predict(fit_lasso, newx = X_test, s = 4)
rmseLasso = sqrt(mean((y_test - predLasso)^2))
cat("RMSE =", rmseLasso)
# Lasso regression all variables (no weight and height)

# Create model
fit = lm(BMI ~ Gender + SMOKE, data = trainData)
X = model.matrix(fit)
y = BMI

# Remove intercept
X = X[, -1]
fit_lasso = glmnet(X, y)
X_test = model.matrix(~Gender + SMOKE, data = testData)
X_test = as.matrix(X_test)
# Remove intercept
X_test = X_test[, -1]
y_test = BMI[1:634]
predLasso = predict(fit_lasso, newx = X_test, s = 0.27)
rmseLasso = sqrt(mean((y_test - predLasso)^2))
cat("RMSE =", rmseLasso)
cat("Minimum =", min(BMI))
cat("Maximum =", max(BMI))
# Checking for multicollinearity
double_vars = sapply(dataset, function(x) is.numeric(x) && !all(x%%1 == 0))
double_data = trainData[, double_vars]
corMatrix = cor(double_data)
abs(corMatrix)
library(infer)
library(class)
library(Metrics)
library(yardstick)
library(randomForest)
library(party)

train = trainData %>%
    mutate(BMI = Weight/(Height^2), Y = as.factor(as.numeric(BMI > 30)))

test = testData %>%
    mutate(BMI = Weight/(Height^2), Y = as.factor(as.numeric(BMI > 30)))


test = test %>%
    dplyr::select(-BMI, -Weight, -Height, -NObeyesdad)
train = train %>%
    dplyr::select(-BMI, -Weight, -Height, -NObeyesdad)
```

```r
# Is data balanced
prop_table = train %>%
    group_by(Y) %>%
    summarize(proportion = n()/(nrow(train)))
kable(prop_table)


# Binary variable indicating whether an individual is obese or not obese
target = train$Y
# The actual classes of the individuals to compare our models results with
actual = test$Y



categorical_table = tibble(Gender = chisq_test(train, Y ~ Gender)$p_value, fhwo = chisq_test(train,
    Y ~ family_history_with_overweight)$p_value, FAVC = chisq_test(train, Y ~ FAVC)$p_value,
    CAEC = chisq_test(train, Y ~ CAEC)$p_value, SMOKE = chisq_test(train, Y ~ SMOKE)$p_value,
    SCC = chisq_test(train, Y ~ SCC)$p_value, CALC = chisq_test(train, Y ~ CALC)$p_value, MTRANS = chis
        Y ~ MTRANS)$p_value)

# convert very small decimals to strings so are in table output
categorical_table = categorical_table %>%
    mutate_if(is.numeric, funs(as.character(signif(., 3))))

Variable = matrix(c("p-value"), nrow = 1)

kable(cbind(Variable, categorical_table), caption = "\\label{tab2}P-values for Chi-Squared Test",
    align = "c", longtable = TRUE)


# chi-square test for categorical variables

# Numerical variables
numerical_table = tibble(Age = t_test(train, Age ~ Y)$p_value, FCVC = t_test(train, FCVC ~
    Y)$p_value, NCP = t_test(train, NCP ~ Y)$p_value, CH2O = t_test(train, CH2O ~ Y)$p_value,
    FAF = t_test(train, FAF ~ Y)$p_value, TUE = t_test(train, TUE ~ Y)$p_value)

numerical_table = numerical_table %>%
    mutate_if(is.numeric, funs(as.character(signif(., 3))))

kable(cbind(Variable, numerical_table), caption = "\\label{tab2}P-values for T-test", align = "c",
    longtable = TRUE)


# Remove independent variables remove independent variables with no difference with mean
# values for two classes of Y
train = train %>%
    dplyr::select(-Gender, -SMOKE, -NCP)
test = test %>%
    dplyr::select(-Gender, -SMOKE, -NCP)
```

```r
log_model = glm(Y ~ ., data = train, family = "binomial")
coefficients(log_model)




# only use numeric values for knn so calculates distance properly
cols = sapply(train, is.numeric)

# keep Y non-numeric target variable
cols[length(cols)] = TRUE

# alter dataset in preparation for knn() function
train_k = train[cols]
test_k = test[cols]


# Find the best odd K value
max_Acc = -Inf
best_k = 0

accuracys = numeric(50)
f_scores = numeric(50)
roc = numeric(50)

max_f = -Inf
f_k = 0

max_a = -Inf
a_k = 0

for (i in seq(from = 0, to = 49, by = 1)) {

    pred_class2 = knn(train = train_k[, -length(train_k)], test = test_k[, -length(test_k)],
        cl = train_k[, length(train_k)], k = (2 * i + 1))
    knn_prob = knn(train = train_k[, -length(train_k)], test = test_k[, -length(test_k)], cl = train_k[
        length(train_k)], k = (2 * i + 1), prob = TRUE)

    accuracys[i + 1] = mean(pred_class2 == actual)

    f_scores[i + 1] = fbeta_score(as.numeric(actual), as.numeric(pred_class2), beta = 1)

    knn_tab = tibble(truth = actual, estimate = attr(knn_prob, "prob"))

    roc[i + 1] = roc_auc(data = knn_tab, truth = truth, estimate, event_level = "second")$.estimate

    if (roc[i + 1] > max_a) {
        max_a = roc[i + 1]
        a_k = (2 * i + 1)
    }
    if (f_scores[i + 1] > max_f) {
        max_f = f_scores[i + 1]
        f_k = (2 * i + 1)
```

```
    }

    if (accuracys[i + 1] > max_Acc) {
        max_Acc = accuracys[i + 1]
        best_k = (2 * i + 1)
    }

}
# plot(accuracys) max_Acc best_k plot(f_scores) max_f f_k plot(roc) max_a a_k

knn_prob = knn(train = train_k[, -length(train_k)], test = test_k[, -length(test_k)], cl = train_k[,
    length(train_k)], k = a_k, prob = TRUE)


pred_class = knn(train = train_k[, -length(train_k)], test = test_k[, -length(test_k)], cl = train_k[,
    length(train_k)], k = a_k)

# table_knn = table(actual, pred_class) accuracy_knn =
# sum(diag(table_knn))/sum(table_knn)
accuracy_knn = mean(pred_class == actual)


tree = ctree(Y ~ ., data = train)

class_pred = predict(tree, newdata = test)

table_t = table(actual, class_pred)

accuracy_t = sum(diag(table_t))/sum(table_t)
accuracy_t



rf = randomForest(Y ~ ., data = train)

p = predict(rf, newdata = test)

table_rf = table(actual, p)

prob_rf = predict(rf, newdata = test, type = "prob")

rf_probs = prob_rf[, 2]

mean(actual == p)
kable(list(table_t, table_rf))
# kable(table_rf)

prob_l = predict(log_model, newdata = test, type = "response")
# pred_class_l = as.numeric(prob_l > 0.5)

log_tab = tibble(truth = actual, estimate = prob_l, model = "Logistic Regression")
```

```r
tree_prob = predict(tree, newdata = test, type = "prob")
probs = unlist(lapply(tree_prob, FUN = function(x) {
    x[2]
}))

knn_tab = tibble(truth = actual, estimate = attr(knn_prob, "prob"), model = "Nearest Neighbors")

tree_tab <- tibble(truth = actual, estimate = probs, model = "Decision tree")

tree_tab2 <- tibble(truth = actual, estimate = rf_probs, model = "Random Forest")

data_pred <- bind_rows(knn_tab, tree_tab, tree_tab2, log_tab)


# *** Since the data is balanced ROC
data_pred %>%
    group_by(model) %>%
    roc_curve(truth, estimate, event_level = "second") %>%
    autoplot()

table_auc = data_pred %>%
    group_by(model) %>%
    roc_auc(truth = truth, estimate, event_level = "second")

# DATA CLEANING to get table in correct format to be displayed
table_auc = table_auc %>%
    dplyr::select(-.metric, -.estimator) %>%
    pivot_wider(names_from = model, values_from = .estimate)
Model = matrix(c("AUC"), nrow = 1)

kable(cbind(Model, table_auc), caption = "\\label{tab2}Area Under the Curve for ROC", align = "c",
    longtable = TRUE)

# data_pred %>% group_by(model) %>% pr_curve(truth, estimate, event_level = 'second') %>%
# autoplot() data_pred %>% group_by(model) %>% pr_auc(truth = truth, estimate,
# event_level = 'second')
```