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 an 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. In recent years, many first world countries are experiencing an epidemic of obesity, including Canada. In 2003, it was reported that Canadian adults had a 30% obesity rate, which was an increase from the 21% reported in 2003 [2]. In order to reverse this trend, it is necessary to be able to diagnose obesity early on for those at risk so that they can be provided with preventative therapies to decrease body fat percentage. To be able to diagnose early obesity we must first have a method of measuring levels of obesity.

There are many different tools for measuring obesity levels, some of which include 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 body mass index (BMI) which is calculated using an individuals weight (in kilograms) and height (in metres), with the following formula: $BMI = \frac{weight}{height^2}$. Unfortunately, in many cases the height and weight of individuals are not available to be used to determine BMI for diagnosing obesity level.

In this paper, different statistical models were created that attempt to predict obesity levels based on variables other than height and weight. These models were trained and tested using data obtained from a study conducted at Coast University in Colombia [1]. The data includes 17 different attributes regarding an individual's lifestyle choices and patterns. This study will be explained in greater depth in the following "Data Analysis" section.

Using data analysis, it was found that the following variables are the most valuable in evaluating an individual's risk of obesity: *insert the variables used.* Various regression models were built using BMI as the target variable and the rest of the variables in the data set as the predictor variables to determine which ones are the most valuable in predicting obesity. In doing so, an individual's risk of obesity is able to be predicted using information about their lifestyle. The regression methods used to create these models were Linear, Spline, Lasso, and Ridge regression. Root mean squared error (RMSE) was calculated for each model as a means of comparing which model had the greatest prediction ability.

To complement the various regression models, classification models were built using the same data. These models perform a binary classification predicting whether or not an individual is obese. We performed a filtering variable selection to determine the features that would be used in training the models. The classification methods used to create these models include logistic regression, nearest neighbors, and decisions trees. For each of these models, techniques were then applied in attempts in improve the models ability to predict obesity. The resulting models are then analyzed by classification metrics such as accuracy and the Receiver Operating Characteristic (ROC) curve to determine the best model.

Data Analysis

The data used as the basis for this paper 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.

In our regression and classification models we use the same training and testing data. To do this we split our data up so that 70% of the original data will be used as the training data and the other 30% will be used to test our models. We first check our data set for missing values and find that there are no missing entries. We then convert all categorical variables to factored variables with levels so that we can involve them in our models. BMI levels are not present in the given dataset so we directly use height and weight with the above formula to add BMI values for each individual to our dataset.

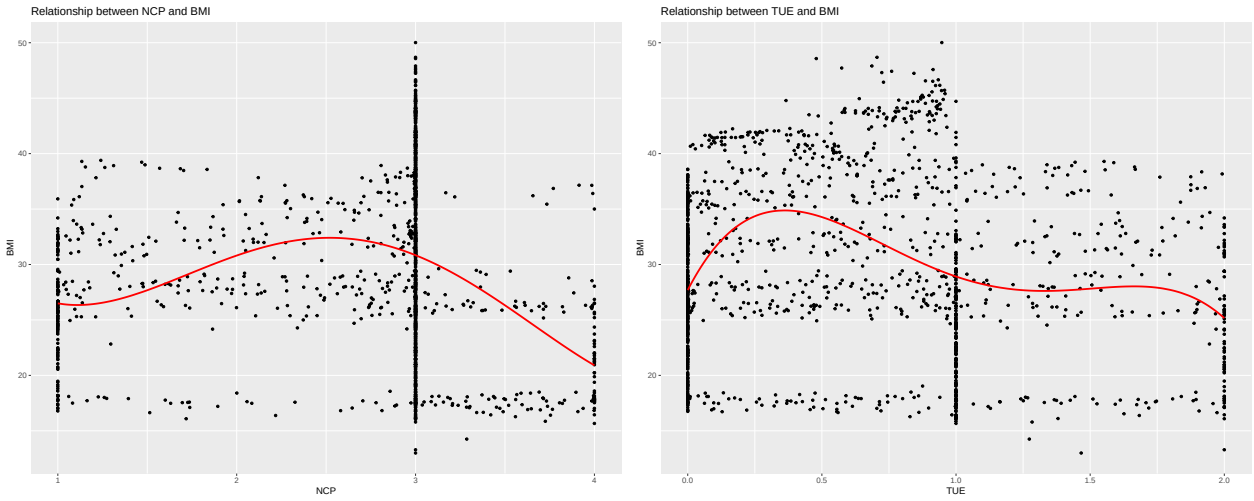
Methods

Regression

To ensure that the most effective regression method was used, many different methods were utilized and subsequently compared using their RSME values. These methods were Linear, Spline, Lasso, and Ridge regression. First, in order to determine which variables in the dataset are the most valuable in predicting BMI, linear regression models were created using each variable available, with the exception of height and weight, which were used directly for calculating BMI, as well as obesity level category, which was categorized directly from BMI. Prior to making the models, the categorical variables in the dataset were factorized to allow them to be used in the model. The following list presents the 14 variables that were considered alongside their respective linear regression model's RSME values in order from least to greatest:

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

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

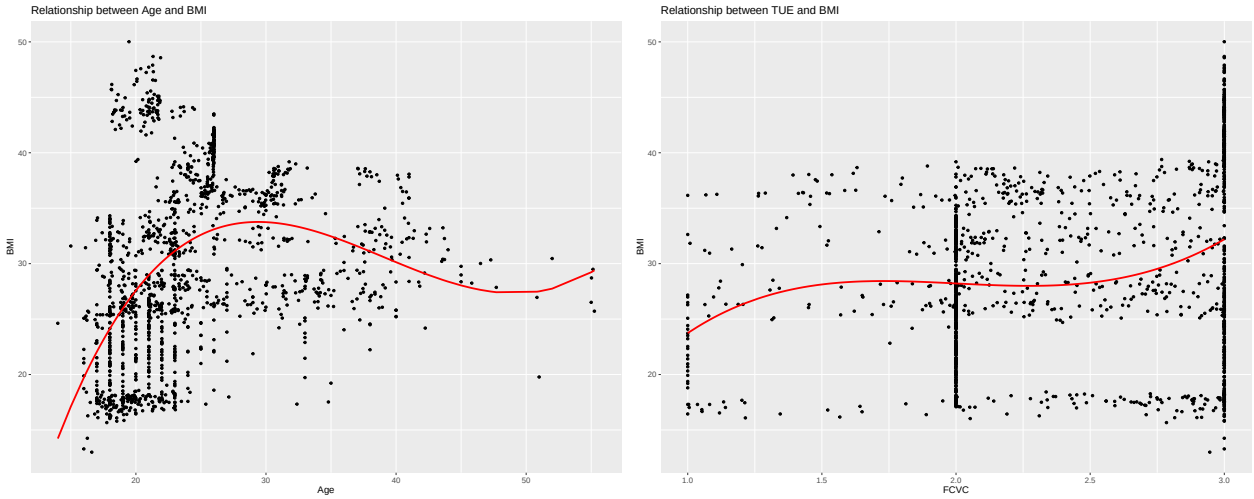


From our previous progress report we noticed that the correlation with Vegetable consumption and Age were quite high as you can see here.

Table 1: Correlation

Variable	Age	Vegetable Consumption	Meals Quantity	Water Consumption	Physical Activity	Screen Time
Correlation	0.4006	0.2605	-0.0526	0.1593	-0.1686	-0.0752

This led us to want to plot those variables too. Here are those plots.



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 = 8.458963
```

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 individuals 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 values of if an individual is obese or not. Based on the CDC's definition of obese, an individual with a BMI level greater than 30 is considered obese [3]. We require these true class values on our dataset for the developing and testing of our models. Therefore we add the following variable to our 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 is essential to the purpose behind building our models as 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 is 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.

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 and simple method. For each feature we perform the appropriate statistical test to determine its relationship with the target variable Y.

For the categorical variables, we 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.

Table 2: 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

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 whether 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 numerical feature we will test whether their mean values within the two groups are significantly different.

Table 3: 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 Table 2 we see that all but two categorical variables have a resulting p-value less than 0.01 from the Chi-Squared tests. Therefore we fail to reject the null hypotheses for both Gender and SMOKE at a 1% significance level. We have insufficient evidence that obesity is dependent of Gender and also SMOKE. From the results in Table 3 we see that only NCP has a p-value greater than 0.01. Therefore, for all numerical variables but NCP we reject the null hypothesis at a 1% significance level. In NCP test, we fail to reject the null hypothesis and conclude that we have insufficient evidence to conclude that the mean NCP differs between the obese and non-obese. From these result we not include the following three variables in the developing of our models: Gender, SMOKE, and NCP.

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, but in this report, we focus on logistic regression, nearest neighbors, and decision trees.

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 compared to 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, that is, predict whether or not that individual is obese given the values of covariates. 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, and adequacy of the model. For simplicity we will assume these properties are satisfied for our dataset.

We created a logistic model using all features of our data set and it resulted in an accuracy of 0.755 when tested on the testing data. To improve our model we explore regularized regression through Lasso regression. However, since our data does not have a large number of features compared to the 2111 total number of observations, and as noted early there are no signs of multicollinearity, we do not expect to see a big improvement in our results. This is exactly what we found when creating a Lasso Logistic regression model. The accuracy of the model did not differ from our original Logistic model. Since Lasso can be used for variable selection we examine the coefficients of all the features created by this model. We find that the category 'no' from CALC variable and the category 'Bike' from MTRANS are the two coefficients that have

gone to zero. This suggests that these two values of these covariates do not have a significant impact on the expected value of the outcome variable Y. We also observe that the category 'yes' from the variable family_history_with_overweight has the largest coefficient absolute value, indicating that it significantly impacts the expected value of indicating obesity.

Nearest Neighbors

While some classification models require assumptions, other models such as Nearest Neighbors do not make any assumptions about the structure of the data. Nearest Neighbors is a simple model that involves determining the class of an observation based on the class of the observations that surround it. Typically, the Euclidean distance is the measure used to calculate the distance between two observations base on values of their 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 them from this model. Therefore we will only consider the 5 numerical features that were selected: AGE, FCVC, CH2O, FAF, adn TUE. However, there are techniques to convert categorical variables to numeric while keeping there relationship consistent and this should be explored in future analysis for improving our models. The class of a given observation is then determined by considering what the majority class of the K observations that are closest. K is a hyper-parameter of the model that determines how many nearest neighbors to consider when classifying a given observation.

Determining the value of K is specific to the data at hand and there is not a single value of K that will work best all of the time. Hence, we will calculate multiple models and compare them to determine the optimal value of K. We proceed by creating 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 because we have a binary classification, so if K were even then the majority class between its K nearest neighbors could result in 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 considering appropriate metrics such as Accuracy and Area Under the Curve (AUC).

Decision Tree

Nearest Neighbors is flexible as there are not assumptions but it is not the most interpretable as it is difficult to tell which features are the most important in determining the resulting class. Like Nearest Neighbors, Decision trees are another classification model that are flexible and do not make assumptions about the data. The main advantage of decision trees is its easy interpretation. Decision 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 consider and evaluate the bias-variance trade-off when consider the size and depth of the tree to avoid perfect classification model that overfit the data.

We use all of the selected categorical and numerical variables in the development of the decision tree. From this tree we 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 history of overweight family maximizes the information gain out of all possible predicates. Since this variable is binary (yes/no), it tells use that this feature does the best job at segregating individuals so that the entropy of the child nodes is minimized based on the classification 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 risks 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. Random forests then combine characteristics of the resulting trees to produce a final model.

We created a Random Forest model on our training dataset in attempt to improve our original decision tree performance. Table 4 and Table **??** display the resulting confusion matrices when we applied these two models to our training data.

Table 4: Decision Tree Confusion Matrix

	0	1
0	273	58
1	48	255

Table 5: Random Forest Confusion Matrix

	0	1
0	296	35
1	42	261

Results

There are various numerical measures of classification models including accuracy, precision, and recall. When deciding on the most effective measure it is important to consider if our dataset is balanced. Since the target variable Y is divided into two classes of obese and non-obese which have proportions 0.45 and 0.55, respectively, we can say that the binary class of obesity has a balanced population. Therefore accuracy is an appropriate measure of our classification models. In our analysis we will compare the following four models that we developed: the original Logistic model, the nearest neighbors model with the optimal K value, the basic decision tree, and the random forest model. Table 6 is a table containing the accuracy results from applying each model to the test data.

Table 6: Accuracy of Models

Logistic Regression	Nearest Neighbors	Decision tree	Random Forest
0.756	0.765	0.833	0.879

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 their effectiveness. 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 the 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 it suggests that we could get a better performance by swapping the binary classes within the model. The following figure displays the ROC curve for all four of our models.

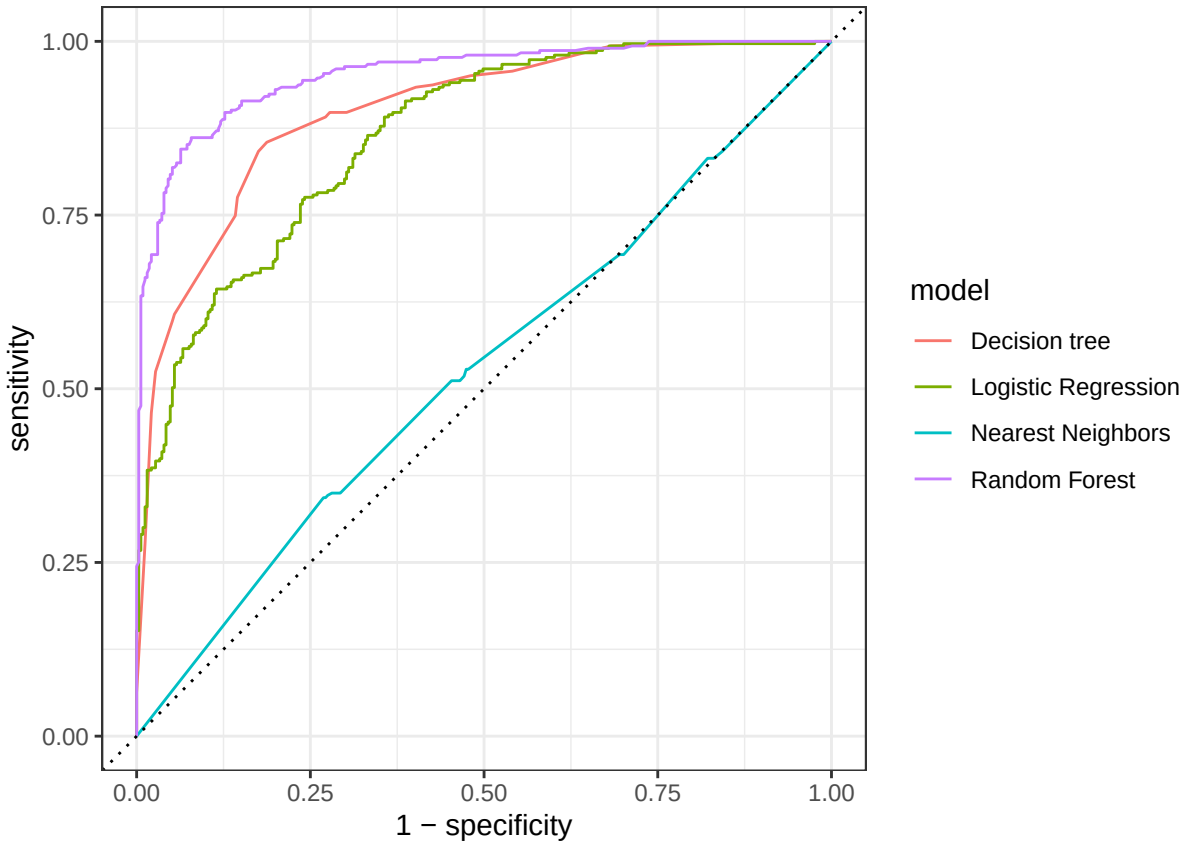


Table 7: Area Under the Curve for ROC

Model	Decision tree	Logistic Regression	Nearest Neighbors	Random Forest
AUC	0.899	0.863	0.527	0.952

From the plot, we see that the nearest neighbor model had the worst performance with an AUC of *0.527*. This is insightful as the nearest neighbor classification was trained only using the numerical features, which suggests that the categorical variables play a strong role in predicting obesity. The logistic regression model involving all features has an AUC of *0.863*, performing much better than nearest neighbors but not nearly as strong as the decision tree model and the random forest model which had AUC values of *0.899* and *0.952*, respectively. The weaker performance of the logistic model may be a result of certain assumptions that were not satisfied, such as the relationship between the features and the log-odds of the target variable being non-linear. The ROC curve and AUC values makes it clear that the random forest classification model has the best capability of predicting obesity in individuals.

In the variable selection for our classification models, we determined through statistical testing that obesity is likely not dependent on the following variables: Gender, SMOKE, and NCP. Furthermore, from developing the these classification models, we discovered from the logistic regression coefficients and from the chosen root predicate of the decision tree that the variable family_history_with_overweight has a significant role in predicting the obesity of an individual.

Conclusion

In conclusion, this report explored possible relationships between obesity and characteristics/behaviours of individuals. Using regression, we searched for the best model to predict BMI levels directly. However, we were unable to reduce the error in the models significantly. This indicates that available variables may not be the best for predicting the BMI of individuals and that other variables should be explored. We then explored classification models to predict the binary outcome of whether or not an individual is obese. We used the following three main methods in developing models: logistic regression, nearest neighbors, and decision trees. The model that had the best performance was the decision tree model, which produced excellent capabilities in predicting whether or not an individual was obese. This model could be used as a tool to predict obesity in situations when weight and height are not available. In screening patient data without weight and height the ability to diagnose someone with obesity allows the early initiation of health protocols to prevent the obesity associated diseases from developing in individuals.

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
3. Defining Adult Overweight & Obesity

University of Manitoba DATA-2010 slides

Appendix

```
knitr::opts_chunk$set(echo = FALSE, warning = FALSE, message=FALSE, tidy.opts=list(width.cutoff=90), ti
# Setup Code Chunk
# Add any libraries etc used here.
library(ggplot2)
library(tidyverse)
library(splines)
library(glmnet)

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

# Is zero indicating no missing values
# sum(is.na(dataset))

# 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
```

```r
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)
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))
```

```r
# 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("\nSmoke =", rmse8, "\nNumber of main meals =", rmse6, "Gender =", rmse1, "\nTime using electronics
# Lets graph the regression models

# 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")

# 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")
library(kableExtra)
ObesityDataset = dataset %>% mutate(BMI = Weight / (Height^2))
table = ObesityDataset %>% summarize("Age" = round(cor(ObesityDataset$BMI,
                                                       ObesityDataset$Age, method = "spearman"), 4),
                                     "Vegetable Consumption" = round(cor(ObesityDataset$BMI,
                                                                         ObesityDataset$FCVC, method = "spearman"),
                                     "Meals Quantity" = round(cor(ObesityDataset$BMI,
                                                                  ObesityDataset$NCP, method = "spearman"), 4),
                                     "Water Consumption" = round(cor(ObesityDataset$BMI, ObesityDataset$CH2O,method =
                                     "Physical Activity" = round(cor(ObesityDataset$BMI, ObesityDataset$FAF,
                                                                     method = "spearman"),4),
                                     "Screen Time" = round(cor(ObesityDataset$BMI, ObesityDataset$TUE,
                                                               method = "spearman"), 4))

Variable = matrix(c("Correlation"), nrow = 1)

kable(cbind(Variable, table), align = "c", longtable = TRUE, caption = "\\label{tab1}Correlation")
# Lets graph the regression models
```

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

# FCVC
m = lm(BMI ~ poly(FCVC, 4), data = trainData)
trainData |>
  mutate(fitted = fitted(m)) |>
  ggplot(aes(x = FCVC)) + 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 +
X = model.matrix(fit)
```

```r
y = BMI

# Remove intercept
X = X[, -1]
fit_lasso = glmnet(X, y)
X_test = model.matrix(~Gender + Age + family_history_with_overweight + FAVC + FCVC + NCP + CAEC + SMOKE
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)
library(glmnet)

# Add class variabke Y
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)))

# Remove unwanted variables
```

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


# Check data balanced
prop_table = train %>%
  group_by(Y) %>%
  summarize(proportion = n()/(nrow(train)))
# population proportions for in text later
prop_y0 = prop_table[1,2]
prop_y1 = prop_table[2,2]


# 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
# chi-square test for categorical variables
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)$p_value,
               MTRANS = chisq_test(train, Y ~ MTRANS)$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{tab1}P-values for Chi-Squared Test",
      align = "c", longtable = TRUE)


# Numerical variables p_values
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 and with no difference with mean values for two classes of Y
train = train %>% dplyr::select(-Gender, -SMOKE, -NCP)
```

```r
test = test %>% dplyr::select(-Gender, -SMOKE, -NCP)


# normal logistic regression model
log_model = glm(Y ~ ., data = train, family = "binomial")

# from all coefficients MTRANS variable with Bike category has the largest absolute value
# coefficients(log_model)

# calculate accuracy and get probabilities
prob_l = predict(log_model, newdata = test, type = "response")
pred_class_l = as.numeric(prob_l > 0.5)
accuracy_l = mean(pred_class_l == actual)


# Perform Lasso Logistic regression
X = model.matrix(Y ~ ., data = train)
y = as.numeric(train$Y)
X = X[,-1]

test_noY = test %>% dplyr::select(-Y)
X_test = model.matrix(~ ., data = test_noY)
X_test = as.matrix(X_test)
X_test = X_test[,-1]

# function that does k-fold cross-validation to determine best value for lambda
min_lambda = cv.glmnet(X, y, alpha = 1, family = "binomial")$lambda.min

lasso_model = glmnet(X, y, alpha = 1, family = "binomial", lambda = min_lambda)

# checked coefficients of the model at the set min lambda value
# coef(lasso_model)

# Checked accuracy of lasso which did not improve fomr our original model above
# probabilities_l = predict(lasso_model, newx = X_test, type = "response")
# pred_lasso = ifelse(probabilities_l > 0.5, "1", "0")
# accuracy_lasso = mean(pred_lasso == actual)




# 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
```

```r
max_Acc = -Inf
best_k = 0

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

max_a = -Inf
a_k = 0

# Within each loop we calculate accuracy and AUC for the roc-curve to determine the optimal
# value of K
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)

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

}
# get probabilities from model using optimal K value
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)

# get predictions and calculate accuracy
pred_class = knn(train = train_k[,-length(train_k)], test = test_k[,-length(test_k)],
                    cl = train_k[,length(train_k)], k = a_k)
accuracy_knn = mean(pred_class == actual)

# normal decision tree
tree = ctree(Y ~ ., data = train)

# printed it to observe its structure
# tree

tree_prob = predict(tree, newdata = test, type = "prob")
```

```r
probs = unlist(lapply(tree_prob, FUN = function(x){x[2]}))

class_pred = predict(tree, newdata = test)

# confusion matrix
table_t = table(actual, class_pred)
accuracy_t = sum(diag(table_t))/sum(table_t)


# Improve model with Random Forest function
rf = randomForest(Y ~ ., data = train)

prob_rf = predict(rf, newdata = test, type = "prob")
rf_probs = prob_rf[,2]

pred_rf = predict(rf, newdata = test)
accuracy_rf = mean(actual == pred_rf)
# confusion matrix
table_rf = table(actual, pred_rf)
kable(table_t, align = "clc", caption = "\\label{tab3}Decision Tree Confusion Matrix") %>%
  kable_styling(full_width = F)

kable(table_rf, align = "clc", caption = "\\label{tab4}Random Forest Confusion Matrix") %>%
  kable_styling(full_width = F)

# c("Decision Tree", "Decision Tree", "Random Forest", "Random Forest")

kable(matrix(c(accuracy_l, accuracy_knn, accuracy_t, accuracy_rf), ncol = 4),
      caption = "\\label{tab5}Accuracy of Models",
      align = "c", col.names = c("Logistic Regression", "Nearest Neighbors",
                    "Decision tree", "Random Forest"),
      digits = 3)


# create individual tables to then join into big one to plot all model results together
log_tab = tibble(truth = actual,
                 estimate = prob_l,
                 model = "Logistic Regression")
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{tab6}Area Under the Curve for ROC",
      align = "c", longtable = TRUE,
      digits = 3)
```