The Study of Obesity with Predictive Performances

Dominic Apolo

2024-03-18

Setting up the Data Set

Link: https://www.kaggle.com/datasets/mrsimple07/obesity-prediction

Background: The Obesity Prediction Dataset is like a big collection of information about people's age, habits, and health stuff. It's useful for scientists and doctors who want to figure out why some people become obese and how to help them better.

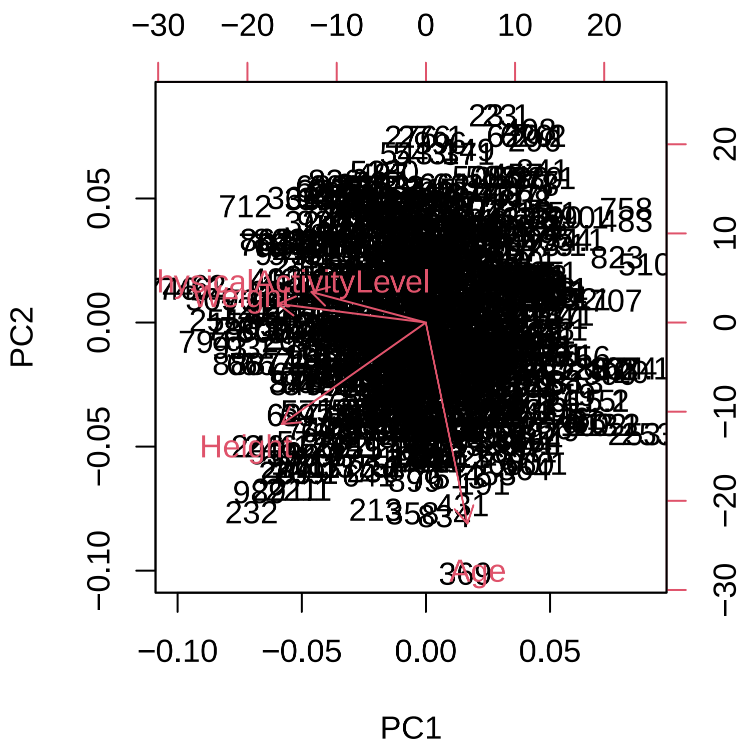
Columns: Age, Gender, Height, Weight, BMI, PhysicalActivityLevel, ObeseityCategory, Obesity Rating, Is_Obese

We'll begin with a Principal Component Analysis (PCA). This is a technique used for dimensionality reduction and data visualization. It's commonly applied to datasets with many variables to identify patterns and relationships among them. Below, you'll see the relationships between the age, physical cctivity level, height, weight. These samples can help us understand the correlation that could affect the BMI level.

```r
# Remove rows with missing values
obesity_data <- na.omit(obesity_data)

# First, create a set of indices to sample from your dataset
indices <- sample(nrow(obesity_data), size = 1000, replace = TRUE)

# Sample 1000 rows from your dataset using the indices
obesity_sample <- obesity_data[indices, c("Age", "PhysicalActivityLevel", "Height", "Weight")]

# Scaling the numeric variables
obesity_scaled <- scale(obesity_sample)

pca_out <- prcomp(obesity_scaled)
summary(pca_out)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4
## Standard deviation     1.0613 1.0109 0.9833 0.9407
## Proportion of Variance 0.2816 0.2555 0.2417 0.2212
## Cumulative Proportion  0.2816 0.5371 0.7788 1.0000
```

```r
# Create a biplot to visualize the principal components and variables
biplot(pca_out, cex=1, cex.axis=1)
```



As you can see in the biplot, the physical activity and weight are clustered together. This can show some relativity between the two columns. However, the age and height are not related. We can also notice that the height has a higher importance than the other traits.

In this case, the standard deviations for the first four principal components (PC1, PC2, PC3, and PC4) are 1.0664, 1.0055, 0.9667, and 0.9577, respectively.PC1 has the highest standard deviation, followed by PC2, PC3, and PC4. This indicates that PC1 captures the most variability in the data, followed by PC2, PC3, and PC4.

Evaluate the Predictive Performance of a model for Predicting a Quantitative Response

```r
obesity_subset <- obesity_data[   , c("BMI", "Age", "PhysicalActivityLevel")]

# Set seed for reproducibility
set.seed(123)

n <- nrow(obesity_subset)

train_indices <- sample(1:n, size = round(0.8 * n))
train_data <- obesity_subset[train_indices, ]
holdout_data <- obesity_subset[-train_indices, ]

# Fit a linear regression model on the training data
lm_model <- lm(formula = BMI ~ Age + PhysicalActivityLevel, data = train_data)

# Predict for training and holdout data
train_predicted <- predict(lm_model, newdata = train_data)
holdout_predicted <- predict(lm_model, newdata = holdout_data)
```

```r
# Assess predictive performance
rmse_train <- sqrt(mean((train_data$BMI - train_predicted)^2))
rmse_holdout <- sqrt(mean((holdout_data$BMI - holdout_predicted)^2))

# Print RMSE for training and holdout data
cat("RMSE on Training Data:", rmse_train, "\n")
```

```
## RMSE on Training Data: 6.098333
```

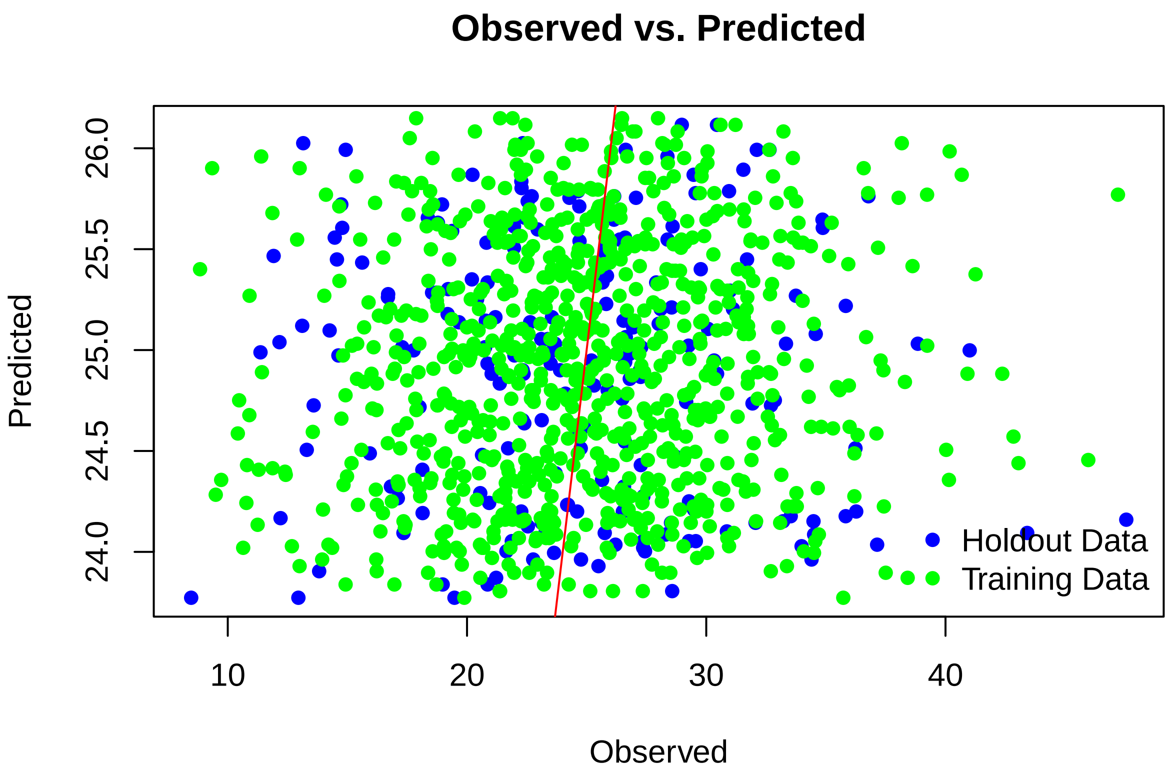
```r
cat("RMSE on Holdout Data:", rmse_holdout, "\n")
```

```
## RMSE on Holdout Data: 6.498254
```

RMSE measures the average magnitude of the errors between predicted and actual values, with lower values indicating better model performance. The RMSE on holdout data should be close to the RMSE on training data, indicating that the model performs well on new data.

This value (6.498254) indicates the average error of the model's predictions when evaluated on new, unseen data (holdout data). It serves as a measure of how well the model generalizes to data it hasn't seen during training.

```r
# Visualize observed vs. predicted on both training and holdout data
plot(holdout_data$BMI, holdout_predicted,
     main = "Observed vs. Predicted",
     xlab = "Observed", ylab = "Predicted",
     col = "blue", pch = 16)

# Add points for observed vs. predicted on training data
points(train_data$BMI, train_predicted,
       col = "green", pch = 16)

# Add reference line for perfect predictions
abline(a = 0, b = 1, col = "red")

# Create a legend
legend("bottomright",
       legend = c("Holdout Data", "Training Data"),
       col = c("blue", "green"),
       pch = 16, cex = 1, bty = "n")
```



This code performs the following steps:

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
# Split the dataset into training and testing sets
set.seed(123) # For reproducibility
train_indices <- sample(nrow(obesity_data), 0.8 * nrow(obesity_data)) # 80% train, 20% test
train_data <- obesity_data[train_indices, ]
test_data <- obesity_data[-train_indices, ]

# Fit logistic regression model
model <- glm(Is_Obese ~ Age + GenderNumbered + Height + PhysicalActivityLevel, data = train_data)
```

```r
predicted <- predict(model, newdata = test_data, type = "response")

# Calculate performance measures
predicted_class <- ifelse(predicted > 0.5, 1, 0)

# Calculate accuracy
accuracy <- mean(predicted_class == test_data$Is_Obese)
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.815"
```

```r
# Create confusion matrix
conf_matrix <- table(Actual = test_data$Is_Obese, Predicted = predicted_class)
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```r
print(conf_matrix)
```

```
##       Predicted
## Actual   0   1
##      0 161   2
##      1  35   2
```

```r
# Calculate sensitivity and specificity
sensitivity <- conf_matrix[2, 2] / sum(test_data$Is_Obese == 1)
specificity <- conf_matrix[1, 1] / sum(test_data$Is_Obese == 0)
print(paste("Sensitivity (True Positive Rate):", sensitivity))
```

```
## [1] "Sensitivity (True Positive Rate): 0.0540540540540541"
```

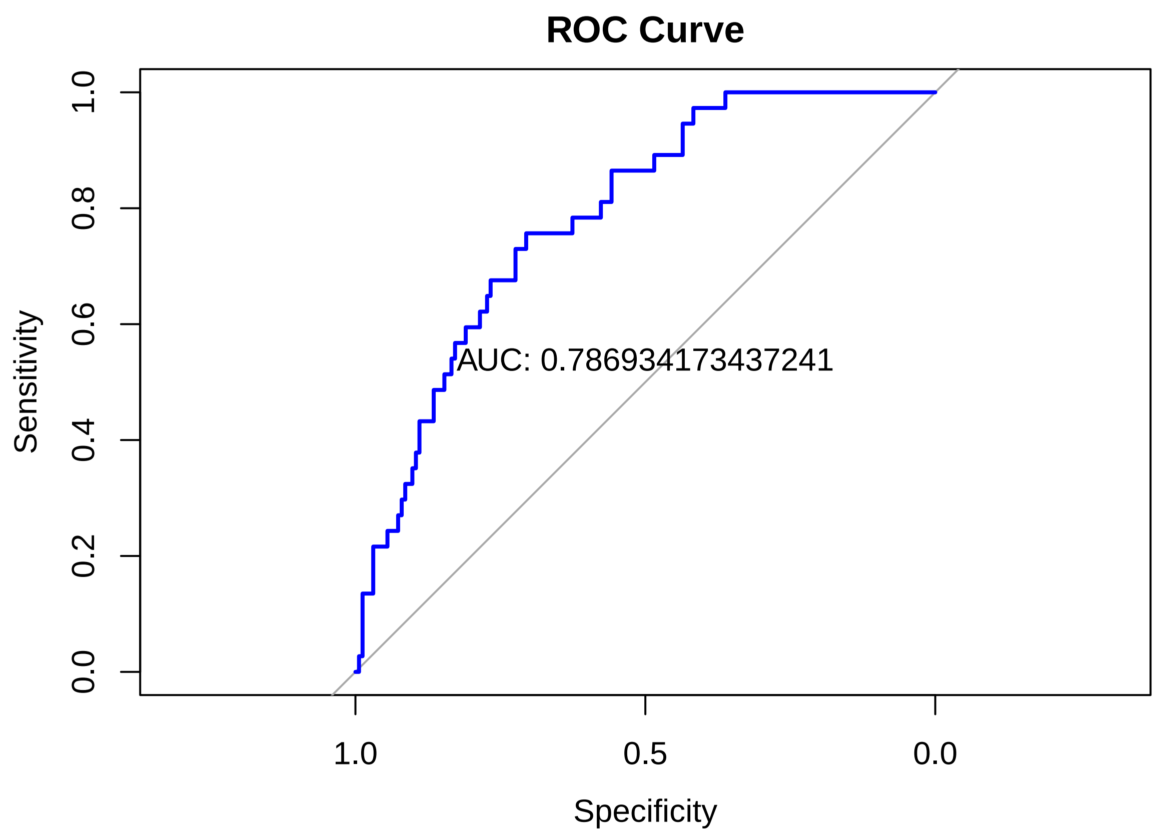
```r
print(paste("Specificity (True Negative Rate):", specificity))
```

```
## [1] "Specificity (True Negative Rate): 0.987730061349693"
```

```r
# 5. Plot ROC curve
roc_curve <- roc(test_data$Is_Obese, predicted)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```r
plot(roc_curve, main = "ROC Curve", col = "blue")
auc_score <- auc(roc_curve)
text(0.5, 0.5, paste("AUC:", auc_score), adj = c(0.5, -0.5), col = "black")
```



```r
print(paste("AUC:", auc_score))
```

```
## [1] "AUC: 0.786934173437241"
```

the accuracy is reported as 81.5%, indicating that the model correctly predicted 81.5% of the instances. AUC values range from 0 to 1, where higher values indicate better model performance. In this case, the AUC is reported as approximately 0.787.We can say that the model is fairyly accurate.

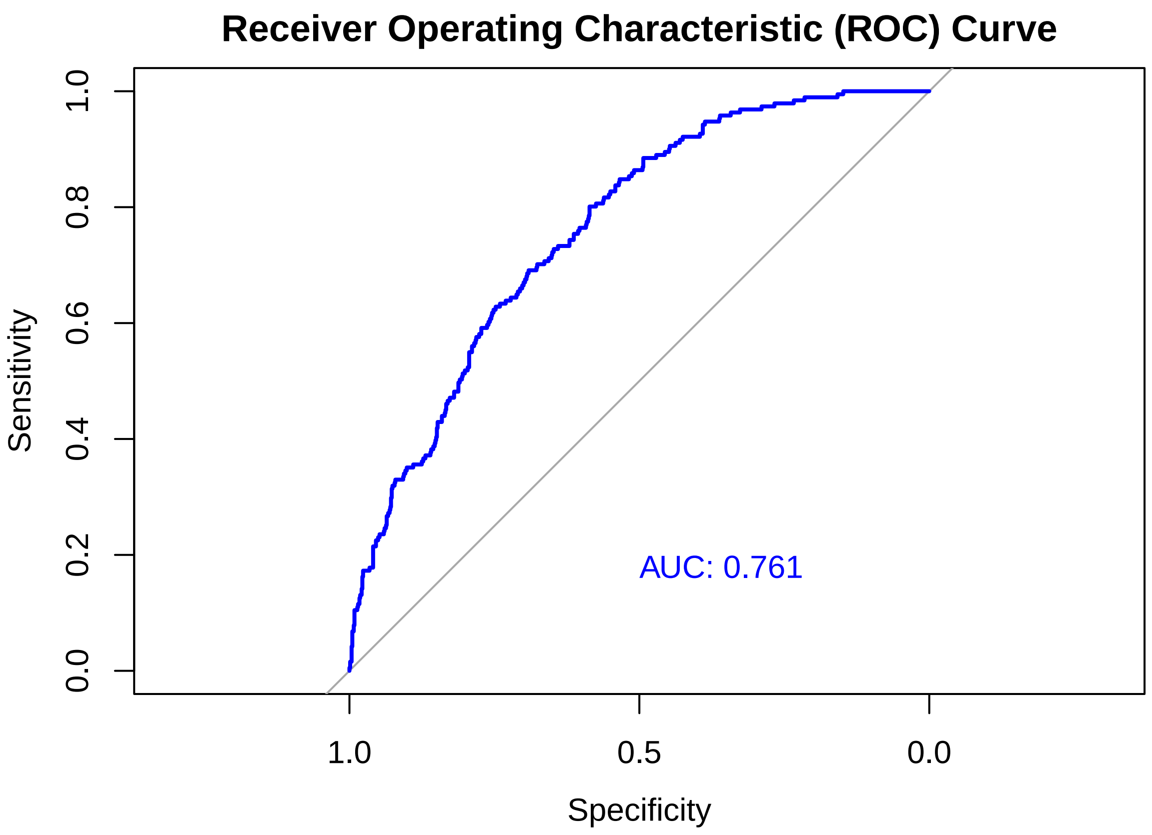
The top-left cell represents True Negatives (TN), indicating the number of instances where the actual class was 0 (negative) and the model predicted it as 0. The top-right cell represents False Positives (FP), indicating the number of instances where the actual class was 0 but the model predicted it as 1.The bottom-left cell represents False Negatives (FN), indicating the number of instances where the actual class was 1 but the model predicted it as 0. The bottom-right cell represents True Positives (TP), indicating the number of instances where the actual class was 1 and the model predicted it as 1.

```r
# Load necessary library
library(pROC)

# Assuming you already have your dataset loaded and it's named obesity_data
# Assign the binary response variable to y_actual
y_actual <- ifelse(obesity_data$obesityCategory == "Obese", 1, 0)

# Fit a logistic regression model
logistic_model <- glm(Is_Obese ~ Age + PhysicalActivityLevel + Height,
                      data = obesity_data,
                      family = binomial)

# Predict probabilities using the fitted model
predicted_probs_model <- predict(logistic_model, newdata = obesity_data, type = "response")
```

```
# Compute ROC curve
roc_curve <- roc(obesity_data$Is_Obese, predicted_probs_model)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
# Plot ROC curve
plot(roc_curve, main = "Receiver Operating Characteristic (ROC) Curve",
     col = "blue", lwd = 2, print.auc = TRUE, print.auc.x = 0.5, print.auc.y = 0.2)
```



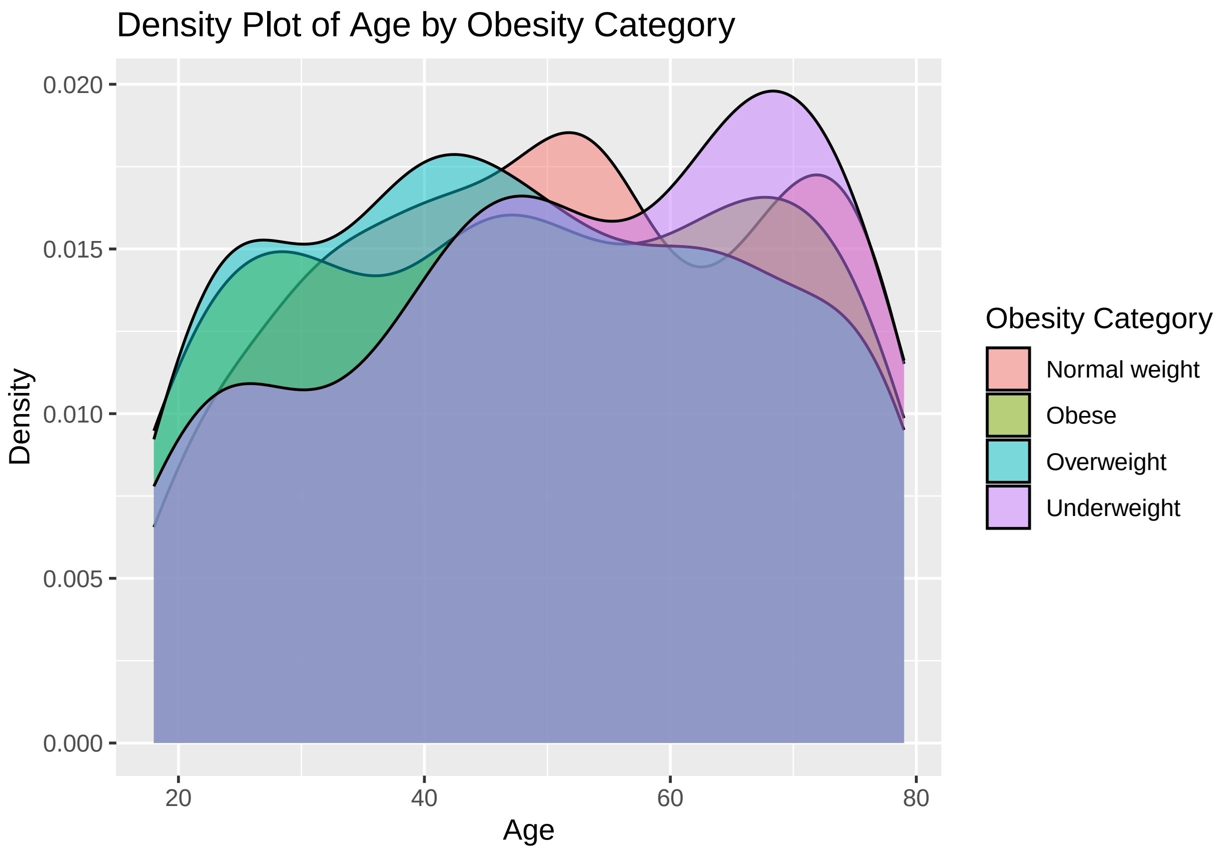
The closer the ROC curve is to the top-left corner, the higher the AUC and the better the model's performance. The further the ROC curve is from the diagonal line, the better the model's discrimination ability. A diagonal line from the bottom-left corner to the top-right corner represents random guessing, with an AUC of 0.5.

```
# Fit logistic regression model
model <- glm(Is_Obese ~ Age + GenderNumbered + Height + Weight + BMI + PhysicalActivityLevel, data = obe

summary(model)
```

```
##
## Call:
## glm(formula = Is_Obese ~ Age + GenderNumbered + Height + Weight +
##     BMI + PhysicalActivityLevel, data = obesity_data)
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -5.710e+00  5.495e-01 -10.391   <2e-16 ***
## Age                     2.614e-04  4.685e-04   0.558    0.577
## GenderNumbered         -1.124e-02  1.701e-02  -0.661    0.509
## Height                   2.847e-02  3.211e-03   8.866   <2e-16 ***
## Weight                  -3.569e-02  3.695e-03  -9.658   <2e-16 ***
## BMI                      1.444e-01  1.052e-02  13.721   <2e-16 ***
## PhysicalActivityLevel -6.105e-05  7.626e-03  -0.008    0.994
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.0715183)
##
##     Null deviance: 154.519  on 999  degrees of freedom
## Residual deviance:  71.018  on 993  degrees of freedom
## AIC: 209.05
##
## Number of Fisher Scoring iterations: 2
```

In this case, the algorithm took 2 iterations to converge, which suggests that the model estimation process was relatively efficient and converged quickly to a solution.

```r
library(ggplot2)

ggplot(obesity_data, aes(x = Age, fill = ObesityCategory)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of Age by Obesity Category", x = "Age", y = "Density", fill = "Obesity Cat
```


Density Plot of Age by Obesity Category
Density
Age
Obesity Category
Normal weight
Obese
Overweight
Underweight