STA363 Project 1 Appendix

Zishan Shao

2022-09-28

```r
knitr::opts_chunk$set(echo = TRUE)
set.seed(114514)
```

```r
# load the data from the dataset
dengue <- read.csv("../Project_1/dengueData.csv", header = TRUE)
```

Section 1: Introduction and Data

Label Analysis & Data Cleaning

```r
# data cleaning & label analysis

# total number of features
numFeatures <- ncol(dengue)

# construct the barplot of the label

# import the ggplot2 for plotting
suppressMessages(library(ggplot2))
ggplot(dengue, aes(x=Y)) + geom_bar(fill='steelblue') +
  labs(x='Dengue Fever (Y = 1: Positive/True ; Y = 0: Negative/False)', main = " ")
```

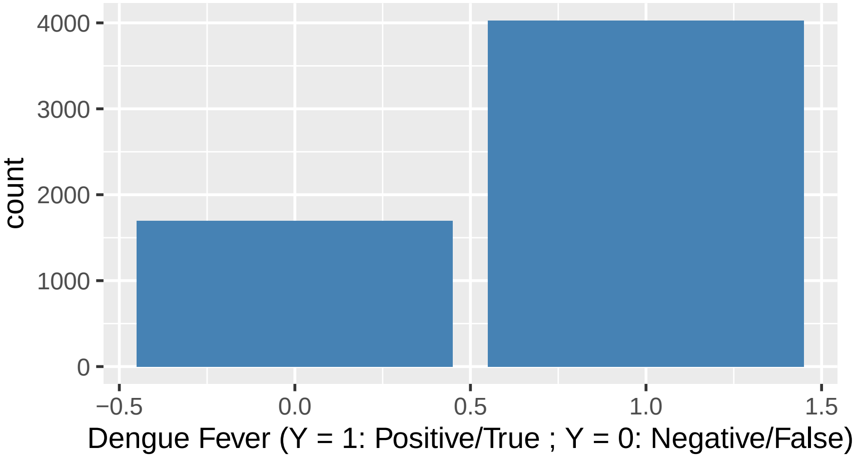


Figure 1.1: Barplot of Label Y

```r
# Data Cleaning

# total number of observations
n <- nrow(dengue)
dengueClean <- na.omit(dengue)
nclean <- nrow(dengueClean)

# number of observations before/after
nobs <- matrix(c(n,nclean), nrow = 2, ncol = 1)
rownames(nobs) <- c("Before Cleaning", "After Cleaning")

# table of observations before/after the cleaning
knitr::kable(nobs, col = "Num Observations",caption = "Data Cleaning")
```

Table 1: Data Cleaning

	Num Observations
Before Cleaning	5726
After Cleaning	5726

Section 2: K-Nearest Neighbor (KNN)

2.1: Brief Example of K-Nearest Neighbor (KNN)

```r
# Example of KNN using Age and Height

# create the exampleSet with the feature: Age and Height with label Y
exampleSet <- dengueClean[,c(2,10,15)]

# reset the names, especially for the label Y
colnames(exampleSet) = c("Age", "Height", "DengueOrNot")
# set the labels as character for later naming in the plot
exampleSet$DengueOrNot <- as.character(exampleSet$DengueOrNot)

# create the train-test set with train-test 80:20 split method
# (I may not use this one in later modeling, but it serve as an efficient example)
set.seed(114514)  # set the seed everytime you need sampling
exampleTestRow <- sample(1:nrow(exampleSet), nrow(exampleSet)*0.2)

exampleTest <- exampleSet[exampleTestRow,]  # create the testing data
exampleTrain <- exampleSet[-exampleTestRow,]  # create the training data

# create the KNN model
suppressMessages(library(class))
# when train the model, read only the features and add the labels for the
# train set for parameter cl the model is to predict the Y value of the test
results <- knn(train = exampleTrain[,c(1,2)],
               test = exampleTest[,c(1,2)],
               cl = exampleTrain$DengueOrNot, k = 5)
```

Prediction with Scatterplot & Model

```
# Prediction Making with Scatterplot

# import the ggplot2 for plotting the empirical logit plot while suppressing
# useless warning information
suppressMessages(library(ggplot2))
ggplot(exampleSet, aes(x=Age, y = Height, color = DengueOrNot)) + geom_point() +
  labs(title =" ", x = "Age (in years)", y = "Height (in cm)") +
  xlim(8,8) + ylim(80, 120) + geom_point(aes(x=8, y=100), colour="black")
```

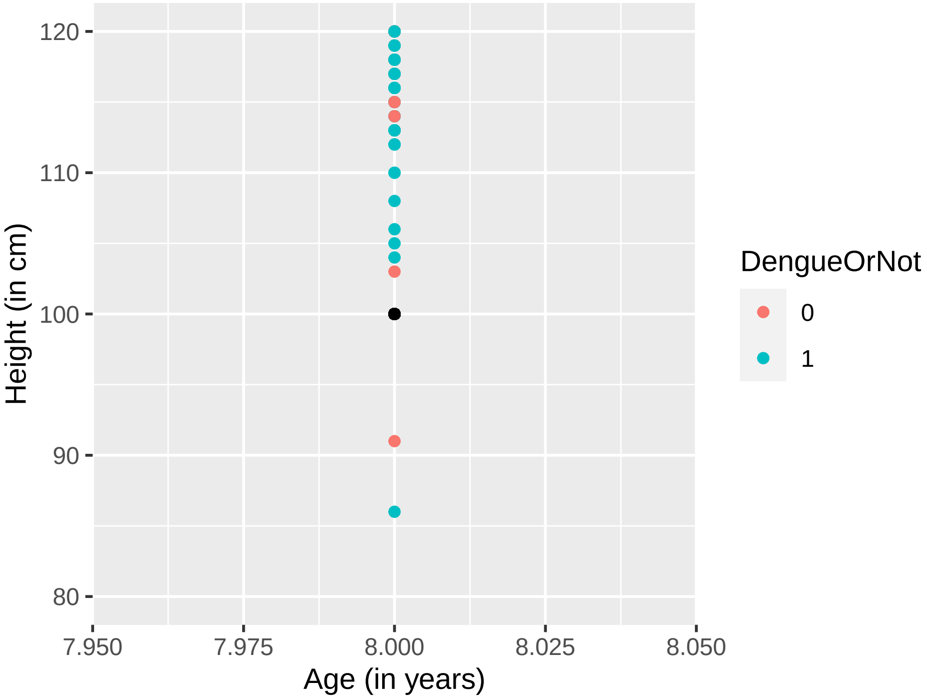


Figure 2.1.1: Scatterplot of Age & Height (Age = 8)

```
### prediction making via KNN model

# Given that a child has age = 8 and 100 cm tall, we could set them as test set
# and make prediction
singleTest <- c(8,100)
# print(singleTest)
 # predict the test set with trained model
results <- knn(train = exampleTrain[,c(1,2)], test = singleTest,
               cl = exampleTrain$DengueOrNot, k = 5)
```

2.2: KNN models (with all features) & train-test split method

Train-Test Spliting

```
set.seed(114514)
# train-test split
# randomly sample the numbers from an ordered vector from 1 to 5726 to represent the
# row indices for the testing set, later we could directly slice the data with the
# indices collected here
testRow <- sample(1:nrow(dengueClean), nrow(dengueClean)*0.2)

# creating the test set with observations with indices in testRow vector
# (20% of total data)
newTest <- dengueClean[testRow,]
# sprintf("Num observations in test set: %d", nrow(newTest))
# creating the training set with observations with indices not in testRow vector
# (80% of total data)
newTrain <- dengueClean[-testRow,]
# sprintf("Num observations in training set: %d", nrow(newTrain))

# find the number of true dengue fever people (Y = 1) and
# true non-dengue people (Y = 0)
# find the number of positive dengue cases in newTest
trueDengueS <- which(newTest$Y == 1)
ntrueDengueS <- length(trueDengueS)
# sprintf("True Dengue Number in test set: %d", ntrueDengueS)

# find the number of negative dengue cases in newTest
trueNotDengueS <- which(newTest$Y == 0)
ntrueNotDengueS <- length(trueNotDengueS)
# sprintf("True Not Dengue Number in test set: %d", ntrueNotDengueS)

# create the table for the observation numbers in each sets
nobsSplit <- c(nrow(newTest), nrow(newTrain), ntrueDengueS, ntrueNotDengueS)
nobsSplit <- matrix(nobsSplit, nrow = 4, ncol = 1)
rownames(nobsSplit) <- c("Num observations in test set",
                         "Num observations in training set",
                         "True Dengue Number in test set",
                         "True Not Dengue Number in test set")
knitr::kable(nobsSplit, col = "Count",
             caption = "Observations in Train/Test Sets & True/False Counts")
```

Table 2: Observations in Train/Test Sets & True/False Counts

	Count
Num observations in test set	1145
Num observations in training set	4581
True Dengue Number in test set	803
True Not Dengue Number in test set	342

Model Creation (via loop)

```r
# create the KNN model
library(class)

# initialize the repository of the result
# because there are totally 4 different values of the k, so the data should
# have 4 sets of K and geometric means
splitResult <- data.frame("K" = rep(NA, 4), "GMean" = rep(NA,4))

# create the index of the splitResult
j = 1

# the train and test data should include all features,
# which is the columns from 1 to 14
for (i in c(3, 5, 7, 9)) {
  # loop over tuning variable k with different values,
  # record the information in splitResult
  results <- knn(train = newTrain[,c(1:14)],
                 test = newTest[,c(1:14)],
                 cl = newTrain$Y, k = i)
  splitResult$K[j] = i

  # sensitivity
  sensitivity = sum(results[trueDengueS] == 1)/ntrueDengueS

  # specificity
  specificity = sum(results[trueNotDengueS] == 0)/ntrueNotDengueS

  # calculate the geometric mean with two metrics above
  splitResult$GMean[j] = round(sqrt(sensitivity*specificity), 3)

  j = j + 1 # so that it sets the index for splitResult
}

# create the table for the results
knitr::kable(splitResult, col.names = c("K", "Geometric Mean"),
             caption= "Models with different k values with train-test split method")
```

Table 3: Models with different k values with train-test split method

K	Geometric Mean
3	0.526
5	0.554
7	0.540
9	0.548

Estimation Variation Test

```r
# resampling the data with a different seed
set.seed(1919810)

testRow <- sample(1:nrow(dengueClean), nrow(dengueClean)*0.2)

# creating the test set with observations with indices in testRow vector
# (20% of total data)
newTest <- dengueClean[testRow,]
# sprintf("Num observations in test set: %d", nrow(newTest))
# creating the training set with observations with indices not in testRow vector
# (80% of total data)
newTrain <- dengueClean[-testRow,]
# sprintf("Num observations in training set: %d", nrow(newTrain))

# find the number of true dengue fever people (Y = 1) and
# true non-dengue people (Y = 0)
# find the number of positive dengue cases in newTest
trueDengueS <- which(newTest$Y == 1)
ntrueDengueS <- length(trueDengueS)
# sprintf("True Dengue Number in test set: %d", ntrueDengueS)

# find the number of negative dengue cases in newTest
trueNotDengueS <- which(newTest$Y == 0)
ntrueNotDengueS <- length(trueNotDengueS)
# sprintf("True Not Dengue Number in test set: %d", ntrueNotDengueS)



# create the KNN model
library(class)

# initialize the repository of the result
# because there are totally 4 different values of the k, so the data
# should have 4 sets of K and geometric means
splitResult <- data.frame("K" = rep(NA, 4), "GMean" = rep(NA,4))

# create the index of the splitResult
j = 1

# the train and test data should include all features, which is the columns from 1 to 14
for (i in c(3, 5, 7, 9)) {
  # loop over tuning variable k with different values, record the
  # information in splitResult
  results <- knn(train = newTrain[,c(1:14)],
                 test = newTest[,c(1:14)],
                 cl = newTrain$Y, k = i)
  splitResult$K[j] = i

  # sensitivity
  sensitivity = sum(results[trueDengueS] == 1)/ntrueDengueS

  # specificity
  specificity = sum(results[trueNotDengueS] == 0)/ntrueNotDengueS
  # calculate the geometric mean with two metrics above
  splitResult$GMean[j] = round(sqrt(sensitivity*specificity), 3)

  j = j + 1 # so that it sets the index for splitResult
}

# create the table for the results
knitr::kable(splitResult, col.names = c("K", "Geometric Mean"),
             caption= "Models with Resampled Data")
```

Table 4: Models with Resampled Data

K	Geometric Mean
3	0.539
5	0.555
7	0.543
9	0.556

Confusion Matrix of KNN (train-test split method) & Metrics Computation

```r
# make predictions with k = 5
resultk5 <- knn(train = newTrain[,c(1:14)],
                test = newTest[,c(1:14)],
                cl = newTrain$Y, k = 5)

# confusion matrix
knitr::kable(table(resultk5, newTest$Y),
             caption= "Confusion Matrix of KNN",
             col = c("True Not Dengue", "True Dengue"))
```

Table 5: Confusion Matrix of KNN

	True Not Dengue	True Dengue
0	120	109
1	220	696

Metric Computation

$$Sensitivity = \frac{TrueDengue \ \& \ PredictedTrueDengue}{Total \ TrueDengue} = \frac{696}{696 + 109} \approx 0.865$$

$$Specificity = \frac{TrueNotDengue \ \& \ Predicted \ TrueNotDengue}{Total \ TrueNotDengue} = \frac{120}{120 + 220} \approx 0.353$$

$$Accuracy = \frac{Total \ Correct \ Predictions}{Total \ Observations} = \frac{696 + 120}{120 + 220 + 696 + 109} \approx 0.713$$

2.3: KNN Models (with All Features) & 10-Fold Cross Validation

```
# K-fold Cross Validation: fold creation
# I will apply 10 fold cross validation because there are 5726 total data, which is very large

# record the validation result for different k values
validResult <- data.frame("K" = rep(NA, 4), "GMean" = rep(NA, 4))

# define the number of observations
n <- nrow(dengueClean)

# create the folds
pool <- rep(1:10, ceiling(n/10))

# for convenience purpose, reset the seed here
set.seed(114514)
# randomly assign elements in pool to folds so that fold could contain index of
# observations to be sampled from dengueClean
folds <- sample(pool,n)

# create the storage space
storageDenK <- data.frame("YHat" = rep(NA,nrow(dengueClean)))
# this one was necessary because it will return all prediction data with dim = [5726, 1]
# however, we don't really need all information here, so I will not print and
# interpret the results
# of storageDenK here.

library(class)  # import the class package for knn

j = 1 # index purpose

for (i in c(3, 5, 7, 9)) {
  validResult$K[j] = i;
  for (f in c(1:10)) {
    # find the data in fold f
    infold <- which(folds ==f)

    # define the train and test data sets
    newTrain <- dengueClean[-infold,]
    newTest <- dengueClean[infold,]

    # find the number of positive & negative dengue cases in newTest
    ntrue <- length(which(newTest$Y == 1))
    ntrueNot <- length(which(newTest$Y == 0))

    # make predictions
    k_preds <- knn(newTrain[,c(1:14)], newTest[,c(1:14)], cl = newTrain$Y, k = i)

    # sensitivity
    sensitivity = sum(k_preds[which(newTest$Y == 1)] == 1)/ntrue

    # specificity
    specificity = sum(k_preds[which(newTest$Y == 0)] == 0)/ntrueNot
    # calculate the geometric mean with two metrics above
    validResult$GMean[j] = round(sqrt(sensitivity*specificity), 3)

    # store predictions
    storageDenK$YHat[infold] <- k_preds
  }
  j = j + 1
}

knitr::kable(validResult, col.names = c("K", "Geometric Mean"),
              caption= "Models with different k values with 10-Fold Cross Validation")
```

Table 6: Models with different k values with 10-Fold Cross Validation

K	Geometric Mean
3	0.533
5	0.529
7	0.523
9	0.538

Confusion Matrix of KNN (10-fold cross validation method)

```
# confusion matrix of k = 9
knitr::kable(table(storageDenK$YHat, dengueClean$Y),
              caption= "Confusion matrix (fold=10) of DengueClean",
              col = c("True Not Dengue", "True Dengue"))
```

Table 7: Confusion matrix (fold=10) of DengueClean

True Not Dengue	True Dengue
525	508
1173	3520

Metrics Computation

$$Sensitivity = \frac{TrueDengue \ \& \ PredictedTrueDengue}{Total \ TrueDengue} = \frac{3520}{3520 + 508} \approx 0.874$$

$$Specificity = \frac{TrueNotDengue \ \& \ Predicted \ TrueNotDengue}{Total \ TrueNotDengue} = \frac{525}{525 + 1173} \approx 0.309$$

$$Accuracy = \frac{Total \ Correct \ Predictions}{Total \ Observations} = \frac{525 + 3520}{525 + 3520 + 508 + 1173} \approx 0.706$$

General Table of Metrics

	Geometric Mean	Sensitivity	Specificity	Accuracy
train-test	0.554	86.5%	35.3%	71.3%
10-fold	0.538	87.4%	30.9%	70.6%

Section 3: Multivariable Logistic Regression

$$Y_i \sim Bernoulli(\pi_i)$$

where in this case

$$\pi_i = P(Y_i = 1|X_i)$$

$$log(\frac{\pi_i}{1-\pi_i}) = \beta_0 + \beta_i X_i, \quad i \in \{1, 2, ..., n\}$$

n is the number of features chosen to be used in the model

3.1 Exploratory Data Analysis (EDA)

EDA (Part 1): Variable Type Analysis & Standardization

```
# print the type of features: categorical or numerical
# aa to store the feature name and range
aa <- data.frame("Feature" = rep(NA, ncol(dengueClean)-1),
                 "RangeBegin" = rep(NA, ncol(dengueClean)-1),
                 "RangeEnd" = rep(NA, ncol(dengueClean)-1))

for (i in c(1:14)) {
  aa$Feature[i] = colnames(dengueClean)[i]
  rr = range(dengueClean[colnames(dengueClean)[i]])
  aa$RangeBegin[i] = rr[1]
  aa$RangeEnd[i] = rr[2]
}

knitr::kable(aa,
             col.names = c("feature", "range begin", "range end"),
             caption= "Feature Range")
```

Table 9: Feature Range

feature	range begin	range end
Sex	1.000000	2.00000
Age	1.000000	15.00000
DayDisease	1.000000	3.00000
Vomiting	1.000000	2.00000
Abdo	1.000000	2.00000
Muco	1.000000	2.00000
Skin	1.000000	2.00000
Temp	35.800000	41.00000
BMI	8.503401	34.70986
Height	58.000000	176.00000
Weight	7.200000	91.00000
Flush	0.000000	1.00000
Hepatomegaly	0.000000	1.00000
Rash	0.000000	1.00000

```
# convert DayDisease to categorical feature
dengueClean$DayDisease <- as.factor(dengueClean$DayDisease)
```

EDA (Part 2): Empirical Logit Plots for Numerical Features

```
# function for the empirical logit plot
emplogitPlot <- function(x, y, binsize = NULL, ci = FALSE, probit = FALSE,
prob = FALSE, main = NULL, xlab = "", ylab = "", lowess.in = FALSE){
  # x          vector with values of the independent variable
  # y          vector of binary responses
  # binsize   integer value specifying bin size (optional)
  # ci        logical value indicating whether to plot approximate
  #           confidence intervals (not supported as of 02/08/2015)
  # probit    logical value indicating whether to plot probits instead
  #           of logits
  # prob      logical value indicating whether to plot probabilities
  #           without transforming
  #
  # the rest are the familiar plotting options

  if(class(y) =="character"){
   y <- as.numeric(as.factor(y))-1
   }

  if (length(x) != length(y))
    stop("x and y lengths differ")
  if (any(y < 0 | y > 1))
    stop("y not between 0 and 1")
  if (length(x) < 100 & is.null(binsize))
    stop("Less than 100 observations: specify binsize manually")

  if (is.null(binsize)) binsize = min(round(length(x)/10), 50)

  if (probit){
    link = qnorm
    if (is.null(main)) main = "Empirical probits"
  } else {
    link = function(x) log(x/(1-x))
    if (is.null(main)) main = "Empirical logits"
  }

  sort = order(x)
  x = x[sort]
  y = y[sort]
  a = seq(1, length(x), by=binsize)
  b = c(a[-1] - 1, length(x))

  prob = xmean = ns = rep(0, length(a)) # ns is for CIs
  for (i in 1:length(a)){
    range = (a[i]):(b[i])
    prob[i] = mean(y[range])
    xmean[i] = mean(x[range])
    ns[i] = b[i] - a[i] + 1 # for CI
  }

  extreme = (prob == 1 | prob == 0)
  prob[prob == 0] = min(prob[!extreme])
  prob[prob == 1] = max(prob[!extreme])

  g = link(prob) # logits (or probits if probit == TRUE)

  linear.fit = lm(g[!extreme] ~ xmean[!extreme])
  b0 = linear.fit$coef[1]
```

```
  b1 = linear.fit$coef[2]

  loess.fit = loess(g[!extreme] ~ xmean[!extreme])

  plot(xmean, g, main=main, xlab=xlab, ylab=ylab)
  abline(b0,b1)
  if(lowess.in ==TRUE){
  lines(loess.fit$x, loess.fit$fitted, lwd=2, lty=2)
  }
}
```

```
# cluster of mosaic plots
op <- par( mfrow = c(2,3))

emplogitPlot(x=dengueClean[,2], y=dengueClean$Y,
             xlab = "Age",
             ylab = "Log Odds Dengue Fever",
             main = "Age")

# emplogitPlot between Temperature and Dengue Fever
  emplogitPlot(x=dengueClean[,8], y=dengueClean$Y,
             xlab = "Temperature (in Celsius)",
             ylab = "Log Odds Dengue Fever",
             main = "Temperature")


# empirical logit plot between BMI and Dengue Fever
  emplogitPlot(x=dengueClean[,9], y=dengueClean$Y,
             xlab = "BMI",
             ylab = "Dengue Fever",
             main = "BMI")


# empirical logit plot between Height and Dengue Fever
  emplogitPlot(x=dengueClean[,10], y=dengueClean$Y,
             xlab = "Height (in cm)",
             ylab = "Dengue Fever",
             main = "Height")


# empirical logit plot between Weight and Dengue Fever
  emplogitPlot(x=dengueClean[,11], y=dengueClean$Y,
             xlab = "Weight (in kg)",
             ylab = "Dengue Fever",
             main = "Weight")
par(op)
```

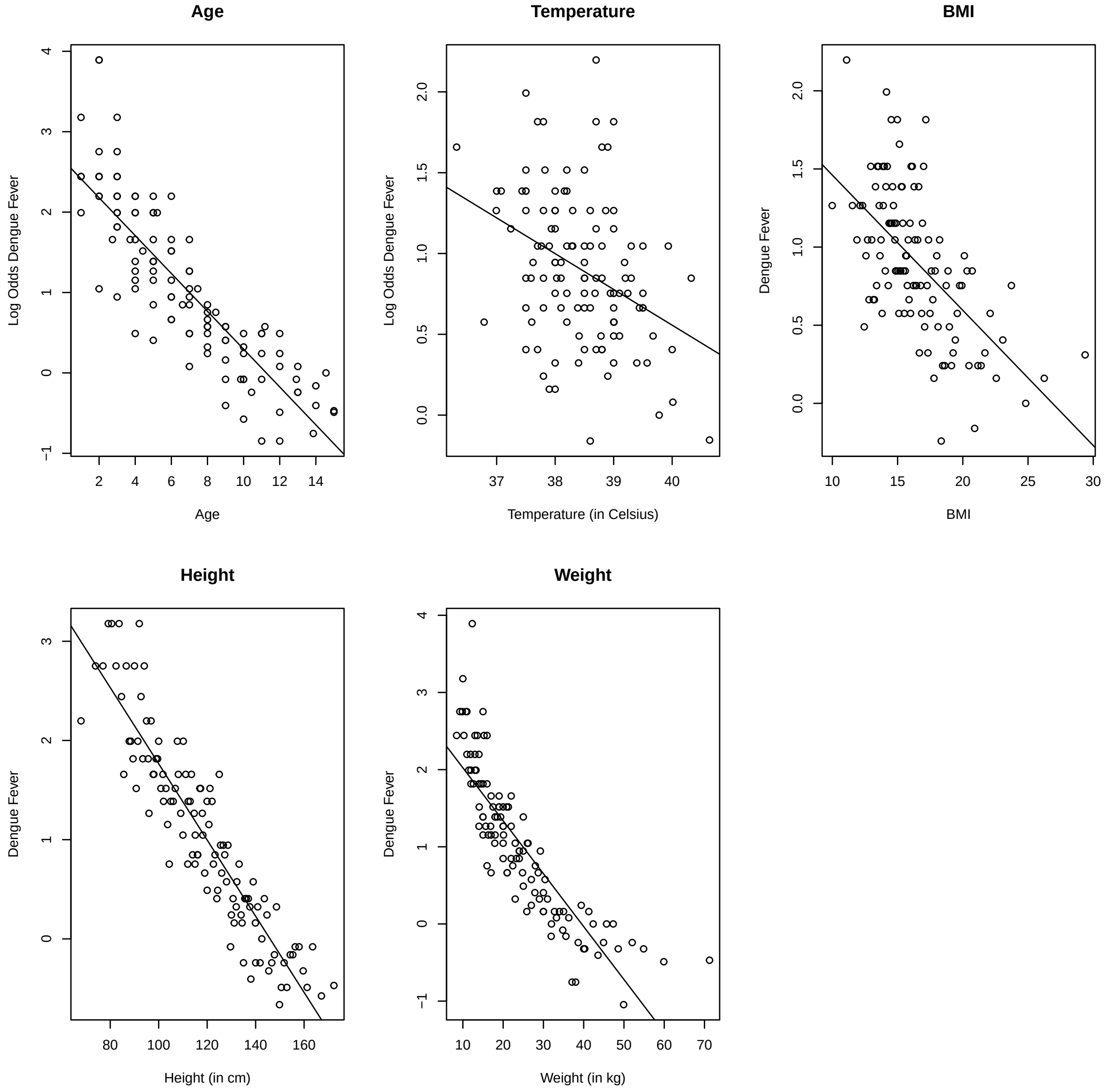


Figure 3.1.1: Relationship Analysis of Numerical Variables vs Log Odds Dengue Fever

```
emplogitPlot(x=log(dengueClean$Weight), y=dengueClean$Y,
             xlab = "Log Weight",
             ylab = "Log Odds Dengue Fever",
             main = " ")
```

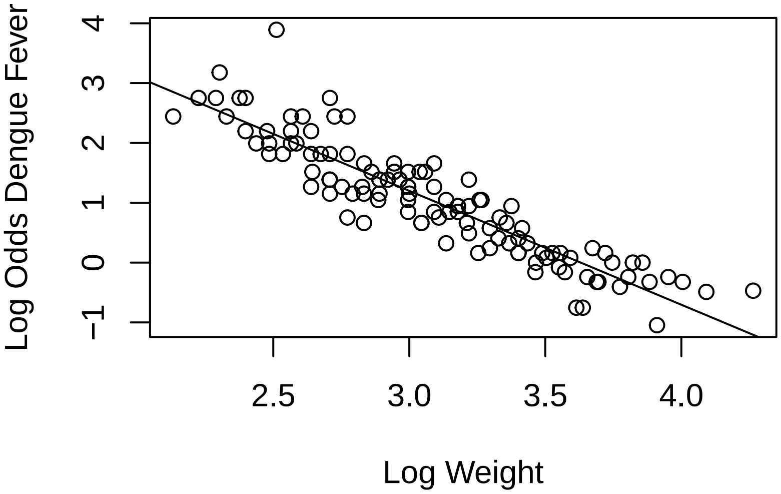


Figure 3.1.2: Log Weight vs Log Odds Dengue Fever

EDA (Part 3): Multicollinearity

```
senpai <- cor(dengueClean[,c(2,8,9,10,11)])
knitr::kable(senpai, caption = "Correlation Matrix of Numerical Variables")
```

Table 10: Correlation Matrix of Numerical Variables

	Age	Temp	BMI	Height	Weight
Age	1.0000000	0.0117088	0.1708381	0.9445484	0.8469863
Temp	0.0117088	1.0000000	0.0408907	0.0229824	0.0335897
BMI	0.1708381	0.0408907	1.0000000	0.1493719	0.5608645
Height	0.9445484	0.0229824	0.1493719	1.0000000	0.8828541
Weight	0.8469863	0.0335897	0.5608645	0.8828541	1.0000000

EDA (Part 4): Categorical variables analysis

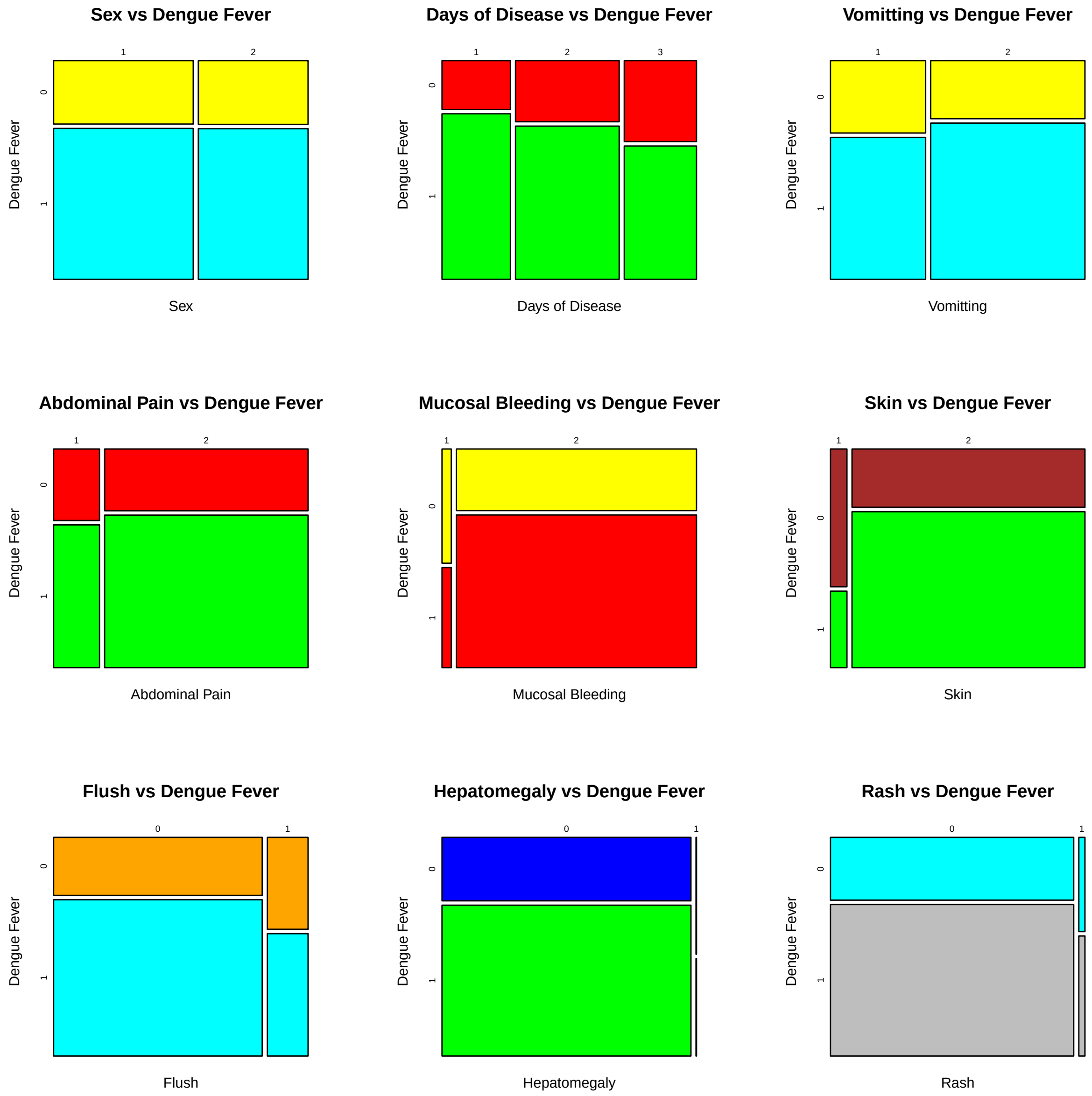
Along with the numerical variables, we use mosaic plot to select categorical features with statistically significant influences of the outcome.

```
op <- par(mfrow=c(3,3))

# dengue fever here is not log odds
mosaicplot(dengueClean$Sex  ~ dengueClean$Y,
           col = c("yellow", "cyan"),xlab = "Sex",
           ylab = "Dengue Fever", main = "Sex vs Dengue Fever")
```

```r
mosaicplot(dengueClean$DayDisease  ~ dengueClean$Y,
           col = c("red", "green"),xlab = "Days of Disease",
           ylab = "Dengue Fever", main = "Days of Disease vs Dengue Fever")
mosaicplot(dengueClean$Vomiting  ~ dengueClean$Y,
           col = c("yellow", "cyan"),xlab = "Vomitting",
           ylab = "Dengue Fever", main = "Vomitting vs Dengue Fever")
mosaicplot(dengueClean$Abdo  ~ dengueClean$Y,
           col = c("red", "green"),xlab = "Abdominal Pain",
           ylab = "Dengue Fever", main = "Abdominal Pain vs Dengue Fever")

mosaicplot(dengueClean$Muco  ~ dengueClean$Y,
           col = c("yellow", "red"),xlab = "Mucosal Bleeding",
           ylab = "Dengue Fever", main = "Mucosal Bleeding vs Dengue Fever")
mosaicplot(dengueClean$Skin  ~ dengueClean$Y,
           col = c("brown", "green"),xlab = "Skin",
           ylab = "Dengue Fever", main = "Skin vs Dengue Fever")
# we need to convert True/False categories into numerical expressions (0/1)
mosaicplot(as.numeric(dengueClean$Flush)  ~ dengueClean$Y,
           col = c("orange", "cyan"),xlab = "Flush",
           ylab = "Dengue Fever", main = "Flush vs Dengue Fever")
mosaicplot(as.numeric(dengueClean$Hepatomegaly)  ~ dengueClean$Y,
           col = c("blue", "green"),xlab = "Hepatomegaly",
           ylab = "Dengue Fever", main = "Hepatomegaly vs Dengue Fever")
mosaicplot(as.numeric(dengueClean$Rash)  ~ dengueClean$Y,
           col = c("cyan", "gray"),xlab = "Rash",
           ylab = "Dengue Fever", main = "Rash vs Dengue Fever")
```



```
par(op)
```

Figure 3.1.3: Relationship Analysis of Categorical Variables and Dengue Fever

3.2: Model building and training

Model 1 Construction & Coefficients Evaluation

```
# Model Building: m1

# make the model and print the coefficient at the end
```

```r
m1 <- glm(as.factor(Y) ~ DayDisease + Vomiting + Abdo +
            Muco + Skin + Flush + Rash + Age + Temp + BMI,
          data = dengueClean, family = "binomial")

# summarize the information of the model
knitr::kable(round(summary(m1)$coefficients,3),
             caption = "Coefficients Table of Logistic Model 1")
```

Table 11: Coefficients Table of Logistic Model 1

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	8.433	1.656	5.093	0.000
DayDisease2	-0.304	0.081	-3.752	0.000
DayDisease3	-0.582	0.086	-6.767	0.000
Vomiting	0.308	0.067	4.602	0.000
Abdo	0.086	0.083	1.036	0.300
Muco	0.166	0.177	0.935	0.350
Skin	1.283	0.133	9.648	0.000
FlushTRUE	-0.617	0.085	-7.261	0.000
RashTRUE	-0.339	0.203	-1.665	0.096
Age	-0.213	0.009	-22.879	0.000
Temp	-0.221	0.042	-5.201	0.000
BMI	-0.033	0.010	-3.198	0.001

Model 2 Construction & Coefficients Evaluation

```r
# Model Building: m2
# Check the deviance
m2 <- glm(as.factor(Y) ~ DayDisease + Vomiting + Skin + Flush + Age +
            Temp + BMI, data = dengueClean, family = "binomial")
knitr::kable(round(summary(m2)$coefficients,3),
             caption = "Coefficients Table of Logistic Model 2")
```

Table 12: Coefficients Table of Logistic Model 2

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	8.476	1.643	5.158	0.000
DayDisease2	-0.309	0.081	-3.817	0.000
DayDisease3	-0.592	0.086	-6.901	0.000
Vomiting	0.321	0.065	4.924	0.000
Skin	1.356	0.123	11.059	0.000
FlushTRUE	-0.643	0.084	-7.672	0.000
Age	-0.212	0.009	-22.877	0.000
Temp	-0.213	0.042	-5.055	0.000
BMI	-0.033	0.010	-3.220	0.001

AIC Table of Model 1 & Model 2

```r
# AIC table between the m1 and m2
aic <-c(AIC(m1),AIC(m2))
aic <- matrix(aic, nrow = 2, ncol = 1)   # create the matrix
rownames(aic) <- c("Model 1", "Model 2")  # rename the columns

# table
knitr::kable(aic, col = "AIC values", caption = "AIC Table of M1 & M2")
```

Table 13: AIC Table of M1 & M2

	AIC values
Model 1	5942.354
Model 2	5941.160

3.3: Prediction & Result Evaluation

```r
# we will put the same train-test data as previous so that it is easier
# to compare the outcomes and thus the relationship between two variables
set.seed(114514)
# train-test split
# randomly sample the numbers from an ordered vector from 1 to 5726 to represent the
# row indices for the testing set, later we could directly slice the data with the
# indices collected here
testRow <- sample(1:nrow(dengueClean), nrow(dengueClean)*0.2)

# creating the test set with observations with indices in testRow vector
# (20% of total data)
newTest <- dengueClean[testRow,]

# creating the training set with observations with indices not in testRow vector
# (80% of total data)
newTrain <- dengueClean[-testRow,]

# find the number of positive dengue cases in newTest
trueDengueS <- which(newTest$Y == 1)
ntrueDengueS <- length(trueDengueS)

# find the number of negative dengue cases in newTest
trueNotDengueS <- which(newTest$Y == 0)
ntrueNotDengueS <- length(trueNotDengueS)
```

```r
# predict the Y of the testing set with threshold 0.5

# If p >= 0.5, it should be dengue fever, if p <= 0.5, not dengue fever
probabilities <- predict(m2, newdata = newTest, type = "resp")
predicted.Y <- ifelse(probabilities >= 0.5, "1", "0") # classify the result
```

```
# print(length(which(predicted.Y == 0)))
# print(length(which(predicted.Y == 1)))
```

Confusion Matrix of Logistic Regression Model (Model 2)

```
# confusion matrix
knitr::kable(table(predicted.Y, newTest$Y),
             caption= "Confusion Matrix of Logistic Regression",
             col = c("True Not Dengue", "True Dengue"))
```

Table 14: Confusion Matrix of Logistic Regression

	True Not Dengue	True Dengue
0	123	86
1	219	717

Metrics Computation

$$Sensitivity = \frac{TrueDengue \ \& \ PredictedTrueDengue}{Total \ TrueDengue} = \frac{717}{717+86} \approx 0.892$$

$$Specificity = \frac{TrueNotDengue \ \& \ Predicted \ TrueNotDengue}{Total \ TrueNotDengue} = \frac{123}{123+219} \approx 0.360$$

$$Accuracy = \frac{Total \ Correct \ Predictions}{Total \ Observations} = \frac{717+123}{717+123+86+219} \approx 0.734$$

Section 4: Compare and Evaluate: Logistic/KNN

Summary Table

Summary Table of Metrics (Logistic vs KNN)

	Sensitivity	Specificity	Accuracy
———	———-	———-	——-
KNN	86.5%	35.3%	71.3%
Logistic	89.2%	36.0%	73.4%

Conclusion