Logistic Regression & Classification in R

VENKATA SESHA SAI ESHWAR VUDHANTHI

2025-04-20

1. Introduction

< Introduce the models being used >

We are using logistic regression and K-Nearest Neighbors (KNN) classification. Logistic regression is used to predict binary outcomes, and KNN is a flexible method that classifies based on proximity to neighboring data points.

2. Data

< Describe the data >

We are using the Default dataset from the ISLR2 package. It includes whether individuals defaulted on their credit card debt, their balance, income, and student status.

```r
data = Default
str(data)
```

```
## 'data.frame':    10000 obs. of  4 variables:
##  $ default: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ student: Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 2 1 1 ...
##  $ balance: num  730 817 1074 529 786 ...
##  $ income : num  44362 12106 31767 35704 38463 ...
```

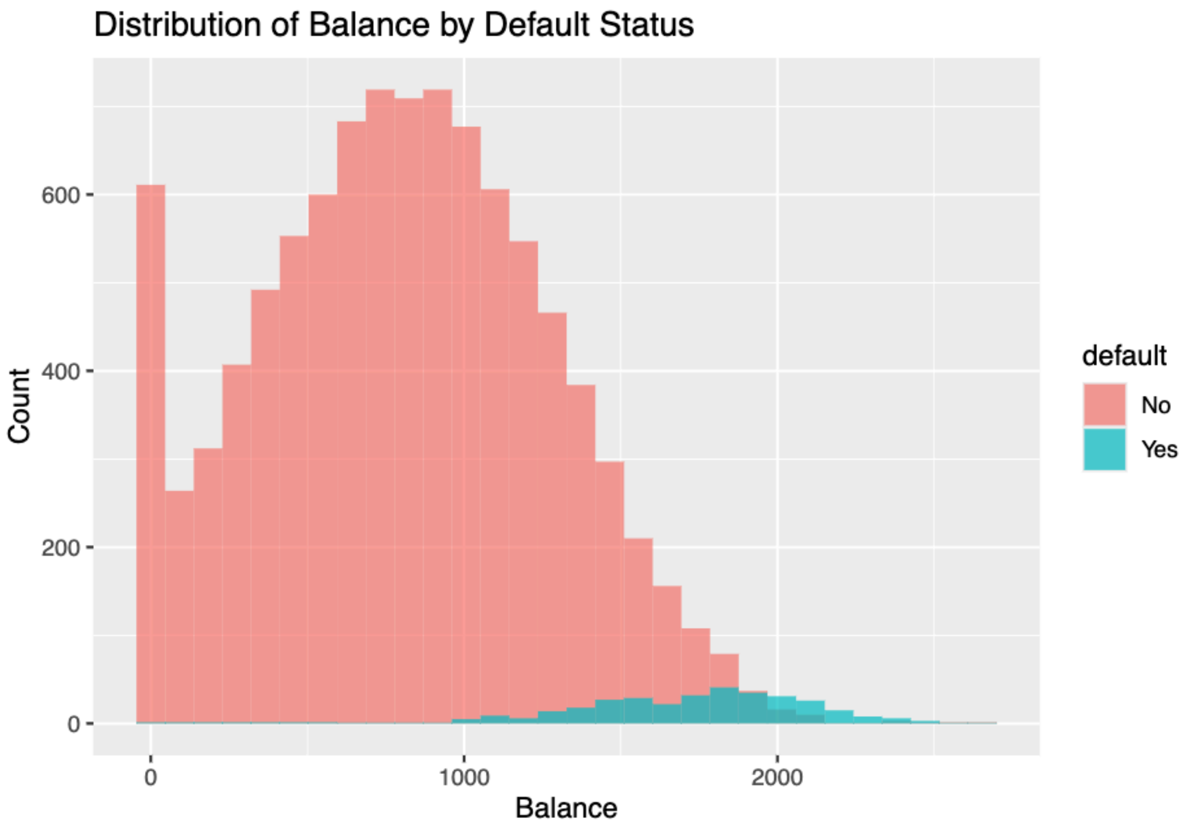
2.1 Visualizing the Data

2.1.1 Distribution of Balance

< What does this figure mean? >

The figure shows how balance amounts differ between people who default and those who do not. Higher balances are associated with more defaults.

```r
ggplot(data, aes(x = balance, fill=default)) +
  geom_histogram(bins = 30, alpha = 0.7, position = "identity") +
  labs(title = "Distribution of Balance by Default Status",
       x= "Balance",
       y = "Count")
```

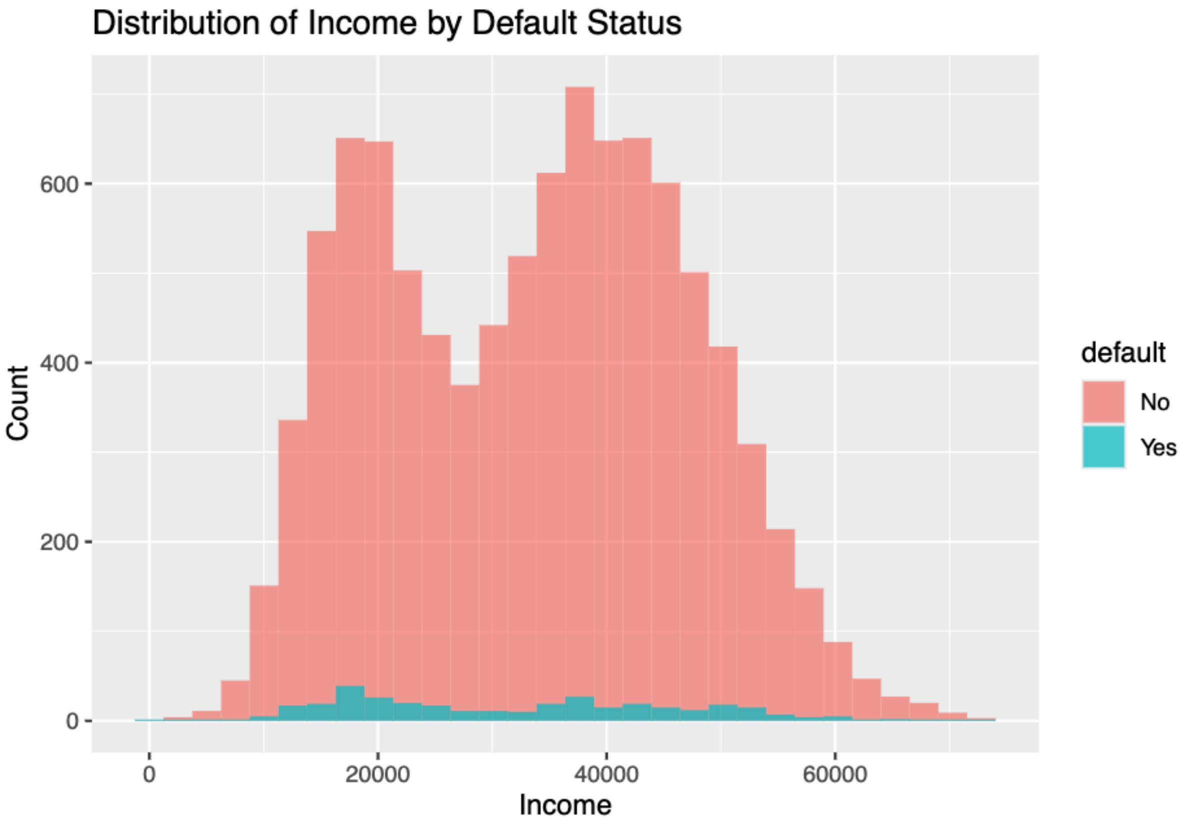


2.1.2 Distribution of Income

< What does this figure mean >

The figure shows income levels by default status. Income seems more evenly distributed across defaulters and non-defaulters.

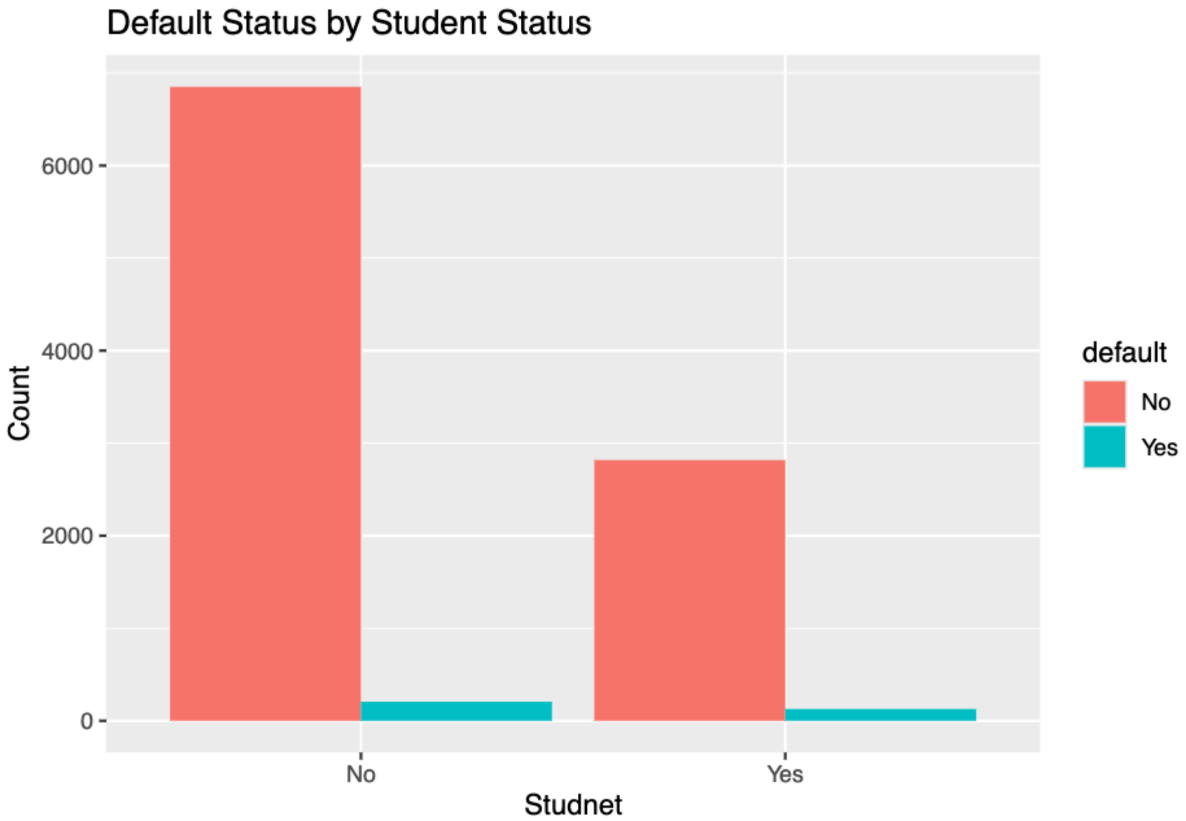
```r
ggplot(data, aes(x = income, fill = default)) +
  geom_histogram(bins = 30, alpha = 0.7, position = 'identity') +
  labs(title = "Distribution of Income by Default Status",
       x= "Income",
       y = "Count")
```



2.1.3 Student Status by Default

This plot shows the number of students vs non-students and their default rates. Students may have different default behaviors than non-students.

```
ggplot(data, aes(x = student, fill = default)) +
  geom_bar(position = 'dodge') +
  labs(title = "Default Status by Student Status",
       x = "Studnet",
       y = "Count")
```



Logistics Regression

4.1 Fitting the Model

< Describe Logistic Regression >

Logistic regression models the probability of default as a function of balance. It predicts a binary outcome using the logistic function.

```
logit_model = glm(default ~ balance, data = data, family = binomial)
summary(logit_model)
```

```
##
## Call:
## glm(formula = default ~ balance, family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2697  -0.1465  -0.0589  -0.0221   3.7589
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -1.065e+01  3.612e-01  -29.49   <2e-16 ***
## balance      5.499e-03  2.204e-04   24.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1596.5  on 9998  degrees of freedom
## AIC: 1600.5
##
## Number of Fisher Scoring iterations: 8
```

```
data$predicted_prob = predict(logit_model, type = "response")
head (data)
```

```
##   default student   balance    income predicted_prob
## 1      No      No  729.5265 44361.625   0.0013056797
## 2      No     Yes  817.1804 12106.135   0.0021125949
## 3      No      No 1073.5492 31767.139   0.0085947405
## 4      No      No  529.2506 35704.494   0.0004344368
## 5      No      No  785.6559 38463.496   0.0017769574
## 6      No     Yes  919.5885  7491.559   0.0037041528
```

4.2 Evaluate Model Performance

< Talk about our model and evaluation metrics >

We evaluate the model using a confusion matrix and look at how well it predicts defaults based on a 0.5 threshold.

```
threshold = 0.5
data$predicted_default = ifelse(data$predicted_prob > threshold, "Yes", "No")
conf_matrix = table(data$predicted_default, data$default)
```

5 Multiple Logistic Regression

Fitting the model

We will include an interaction term between income and student to differ between student and non-student

```
logit_mult_model = glm(default ~ balance + income * student, data=data, family=binomial)
summary(logit_mult_model)
```

```
##
## Call:
```

```
## glm(formula = default ~ balance + income * student, family = binomial,
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4638  -0.1420  -0.0558  -0.0203   3.7406
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.081e+01  5.031e-01 -21.495   <2e-16 ***
## balance             5.737e-03  2.319e-04  24.736   <2e-16 ***
## income              1.644e-06  8.633e-06   0.190    0.849
## studentYes         -9.343e-01  6.067e-01  -1.540    0.124
## income:studentYes   1.429e-05  2.772e-05   0.516    0.606
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1571.3  on 9995  degrees of freedom
## AIC: 1581.3
##
## Number of Fisher Scoring iterations: 8
```

5.2 Evaluating the Model

< Talk about evaluation metrics / interpretation >

We again use a confusion matrix and accuracy to measure performance. A better model will have higher accuracy.

```
data$mult_predicted_prob = predict(logit_mult_model, type = "response")
data$mult_predicted_default = ifelse(data$mult_predicted_prob > threshold, "Yes", "No")
conf_matrix_mult = table(data$mult_predicted_default, data$default)
conf_matrix_mult
```

```
##
##        No  Yes
##   No  9628  227
##   Yes   39  106
```

```
accuracy_mult = sum(diag(conf_matrix_mult)) / sum(conf_matrix_mult)
accuracy_mult = sum(diag(conf_matrix_mult)) / sum(conf_matrix_mult)
accuracy_mult
```

```
## [1] 0.9734
```

6. Multinomial Logistic Regression

6.1 Load the Data

```
data2 = Carseats
data2$SalesCategory = cut(data2$Sales, breaks = 3, lables = c("Low", "Medium", "High"))
```

```
multi_model = multinom(SalesCategory ~ Price + Income + Advertising, data=data2)
```

```
## # weights:  15 (8 variable)
## initial  value 439.444915
## iter  10 value 320.494096
## final  value 320.396998
## converged
```

```
summary(multi_model)
```

```
## Call:
## multinom(formula = SalesCategory ~ Price + Income + Advertising,
##     data = data2)
##
## Coefficients:
##             (Intercept)       Price      Income Advertising
## (5.42,10.8]    3.645096 -0.02909122 0.004288951   0.0834958
## (10.8,16.3]    4.839031 -0.06414212 0.011164722   0.1417921
##
## Std. Errors:
##             (Intercept)       Price      Income Advertising
## (5.42,10.8]   0.7937487 0.006004112 0.004488935  0.02137942
## (10.8,16.3]   1.1240727 0.009264365 0.006892574  0.03025054
##
## Residual Deviance: 640.794
## AIC: 656.794
```

6.2 Make Predictions

```
data2$nomial_predicted_salesCat = predict(multi_model)
head(data2)
```

```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age Education
## 1  9.50       138     73          11        276   120       Bad  42        17
## 2 11.22       111     48          16        260    83      Good  65        10
## 3 10.06       113     35          10        269    80    Medium  59        12
## 4  7.40       117    100           4        466    97    Medium  55        14
## 5  4.15       141     64           3        340   128       Bad  38        13
## 6 10.81       124    113          13        501    72       Bad  78        16
##   Urban  US SalesCategory nomial_predicted_salesCat
## 1   Yes Yes   (5.42,10.8]               (5.42,10.8]
```

```
## 2   Yes Yes    (10.8,16.3]               (5.42,10.8]
## 3   Yes Yes    (5.42,10.8]               (5.42,10.8]
## 4   Yes Yes    (5.42,10.8]               (5.42,10.8]
## 5   Yes  No (-0.0163,5.42]               (5.42,10.8]
## 6    No Yes    (5.42,10.8]               (10.8,16.3]
```

6.3 Evalute Model

```r
conf_matrix_mult = table(data2$nomial_predicted_salesCat, data2$SalesCategory)
conf_matrix_mult
```

```
##
##                  (-0.0163,5.42] (5.42,10.8] (10.8,16.3]
##   (-0.0163,5.42]             25          17           0
##   (5.42,10.8]                77         224          48
##   (10.8,16.3]                 0           6           3
```

```r
accuracy_mult = sum(diag(conf_matrix_mult)) / sum(conf_matrix_mult)
accuracy_mult
```

```
## [1] 0.63
```

Assignment Section

Background

Diabetes is a chronic disesse affecting millions of individuals worldwide. Early detection through predictive modeling can help guide prevention and treatment. In this assignment, you will use logistic regression to predict whether an individual has diabetes using basic health information.

We will use the Pima indians Diabetes Dataset, a commonly used dataset in health Informatics available from the UCI Machine Learning Repository and built into the mlbench R package.

Simple Logistic Regression

```r
####install.packages("mlbench", dependencies = TRUE)

library(mlbench)
data("PimaIndiansDiabetes")
df = PimaIndiansDiabetes
```

Data Exploration and Summary Figures

```r
glimpse(df)
```

```
## Rows: 768
## Columns: 9
## $ pregnant <dbl> 6, 1, 8, 1, 0, 5, 3, 10, 2, 8, 4, 10, 10, 1, 5, 7, 0, 7, 1, 1~
## $ glucose  <dbl> 148, 85, 183, 89, 137, 116, 78, 115, 197, 125, 110, 168, 139,~
## $ pressure <dbl> 72, 66, 64, 66, 40, 74, 50, 0, 70, 96, 92, 74, 80, 60, 72, 0,~
## $ triceps  <dbl> 35, 29, 0, 23, 35, 0, 32, 0, 45, 0, 0, 0, 0, 23, 19, 0, 47, 0~
## $ insulin  <dbl> 0, 0, 0, 94, 168, 0, 88, 0, 543, 0, 0, 0, 0, 846, 175, 0, 230~
## $ mass     <dbl> 33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31.0, 35.3, 30.5, 0.0, 37~
## $ pedigree <dbl> 0.627, 0.351, 0.672, 0.167, 2.288, 0.201, 0.248, 0.134, 0.158~
## $ age      <dbl> 50, 31, 32, 21, 33, 30, 26, 29, 53, 54, 30, 34, 57, 59, 51, 3~
## $ diabetes <fct> pos, neg, pos, neg, pos, neg, pos, neg, pos, pos, neg, pos, n~
```

```r
summary(df)
```

```
##     pregnant         glucose         pressure         triceps     
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00  
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00  
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00  
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54  
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00  
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00  
##     insulin          mass          pedigree           age        diabetes
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00   neg:500 
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437   1st Qu.:24.00   pos:268 
##  Median : 30.5   Median :32.00   Median :0.3725   Median :29.00           
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719   Mean   :33.24           
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00           
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00           
```

Fit Simple Logistic Regression Model (Train & Test Split)

```r
set.seed(123)
train_idx <- sample(seq_len(nrow(df)), size = 0.75 * nrow(df))
train_data <- df[train_idx, ]
test_data  <- df[-train_idx, ]

# Simple logistic regression on glucose
simple_logit_model <- glm(diabetes ~ glucose,
                          data = train_data,
                          family = binomial)
summary(simple_logit_model)
```

```
## 
## Call:
## glm(formula = diabetes ~ glucose, family = binomial, data = train_data)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1671  -0.7854  -0.5169   0.8200   3.3759
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.694847   0.509353  -11.18   <2e-16 ***
## glucose      0.040318   0.003912   10.31   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 745.11  on 575  degrees of freedom
## Residual deviance: 595.64  on 574  degrees of freedom
## AIC: 599.64
##
## Number of Fisher Scoring iterations: 4
```

Interpret Coefficients & Apply the Model for Prediction on Test Data

Higher glucose levels increase the probability of having diabetes.

Intercept: -5.695 (baseline log-odds when glucose $= 0$)

Glucose: $+0.0403$ per unit increase ($p < 2\text{e-}16$), meaning each additional point of blood glucose raises the log-odds of diabetes by 0.0403.

```r
test_data$predicted_prob <- predict(simple_logit_model,
                                    newdata = test_data,
                                    type = "response")
test_data$predicted_diabetes <- ifelse(test_data$predicted_prob > 0.5,
                                       "pos", "neg")

conf_matrix_simple <- table(test_data$predicted_diabetes,
                            test_data$diabetes)
conf_matrix_simple
```

```
##
##       neg pos
##   neg 107  39
##   pos  18  28
```

```r
accuracy_simple <- sum(diag(conf_matrix_simple)) / sum(conf_matrix_simple)
accuracy_simple
```

```
## [1] 0.703125
```

Accuracy: 70.31% of cases correctly classified.

Multiple Logistic Regression

We now include glucose, age, BMI, and pregnancies to predict diabetes more accurately.

```
multi_logit_model <- glm(diabetes ~ glucose + age + mass + pregnant,
                         data = train_data,
                         family = binomial)
summary(multi_logit_model)
```

```
##
## Call:
## glm(formula = diabetes ~ glucose + age + mass + pregnant, family = binomial,
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1083  -0.7164  -0.4321   0.7433   2.8929
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.492123   0.771788 -11.003  < 2e-16 ***
## glucose      0.034569   0.004022   8.594  < 2e-16 ***
## age          0.013773   0.010590   1.300  0.19344
## mass         0.080047   0.016001   5.003 5.66e-07 ***
## pregnant     0.106275   0.036467   2.914  0.00357 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 745.11  on 575  degrees of freedom
## Residual deviance: 550.39  on 571  degrees of freedom
## AIC: 560.39
##
## Number of Fisher Scoring iterations: 5
```

Interpretation of Coefficients

Glucose: +0.0346 ($p < 2\text{e-}16$) — higher glucose increases risk.

Age: +0.0138 ($p = 0.193$) — not statistically significant.

Mass (BMI): +0.0800 ($p < 1\text{e-}6$) — higher BMI increases risk.

Pregnant: +0.1063 ($p = 0.0036$) — each additional pregnancy raises risk.

< Fit a Multiple Logistic Regression Model (Train & Test Split)

< Fit a logistic regression using the glucose, age, BMI, and pregnant as predictors of diabetes>

```r
test_data$predicted_prob_multi <- predict(multi_logit_model, newdata = test_data, type = "response")
test_data$predicted_diabetes_multi <- ifelse(test_data$predicted_prob_multi > 0.5, "pos", "neg")

conf_matrix_multi <- table(test_data$predicted_diabetes_multi, test_data$diabetes)
conf_matrix_multi
```

```
##
##        neg pos
##    neg 106  32
##    pos  19  35
```

```r
accuracy_multi <- sum(diag(conf_matrix_multi)) / sum(conf_matrix_multi)
accuracy_multi
```

```
## [1] 0.734375
```

Interpret Coefficients & Apply the Model for Prediction on Test Data

Glucose: Positive coefficient—higher glucose increases diabetes risk.

Age: Positive coefficient—older age increases risk.

Mass: Positive coefficient—higher BMI increases risk.

Pregnant: Positive coefficient—each additional pregnancy slightly raises the log-odds of diabetes.

```r
test_data$predicted_prob_multi <- predict(multi_logit_model,
                                          newdata = test_data,
                                          type = "response")
test_data$predicted_diabetes_multi <- ifelse(test_data$predicted_prob_multi > 0.5,
                                             "pos", "neg")

conf_matrix_multi <- table(test_data$predicted_diabetes_multi,
                           test_data$diabetes)
conf_matrix_multi
```

```
##
##       neg pos
##   neg 106  32
##   pos  19  35
```

```
accuracy_multi <- sum(diag(conf_matrix_multi)) / sum(conf_matrix_multi)
accuracy_multi
```

```
## [1] 0.734375
```

Accuracy: 73.44%, improved over simple model.

Sensitivity: 35 / (35 + 32) = 52.24% (better at identifying diabetics).

Specificity: 106 / (106 + 19) = 84.84% (non-diabetics still well identified).

K-Nearest Neighbors Classification

KNN is a non-parametric method that classifies based on the majority vote of nearest neighbors.

K-Nearest Neighbors (KNN) is a simple, flexable algorithm that makes predictions based on the majority class of the closest data points.

Use the caret and class libraries with the knn() function. See our in-class lab for a worked example.

Prepare the Data

Fit a KNN Classifier Model (Train & test Split)

Interpret & Apply to Test Data

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(class)

# Normalize numeric predictors
normalize <- function(x) (x - min(x)) / (max(x) - min(x))
df_norm <- df %>%
```

```r
  mutate(across(c(glucose, age, mass, pregnant), normalize))

train_norm <- df_norm[train_idx, ]
test_norm  <- df_norm[-train_idx, ]

train_labels <- train_norm$diabetes
test_labels  <- test_norm$diabetes

# Fit KNN model with k = 5
knn_pred <- knn(train     = train_norm[, c("glucose","age","mass","pregnant")],
                test      = test_norm[,  c("glucose","age","mass","pregnant")],
                cl        = train_labels,
                k         = 5)

conf_matrix_knn <- table(knn_pred, test_labels)
conf_matrix_knn
```

```
##          test_labels
## knn_pred neg pos
##      neg 105  30
##      pos  20  37
```

```r
accuracy_knn <- sum(diag(conf_matrix_knn)) / sum(conf_matrix_knn)
accuracy_knn
```

```
## [1] 0.7395833
```

Model Comparison and Discussion

Simple logistic regression captures the main trend but may miss complex patterns.

Multiple logistic regression adds more predictors, improving accuracy.

KNN is flexible but may be sensitive to the choice of k and noise in the data.

Accuracy: 73.96% (slightly higher than multiple logistic).