HW#3:LOGISTIC_REGRESSION

Lewris Mota Sanchez, Warner Alexis, Saloua Daouki, Souleymane Doumbia, Fomba Kassoh

2024-10-31

Contents

Our objective is to build a binary logistic regression model on the training data set to predict whether the neighborhood will be at risk for high crime levels. We will provide classifications and probabilities for the evaluation data set using our binary logistic regression model. We can only use the variables given to us (or, variables that we derive from the variables provided). Below is a short description of the variables of interest in the data set:

- zn: proportion of residential land zoned for large lots (over 25000 square feet) (predictor variable)
- indus: proportion of non-retail business acres per suburb (predictor variable)
- chas: a dummy var. for whether the suburb borders the Charles River (1) or not (0) (predictor variable)
- nox: nitrogen oxides concentration (parts per 10 million) (predictor variable)
- rm: average number of rooms per dwelling (predictor variable)
- age: proportion of owner-occupied units built prior to 1940 (predictor variable)
- dis: weighted mean of distances to five Boston employment centers (predictor variable)
- rad: index of accessibility to radial highways (predictor variable)
- tax: full-value property-tax rate per $10,000 (predictor variable)
- ptratio: pupil-teacher ratio by town (predictor variable)
- lstat: lower status of the population (percent) (predictor variable)
- medv: median value of owner-occupied homes in $1000s (predictor variable)
- target: whether the crime rate is above the median crime rate (1) or not (0) (response variable)

1. DATA EXPLORATION

a. Load the Data and Check Size/Variables

```
# Load necessary libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# Load the datasets
train_data <- read.csv("crime-training-data_modified.csv")  # Training data
eval_data <- read.csv("crime-evaluation-data_modified.csv")  # Evaluation data

# Check the dimensions and structure of the datasets
dim(train_data)
```

```
## [1] 466  13
```

```
str(train_data)
```

```
## 'data.frame':    466 obs. of  13 variables:
##  $ zn     : num  0 0 0 30 0 0 0 0 0 80 ...
##  $ indus  : num  19.58 19.58 18.1 4.93 2.46 ...
##  $ chas   : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.605 0.871 0.74 0.428 0.488 0.52 0.693 0.693 0.515 0.392 ...
##  $ rm     : num  7.93 5.4 6.49 6.39 7.16 ...
##  $ age    : num  96.2 100 100 7.8 92.2 71.3 100 100 38.1 19.1 ...
##  $ dis    : num  2.05 1.32 1.98 7.04 2.7 ...
##  $ rad    : int  5 5 24 6 3 5 24 24 5 1 ...
##  $ tax    : int  403 403 666 300 193 384 666 666 224 315 ...
##  $ ptratio: num  14.7 14.7 20.2 16.6 17.8 20.9 20.2 20.2 20.2 16.4 ...
##  $ lstat  : num  3.7 26.82 18.85 5.19 4.82 ...
##  $ medv   : num  50 13.4 15.4 23.7 37.9 26.5 5 7 22.2 20.9 ...
##  $ target : int  1 1 1 0 0 0 1 1 0 0 ...
```

b. Calculate Mean, Standard Deviation, and Median

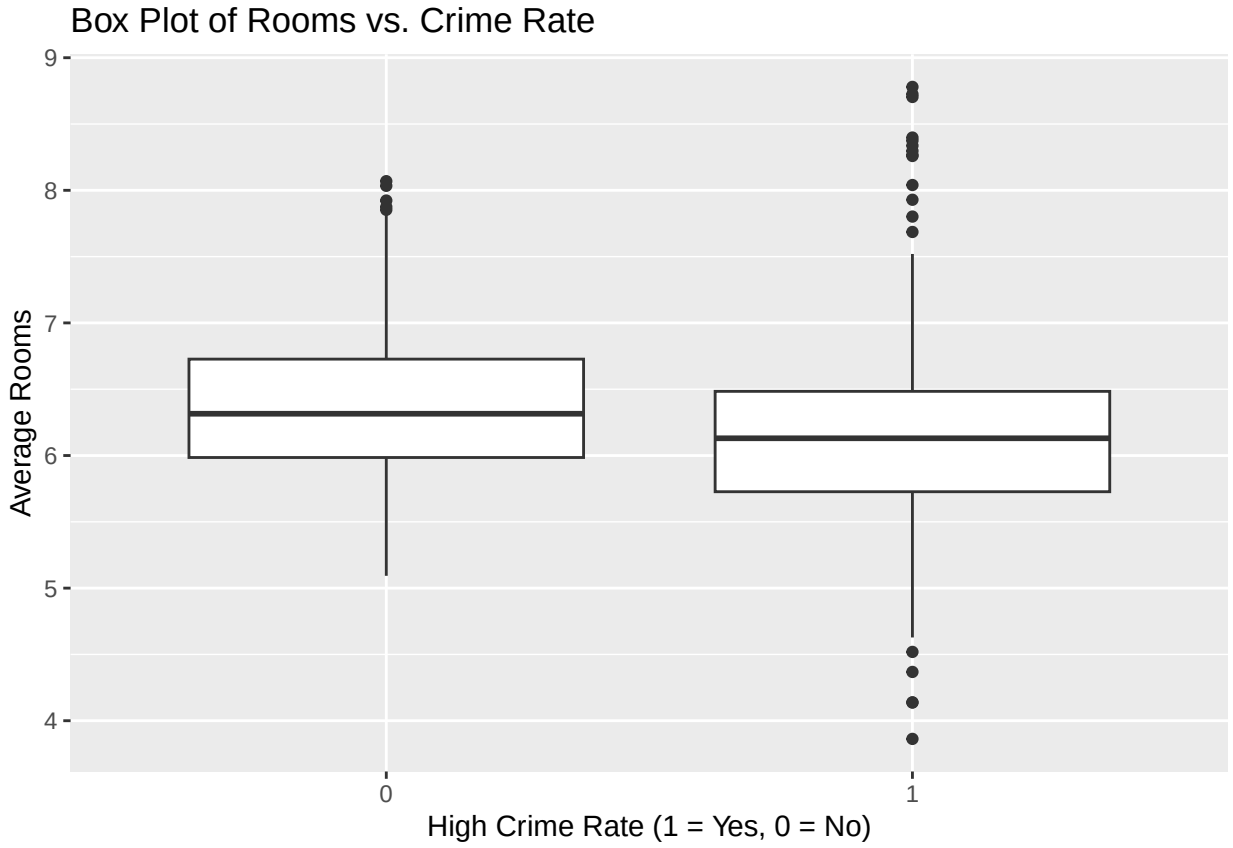
```
# Summary statistics for training data
summary_stats <- train_data %>% summarise(across(everything(), list(mean = ~mean(.x, na.rm = TRUE),
                                                                     sd = ~sd(.x, na.rm = TRUE),
                                                                     median = ~median(.x, na.rm = TRUE)
print(summary_stats)
```

```
##    zn_mean    zn_sd zn_median indus_mean indus_sd indus_median  chas_mean
## 1 11.57725 23.36465         0   11.10502 6.845855         9.69 0.07081545
##    chas_sd chas_median  nox_mean    nox_sd nox_median  rm_mean     rm_sd
## 1 0.256792           0 0.5543105 0.1166667      0.538 6.290674 0.7048513
```

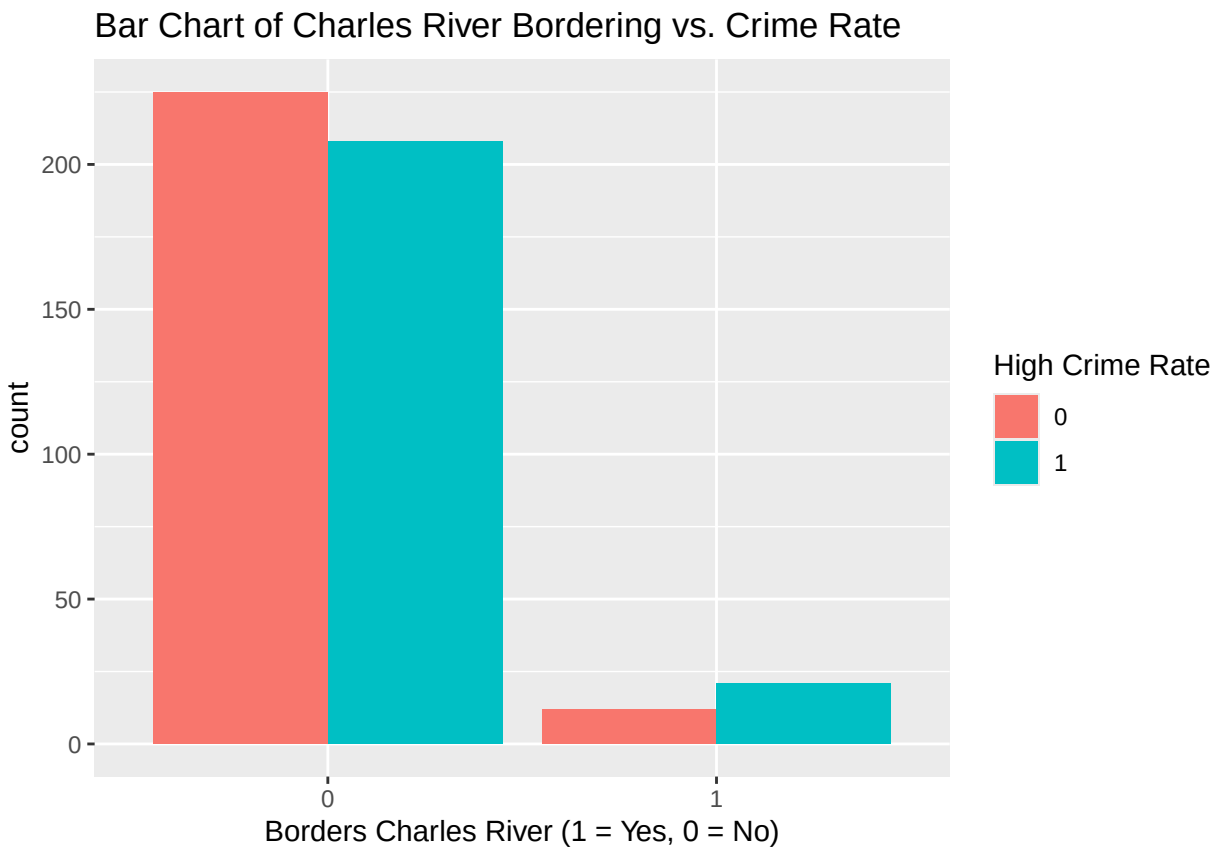
```
##   rm_median age_mean   age_sd age_median dis_mean  dis_sd dis_median rad_mean
## 1      6.21  68.3676 28.32138      77.15 3.795693 2.10695    3.19095 9.530043
##     rad_sd rad_median tax_mean   tax_sd tax_median ptratio_mean ptratio_sd
## 1 8.685927          5 409.5021 167.9001      334.5      18.3985   2.196845
##   ptratio_median lstat_mean lstat_sd lstat_median medv_mean  medv_sd
## 1           18.9   12.63146 7.101891        11.35  22.58927 9.239681
##   medv_median target_mean target_sd target_median
## 1        21.2   0.4914163 0.5004636             0
```

c. Bar Chart and Box Plot

```
# Box plot for continuous variables vs. target
ggplot(train_data, aes(x = factor(target), y = rm)) +
  geom_boxplot() +
  labs(title = "Box Plot of Rooms vs. Crime Rate", x = "High Crime Rate (1 = Yes, 0 = No)", y = "Averag
```




```
# Bar chart for categorical variable (e.g., chas)
ggplot(train_data, aes(x = factor(chas), fill = factor(target))) +
  geom_bar(position = "dodge") +
  labs(title = "Bar Chart of Charles River Bordering vs. Crime Rate", x = "Borders Charles River (1 = Y
```



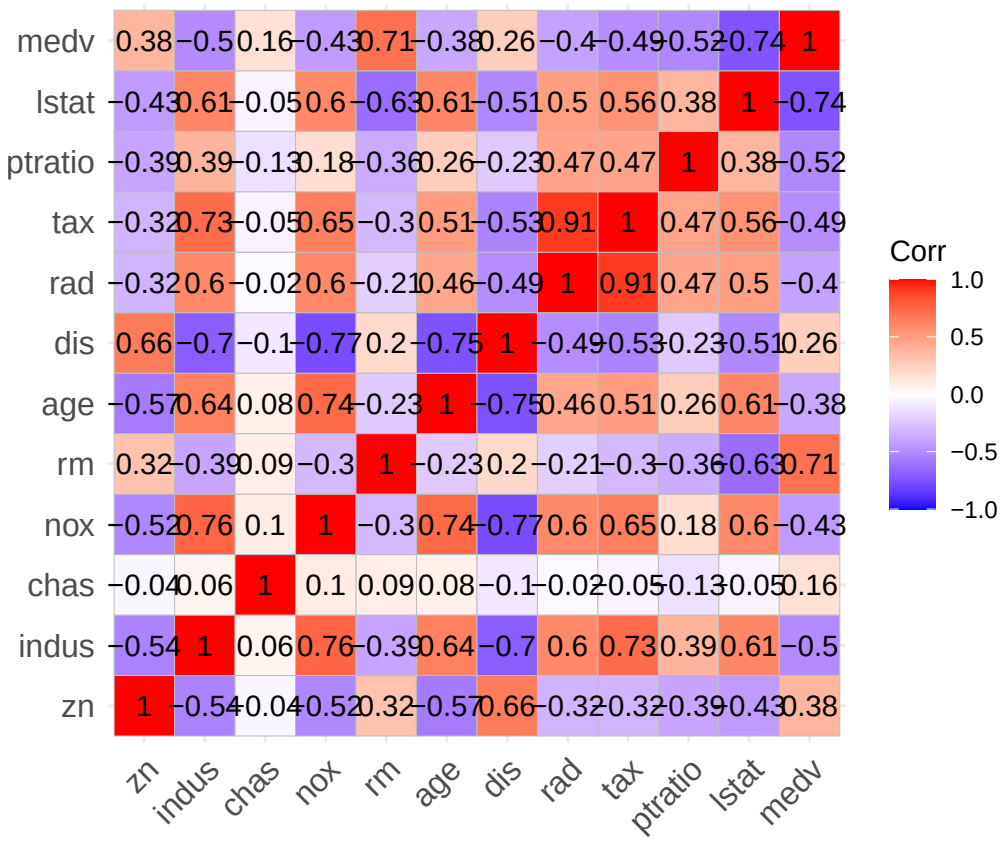
d. Correlation Matrix

```r
# Correlation matrix for training data
correlation_matrix <- cor(train_data %>% select(-target), use = "complete.obs")
print(correlation_matrix)
```

```
##                zn       indus        chas         nox          rm         age
## zn       1.00000000 -0.53826643 -0.04016203 -0.51704518  0.31981410 -0.57258054
## indus   -0.53826643  1.00000000  0.06118317  0.75963008 -0.39271181  0.63958182
## chas    -0.04016203  0.06118317  1.00000000  0.09745577  0.09050979  0.07888366
## nox     -0.51704518  0.75963008  0.09745577  1.00000000 -0.29548972  0.73512782
## rm       0.31981410 -0.39271181  0.09050979 -0.29548972  1.00000000 -0.23281251
## age     -0.57258054  0.63958182  0.07888366  0.73512782 -0.23281251  1.00000000
## dis      0.66012434 -0.70361886 -0.09657711 -0.76888404  0.19901584 -0.75089759
## rad     -0.31548119  0.60062839 -0.01590037  0.59582984 -0.20844570  0.46031430
## tax     -0.31928408  0.73222922 -0.04676476  0.65387804 -0.29693430  0.51212452
## ptratio -0.39103573  0.39468980 -0.12866058  0.17626871 -0.36034706  0.25544785
## lstat   -0.43299252  0.60711023 -0.05142322  0.59624264 -0.63202445  0.60562001
## medv     0.37671713 -0.49617432  0.16156528 -0.43012267  0.70533679 -0.37815605
##                dis         rad         tax    ptratio       lstat       medv
## zn       0.66012434 -0.31548119 -0.31928408 -0.3910357 -0.43299252  0.3767171
## indus   -0.70361886  0.60062839  0.73222922  0.3946898  0.60711023 -0.4961743
## chas    -0.09657711 -0.01590037 -0.04676476 -0.1286606 -0.05142322  0.1615653
## nox     -0.76888404  0.59582984  0.65387804  0.1762687  0.59624264 -0.4301227
## rm       0.19901584 -0.20844570 -0.29693430 -0.3603471 -0.63202445  0.7053368
## age     -0.75089759  0.46031430  0.51212452  0.2554479  0.60562001 -0.3781560
## dis      1.00000000 -0.49499193 -0.53425464 -0.2333394 -0.50752800  0.2566948
```

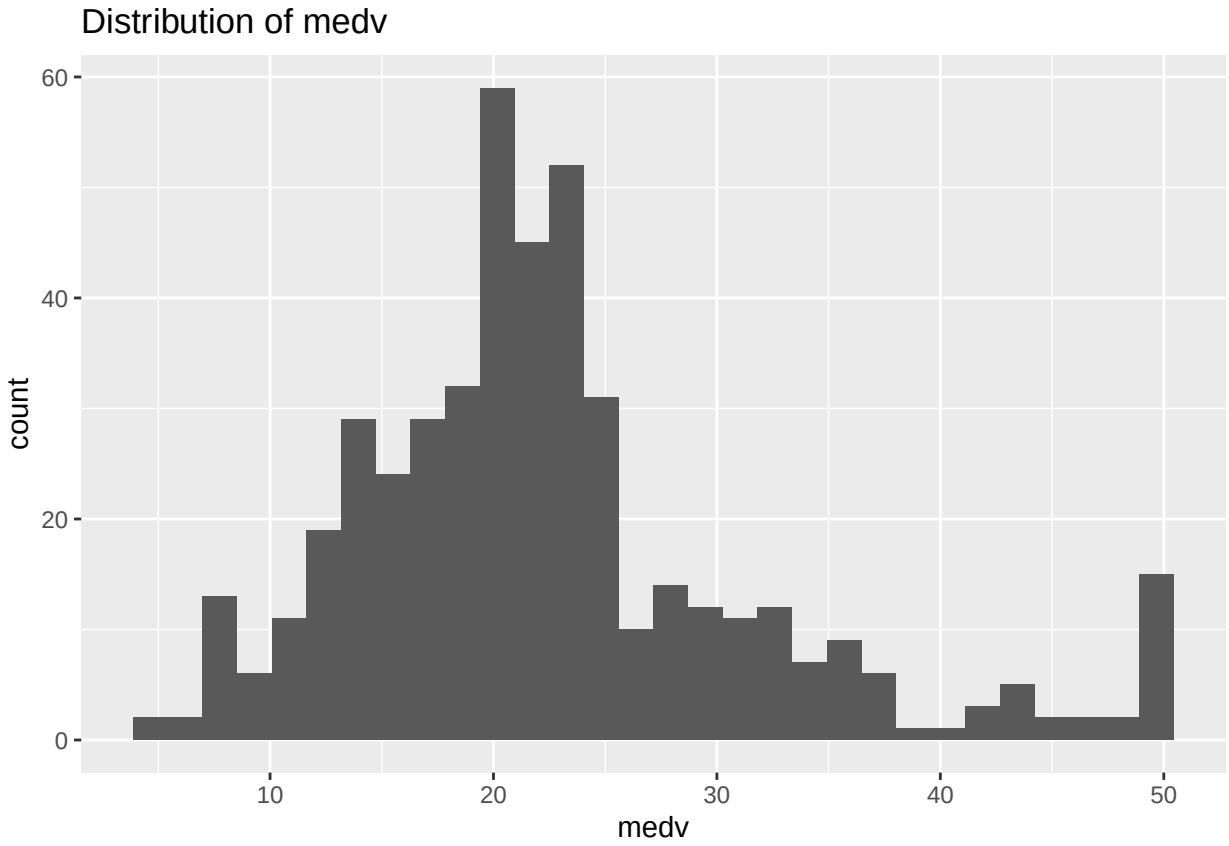
```
## rad      -0.49499193  1.00000000  0.90646323  0.4714516  0.50310125 -0.3976683
## tax      -0.53425464  0.90646323  1.00000000  0.4744223  0.56418864 -0.4900329
## ptratio -0.23333940  0.47145160  0.47442229  1.0000000  0.37735605 -0.5159153
## lstat    -0.50752800  0.50310125  0.56418864  0.3773560  1.00000000 -0.7358008
## medv      0.25669476 -0.39766826 -0.49003287 -0.5159153 -0.73580078  1.0000000
```

```
# Visualization of correlation matrix
library(ggcorrplot)
ggcorrplot(correlation_matrix, lab = TRUE)
```



e. Calculating skewness and checking nonlinearity

```
library(ggplot2)
ggplot(train_data, aes(x = medv)) + geom_histogram(bins = 30) + ggtitle("Distribution of medv")
```



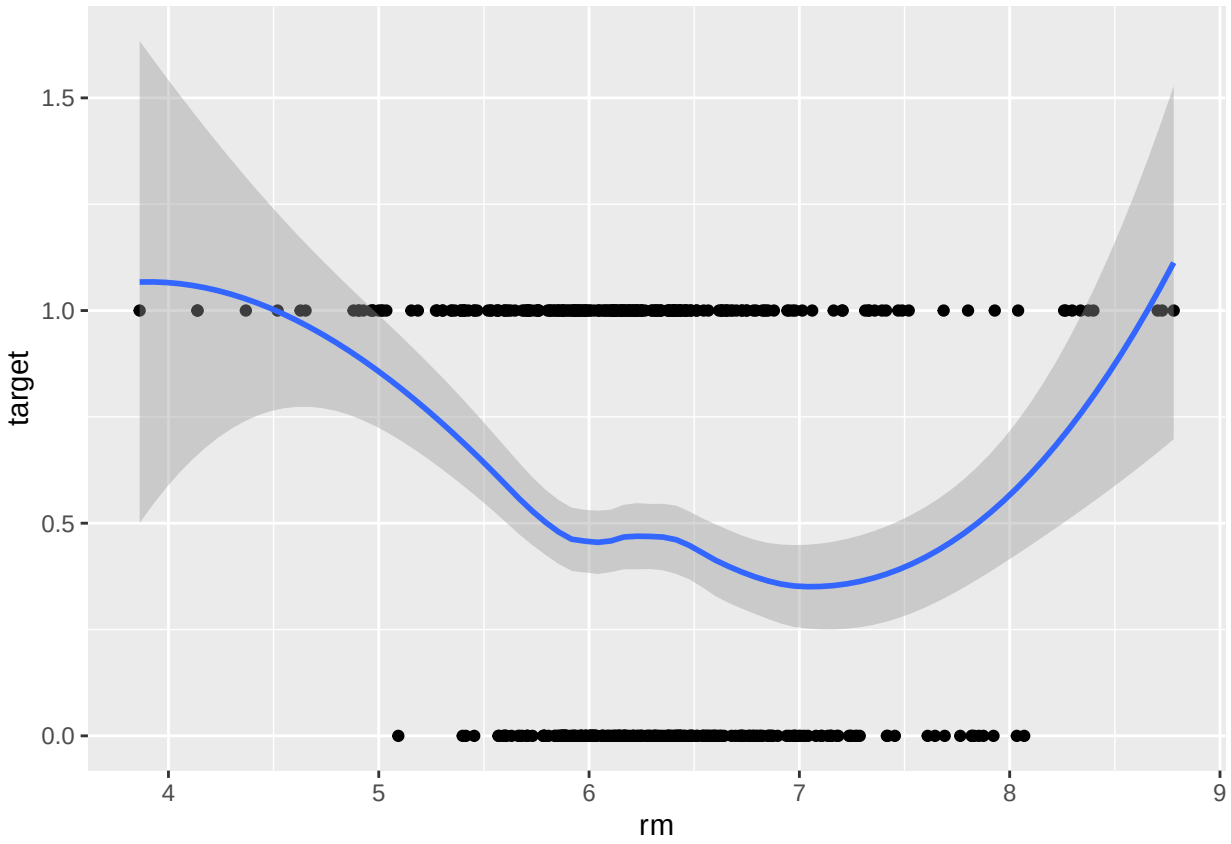

```
# Calculate skewness
library(moments)
skewness_value <- skewness(train_data$medv)
print(skewness_value)
```

```
## [1] 1.080167
```

The medv variable is positively skewed; we can see it from the histogram above where the right hand side tail is longer and also from the skewness value of 1.080167, this tells us that log transformation is suitable for the variable 'medv'.

```
#checking nonlinearity between rm and target
ggplot(train_data, aes(x = rm, y = target)) + geom_point() + geom_smooth(method = "loess")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Baed on the scatter plot above, the relationship between the rm and target is non-linear, we can use polynomial terms, perhaps square of rm.

f. Check for Missing Values

```
# Check for missing values in training data
missing_values <- sapply(train_data, function(x) sum(is.na(x)))
print(missing_values)
```

```
##      zn   indus    chas     nox      rm     age     dis     rad     tax ptratio
##       0       0       0       0       0       0       0       0       0       0
##   lstat    medv  target
##       0       0       0
```

2. DATA PREPARATION

Since there is no missing values in the provided data, we can skip handling them in this section.

a. Mathematical Transformations

```
# Log transformation of medv
train_data <- train_data %>%
  mutate(log_medv = log(medv + 1))  # Adding 1 to avoid log(0)

# Adding a quadratic term for rm
train_data <- train_data %>%
  mutate(rm_squared = rm^2)
```

3. BUILD MODELS

a. Build Logistic Regression Models

```
model1 <- glm(target ~ zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + lstat,
              data = train_data,
              family = "binomial")
summary(model1)
```

a.1. Model 1: Using All Predictors

```
##
## Call:
## glm(formula = target ~ zn + indus + chas + nox + rm + age + dis +
##     rad + tax + ptratio + lstat, family = "binomial", data = train_data)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -37.867486   6.239330  -6.069 1.29e-09 ***
## zn           -0.062365   0.032297  -1.931   0.0535 .
## indus        -0.055502   0.046250  -1.200   0.2301
## chas          0.990769   0.771749   1.284   0.1992
## nox          44.022991   7.257684   6.066 1.31e-09 ***
## rm            1.001582   0.440092   2.276   0.0229 *
## age           0.016530   0.011398   1.450   0.1470
## dis           0.492199   0.197144   2.497   0.0125 *
## rad           0.635987   0.154415   4.119 3.81e-05 ***
## tax          -0.007331   0.002857  -2.566   0.0103 *
## ptratio       0.220199   0.099458   2.214   0.0268 *
## lstat         0.046643   0.051801   0.900   0.3679
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 645.88  on 465  degrees of freedom
## Residual deviance: 199.95  on 454  degrees of freedom
## AIC: 223.95
##
## Number of Fisher Scoring iterations: 9
```

Here are the key coefficient of model 1

- nox (Nitrogen oxides concentration): 44.02299 indicates that for each unit increase in nox, the log-odds of high crime increase significantly, which is very strong evidence ($p < 0.001$); highly significant.
- rm (Average number of rooms): 1.001582 indicates that an increase of one room is associated with a significant increase in the likelihood of high crime ($p < 0.05$), statistically significant.
- dis (Weighted mean distance to employment centers): 0.4922 indicates that as distance increases, the likelihood of high crime increases ($p < 0.05$), statistically significant.
- rad (Accessibility to radial highways): 0.6359 shows that increased access is positively associated with high crime levels ($p < 0.001$); highly significant.
- tax (Property-tax rate): -0.007331 suggests that higher property tax rates are associated with lower likelihood of high crime ($p < 0.05$), statistically significant.

- ptratio (Pupil-teacher ratio): 0.2202 indicates a positive association with high crime rates ($p < 0.05$)statistically significant.
- lstat (Lower status of the population): The coefficient is not statistically significant ($p = 0.3679$), suggesting it may not be a useful predictor in this model because it is not significant.

```
model2 <- glm(target ~ log_medv + rm + rm_squared + lstat, data = train_data, family = "binomial")
summary(model2)
```

a.2. Model 2: With Transformations and Selected Predictors

```
##
## Call:
## glm(formula = target ~ log_medv + rm + rm_squared + lstat, family = "binomial",
##     data = train_data)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 28.34213    9.41648   3.010 0.002614 **
## log_medv    -1.20293    0.65507  -1.836 0.066306 .
## rm          -9.29450    2.62474  -3.541 0.000398 ***
## rm_squared   0.78630    0.20027   3.926 8.63e-05 ***
## lstat        0.19639    0.03523   5.575 2.47e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 645.88  on 465  degrees of freedom
## Residual deviance: 483.96  on 461  degrees of freedom
## AIC: 493.96
##
## Number of Fisher Scoring iterations: 5
```

Here are the interpretations of each of the predictors used in model 2:

- log_medv: -1.20293 indicates that for every one-unit increase in the log of the median value of owner-occupied homes, the log-odds of high crime decreases. The p-value is 0.066306, which is marginally significant ($p < 0.1$), suggesting this predictor might still have some importance but is not conventionally significant at the 0.05 level.
- rm: -9.29450 indicates that for each additional room in a dwelling, the log-odds of high crime decreases significantly ($p < 0.001$). This suggests a strong negative relationship between the average number of rooms and high crime levels.
- rm_squared: 0.78630 indicates a positive association with the log-odds of high crime, suggesting that as the number of rooms increases, the effect of rooms on high crime becomes more positive at higher levels of rm ($p < 0.001$).
- lstat: 0.19639 indicates that for each one-unit increase in the lower status of the population, the log-odds of high crime increase significantly ($p < 0.001$).

```
library(MASS)
```

a.3. Model 3: Stepwise Regression for Variable Selection

```
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
model3 <- stepAIC(glm(target ~ ., data = train_data, family = "binomial"), direction = "both")

## Start:  AIC=210.28
## target ~ zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + lstat + medv + log_medv + rm_squared
##
##              Df Deviance    AIC
## - medv        1   180.34 208.34
## - lstat       1   180.48 208.48
## - log_medv    1   180.94 208.94
## - chas        1   181.71 209.71
## - indus       1   181.96 209.96
## <none>            180.28 210.28
## - zn          1   185.19 213.19
## - tax         1   186.58 214.58
## - ptratio     1   189.12 217.12
## - age         1   190.28 218.28
## - rm_squared  1   190.78 218.78
## - rm          1   191.42 219.42
## - dis         1   191.54 219.54
## - rad         1   224.82 252.82
## - nox         1   248.40 276.40
##
## Step:  AIC=208.34
## target ~ zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + lstat + log_medv + rm_squared
##
##              Df Deviance    AIC
## - lstat       1   180.57 206.57
## - chas        1   181.84 207.84
## - indus       1   182.01 208.01
## <none>            180.34 208.34
## - log_medv    1   182.35 208.35
## + medv        1   180.28 210.28
## - zn          1   185.22 211.22
## - tax         1   186.83 212.83
## - ptratio     1   189.62 215.62
## - age         1   190.42 216.42
## - dis         1   191.65 217.65
## - rm          1   195.70 221.70
## - rm_squared  1   196.17 222.17
## - rad         1   225.33 251.33
## - nox         1   248.60 274.60
##
## Step:  AIC=206.57
## target ~ zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + log_medv + rm_squared
##
```

```
##              Df Deviance    AIC
## - chas        1   181.94 205.94
## - indus       1   182.35 206.35
## <none>            180.57 206.57
## - log_medv    1   183.12 207.12
## + lstat       1   180.34 208.34
## + medv        1   180.48 208.48
## - zn          1   185.70 209.70
## - tax         1   187.00 211.00
## - ptratio     1   189.81 213.81
## - age         1   191.31 215.31
## - dis         1   191.81 215.81
## - rm          1   197.60 221.60
## - rm_squared  1   197.78 221.78
## - rad         1   225.37 249.37
## - nox         1   248.92 272.92
##
## Step:  AIC=205.94
## target ~ zn + indus + nox + rm + age + dis + rad + tax + ptratio +
##     log_medv + rm_squared
##
##              Df Deviance    AIC
## - indus       1   183.14 205.14
## <none>            181.94 205.94
## - log_medv    1   184.55 206.55
## + chas        1   180.57 206.57
## + medv        1   181.78 207.78
## + lstat       1   181.84 207.84
## - zn          1   188.22 210.22
## - tax         1   190.29 212.29
## - ptratio     1   190.43 212.43
## - dis         1   192.72 214.72
## - age         1   193.90 215.90
## - rm          1   199.50 221.50
## - rm_squared  1   199.57 221.57
## - rad         1   234.16 256.15
## - nox         1   249.05 271.05
##
## Step:  AIC=205.14
## target ~ zn + nox + rm + age + dis + rad + tax + ptratio + log_medv +
##     rm_squared
##
##              Df Deviance    AIC
## <none>            183.14 205.14
## - log_medv    1   185.42 205.42
## + indus       1   181.94 205.94
## + chas        1   182.35 206.35
## + lstat       1   182.95 206.95
## + medv        1   183.02 207.02
## - zn          1   190.00 210.00
## - ptratio     1   191.25 211.25
## - dis         1   193.56 213.56
## - age         1   194.36 214.36
## - tax         1   196.59 216.59
```

```
## - rm          1   200.65 220.65
## - rm_squared  1   200.90 220.90
## - rad         1   242.37 262.37
## - nox         1   255.78 275.79
```

```
summary(model3)
```

```
##
## Call:
## glm(formula = target ~ zn + nox + rm + age + dis + rad + tax +
##     ptratio + log_medv + rm_squared, family = "binomial", data = train_data)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  23.788430  17.639429   1.349 0.177467
## zn           -0.085003   0.037037  -2.295 0.021730 *
## nox          45.765944   7.231196   6.329 2.47e-10 ***
## rm          -21.359235   5.674501  -3.764 0.000167 ***
## age           0.041338   0.012999   3.180 0.001473 **
## dis           0.717434   0.238258   3.011 0.002602 **
## rad           0.845234   0.172736   4.893 9.92e-07 ***
## tax          -0.009701   0.002882  -3.367 0.000761 ***
## ptratio       0.351166   0.128985   2.723 0.006478 **
## log_medv      2.468045   1.680733   1.468 0.141986
## rm_squared    1.629235   0.434383   3.751 0.000176 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 645.88  on 465  degrees of freedom
## Residual deviance: 183.14  on 455  degrees of freedom
## AIC: 205.14
##
## Number of Fisher Scoring iterations: 9
```

Model 3 reveals that the following predictors are statistically significant with $p < 0.05$;

- zn (*)
- nox (***)
- rm (***)
- age (**)
- dis (**)
- rad (***)
- tax (***)
- ptratio (**)
- rm_squared (***)

Now that we have three models, we can use their AIC values to compare them and conclude which model is better fit: the AIC value of 205.14 for model 3 shows a better fit than both Model 1 (AIC = 223.95) and Model 2 (AIC = 493.96). This suggests that Model 3 is a more efficient model in terms of balancing goodness of fit and complexity.

4. SELECT MODELS

a. Predictions on Evaluation Data

Now that we decided which model is the best, we can predict on evaluation data :

```
# Calculate log_medv and rm_squared in evaluation data
eval_data$log_medv <- log(eval_data$medv)                # Create log_medv
eval_data$rm_squared <- eval_data$rm^2                    # Create rm_squared

# Predictions on evaluation data using the best model (model3 in this case)
eval_data$pred_prob <- predict(model3, newdata = eval_data, type = "response")
eval_data$pred_class <- ifelse(eval_data$pred_prob > 0.5, 1, 0)

# Display first few rows to confirm
head(eval_data[, c("log_medv", "rm_squared", "pred_prob", "pred_class")])
```

```
##   log_medv rm_squared  pred_prob pred_class
## 1 3.546740   51.62422 0.02486662          0
## 2 2.901422   37.16122 0.59415864          1
## 3 2.912351   42.18503 0.61653435          1
## 4 2.580217   35.40250 0.35425720          0
## 5 3.044522   34.22250 0.09489003          0
## 6 2.928524   32.95908 0.38766297          0
```

b. Confusion Matrix and Metrics

```
length(eval_data$target)
```

```
## [1] 0
```

```
length(eval_data$pred_class)
```

```
## [1] 40
```

```
# Inspect eval_data
str(eval_data)
```

```
## 'data.frame':    40 obs. of  16 variables:
##  $ zn        : int  0 0 0 0 0 25 25 0 0 0 ...
##  $ indus     : num  7.07 8.14 8.14 8.14 5.96 5.13 5.13 4.49 4.49 2.89 ...
##  $ chas      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox       : num  0.469 0.538 0.538 0.538 0.499 0.453 0.453 0.449 0.449 0.445 ...
##  $ rm        : num  7.18 6.1 6.5 5.95 5.85 ...
##  $ age       : num  61.1 84.5 94.4 82 41.5 66.2 93.4 56.1 56.8 69.6 ...
##  $ dis       : num  4.97 4.46 4.45 3.99 3.93 ...
##  $ rad       : int  2 4 4 4 5 8 8 3 3 2 ...
##  $ tax       : int  242 307 307 307 279 284 284 247 247 276 ...
##  $ ptratio   : num  17.8 21 21 21 19.2 19.7 19.7 18.5 18.5 18 ...
##  $ lstat     : num  4.03 10.26 12.8 27.71 8.77 ...
##  $ medv      : num  34.7 18.2 18.4 13.2 21 18.7 16 26.6 22.2 21.4 ...
##  $ log_medv  : num  3.55 2.9 2.91 2.58 3.04 ...
##  $ rm_squared: num  51.6 37.2 42.2 35.4 34.2 ...
##  $ pred_prob : num  0.0249 0.5942 0.6165 0.3543 0.0949 ...
##  $ pred_class: num  0 1 1 0 0 0 0 0 0 0 ...
```

```
summary(eval_data)
```

```
##        zn              indus             chas          nox
##  Min.   : 0.000   Min.   : 1.760   Min.   :0.00   Min.   :0.3850
##  1st Qu.: 0.000   1st Qu.: 5.692   1st Qu.:0.00   1st Qu.:0.4713
##  Median : 0.000   Median : 8.915   Median :0.00   Median :0.5380
##  Mean   : 8.875   Mean   :11.507   Mean   :0.05   Mean   :0.5592
##  3rd Qu.: 0.000   3rd Qu.:18.100   3rd Qu.:0.00   3rd Qu.:0.6258
##  Max.   :90.000   Max.   :25.650   Max.   :1.00   Max.   :0.7400
##        rm              age              dis             rad
##  Min.   :3.561   Min.   :  6.80   Min.   :1.202   Min.   : 1.000
##  1st Qu.:5.874   1st Qu.: 56.62   1st Qu.:2.041   1st Qu.: 4.000
##  Median :6.143   Median : 83.25   Median :3.373   Median : 5.000
##  Mean   :6.214   Mean   : 70.99   Mean   :3.787   Mean   : 9.775
##  3rd Qu.:6.532   3rd Qu.: 93.10   3rd Qu.:4.527   3rd Qu.:24.000
##  Max.   :8.247   Max.   :100.00   Max.   :9.089   Max.   :24.000
##       tax           ptratio          lstat            medv
##  Min.   :188.0   Min.   :14.70   Min.   : 2.960   Min.   : 8.40
##  1st Qu.:276.8   1st Qu.:18.40   1st Qu.: 6.435   1st Qu.:16.98
##  Median :307.0   Median :19.60   Median :11.685   Median :20.55
##  Mean   :393.5   Mean   :19.12   Mean   :12.905   Mean   :21.88
##  3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:17.363   3rd Qu.:25.00
##  Max.   :666.0   Max.   :21.20   Max.   :34.020   Max.   :50.00
##     log_medv       rm_squared      pred_prob         pred_class
##  Min.   :2.128   Min.   :12.68   Min.   :0.00000   Min.   :0.000
##  1st Qu.:2.831   1st Qu.:34.50   1st Qu.:0.08268   1st Qu.:0.000
##  Median :3.023   Median :37.73   Median :0.57002   Median :1.000
##  Mean   :3.016   Mean   :39.07   Mean   :0.50726   Mean   :0.525
##  3rd Qu.:3.219   3rd Qu.:42.66   3rd Qu.:1.00000   3rd Qu.:1.000
##  Max.   :3.912   Max.   :68.01   Max.   :1.00000   Max.   :1.000
```

```
# Check if target variable is present
if ("target" %in% colnames(eval_data)) {
    cat("Target variable exists in eval_data.\n")
} else {
    cat("Target variable is missing from eval_data.\n")
}
```

```
## Target variable is missing from eval_data.
```

```
# Summary of predicted probabilities
summary(eval_data$pred_prob)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.08268 0.57002 0.50726 1.00000 1.00000
```

```
# Summary of predicted classes
table(eval_data$pred_class)
```

```
##
##  0  1
## 19 21
```

```
# Make predictions on training data
train_data$pred_prob <- predict(model3, newdata = train_data, type = "response")
train_data$pred_class <- ifelse(train_data$pred_prob > 0.5, 1, 0)

# Confusion Matrix
confusion_matrix <- table(train_data$target, train_data$pred_class)
```

```r
cat("Confusion Matrix:\n")
```

```
## Confusion Matrix:
```

```r
print(confusion_matrix)  # Ensure the confusion matrix displays
```

```
##
##       0   1
##   0 218  19
##   1  20 209
```

```r
# Accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.916309
```

```r
# Classification Error Rate
classification_error_rate <- 1 - accuracy
cat("Classification Error Rate:", classification_error_rate, "\n")
```

```
## Classification Error Rate: 0.08369099
```

```r
# Sensitivity and Specificity
sensitivity <- confusion_matrix[2, 2] / sum(confusion_matrix[2, ])
specificity <- confusion_matrix[1, 1] / sum(confusion_matrix[1, ])
cat("Sensitivity:", sensitivity, "\n")
```

```
## Sensitivity: 0.9126638
```

```r
cat("Specificity:", specificity, "\n")
```

```
## Specificity: 0.9198312
```

```r
# Precision
precision <- confusion_matrix[2, 2] / sum(confusion_matrix[, 2])
cat("Precision:", precision, "\n")
```

```
## Precision: 0.9166667
```

```r
# F1 Score
f1_score <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1 Score:", f1_score, "\n")
```

```
## F1 Score: 0.9146608
```

```r
# ROC Curve and AUC
library(pROC)
```

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

```
##
## Attaching package: 'pROC'
```

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

```r
roc_obj <- roc(train_data$target, train_data$pred_prob)
```

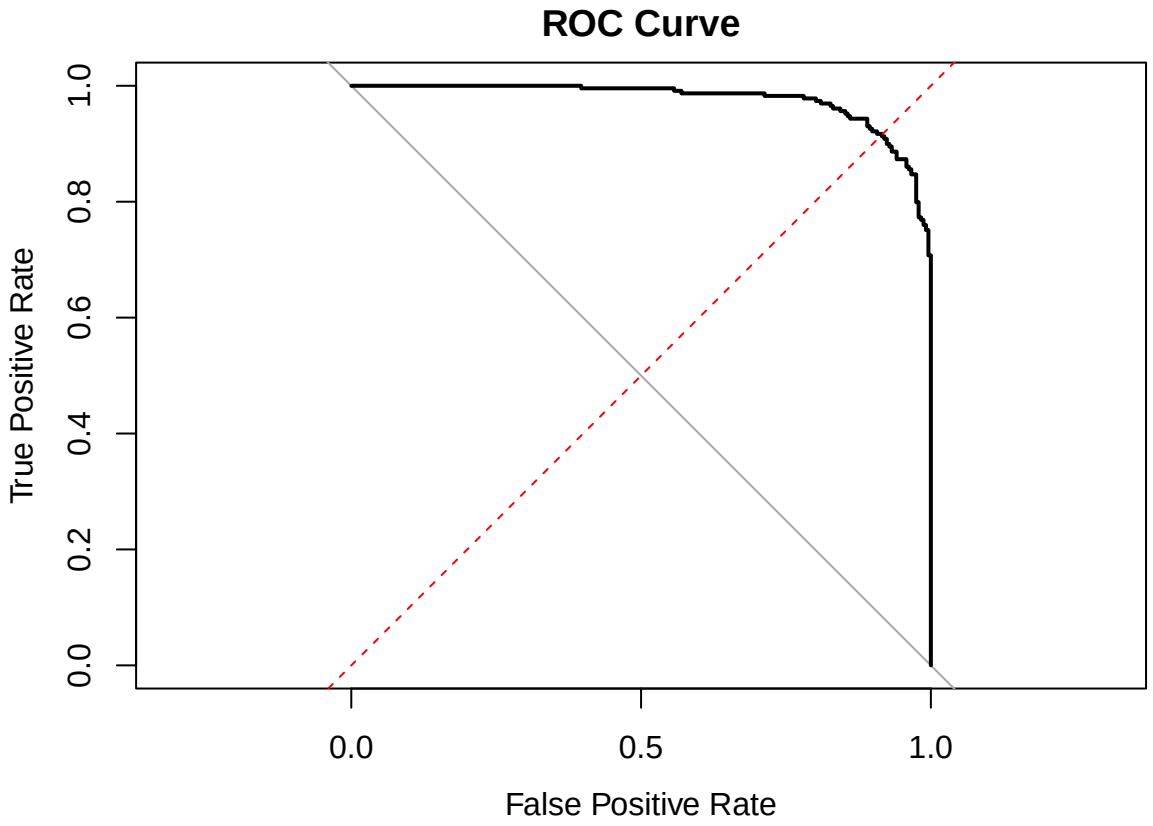
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
auc_value <- auc(roc_obj)
cat("AUC:", auc_value, "\n")
```

```
## AUC: 0.9765629
```

```
# Plot ROC Curve
plot(roc_obj, main = "ROC Curve", xlim = c(0, 1), ylim = c(0, 1),
     xlab = "False Positive Rate", ylab = "True Positive Rate")
abline(a = 0, b = 1, lty = 2, col = "red") # Add a diagonal line for reference
```



- Confusion Matrix:

True Negatives (0 predicted as 0): 218

False Positives (0 predicted as 1): 19

False Negatives (1 predicted as 0): 20

True Positives (1 predicted as 1): 209

- Accuracy (91.63%): Indicates that the model correctly predicts the class for about 92% of the training samples, a strong result.
- Classification Error Rate (8.37%): Complements the accuracy, showing the proportion of misclassifications, which is low.
- Sensitivity (91.27%): Represents the model's ability to correctly identify positive cases (target = 1). This high sensitivity means the model is effective in identifying positives.
- Specificity (91.98%): Indicates the model's ability to correctly identify negative cases (target = 0). A similarly high value here shows balanced performance across classes.

- Precision (91.67%): Shows how many of the instances predicted as positive are actually positive, which is crucial if the cost of false positives is high. F1 Score (91.47%): Balances precision and sensitivity, confirming the model's consistent performance.
- AUC (0.9766): Reflects excellent overall separability between classes, meaning the model is likely distinguishing well between positive and negative cases.

These metrics suggest a well-performing model, especially given the high AUC and balanced sensitivity and specificity.