GroupM Analysis Notebook MARS

GroupM

2025-02-11

This R notebook details the implementation of MARS on the 3 different datasets, using the earth package to set up the models.

Data Import and Cleaning

```r
# Install packages if not already installed
if (!requireNamespace("Metrics", quietly = TRUE)) install.packages("Metrics", quiet = TRUE)
if (!requireNamespace("earth", quietly = TRUE)) install.packages("earth", quiet = TRUE)
if (!requireNamespace("MASS", quietly = TRUE)) install.packages("MASS", quiet = TRUE)

# Load packages with suppressed warnings
suppressWarnings({
  suppressPackageStartupMessages({
    library(MASS)    # Load the MASS package for robust linear models
    library(earth)   # Load earth package for MARS
    library(Metrics) # Load Metrics package for MAE, RMSE, MAPE
  })
})
```

```r
# Load the CleanDataset.csv file
clean_dataset <- read.csv('data/Clean_Dataset.csv')
clean_dataset <- clean_dataset[, !(names(clean_dataset) %in% c("X", "flight"))]

allclasses_dataset <- clean_dataset
economy_dataset <- clean_dataset[clean_dataset$class == "Economy", ]
business_dataset <- clean_dataset[clean_dataset$class == "Business", ]
```

```r
# Scale the values of the columns duration and days_left
allclasses_dataset$duration <- scale(allclasses_dataset$duration)
allclasses_dataset$days_left <- scale(allclasses_dataset$days_left)
economy_dataset$duration <- scale(economy_dataset$duration)
economy_dataset$days_left <- scale(economy_dataset$days_left)
business_dataset$duration <- scale(business_dataset$duration)
business_dataset$days_left <- scale(business_dataset$days_left)
```

```r
# Use log transformation for price
allclasses_dataset$price <- log(allclasses_dataset$price)
economy_dataset$price <- log(economy_dataset$price)
business_dataset$price <- log(business_dataset$price)
```

```r
# Remove the class column for the separate models
economy_dataset$class <- NULL
business_dataset$class <- NULL
```

```r
# Show the dimensions and the first rows of the datasets
cat("\nDimensions of the full dataset (all classes):\n")
```

```
##
## Dimensions of the full dataset (all classes):
```

```r
dim(allclasses_dataset)
```

```
## [1] 300153     10
```

```r
cat("\nFirst rows of the full dataset (all classes):\n")
```

```
##
## First rows of the full dataset (all classes):
```

```r
head(allclasses_dataset)
```

```
##    airline source_city departure_time stops  arrival_time destination_city
## 1 SpiceJet       Delhi        Evening  zero         Night           Mumbai
## 2 SpiceJet       Delhi  Early_Morning  zero       Morning           Mumbai
## 3  AirAsia       Delhi  Early_Morning  zero Early_Morning           Mumbai
## 4  Vistara       Delhi        Morning  zero     Afternoon           Mumbai
## 5  Vistara       Delhi        Morning  zero       Morning           Mumbai
## 6  Vistara       Delhi        Morning  zero     Afternoon           Mumbai
##     class  duration days_left    price
## 1 Economy -1.397528 -1.843872 8.691651
## 2 Economy -1.375282 -1.843872 8.691651
## 3 Economy -1.397528 -1.843872 8.692154
## 4 Economy -1.386405 -1.843872 8.691986
## 5 Economy -1.375282 -1.843872 8.691986
## 6 Economy -1.375282 -1.843872 8.691986
```

```r
cat("\nDimensions of the economy class dataset:\n")
```

```
##
## Dimensions of the economy class dataset:
```

```r
dim(economy_dataset)
```

```
## [1] 206666      9
```

```r
cat("\nDimensions of the business class dataset:\n")
```

```
##
## Dimensions of the business class dataset:
```

```r
dim(business_dataset)
```

```
## [1] 93487     9
```

```r
cat("\nFirst rows of the economy class dataset:\n")
```

```
##
## First rows of the economy class dataset:
```

```r
head(economy_dataset)
```

```
##    airline source_city departure_time stops  arrival_time destination_city
## 1 SpiceJet       Delhi        Evening  zero         Night           Mumbai
## 2 SpiceJet       Delhi  Early_Morning  zero       Morning           Mumbai
## 3  AirAsia       Delhi  Early_Morning  zero Early_Morning           Mumbai
## 4  Vistara       Delhi        Morning  zero     Afternoon           Mumbai
## 5  Vistara       Delhi        Morning  zero       Morning           Mumbai
## 6  Vistara       Delhi        Morning  zero     Afternoon           Mumbai
##    duration days_left    price
## 1 -1.295359  -1.85694 8.691651
## 2 -1.273263  -1.85694 8.691651
## 3 -1.295359  -1.85694 8.692154
## 4 -1.284311  -1.85694 8.691986
## 5 -1.273263  -1.85694 8.691986
## 6 -1.273263  -1.85694 8.691986
```

```r
cat("\nFirst rows of the business class dataset:\n")
```

```
##
## First rows of the business class dataset:
```

```r
head(business_dataset)
```

```
##          airline source_city departure_time stops arrival_time destination_city
## 206667 Air_India       Delhi        Evening  zero      Evening           Mumbai
## 206668 Air_India       Delhi        Evening  zero        Night           Mumbai
## 206669 Air_India       Delhi        Evening   one        Night           Mumbai
## 206670 Air_India       Delhi          Night   one        Night           Mumbai
## 206671 Air_India       Delhi        Evening   one        Night           Mumbai
## 206672   Vistara       Delhi        Evening  zero        Night           Mumbai
##         duration days_left    price
## 206667 -1.708016 -1.815711 10.15082
## 206668 -1.671533 -1.815711 10.15082
## 206669  1.611913 -1.815711 10.65065
## 206670  1.867293 -1.815711 10.70212
## 206671 -1.026518 -1.815711 10.75129
## 206672 -1.683208 -1.815711 10.82504
```

```r
set.seed(123)  # For reproducibility
```

```r
# Create train and test sets for allclasses dataset
train_indices_allclasses <- sample(1:nrow(allclasses_dataset),
                                    size = 0.8 * nrow(allclasses_dataset))
train_allclasses <- allclasses_dataset[train_indices_allclasses, ]
test_allclasses <- allclasses_dataset[-train_indices_allclasses, ]
```

```r
# Create train and test sets for economy dataset
train_indices_economy <- sample(1:nrow(economy_dataset),
                                 size = 0.8 * nrow(economy_dataset))
train_economy <- economy_dataset[train_indices_economy, ]
test_economy <- economy_dataset[-train_indices_economy, ]
```

```r
# Create train and test sets for business dataset
train_indices_business <- sample(1:nrow(business_dataset),
                                  size = 0.8 * nrow(business_dataset))
train_business <- business_dataset[train_indices_business, ]
test_business <- business_dataset[-train_indices_business, ]
```

```r
# Show the dimensions and the first rows of the train and test sets
cat("Dimensions of training set for all classes (80% of data):\n")
```

```
## Dimensions of training set for all classes (80% of data):
```

```r
dim(train_allclasses)
```

```
## [1] 240122     10
```

```r
cat("\nDimensions of test set for all classes (20% of data):\n")
```

```
##
## Dimensions of test set for all classes (20% of data):
```

```r
dim(test_allclasses)
```

```
## [1] 60031    10
```

```r
cat("\nDimensions of training set for economy class (80% of data):\n")
```

```
##
## Dimensions of training set for economy class (80% of data):
```

```r
dim(train_economy)
```

```
## [1] 165332      9
```

```r
cat("\nDimensions of test set for economy class (20% of data):\n")
```

```
##
## Dimensions of test set for economy class (20% of data):
```

```r
dim(test_economy)
```

```
## [1] 41334     9
```

```r
cat("\nDimensions of training set for business class (80% of data):\n")
```

```
##
## Dimensions of training set for business class (80% of data):
```

```r
dim(train_business)
```

```
## [1] 74789     9
```

```r
cat("\nDimensions of test set for business class (20% of data):\n")
```

```
##
## Dimensions of test set for business class (20% of data):
```

```r
dim(test_business)
```

```
## [1] 18698     9
```

MARS - Complete model

```r
# Construct the formula for the model with grade 2 numerical variables
formula <- as.formula(paste("price ~",
                            paste(c(names(train_allclasses)[-which(names(train_allclasses) == "price")]
    "I(duration^2)", "I(days_left^2)"), collapse = " + ")))
cat("Constructed formula for the model:\n")
```

```
## Constructed formula for the model:
```

```r
print(formula)
```

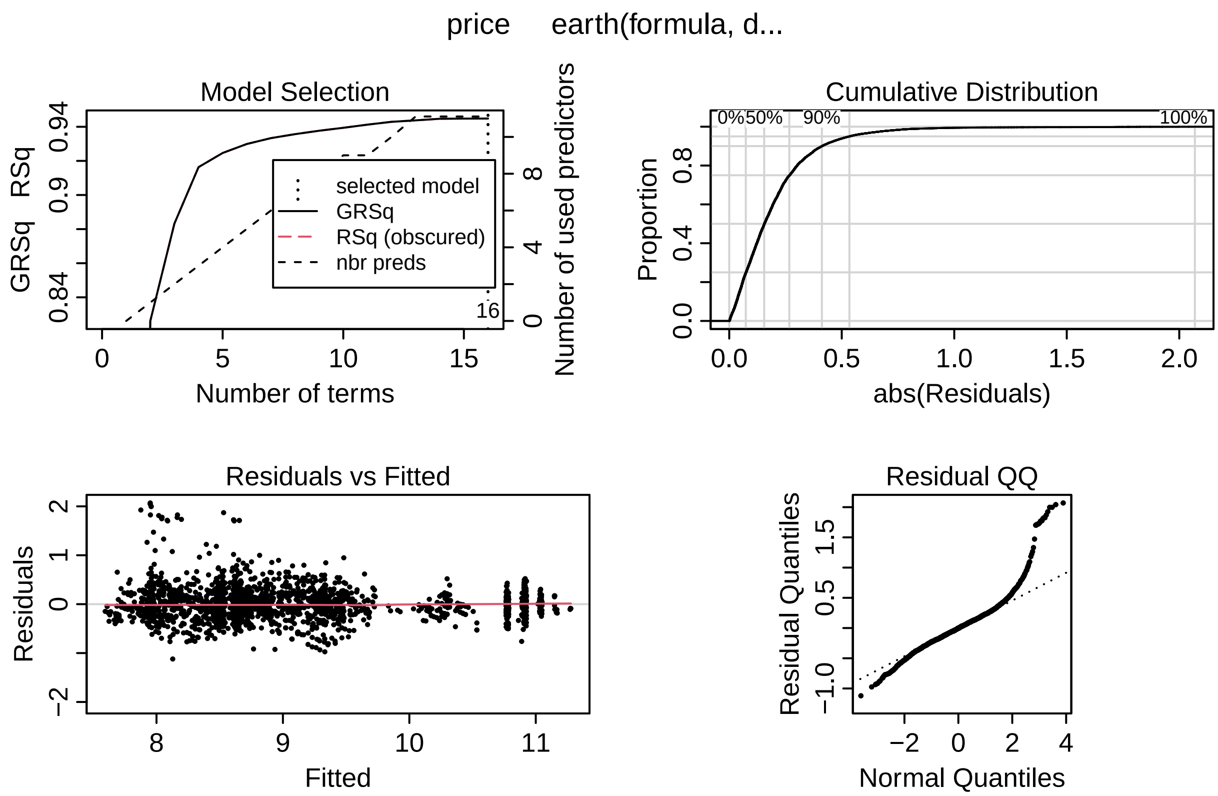
```
## price ~ airline + source_city + departure_time + stops + arrival_time +
##     destination_city + class + duration + days_left + I(duration^2) +
##     I(days_left^2)
```

Train the model using the internal algorithm to automatically choose terms and predictors:

```r
# Create MARS model for allclasses
mars_model_allclasses <- earth(formula,
                              data = train_allclasses,
                              degree = 4,  # Fixed degree
                             )
# Print model results
summary(mars_model_allclasses)
```

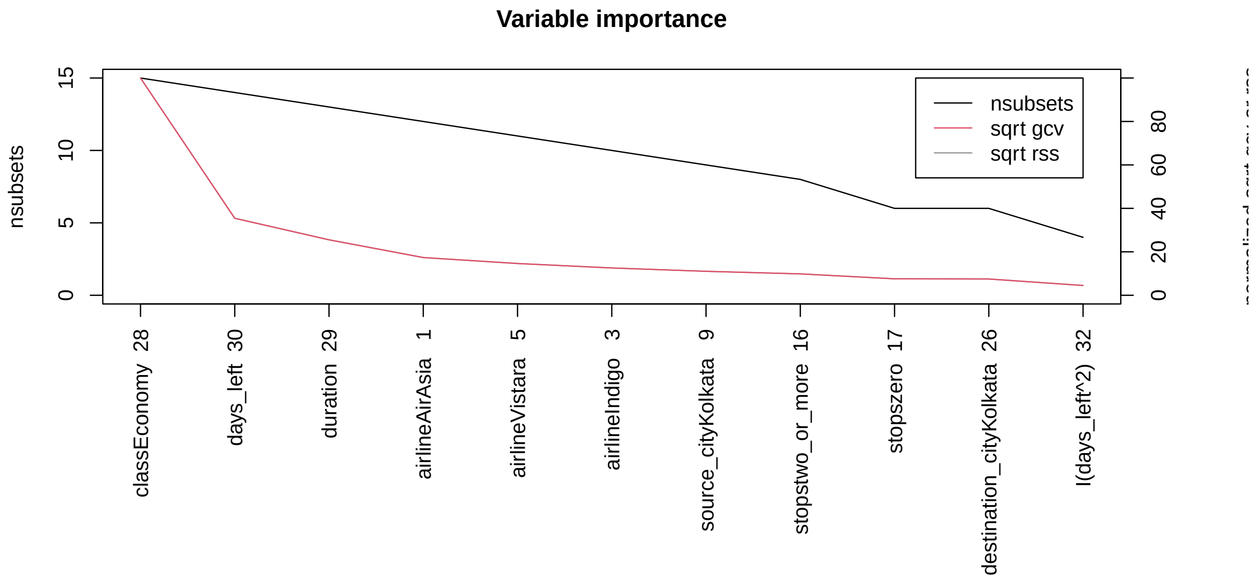
```
## Call: earth(formula=formula, data=train_allclasses, degree=4)
##
##                                                                    coefficients
## (Intercept)                                                          10.7648782
## airlineAirAsia                                                       -0.5495499
## airlineVistara                                                        0.1407637
## stopstwo_or_more                                                      0.2384509
## destination_cityKolkata                                               0.1236809
## classEconomy                                                         -2.2399891
## h(-1.04992-duration)                                                 -1.8028876
## h(duration- -1.04992)                                                 0.0063264
## airlineAirAsia * stopszero                                            0.5932892
## source_cityKolkata * classEconomy                                     0.1961968
## classEconomy * h(days_left- -0.369055)                               -0.0306106
## classEconomy * h(-0.369055-days_left)                                 0.8671968
## airlineIndigo * classEconomy * h(days_left- -0.369055)               -0.2688163
## classEconomy * h(-0.369055-days_left) * h(I(days_left^2)-0.658532)   -0.1183905
## classEconomy * h(-0.369055-days_left) * h(0.658532-I(days_left^2))   -2.7642917
## airlineIndigo * stopszero * classEconomy * h(days_left- -0.369055)    0.3346818
##
## Selected 16 of 16 terms, and 11 of 32 predictors
## Termination condition: RSq changed by less than 0.001 at 16 terms
## Importance: classEconomy, days_left, duration, airlineAirAsia, ...
## Number of terms at each degree of interaction: 1 7 4 3 1
## GCV 0.06839973    RSS 16419.01    GRSq 0.94483    RSq 0.9448472
```

```r
par(mfrow = c(2, 2))  # Set up the plotting area
plot(mars_model_allclasses)
```



We will use the evimp function from the 'earth' package to obtain the variable importance from the MARS model.

```
evimp <- evimp(mars_model_allclasses, trim=TRUE, sqrt.=TRUE)
plot(evimp)
```



The plot illustrates the importance of the variables in the MARS model, highlighting that the most important predictor is the class, followed by the two numerical variables of degree 1.

```
# Make predictions first
test_allclasses$duration_squared <- test_allclasses$duration^2
test_allclasses$days_left_squared <- test_allclasses$days_left^2
predictions_allclasses <- predict(mars_model_allclasses, newdata = test_allclasses)

# Calculate the residuals
residuals_mars_allclasses <- test_allclasses$price - predictions_allclasses

# Calculate the log-likelihood (fixed array conformability issue)
log_likelihood <- -0.5 * length(residuals_mars_allclasses) * (log(2 * pi) +
                  log(var(residuals_mars_allclasses))) -
                  sum(residuals_mars_allclasses^2) / (2 * var(residuals_mars_allclasses))

# Calculate AIC and BIC
n <- length(residuals_mars_allclasses)  # Number of observations
k <- length(mars_model_allclasses$coefficients)  # Number of parameters

aic_value <- -2 * log_likelihood + 2 * k
bic_value <- -2 * log_likelihood + log(n) * k

# Print the results
cat("allclasses Model Performance:\n")
```

```
## allclasses Model Performance:
```

```r
cat("AIC:", aic_value, "\n")
```

```
## AIC: 9152.888
```

```r
cat("BIC:", bic_value, "\n")
```

```
## BIC: 9296.93
```

```r
# Performance metrics for allclasses model using test set
# R^2 for log-transformed data
r2_allclasses_log <- 1 - sum((residuals_mars_allclasses)^2) /
  sum((test_allclasses$price - mean(test_allclasses$price))^2)

# R^2 for original scale data
r2_allclasses_orig <- 1 - sum((exp(test_allclasses$price) - exp(predictions_allclasses))^2) /
                      sum((exp(test_allclasses$price) - mean(exp(test_allclasses$price)))^2)

mae_allclasses <- mae(exp(test_allclasses$price), exp(predictions_allclasses))
rmse_allclasses <- rmse(exp(test_allclasses$price), exp(predictions_allclasses))
mape_allclasses <- mape(exp(test_allclasses$price), exp(predictions_allclasses))

cat("R^2 (log scale):", r2_allclasses_log, "\n")
```

```
## R^2 (log scale): 0.9446884
```

```r
cat("R^2 (original scale):", r2_allclasses_orig, "\n")
```

```
## R^2 (original scale): 0.9366895
```

```r
cat("MAE:", mae_allclasses, "\n")
```

```
## MAE: 3269.506
```

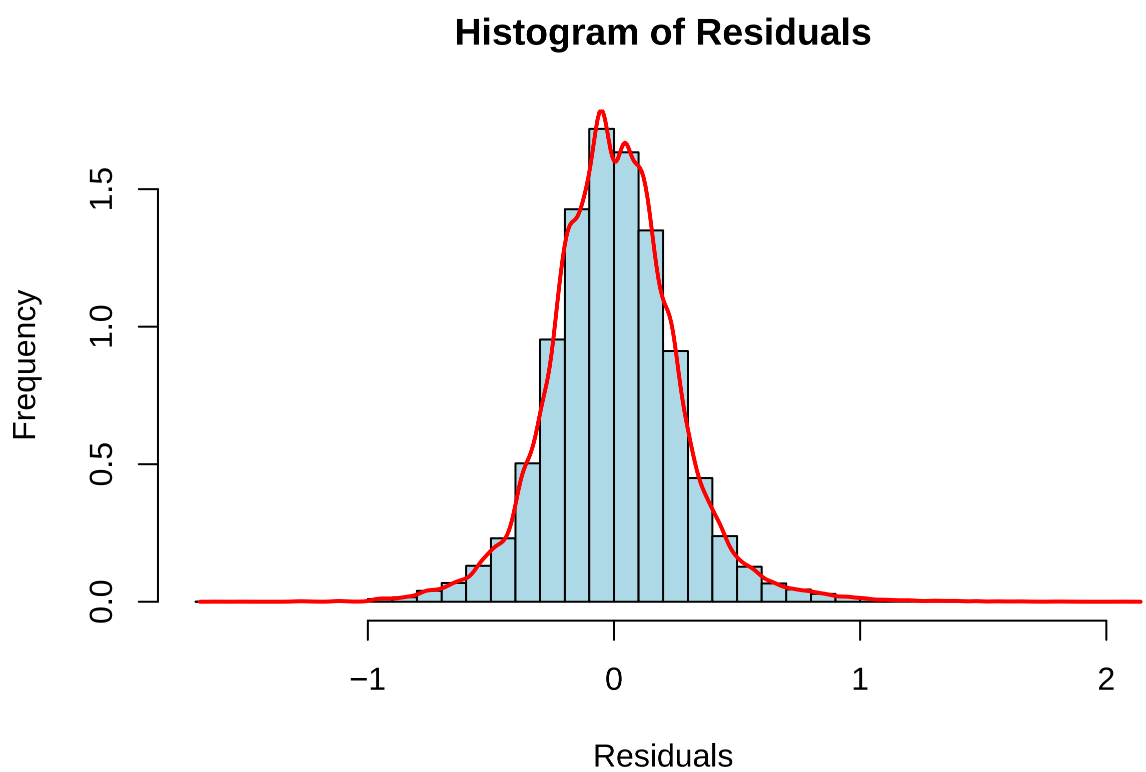
```r
cat("RMSE:", rmse_allclasses, "\n")
```

```
## RMSE: 5697.624
```

```r
cat("MAPE:", mape_allclasses * 100, "%\n")
```

```
## MAPE: 20.0425 %
```

```r
# Create a histogram of the residuals
hist(residuals_mars_allclasses, breaks = 30, main = "Histogram of Residuals",
     xlab = "Residuals", ylab = "Frequency", col = "lightblue", border = "black", freq = FALSE)

# Add a density curve
lines(density(residuals_mars_allclasses), col = "red", lwd = 2)
```



MARS - Economy model

```
# Construct the formula for the model, excluding the 'class' column
formula <- as.formula(paste("price ~",
          paste(c(names(train_allclasses)[-which(names(train_allclasses) %in% c("price", "class"))],
    "I(duration^2)", "I(days_left^2)"), collapse = " + ")))
cat("Constructed formula for the model:\n")
```

```
## Constructed formula for the model:
```

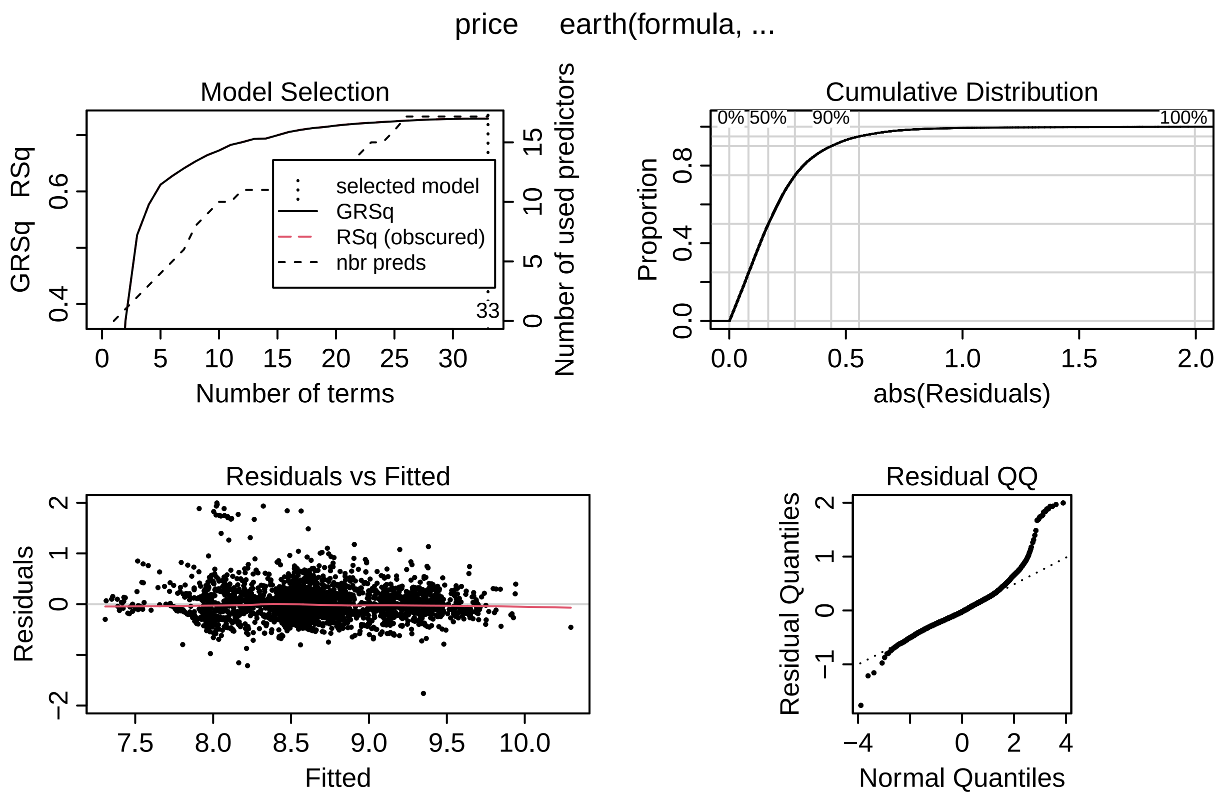
```
print(formula)
```

```
## price ~ airline + source_city + departure_time + stops + arrival_time +
##     destination_city + duration + days_left + I(duration^2) +
##     I(days_left^2)
```

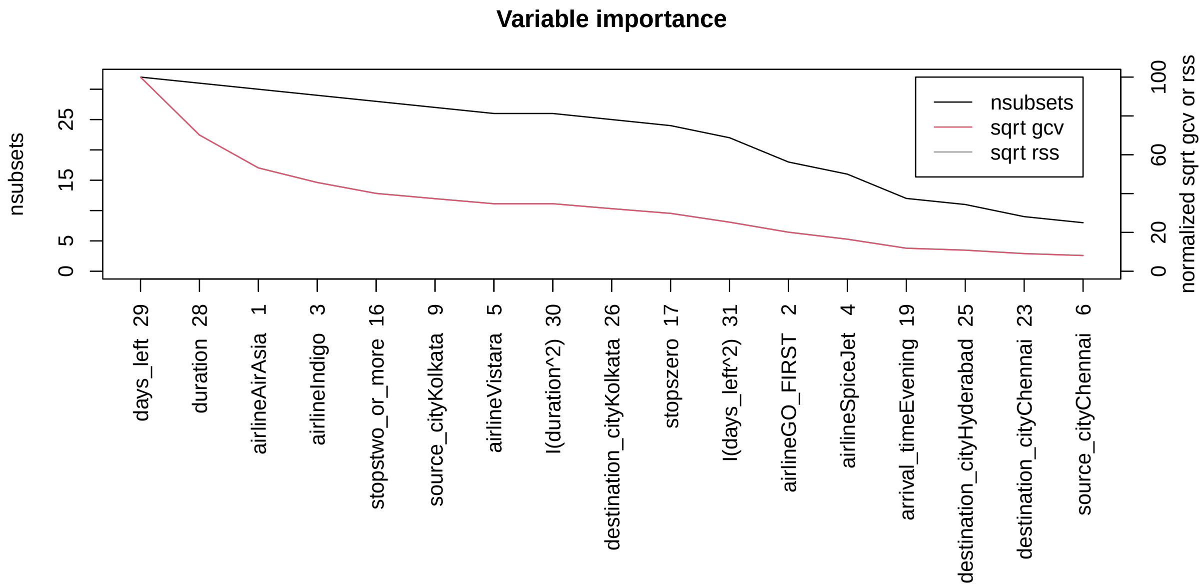
```
# Create MARS model for economy dataset with fixed parameters
mars_model_economy <- earth(formula,
                            data = train_economy,  # Changed to use train dataset
                            degree = 4,   # Fixed degree
                           )

# Print model results
summary(mars_model_economy)
```

```
## Call: earth(formula=formula, data=train_economy, degree=4)
##
##                                                                          coefficients
## (Intercept)                                                                 5.8065122
## airlineAirAsia                                                             -0.5530625
## airlineVistara                                                             -0.0130203
## source_cityKolkata                                                          0.1906196
## stopstwo_or_more                                                            0.2714631
## arrival_timeEvening                                                         0.0496810
## destination_cityHyderabad                                                  -0.0711138
## destination_cityKolkata                                                     0.1214159
## h(-0.973592-duration)                                                      -1.3054437
## h(duration- -0.973592)                                                      0.0256313
## h(days_left- -1.56129)                                                      2.2954781
## h(-0.378701-days_left)                                                      3.4487140
## h(days_left- -0.378701)                                                    -2.3101676
## airlineAirAsia * stopstwo_or_more                                          -0.2306201
## airlineAirAsia * stopszero                                                  0.5248049
## airlineGO_FIRST * h(-0.378701-days_left)                                   -0.6312378
## airlineIndigo * h(duration- -0.973592)                                     -0.1375326
## airlineIndigo * h(days_left- -0.378701)                                    -0.2328032
## airlineIndigo * h(-0.378701-days_left)                                     -0.1644052
## airlineSpiceJet * h(-0.378701-days_left)                                   -0.2787310
## airlineVistara * h(I(duration^2)-3.23783)                                   0.0628479
## airlineVistara * h(3.23783-I(duration^2))                                   0.0403096
## source_cityChennai * h(days_left- -0.378701)                               -0.0626020
## destination_cityChennai * h(days_left- -1.56129)                           -0.0375661
## h(-1.16969-duration) * h(-0.378701-days_left)                               0.9524781
## h(duration- -1.16969) * h(-0.378701-days_left)                             -0.0329539
## h(-0.973592-duration) * h(I(duration^2)-1.21141)                           -1.0306985
## h(-0.973592-duration) * h(1.21141-I(duration^2))                          -13.6655376
## h(-0.378701-days_left) * h(I(days_left^2)-0.675968)                        -0.3033771
## h(-0.378701-days_left) * h(0.675968-I(days_left^2))                        -3.1693493
## airlineIndigo * stopszero * h(days_left- -0.378701)                         0.2912986
## airlineGO_FIRST * h(-0.378701-days_left) * h(I(days_left^2)-0.559894)       0.1586509
## airlineGO_FIRST * h(-0.378701-days_left) * h(0.559894-I(days_left^2))       1.6510959
##
## Selected 33 of 33 terms, and 17 of 31 predictors
## Termination condition: RSq changed by less than 0.001 at 33 terms
## Importance: days_left, duration, airlineAirAsia, airlineIndigo, ...
## Number of terms at each degree of interaction: 1 12 17 3
## GCV 0.07528554    RSS 12434.92    GRSq 0.7291458    RSq 0.7294079
```

```r
par(mfrow = c(2, 2))  # Set up the plotting area
plot(mars_model_economy)
```




```
evimp <- evimp(mars_model_economy, trim=TRUE, sqrt.=TRUE)
plot(evimp)
```




```r
# Predictions and performance metrics for economy model
test_economy$duration_squared <- test_economy$duration^2
test_economy$days_left_squared <- test_economy$days_left^2
predictions_economy <- predict(mars_model_economy, newdata = test_economy)

# Calculate the residuals
residuals_mars_economy <- test_economy$price - predictions_economy

# Calculate the log-likelihood (fixed array conformability issue)
log_likelihood <- -0.5 * length(residuals_mars_economy) * (log(2 * pi) +
                  log(var(residuals_mars_economy))) -
                  sum(residuals_mars_economy^2) / (2 * var(residuals_mars_economy))

# Calculate AIC and BIC
n <- length(residuals_mars_economy)  # Number of observations
k <- length(mars_model_economy$coefficients)  # Number of parameters

aic_value <- -2 * log_likelihood + 2 * k
bic_value <- -2 * log_likelihood + log(n) * k

# Print the results
cat("Economy Model Performance:\n")
```

```
## Economy Model Performance:
```

```r
cat("AIC:", aic_value, "\n")
```

```
## AIC: 9439.121
```

```r
cat("BIC:", bic_value, "\n")
```

```
## BIC: 9723.892
```

```r
# Performance metrics for allclasses model using test set
# R^2 for log-transformed data
r2_economy_log <- 1 - sum((residuals_mars_economy)^2) /
  sum((test_economy$price - mean(test_economy$price))^2)

# R^2 for original scale data
r2_economy_orig <- 1 - sum((exp(test_economy$price) - exp(predictions_economy))^2) /
                      sum((exp(test_economy$price) - mean(exp(test_economy$price)))^2)

# Calculate performance metrics
mae_economy <- mae(exp(test_economy$price), exp(predictions_economy))
rmse_economy <- rmse(exp(test_economy$price), exp(predictions_economy))
mape_economy <- mape(exp(test_economy$price), exp(predictions_economy))


# Print performance results
cat("R^2 Test (log scale):", r2_economy_log, "\n")
```

```
## R^2 Test (log scale): 0.7365014
```

```r
cat("R^2 Test (original scale):", r2_economy_orig, "\n")
```

```
## R^2 Test (original scale): 0.6780231
```

```r
cat("MAE:", mae_economy, "\n")
```

```
## MAE: 1347.961
```

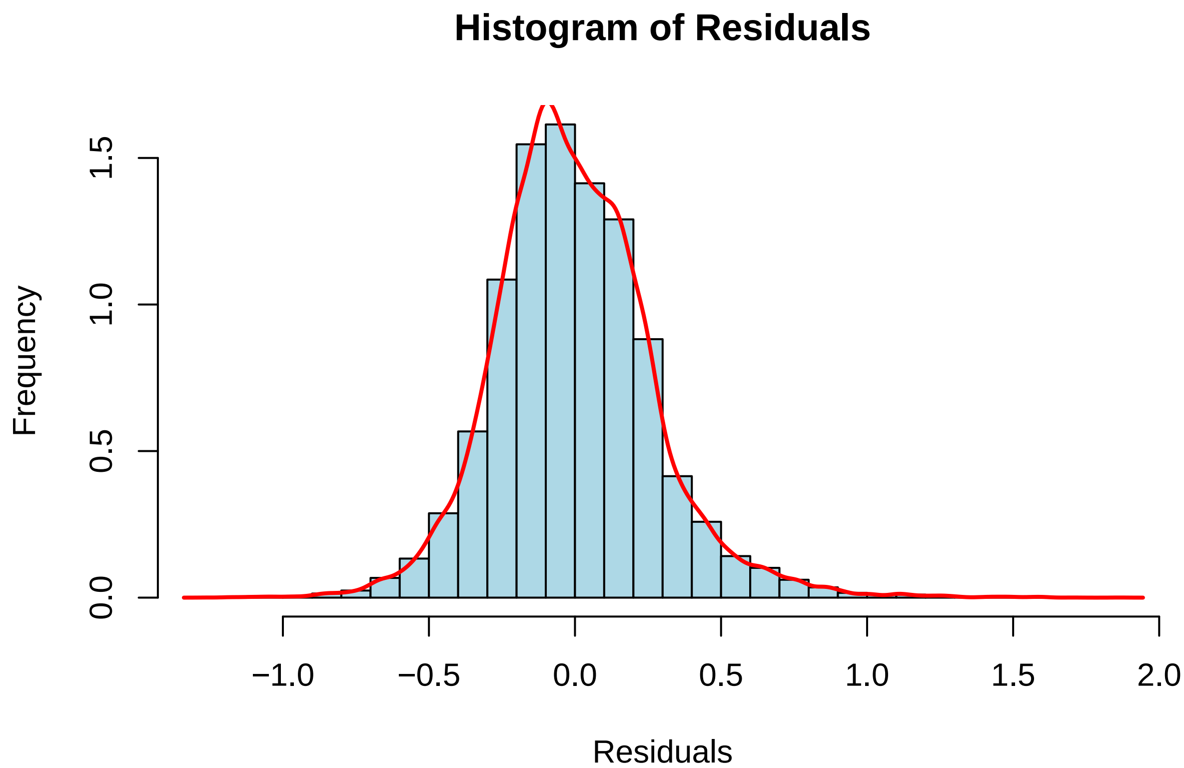
```r
cat("RMSE:", rmse_economy, "\n")
```

```
## RMSE: 2119.797
```

```r
cat("MAPE:", mape_economy * 100, "%\n")
```

```
## MAPE: 20.94346 %
```

```r
# Create a histogram of the residuals
hist(residuals_mars_economy, breaks = 30, main = "Histogram of Residuals",
     xlab = "Residuals", ylab = "Frequency", col = "lightblue", border = "black", freq = FALSE)

# Add a density curve
lines(density(residuals_mars_economy), col = "red", lwd = 2)
```



MARS - Business Model

```
# Construct the formula for the model, excluding the 'class' column
formula <- as.formula(paste("price ~",
          paste(c(names(train_allclasses)[-which(names(train_allclasses) %in% c("price", "class"))],
    "I(duration^2)", "I(days_left^2)"), collapse = " + ")))
cat("Constructed formula for the model:\n")
```

```
## Constructed formula for the model:
```

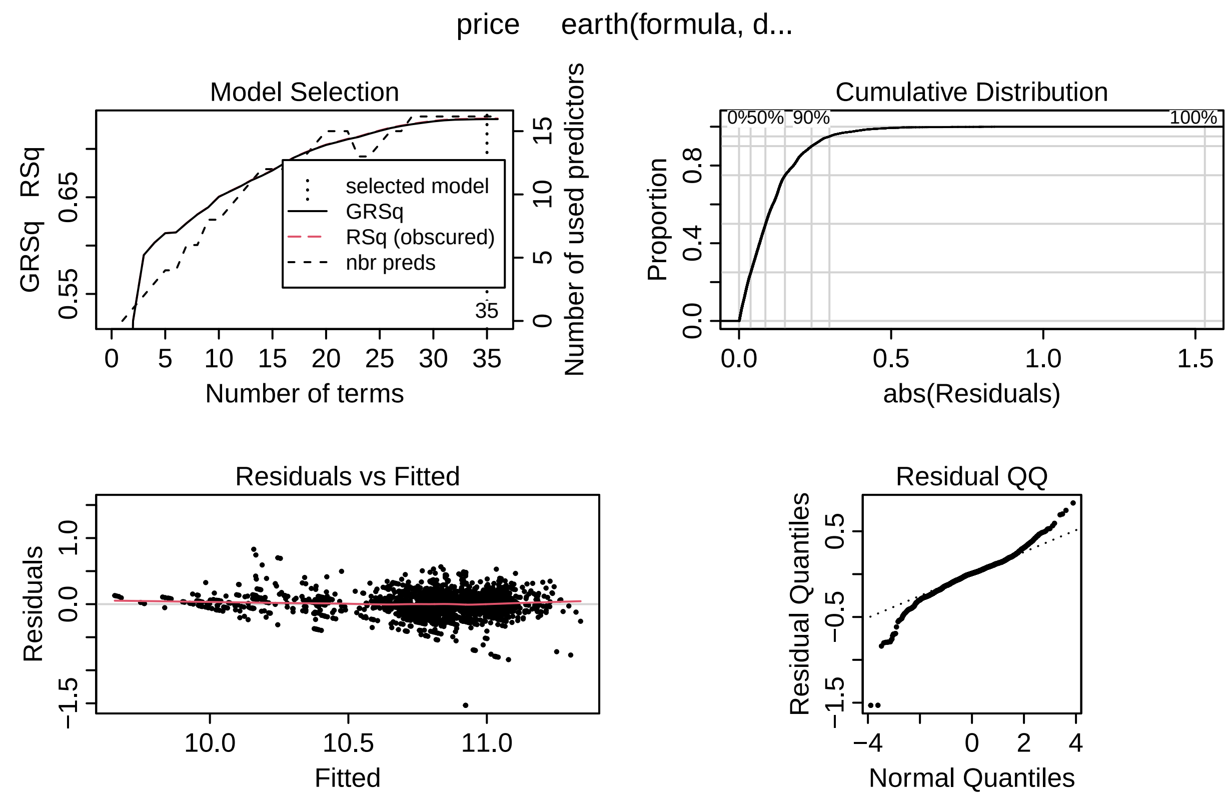
```
print(formula)
```

```
## price ~ airline + source_city + departure_time + stops + arrival_time +
##     destination_city + duration + days_left + I(duration^2) +
##     I(days_left^2)
```

```
# Create MARS model for business dataset with fixed parameters
mars_model_business <- earth(formula,
                             data = train_business,
                             degree = 4,   # Fixed degree
                            )

# Print model results
summary(mars_model_business)
```

```
## Call: earth(formula=formula, data=train_business, degree=4)
##
##                                                                               coefficients
## (Intercept)                                                                     10.7424783
## airlineVistara                                                                   0.0929925
## source_cityKolkata                                                               0.1144888
## stopstwo_or_more                                                                 0.1447733
## destination_cityDelhi                                                           -0.0984445
## destination_cityKolkata                                                          0.1067816
## h(-1.13597-duration)                                                            -2.5208500
## h(duration- -1.13597)                                                            0.0083701
## h(-1.15524-days_left)                                                            0.1253181
## h(days_left- -1.15524)                                                          -0.0057789
## h(3.53052-I(duration^2))                                                         0.0203501
## h(I(duration^2)-3.53052)                                                         0.0275178
## airlineVistara * source_cityHyderabad                                           -0.0668712
## airlineVistara * source_cityMumbai                                               0.1610923
## airlineVistara * destination_cityDelhi                                           0.1489986
## airlineVistara * destination_cityHyderabad                                      -0.0835799
## airlineVistara * destination_cityMumbai                                          0.1828583
## source_cityKolkata * destination_cityMumbai                                     -0.1215814
## source_cityMumbai * destination_cityKolkata                                     -0.1064902
## airlineVistara * h(duration- -1.68321)                                          -0.0136079
## airlineVistara * h(-1.68321-duration)                                           -2.5184532
## airlineVistara * h(-1.44878-days_left)                                           0.4934502
## source_cityDelhi * h(duration- -1.13597)                                        -0.0422451
## stopszero * h(-1.13597-duration)                                                 1.2507017
## arrival_timeEarly_Morning * h(duration- -1.13597)                               -0.0514523
## h(-1.13597-duration) * h(I(duration^2)-2.75513)                                 -0.5885591
## h(-1.13597-duration) * h(2.75513-I(duration^2))                                  1.1775784
## airlineVistara * source_cityDelhi * destination_cityMumbai                      -0.2798514
## airlineVistara * source_cityMumbai * destination_cityDelhi                      -0.1894222
## airlineVistara * arrival_timeEvening * destination_cityHyderabad                 0.0863152
## airlineVistara * source_cityDelhi * h(duration- -1.68321)                        0.0739126
## airlineVistara * source_cityDelhi * destination_cityMumbai * h(duration- -0.844105)   -0.0927810
## airlineVistara * source_cityDelhi * destination_cityMumbai * h(-0.844105-duration)     0.4299237
## airlineVistara * source_cityMumbai * destination_cityDelhi * h(duration- -1.34319)    -0.0715444
## airlineVistara * source_cityMumbai * destination_cityDelhi * h(-1.34319-duration)      0.7454559
##
## Selected 35 of 36 terms, and 16 of 27 predictors
## Termination condition: Reached nk 55
## Importance: duration, airlineVistara, days_left, I(duration^2), ...
## Number of terms at each degree of interaction: 1 11 15 4 4
## GCV 0.02122194    RSS 1583.519    GRSq 0.7308156    RSq 0.7314271
```

```r
par(mfrow = c(2, 2))  # Set up the plotting area
plot(mars_model_business)
```




```
# Make predictions first
test_business$duration_squared <- test_business$duration^2
test_business$days_left_squared <- test_business$days_left^2
predictions_business <- predict(mars_model_business, newdata = test_business)

# Calculate the residuals
residuals_mars_business <- test_business$price - predictions_business

# Calculate the log-likelihood (fixed array conformability issue)
log_likelihood <- -0.5 * length(residuals_mars_business) * (log(2 * pi) +
                  log(var(residuals_mars_business))) -
                  sum(residuals_mars_business^2) / (2 * var(residuals_mars_business))

# Calculate AIC and BIC
n <- length(residuals_mars_business)  # Number of observations
k <- length(mars_model_business$coefficients)  # Number of parameters

aic_value <- -2 * log_likelihood + 2 * k
bic_value <- -2 * log_likelihood + log(n) * k

# Print the results
cat("Business Model Performance:\n")
```

```
## Business Model Performance:
```

```r
cat("AIC:", aic_value, "\n")
```

```
## AIC: -18591.92
```

```r
cat("BIC:", bic_value, "\n")
```

```
## BIC: -18317.66
```

```r
# Performance metrics for allclasses model using test set
# R^2 for log-transformed data
r2_business_log <- 1 - sum((residuals_mars_business)^2) /
  sum((test_business$price - mean(test_business$price))^2)

# R^2 for original scale data
r2_business_orig <- 1 - sum((exp(test_business$price) - exp(predictions_business))^2) /
                      sum((exp(test_business$price) - mean(exp(test_business$price)))^2)

mae_business <- mae(exp(test_business$price), exp(predictions_business))
rmse_business <- rmse(exp(test_business$price), exp(predictions_business))
mape_business <- mape(exp(test_business$price), exp(predictions_business))


cat("R^2 TEST (log scale):", r2_business_log, "\n")
```

```
## R^2 TEST (log scale): 0.7210716
```

```r
cat("R^2 TEST (original scale):", r2_business_orig, "\n")
```

```
## R^2 TEST (original scale): 0.6321119
```

```r
cat("MAE:", mae_business, "\n")
```

```
## MAE: 5744.135
```

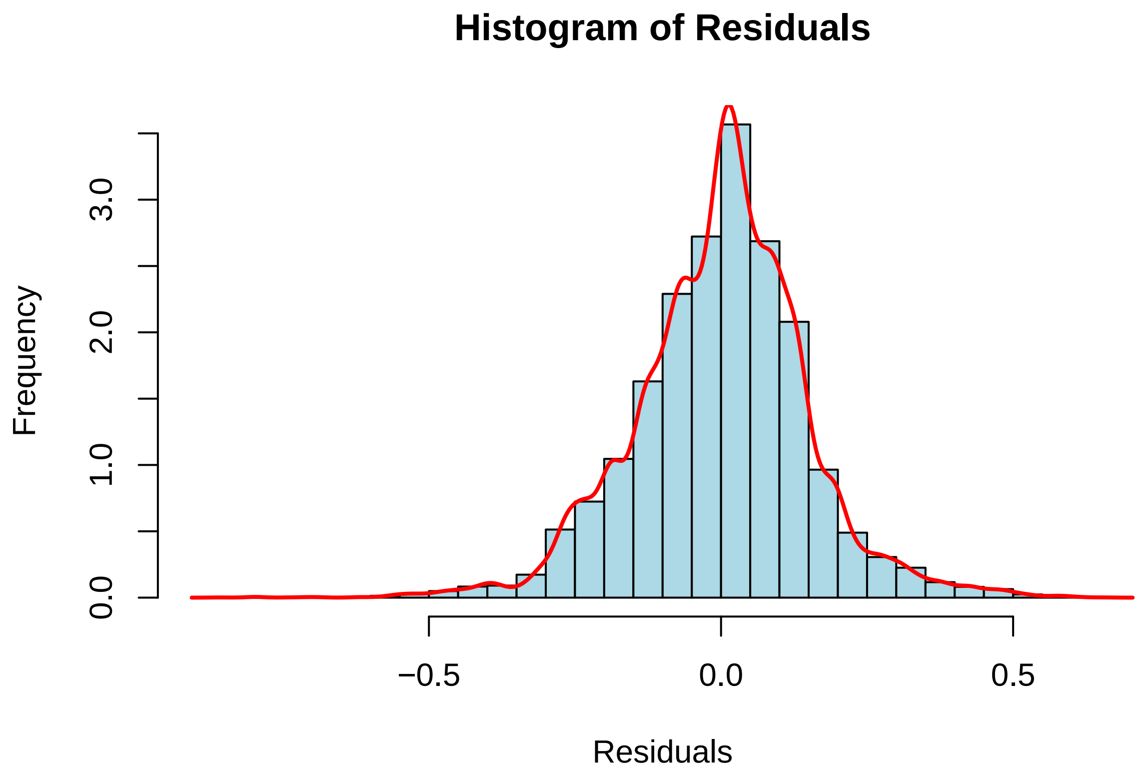
```r
cat("RMSE:", rmse_business, "\n")
```

```
## RMSE: 7867.807
```

```r
cat("MAPE:", mape_business * 100, "%\n")
```

```
## MAPE: 11.16284 %
```

```r
# Create a histogram of the residuals
hist(residuals_mars_business, breaks = 30, main = "Histogram of Residuals",
     xlab = "Residuals", ylab = "Frequency", col = "lightblue", border = "black", freq = FALSE)

# Add a density curve
lines(density(residuals_mars_business), col = "red", lwd = 2)
```


Histogram of Residuals
Frequency
0.0
1.0
2.0
3.0
−0.5
0.0
0.5
Residuals