Assignment 6

Shinjon Ghosh

2025-02-21

Loading Libraries

```
library(MASS)
library(ggplot2)
library(randomForest)

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

library(xgboost)
library(lattice)
library(caret)
```

Setting seed for constant reproduction.

```
set.seed(1234)
```

Question 1. Simulate ten variables from standard normal distributions

```
n <- 100
X <- as.data.frame(matrix(rnorm(n * 10), nrow = n, ncol = 10))
colnames(X) <- paste0("X", 1:10)
```

Question 2. compute the mean parameter

```
mu <- 1 + 2 * X$X1 + X$X2 + 0.5 * X$X5 + 1.5 * X$X10
```

Question 3. Generate Y count response from poisson distribution

```
Y <- rpois(n, lambda = exp(mu))
```

Combining the data into a single data frame

```
data <- cbind(X, Y)
```

Question 4. Randomly splitting data into 80% training and 20% testing

```r
set.seed(123)
train_indices <- sample(1:n, size = 0.8 * n)
train_data <- data[train_indices, ]
test_data <- data[-train_indices, ]

# Separate predictors and response
X_train <- train_data[, -ncol(train_data)]
Y_train <- train_data$Y
X_test <- test_data[, -ncol(test_data)]
Y_test <- test_data$Y
```

Question 5. Fit Poisson regression model

```r
# Fit a Poisson regression model
poisson_model <- glm(Y_train ~ ., data = train_data, family = poisson())

# Check if the estimated coefficients match the actual ones
summary(poisson_model)
```

```
##
## Call:
## glm(formula = Y_train ~ ., family = poisson(), data = train_data)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  9.583e-01  8.674e-02  11.048   <2e-16 ***
## X1           2.064e+00  6.641e-02  31.083   <2e-16 ***
## X2           9.893e-01  4.442e-02  22.270   <2e-16 ***
## X3          -7.493e-03  4.966e-02  -0.151    0.880
## X4           1.597e-02  3.682e-02   0.434    0.664
## X5           5.215e-01  4.691e-02  11.117   <2e-16 ***
## X6          -2.777e-02  5.106e-02  -0.544    0.587
## X7           7.147e-03  3.872e-02   0.185    0.854
## X8          -1.004e-02  5.512e-02  -0.182    0.855
## X9          -2.568e-02  6.014e-02  -0.427    0.669
## X10          1.503e+00  5.996e-02  25.068   <2e-16 ***
## Y           -7.318e-05  1.867e-04  -0.392    0.695
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 21135.837  on 79  degrees of freedom
## Residual deviance:    64.671  on 68  degrees of freedom
## AIC: 311.93
```

```
##
## Number of Fisher Scoring iterations: 5

# Make predictions on the test dataset
poisson_predictions <- predict(poisson_model, newdata = test_data, type =
"response")

comparison <- data.frame(Actual = Y_test, Predicted = poisson_predictions)
print(comparison)

##    Actual    Predicted
## 1       0  0.002244749
## 2       0  1.613400216
## 10      0  0.231407519
## 11      2  0.523272296
## 24     19 20.379697587
## 28      0  0.207373825
## 35      0  0.039345849
## 37      0  0.007540850
## 44      1  0.145468088
## 45      0  0.166715546
## 48      0  0.038117843
## 52      1  2.511995063
## 55      2  3.294772428
## 56     55 56.650014145
## 59     52 46.779681279
## 68     42 39.751403047
## 74      0  1.598799276
## 81      2  1.503605203
## 88      0  0.541938736
## 98     10 16.411241334
```

Question 6. Fitting a Negative Binomial regression

```
# Fit a negative binomial regression model
nb_model <- glm.nb(Y_train ~ ., data = train_data)

# Make predictions on the test dataset
nb_predictions <- predict(nb_model, newdata = test_data, type = "response")
```

Question 7. Fit a Random forest model

```
# Fit a random forest model
rf_model <- randomForest(Y_train ~ ., data = train_data)

# Make predictions on the test dataset
rf_predictions <- predict(rf_model, newdata = test_data)
```

Question 8. Fit an Extreme Gradient Boosting (XGBoost) model

```
# Prepare the data for XGBoost
dtrain <- xgb.DMatrix(data = as.matrix(X_train), label = Y_train)
dtest <- xgb.DMatrix(data = as.matrix(X_test))

# Define parameters
params <- list(objective = "count:poisson", eval_metric = "rmse")

# Train the XGBoost model
xgb_model <- xgb.train(params, dtrain, nrounds = 100)

# Make predictions on the test dataset
xgb_predictions <- predict(xgb_model, dtest)
```

Question 9. Print Predictions

```
print("poisson_predictions:")
```

```
## [1] "poisson_predictions:"
```

```
print(poisson_predictions)
```

```
##            1            2           10           11           24
28
##  0.002244749  1.613400216  0.231407519  0.523272296 20.379697587
0.207373825
##           35           37           44           45           48
52
##  0.039345849  0.007540850  0.145468088  0.166715546  0.038117843
2.511995063
##           55           56           59           68           74
81
##  3.294772428 56.650014145 46.779681279 39.751403047  1.598799276
1.503605203
##           88           98
##  0.541938736 16.411241334
```

```
print("Negative Binomial Predictions:")
```

```
## [1] "Negative Binomial Predictions:"
```

```
print(nb_predictions)
```

```
##            1            2           10           11           24
28
##  0.002245044  1.613460527  0.231435680  0.523337240 20.378061811
0.207380552
```

```
##           35           37           44           45           48
52
##  0.039348839  0.007541027  0.145486252  0.166740953  0.038119923
2.512062692
##           55           56           59           68           74
81
##  3.294717444 56.651194337 46.777997461 39.752393337  1.598829043
1.503611990
##           88           98
##  0.541974457 16.410810316

print("Random Forest Predictions:")

## [1] "Random Forest Predictions:"

print(rf_predictions)

##          1          2         10         11         24         28
35
##   4.423933  15.789433   0.871800  23.602033  19.962933   8.176167
15.008600
##         37         44         45         48         52         55
56
## 143.873900   2.374367   1.476500   8.401700   3.004433   7.026933
42.828033
##         59         68         74         81         88         98
##  98.278733  41.845733  17.810533  48.308400  14.719633  38.575800

print("XGBoost Predictions:")

## [1] "XGBoost Predictions:"

print(xgb_predictions)

##  [1]   0.02093426   3.66479945   0.06539036   0.53622389   7.45058250
##  [6]   0.02909360   0.05764790   0.88766617   0.32350263   0.03736582
## [11]   0.04981174   1.33992612   4.89910030  23.39458656 118.60705566
## [16]  77.54990387  28.61275291   1.45776331   3.17605925 982.90637207
```