Homework 4

Group-1 [Prashanna Raj Pandit | Asha Shah| Nazma Vali Shaik | Hema Sai Paruchuri]

2025-04-28

Phase 1: Multiple Linear Regression

Pre-procesing:

```
absent = read_excel("Absenteeism_at_work.xls")
# Convert relevant variables to factors BEFORE modeling
absent$Month_of_absence <- as.factor(absent$Month_of_absence)
absent$Day_of_the_week <- factor(absent$Day_of_the_week,
                                 levels = 2:6,
                                 labels = c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday"))
absent$Seasons <- factor(absent$Seasons,
                         levels = 1:4,
                         labels = c("Summer", "Autumn", "Winter", "Spring"))
absent$Education <- factor(absent$Education,
                           levels = 1:4,
                           labels = c("High School", "Graduate", "Postgraduate", "Master/Doctor"))
#str(absent)
```

Question 1: Split the data into training and test set. How did you do your data split?

```
set.seed(123)
train_size <- floor(0.7 * nrow(absent))
train_indices <- sample(seq_len(nrow(absent)), size = train_size)

# Split outcome and predictors
x <- model.matrix(Absenteeism_time_in_hours ~ . - ID, data = absent)[, -1]  # Remove intercept
y <- absent$Absenteeism_time_in_hours

x_train <- x[train_indices, ]
x_test <- x[-train_indices, ]
y_train <- y[train_indices]
y_test <- y[-train_indices]

table(absent$Month_of_absence)

##
##  0  1  2  3  4  5  6  7  8  9 10 11 12
##  3 50 72 87 53 64 54 67 54 53 71 63 49
```

Interpretation
We split the dataset into 70% training and 30% testing sets using random sampling with a fixed seed for reproducibility. The categorical variables were converted into factors before splitting.

Question 2: Fit a Lasso regression model in R using the glmnet package using one choice of alpha. Report the test error.

Answer: Lasso Test MSE: 185.358

```
set.seed(123)
cv.out.lasso <- cv.glmnet(x_train, y_train, alpha = 1, nlambda = 1000)
bestlam_lasso <- cv.out.lasso$lambda.min
lasso.mod <- glmnet(x_train, y_train, alpha = 1, lambda = bestlam_lasso)
lasso.pred <- predict(lasso.mod, s = bestlam_lasso, newx = x_test)
lasso.error <- mean((lasso.pred - y_test)^2)
print(paste("Lasso Test MSE:", lasso.error))

## [1] "Lasso Test MSE: 185.358460763272"
```

Question 3: Perform Ridge regression on the same dataset using one choice of alpha. Report the test error.

Answer: Lasso Test MSE: 181.5083

```
# Alpha = 0 for Ridge
set.seed(123)
cv.out.ridge <- cv.glmnet(x_train, y_train, alpha = 0, nfolds = 10, nlambda = 1000)
bestlam_ridge <- cv.out.ridge$lambda.min
ridge.mod <- glmnet(x_train, y_train, alpha = 0, lambda = bestlam_ridge)
ridge.pred <- predict(ridge.mod, s = bestlam_ridge, newx = x_test)
ridge.error <- mean((ridge.pred - y_test)^2)
print(paste("Ridge Test MSE:", ridge.error))

## [1] "Ridge Test MSE: 181.508359466857"
```

Question 4: Now fit an Elastic Net model to the data using your own choice of hyper parameters. Report the test error.

Answer: Elastic Net Test MSE: 184.945

```
set.seed(123)
cv.out.elastic <- cv.glmnet(x_train, y_train, alpha = 0.5, nfolds = 10, nlambda = 1000)
bestlam_elastic <- cv.out.elastic$lambda.min
elastic.mod <- glmnet(x_train, y_train, alpha = 0.5, lambda = bestlam_elastic)
elastic.pred <- predict(elastic.mod, s = bestlam_elastic, newx = x_test)
elastic.error <- mean((elastic.pred - y_test)^2)
print(paste("Elastic Net Test MSE:", elastic.error))

## [1] "Elastic Net Test MSE: 184.945144359402"
```

Question 5: Use cross-validation to select optimal values of alpha and or lambda in each of the methods in 2-4. Report the optimal hyper parameter values you used for these methods in a Table.

Optimal Hyperparameters for Lasso, Ridge, and Elastic Net (Phase 1)

Model	Alpha	Lambda
Lasso	1.00	0.2808118
Ridge	0.00	9.3527189
Elastic Net (tuned)	0.05	4.3287613

```r
library(glmnet)

# Define grid for alpha and lambda
alpha_grid <- seq(0, 1, by = 0.05)
lambda_grid <- 10^seq(3, -3, length = 100)

cv_errors <- matrix(NA, length(alpha_grid), 2)

# Cross-validation over alpha
set.seed(123)
for (i in 1:length(alpha_grid)) {
  alpha_val <- alpha_grid[i]

  cv_model <- cv.glmnet(x_train, y_train, alpha = alpha_val, lambda = lambda_grid, nfolds = 10)

  cv_errors[i, ] <- c(cv_model$lambda.min, min(cv_model$cvm))
}

# Find best alpha and lambda
best_index <- which.min(cv_errors[, 2])
best_alpha <- alpha_grid[best_index]
best_lambda <- cv_errors[best_index, 1]

cat("Best alpha:", best_alpha, "\n")
```

```
## Best alpha: 0.05
```

```r
cat("Best lambda:", best_lambda, "\n")
```

```
## Best lambda: 4.328761
```

```r
# Fit Elastic Net model with best alpha and lambda
elastic_best_model <- glmnet(x_train, y_train, alpha = best_alpha, lambda = best_lambda)

# Predict and calculate Test MSE
elastic_best_pred <- predict(elastic_best_model, s = best_lambda, newx = x_test)
elastic_best_mse <- mean((elastic_best_pred - y_test)^2)
```

```
cat("Elastic Net Test MSE:", elastic_best_mse, "\n")

## Elastic Net Test MSE: 182.5254

# Collect all test errors
error_table <- data.frame(
  Model = c("Lasso", "Ridge", "Elastic Net (tuned)"),
  Alpha = c(1, 0, best_alpha),
  Lambda = c(bestlam_lasso, bestlam_ridge, best_lambda),
  Test_MSE = c(lasso.error, ridge.error, elastic_best_mse)
)
print(error_table)

##                 Model Alpha    Lambda Test_MSE
## 1               Lasso  1.00 0.2808118 185.3585
## 2               Ridge  0.00 9.3527189 181.5084
## 3 Elastic Net (tuned)  0.05 4.3287613 182.5254
```

Question 6: Tabulate the test error for each of these models in 5 and compare with their corresponding models you fit in 2,3,4. What does this tell you about the model's performance?

Comparison of Test MSE for Original and Tuned Models (Phase 1)

Model	Test_MSE_Original	Test_MSE_Tuned
Lasso	185.3580	185.3585
Ridge	181.5083	181.5084
Elastic Net	184.9450	182.5254

Interpretation

Lasso Regression: The test error remained almost identical before and after tuning (185.36), indicating that cross-validation did not significantly improve the model.

Ridge Regression: The test error also remained essentially the same (~181.51), suggesting that the Ridge model was already close to optimal without further tuning.

Elastic Net Regression: Tuning improved the test error slightly, from 184.95 down to 182.53, although the improvement was modest.

Conclusion: The Ridge Regression model consistently achieved the lowest test error across both original and tuned models, confirming its superior predictive performance on this dataset.

Question 7: For the models in 5, which one will you choose as the final model based on the test errors?

Based on the test errors from the models fitted in Question 5, I would choose the **Ridge Regression** model as the final model. It achieved the lowest test MSE (181.51) compared to Lasso (185.36) and Elastic Net (182.53), indicating the best predictive performance on the test set. The Ridge model's ability to retain all features while shrinking their coefficients made it more effective for capturing the small, distributed effects present in the absenteeism data.

Question 8: Describe your next steps in the modeling process now that you have selected your final model from 7.

Next steps:

- **Model Evaluation and Refinement**: Test performance on additional datasets to confirm robustness.
- **Model Interpretability**: Analyze coefficients of the Ridge model to identify important predictors.
- **Deployment**: Deploy the model for real-time predictions and decision-making.
- **Monitoring**: Continuously monitor the model's performance and update it as needed.

Question 9: Based on the final model's output, which factors are most predictive of absenteeism in the workplace? How did you decide on those features?

```r
# Extract coefficients of the final Ridge model
final_ridge_model <- glmnet(x_train, y_train, alpha = 0, lambda = bestlam_ridge)
coef(final_ridge_model)
```

```
## 33 x 1 sparse Matrix of class "dgCMatrix"
##                                    s0
## (Intercept)              -1.732223e+01
## Month_of_absence1         -1.781113e+00
## Month_of_absence2         -1.311031e+00
## Month_of_absence3          1.895338e+00
## Month_of_absence4          3.030312e-01
## Month_of_absence5         -1.437234e+00
## Month_of_absence6          2.895357e-01
## Month_of_absence7          2.002192e+00
## Month_of_absence8         -3.484622e-01
## Month_of_absence9         -6.606962e-01
## Month_of_absence10        -1.435256e+00
## Month_of_absence11        -6.201665e-01
## Month_of_absence12         2.407882e+00
## Day_of_the_weekTuesday     6.171047e-01
## Day_of_the_weekWednesday       -2.806411e-02
## Day_of_the_weekThursday        -1.943534e+00
## Day_of_the_weekFriday          -2.294788e+00
## SeasonsAutumn                  -3.728258e-02
## SeasonsWinter                   7.899793e-01
## SeasonsSpring                  -3.308433e-01
## Transportation_expense          6.884381e-03
## Distance_from_Residence_to_Work -4.191029e-02
## Service_time                    4.327130e-02
## Age                             1.685968e-01
## Work_load_Average_in_days       3.147002e-06
## EducationGraduate              -5.889970e-01
## EducationPostgraduate          -1.379527e+00
## EducationMaster/Doctor         -1.419080e+00
## Son                             5.911816e-01
## Pet                            -8.514763e-02
## Weight                         -9.119567e-04
## Height                          1.113878e-01
## Body_mass_index                -9.336706e-02
```

Key Predictors

Positive Predictors (Increase Absenteeism):

- **Month_of_absence12 (December)**: Strongest predictor of absenteeism (+2.4079).
- **Month_of_absence7 (July)** and **Month_of_absence3 (March)**: Higher absenteeism (+2.0022, +1.8953).
- **Age**: Older employees slightly more likely to be absent (+0.1686).
- **Son**: More children increase absenteeism (+0.5912).
- **SeasonsWinter:** Higher absenteeism observed in winter (+0.7899).

Negative Predictors (Decrease Absenteeism):

- **Day_of_the_weekFriday:** Strong reduction in absenteeism (-2.2948).
- **Day_of_the_weekThursday:** Reduced absenteeism (-1.9435).
- **Month_of_absence10 (October) and Month_of_absence5 (May):** Lower absenteeism (-1.4353, -1.4372).
- **EducationPostgraduate and EducationMaster/Doctor:** Higher education levels associated with less absenteeism (-1.3795, -1.4191).

How These Were Identified:

- **Magnitude of Coefficients:** I selected the most important predictors based on the absolute size of their coefficients in the final Ridge Regression model; larger magnitudes indicate a stronger effect on absenteeism.

- **Sign of Coefficients:** Positive coefficients indicate variables that increase absenteeism, while negative coefficients indicate variables that decrease absenteeism.

Question 10: Discuss the potential implications of your findings for the management of the company.

Key Implications for Management

1. **Workload Management**: Redistribute tasks to reduce stress and absenteeism.
2. **Seasonal Trends**: Plan resources for months with high absenteeism.
3. **Transportation**: Address commuting challenges with remote work or reimbursements.
4. **Employee Demographics**: Offer flexible work hours for employees with family responsibilities.
5. **Health Programs**: Introduce wellness initiatives to improve employee well-being.

Recommendations

- Use predictive insights for staffing and policy adjustments.
- Invest in employee engagement programs for long-term retention.

Question 11: How might the company use the insights from your final model to reduce absenteeism rates?

Key Actions for Management

1. **Targeted Interventions**:
    - Flexible schedules for high-risk employees (e.g., those with family responsibilities).
    - Redistribute workloads to reduce stress.
2. **Seasonal Staffing**:
    - Address absenteeism spikes in months like December and July by planning temporary staffing.
3. **Wellness Programs**:
    - Health initiatives to address absenteeism linked to BMI and other health factors.
4. **Remote Work Options**:
    - Offer remote or hybrid work to mitigate commuting-related absenteeism.

Summary

The company can proactively use these insights to optimize staffing, improve employee well-being, and reduce absenteeism rates effectively.

Phase 2: Logistic Regression

1. Prepare and split Dataset

```r
# Already recoded previously
absent <- absent %>%
  mutate(absenteeism = case_when(
    Absenteeism_time_in_hours >= 0 & Absenteeism_time_in_hours <= 20 ~ "Low",
    Absenteeism_time_in_hours > 20 & Absenteeism_time_in_hours <= 40 ~ "Moderate",
    Absenteeism_time_in_hours > 40 ~ "High"
  ))
absent$absenteeism <- factor(absent$absenteeism, levels = c("Low", "Moderate", "High"))
#View(absent)

set.seed(123)
train_size <- floor(0.7 * nrow(absent))
train_indices <- sample(seq_len(nrow(absent)), size = train_size)

# Predictors
x <- model.matrix(absenteeism ~ . - ID - Absenteeism_time_in_hours, data = absent)[, -1]
y <- absent$absenteeism

x_train <- x[train_indices, ]
x_test <- x[-train_indices, ]
y_train <- y[train_indices]
y_test <- y[-train_indices]
```

---

2. Lasso Logistic Regression (alpha = 1)

```r
set.seed(123)
cv.out.lasso.cat <- cv.glmnet(x_train, y_train, alpha = 1, family = "multinomial", type.measure = "class")
bestlam_lasso_cat <- cv.out.lasso.cat$lambda.min
lasso.mod.cat <- glmnet(x_train, y_train, alpha = 1, lambda = bestlam_lasso_cat, family = "multinomial")

# Predict on test set
lasso.pred.cat <- predict(lasso.mod.cat, newx = x_test, s = bestlam_lasso_cat, type = "class")

# Classification Error
lasso.error.cat <- mean(lasso.pred.cat != y_test)
print(paste("Lasso Classification Error:", lasso.error.cat))

## [1] "Lasso Classification Error: 0.0630630630630631"
```

3.Ridge Logistic Regression (alpha = 0)

```r
set.seed(123)
cv.out.ridge.cat <- cv.glmnet(x_train, y_train, alpha = 0, family = "multinomial", type.measure = "class")
bestlam_ridge_cat <- cv.out.ridge.cat$lambda.min
```

```
ridge.mod.cat <- glmnet(x_train, y_train, alpha = 0, lambda = bestlam_ridge_cat, family =
"multinomial")

ridge.pred.cat <- predict(ridge.mod.cat, newx = x_test, s = bestlam_ridge_cat, type = "class")

ridge.error.cat <- mean(ridge.pred.cat != y_test)
print(paste("Ridge Classification Error:", ridge.error.cat))

## [1] "Ridge Classification Error: 0.0630630630630631"
```

---

4. Elastic Net Logistic Regression (alpha = 0.5)

```
set.seed(123)
cv.out.elastic.cat <- cv.glmnet(x_train, y_train, alpha = 0.5, family = "multinomial", type.measure =
"class")
bestlam_elastic_cat <- cv.out.elastic.cat$lambda.min
elastic.mod.cat <- glmnet(x_train, y_train, alpha = 0.5, lambda = bestlam_elastic_cat, family =
"multinomial")

elastic.pred.cat <- predict(elastic.mod.cat, newx = x_test, s = bestlam_elastic_cat, type = "class")

elastic.error.cat <- mean(elastic.pred.cat != y_test)
print(paste("Elastic Net Classification Error:", elastic.error.cat))

## [1] "Elastic Net Classification Error: 0.0630630630630631"
```

Observation:

The Lasso, Ridge, and Elastic Net models all achieved the same classification error of approximately 6.3% on the test set. This suggests that in this dataset, the choice of regularization (whether Lasso, Ridge, or Elastic Net) did not significantly impact the model's predictive performance. This could be due to the dataset being relatively simple to classify, or key predictors being consistently selected across all models.

---

5. Elastic Net Tuning (Cross-Validation for Alpha and Lambda)

We performed cross-validation to select the optimal hyperparameter values for Lasso, Ridge, and Elastic Net models.

- For **Lasso Regression** (alpha = 1), cross-validation was used to select the best value of lambda.
- For **Ridge Regression** (alpha = 0), cross-validation was used to select the best value of lambda.
- For **Elastic Net Regression**:
    - First, we fixed alpha = 0.5 and tuned lambda.
    - Then, we performed a full grid search over both alpha (0 to 1 by 0.1 steps) and lambda (logarithmic grid) to find the optimal combination.

Optimal Hyperparameters for Each Model

Model	Alpha	Lambda
Lasso	1.0	0.027637
Ridge	0.0	27.637020
Elastic Net (fixed alpha=0.5)	0.5	0.055274
Elastic Net (tuned)	0.0	1000.000000

```r
# Grid search over alpha
alpha_grid <- seq(0, 1, by = 0.1)
lambda_grid <- 10^seq(3, -3, length = 100)

cv_errors_cat <- matrix(NA, length(alpha_grid), 2)

set.seed(123)
for (i in 1:length(alpha_grid)) {
  alpha_val <- alpha_grid[i]

  cv_model <- cv.glmnet(x_train, y_train, alpha = alpha_val, family = "multinomial", type.measure =
"class", lambda = lambda_grid)

  cv_errors_cat[i, ] <- c(cv_model$lambda.min, min(cv_model$cvm))
}
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```r
best_index_cat <- which.min(cv_errors_cat[,2])
best_alpha_cat <- alpha_grid[best_index_cat]
best_lambda_cat <- cv_errors_cat[best_index_cat,1]

cat("Best alpha (categorical):", best_alpha_cat, "\n")
```

```
## Best alpha (categorical): 0
```

```r
cat("Best lambda (categorical):", best_lambda_cat, "\n")
```

```
## Best lambda (categorical): 1000
```

```
# Final tuned Elastic Net Model
elastic_best_model_cat <- glmnet(x_train, y_train, alpha = best_alpha_cat, lambda = best_lambda_cat,
family = "multinomial")

elastic_best_pred_cat <- predict(elastic_best_model_cat, s = best_lambda_cat, newx = x_test, type =
"class")

elastic_best_error_cat <- mean(elastic_best_pred_cat != y_test)

cat("Elastic Net Tuned Classification Error:", elastic_best_error_cat, "\n")

## Elastic Net Tuned Classification Error: 0.06306306

# Tabulate all results
error_table_cat <- data.frame(
  Model = c("Lasso", "Ridge", "Elastic Net (fixed alpha=0.5)", "Elastic Net (tuned)"),
  Alpha = c(1, 0, 0.5, best_alpha_cat),
  Lambda = c(bestlam_lasso_cat, bestlam_ridge_cat, bestlam_elastic_cat, best_lambda_cat),
  Classification_Error = c(lasso.error.cat, ridge.error.cat, elastic.error.cat, elastic_best_error_cat)
)

print(error_table_cat)

##                           Model Alpha       Lambda Classification_Error
## 1                         Lasso   1.0 2.763702e-02           0.06306306
## 2                         Ridge   0.0 2.763702e+01           0.06306306
## 3 Elastic Net (fixed alpha=0.5)   0.5 5.527405e-02           0.06306306
## 4           Elastic Net (tuned)   0.0 1.000000e+03           0.06306306
```

6. Model comparison and evaluation

The table below summarizes the optimal hyperparameters and classification errors for each model:

Comparison of Classification Errors for Original and Tuned Models

Model	Classification_Error_Original	Classification_Error_Tuned
Lasso	0.0630631	0.0630631
Ridge	0.0630631	0.0630631
Elastic Net	0.0630631	0.0630631

Interpretation:

- All models — Lasso, Ridge, Elastic Net (fixed alpha = 0.5), and Elastic Net (tuned) — achieved the same classification error of approximately 6.31% on the test set.
- This indicates that tuning alpha and lambda did not lead to significant performance improvements in terms of classification accuracy.

- The fact that Elastic Net tuning chose an alpha value of 0 (essentially Ridge behavior) suggests that Ridge-style regularization fits this data structure well.
- Since all models performed similarly, Lasso Regression is preferred following the parsimony principle, because it leads to a simpler, more interpretable model with potentially fewer nonzero coefficients.

7. Selection of Final Model Based on Test Errors

All models (Lasso, Ridge, and Elastic Net) achieved very similar classification errors (~6.3%). The differences in performance across Lasso, Ridge, and Elastic Net were very small.

Following the principle of parsimony, it is best to choose the simplest model that achieves similar performance.

Thus, we select **Lasso Regression (alpha = 1) as the final model** because it produces a simpler and more interpretable model by setting some coefficients exactly to zero, while still maintaining a high level of predictive accuracy.

This allows for better understanding of the important predictors of absenteeism and facilitates easier model deployment.

---

8. Next Steps in the Modeling Process

After selecting the Ridge Regression model, the following next steps would be taken:

- **Model Refinement**:
  Explore possible interactions or non-linear effects between variables to further improve performance.
- **Threshold Adjustment**:
  Evaluate different classification thresholds (not just 0.5) to optimize sensitivity/specificity trade-offs for business priorities.
- **Feature Engineering**:
  Create new variables or transformations (e.g., group similar education levels) to potentially enhance predictive power.
- **Deployment**:
  Finalize the model, validate on any additional hold-out datasets, and implement the model for predicting absenteeism risks in operational settings.

---

9. Most Predictive Factors for Absenteeism

```r
final_ridge.cat <- cv.glmnet(x_train, y_train, alpha = 0, family = "multinomial", type.measure = "class")
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
coef(final_ridge.cat)
```

```
## $Low
## 33 x 1 sparse Matrix of class "dgCMatrix"
##                                          1
## (Intercept)                   2.360771e+00
## Month_of_absence1             3.360003e-38
## Month_of_absence2             6.406560e-38
## Month_of_absence3            -4.129956e-38
## Month_of_absence4            -3.867315e-38
## Month_of_absence5             5.759348e-39
## Month_of_absence6            -3.426323e-38
## Month_of_absence7            -6.570653e-39
## Month_of_absence8             2.870786e-38
## Month_of_absence9            -3.858025e-38
## Month_of_absence10            6.475006e-38
## Month_of_absence11            1.131766e-38
## Month_of_absence12           -4.348852e-38
## Day_of_the_weekTuesday       -2.764487e-38
## Day_of_the_weekWednesday     -1.741553e-38
## Day_of_the_weekThursday       7.096728e-38
## Day_of_the_weekFriday         6.287279e-38
## SeasonsAutumn                 1.987842e-38
## SeasonsWinter                -4.554689e-38
## SeasonsSpring                 9.900685e-39
## Transportation_expense       -2.325917e-41
## Distance_from_Residence_to_Work  1.293657e-39
## Service_time                 -2.839811e-39
## Age                          -3.859311e-39
## Work_load_Average_in_days    -4.514145e-43
## EducationGraduate            -2.903133e-39
## EducationPostgraduate         6.530825e-38
## EducationMaster/Doctor        5.872680e-38
## Son                          -8.476924e-39
## Pet                           1.131771e-38
## Weight                       -6.942842e-40
## Height                       -4.444532e-39
## Body_mass_index               8.844336e-40
##
## $Moderate
## 33 x 1 sparse Matrix of class "dgCMatrix"
##                                          1
## (Intercept)                  -8.338120e-01
## Month_of_absence1            -1.260001e-38
## Month_of_absence2            -4.271040e-38
## Month_of_absence3             2.753304e-38
## Month_of_absence4             3.432118e-38
## Month_of_absence5            -1.340156e-38
## Month_of_absence6               3.103623e-38
## Month_of_absence7              -1.925566e-38
## Month_of_absence8              -7.923889e-39
## Month_of_absence9               5.936422e-38
## Month_of_absence10             -4.316671e-38
## Month_of_absence11              9.902951e-39
## Month_of_absence12             -1.559802e-38
## Day_of_the_weekTuesday         -1.077763e-38
## Day_of_the_weekWednesday        1.928148e-38
## Day_of_the_weekThursday        -4.731152e-38
## Day_of_the_weekFriday          -3.805458e-38
## SeasonsAutumn                  -1.325228e-38
## SeasonsWinter                   3.036460e-38
## SeasonsSpring                  -3.274546e-39
## Transportation_expense          1.607139e-40
## Distance_from_Residence_to_Work -6.563490e-42
## Service_time                    1.363324e-39
## Age                             1.008385e-39
## Work_load_Average_in_days       5.692575e-43
## EducationGraduate               2.372996e-38
## EducationPostgraduate          -4.353884e-38
## EducationMaster/Doctor         -3.915120e-38
## Son                             1.009584e-39
## Pet                            -6.266221e-39
## Weight                          6.548647e-41
## Height                          1.727316e-39
## Body_mass_index                -9.914966e-40
##
## $High
## 33 x 1 sparse Matrix of class "dgCMatrix"
##                                            1
## (Intercept)                    -1.526959e+00
## Month_of_absence1              -2.100002e-38
## Month_of_absence2              -2.135520e-38
## Month_of_absence3               1.376652e-38
## Month_of_absence4               4.351965e-39
## Month_of_absence5               7.642212e-39
## Month_of_absence6               3.227008e-39
## Month_of_absence7               2.582632e-38
## Month_of_absence8              -2.078397e-38
## Month_of_absence9              -2.078397e-38
## Month_of_absence10             -2.158335e-38
## Month_of_absence11             -2.122061e-38
## Month_of_absence12              5.908654e-38
## Day_of_the_weekTuesday          3.842250e-38
## Day_of_the_weekWednesday       -1.865950e-39
## Day_of_the_weekThursday        -2.365576e-38
## Day_of_the_weekFriday          -2.481821e-38
## SeasonsAutumn                  -6.626139e-39
## SeasonsWinter                   1.518230e-38
## SeasonsSpring                  -6.626139e-39
## Transportation_expense         -1.374547e-40
## Distance_from_Residence_to_Work -1.287093e-39
## Service_time                    1.476487e-39
## Age                             2.850926e-39
## Work_load_Average_in_days      -1.178430e-43
## EducationGraduate              -2.082682e-38
## EducationPostgraduate          -2.176942e-38
## EducationMaster/Doctor         -1.957560e-38
## Son                             7.467339e-39
## Pet                            -5.051493e-39
## Weight                          6.287977e-40
## Height                          2.717216e-39
## Body_mass_index                 1.070630e-40
```

We extracted the coefficients from the final Ridge Logistic Regression model (alpha = 0, family = “multinomial”) for each absenteeism category: “Low”, “Moderate”, and “High”.

Since we are primarily interested in predicting **High absenteeism**, we focused on the coefficients from the “High” class.

Key Predictors for High Absenteeism:

- Positive Predictors (increase the likelihood of High absenteeism):
  - **Month of Absence 12 (December):** Strong positive effect (+5.9087e-38)
  - **Month of Absence 7 (July):** Positive effect (+2.5826e-38)
  - **Day of the Week Tuesday:** Positive effect (+3.8422e-38)
  - **Seasons Winter:** Positive effect (+1.5182e-38)
  - **Age:** Slight positive effect (+2.8509e-39)
- Negative Predictors (decrease the likelihood of High absenteeism):
  - **Day of the Week Friday:** Negative effect (-2.4818e-38)
  - **Day of the Week Thursday:** Negative effect (-2.3656e-38)
  - **Month of Absence 10 (October):** Negative effect (-2.1583e-38)
  - **Education (Graduate, Postgraduate, Master/Doctor):** Higher education levels associated with lower absenteeism.

How These Were Identified:

- **Magnitude of Coefficients:**
  We selected important predictors based on the absolute size of their coefficients in the final model. Larger magnitudes indicate a stronger influence on absenteeism risk.

- **Sign of Coefficients:**

  - Positive coefficients indicate an increase in the probability of being classified into the “High” absenteeism category.
  - Negative coefficients indicate a decrease in that probability.

Thus, factors such as **month of absence, day of the week, season, and education level** are key drivers for predicting high absenteeism risk in the workplace.

12. Comparison of Models Based on AUC and F1 Score

```
library(pROC)
library(caret)

get_metrics <- function(true_labels, pred_labels, pred_probs, positive_class = "High") {
  pred_labels <- factor(pred_labels, levels = levels(true_labels))

  auc_val <- multiclass.roc(true_labels, pred_probs)$auc
  confusion <- confusionMatrix(pred_labels, true_labels)
  f1_val <- mean(confusion$byClass[, "F1"], na.rm = TRUE)  # Average across all classes

  return(c(AUC = auc_val, F1 = f1_val))
}



# Predict class probabilities
lasso.prob.cat <- predict(lasso.mod.cat, newx = x_test, s = bestlam_lasso_cat, type = "response")
ridge.prob.cat <- predict(ridge.mod.cat, newx = x_test, s = bestlam_ridge_cat, type = "response")
elastic.prob.cat <- predict(elastic.mod.cat, newx = x_test, s = bestlam_elastic_cat, type = "response")

# Convert probabilities into data frames
lasso.prob.df <- as.data.frame(lasso.prob.cat)
ridge.prob.df <- as.data.frame(ridge.prob.cat)
elastic.prob.df <- as.data.frame(elastic.prob.cat)

# Check column names (they should be "Low", "Moderate", "High")
colnames(lasso.prob.df)
```

```
## [1] "Low.1"      "Moderate.1" "High.1"
```

```
# Use the correct column for "High" class

lasso.prob.high <- lasso.prob.df$High.1
ridge.prob.high <- ridge.prob.df$High.1
elastic.prob.high <- elastic.prob.df$High.1

# Calculate metrics
lasso_metrics <- get_metrics(y_test, lasso.pred.cat, lasso.prob.high)
```

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

```
ridge_metrics <- get_metrics(y_test, ridge.pred.cat, ridge.prob.high)
```

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

## Setting direction: controls > cases

elastic_metrics <- get_metrics(y_test, elastic.pred.cat, elastic.prob.high)

## Setting direction: controls < cases

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

# Create a table
metrics_table <- data.frame(
  Model = c("Lasso", "Ridge", "Elastic Net (alpha=0.5)"),
  AUC = c(lasso_metrics["AUC"], ridge_metrics["AUC"], elastic_metrics["AUC"]),
  F1_Score = c(lasso_metrics["F1"], ridge_metrics["F1"], elastic_metrics["F1"])
)

print(metrics_table)

##                     Model       AUC  F1_Score
## 1                   Lasso 0.5000000 0.9674419
## 2                   Ridge 0.6168981 0.9674419
## 3 Elastic Net (alpha=0.5) 0.5000000 0.9674419
```

Comparison of Models: AUC and F1 Score

Model	AUC	F1_Score
Lasso	0.5000	0.9674
Ridge	0.6169	0.9674
Elastic Net (alpha=0.5)	0.5000	0.9674

Interpretation:

- F1 Score: All three models achieved a very high F1 Score (~0.9674), indicating strong balance between precision and recall for classifying absenteeism categories.
- AUC (Area Under the Curve): Ridge Logistic Regression achieved a higher AUC (0.6169) compared to Lasso (0.5000) and Elastic Net (0.5000). AUC values of 0.5 indicate random guessing, whereas higher AUC values reflect better class separation. Thus, Ridge performs better at distinguishing between absenteeism categories.

Conclusion:

Based on the comparison:

- While F1 Scores are very similar across all models,
- Ridge Logistic Regression achieves the best AUC.

- Therefore, Ridge Regression is the best model to choose, as it offers both high classification accuracy and better discrimination between classes.