Predicting 30-Year Survival in the Framingham Cohort: Classical and Regularized Logistic Models

Alan Lin

August 29 2025

Motivation

The Framingham Heart Study is a well-known, ongoing study of chronic circulatory disease. Data have been collected every two years to monitor indicators such as cholesterol levels, smoking habits, and blood pressure, which are associated with coronary heart disease. This project compares classical logistic regression, logistic regression with an L_1 penalty (LASSO), and logistic regression with an L_2 penalty (Ridge) in predicting 30-year survival after enrollment. Additionally, subset selection methods are applied to examine how a reduced model compares to the regularized and classical logistic models in terms of predictive performance. The goal is to demonstrate how modern statistical and machine learning techniques can be applied to health data to inform and potentially improve healthcare outcomes. For this project, we use a course-provided dataset to illustrate these methods; the focus is on model application, comparison, and interpretation rather than data collection.

Data Setup

Data Upload

We begin by loading the `framingham.csv` dataset into our work environment using `read.csv()` to ensure reproducibility. For model fitting and evaluation, we use R's `glmnet` library for regularized logistic regression, `leaps` for stepwise variable selection, `ggplot2` for data visualization, and `caret` for computing confusion matrices and other scoring metrics.

```r
# Load required libraries
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
library(leaps)
library(ggplot2)
library(caret)
```

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

```r
# Load dataset
data = read.csv("framingham.csv")

# Show first 5 entries of dataset
head(data, 5)
```

```
##   age sex BMI0 BMI10 cigarettes chol0 chol2 chol4 chol6 chol8 chol10 death
## 1  45   2   22    22          0   220   217   217   200   219    240     0
## 2  48   1   23    23          0   340   267   285   266   285    246     0
## 3  38   2   24    26          5   200   194   210    -9   231    222     0
## 4  50   1   27    28          0   224   225   232   235   235    246     0
## 5  41   1   21    23         20   200   209   207   220   213    225     0
```

Data Cleaning

We note the -9 value in the `chol6` column for the third row. In this dataset, elements with value -9 are considered missing data, so we will next replace these values with corresponding `NA` values and then drop any rows with missing data for further analysis. Replacing -9 with NA and removing incomplete rows ensures that our models are trained on accurate and complete observations.

```r
# Replace data entries with value -9 with NA
data[data == -9] = NA

# Assign cleaned dataframe to data_clean
data_clean = na.omit(data)

# Check if the cleaned dataframe has any NA values
anyNA(data_clean)
```

```
## [1] FALSE
```

```r
# Calculate and print number of rows removed during cleaning
n_removed = nrow(data) - nrow(data_clean)
cat("Number of rows removed due to missing values:", n_removed, "\n")
```

```
## Number of rows removed due to missing values: 911
```

```r
# Show first 5 entries of cleaned dataframe
head(data_clean, 5)
```

```
##   age sex BMI0 BMI10 cigarettes chol0 chol2 chol4 chol6 chol8 chol10 death
## 1  45   2   22    22          0   220   217   217   200   219    240     0
## 2  48   1   23    23          0   340   267   285   266   285    246     0
## 4  50   1   27    28          0   224   225   232   235   235    246     0
## 5  41   1   21    23         20   200   209   207   220   213    225     0
## 6  48   1   21    21         25   229   211   215   185   189    218     0
```

The dataset has 12 columns:

- `age` gives the age of the individual when they entered the study.

- `sex` provides the assigned-sex-at-birth of the individual (1 = male, 2 = female)
- `BMI0` provides the body mass index (BMI) of the individual initially
- `BMI10` provides the body mass index (BMI) of the individual 10 years after entering the study
- `cigarettes` provides the number of cigarettes the individual smoked per day initially
- `chol0`provides the individual's serum cholesterol levels initially
- `chol2` provides the individual's serum cholesterol levels 2 years into the study
- `chol4` provides the individual's serum cholesterol levels 4 years into the study
- `chol6` provides the individual's serum cholesterol levels 6 years into the study
- `chol8` provides the individual's serum cholesterol levels 8 years into the study
- `chol10` provides the individual's serum cholesterol levels 10 years into the study
- `death` indicates whether the individual is alive (0) or dead (1) at the end of 30 years since enrollment. The dataset excludes those who died during the decade-long data collection period.

Splitting Data into Train and Test Sets

In order to have a benchmark to reference and compare performances across the four models, we choose to randomly split the dataset into 80% training and 20% test observations.

```r
# Set seed for reproducibility
set.seed(22)

# Get number of rows in cleaned dataframe
n = nrow(data_clean)

# Randomly sample 80% of the row indices for the training dataset
train_indices = sample(seq_len(n), size = 0.8 * n)

# Use the 80% randomly selected indices to form the training dataset
train = data_clean[train_indices, ]

# Use the remaining 20% indices to form the testing dataset
test = data_clean[-train_indices, ]
```

We are now ready to move on to constructing models for prediction!

Logistic Regression with L1 Penalty

Our first model is logistic regression with an L_1 penalty, equivalent to applying LASSO to logistic regression. Coefficients are shrunk toward zero by penalizing their absolute values, similar to linear LASSO, but using the logistic loss function. LASSO also performs variable selection by driving some coefficients exactly to zero, helping identify the most important predictors. We will define a function that takes the dataset, the outcome variable (death), and a list of potential covariates, then returns the model chosen by LASSO logistic regression using 5-fold cross-validation and accuracy (minimizing misclassification error) as the criterion to select the optimal regularization parameter λ. We will then apply this function to our dataset and summarize the results.

```r
fit_logistic_l1_cv = function(data, outcome, covariates, nfolds=5) {
  set.seed(44) # ensures reproducibility of CV folds

  # Prepare response and predictor data
  y = data[[outcome]]
  X = data[, covariates]
  # Turn selected data into matrices
  X_mat = as.matrix(X)
  y_vec = as.numeric(as.character(y))

  # Perform cross-validated LASSO logistic regression
  cv_fit = cv.glmnet(
    X_mat, y_vec,
    family = "binomial",
    alpha = 1,                # L1 penalty
    nfolds = nfolds,
    type.measure = "class"  # min misclassification error
  )

  cat("Best lambda chosen by CV (min misclassification error):",
      cv_fit$lambda.min, "\n")

  # Fit final model using optimal lambda
  final_model = glmnet(X_mat, y_vec, family = "binomial", alpha = 1,
                       lambda = cv_fit$lambda.min)

  # Print selected coefficients
  cat("\nSelected coefficients at optimal lambda:\n")
  print(coef(final_model))

  return(final_model)
}

# Use function for our goal
outcome = "death"
covariates = names(data_clean)[!(names(data_clean) %in% outcome)]

# Apply optimal model for predictions
result_l1 = fit_logistic_l1_cv(train, outcome, covariates, nfolds=5)
```

```
## Best lambda chosen by CV (min misclassification error): 0.006509127
##
## Selected coefficients at optimal lambda:
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept) -5.9509900933
## age          0.1059812182
## sex         -0.2727152169
## BMI0         .
## BMI10        .
## cigarettes   0.0269800968
## chol0        0.0008736791
## chol2        .
## chol4        .
## chol6       -0.0006662772
## chol8       -0.0003445924
## chol10       .
```

The optimal lambda chosen by 5-fold cross-validation was approximately 0.0065. At this level of regularization, LASSO logistic regression shrinks some coefficients exactly to zero, effectively performing variable selection. According to the model, the most influential predictors (based on the magnitude of their coefficients) are:

1. Assigned sex at birth
2. Age of the individual
3. Number of cigarettes initially smoked per day
4. Initial serum cholesterol level
5. Serum cholesterol level 6 years into the study
6. Serum cholesterol level 8 years into the study

All other predictors were excluded from the model through penalization, indicating that LASSO considered them less relevant for predicting survival 30 years after enrollment.

Logistic Regression with L2 Penalty

Our second model is logistic regression with an L_2 penalty, commonly known as Ridge logistic regression. Coefficients are shrunk toward zero by penalizing their squared values, using the logistic loss function, but unlike LASSO, Ridge does not set any coefficients exactly to zero. This approach helps to reduce variance and mitigate multicollinearity while retaining all predictors. We will write a function that takes the dataset, outcome variable (death), and a list of potential covariates, then returns the Ridge logistic regression model chosen by 5-fold cross-validation, using misclassification error (accuracy) to select the optimal regularization parameter λ. We then apply this function to the dataset and print the results.

```r
fit_logistic_l2_cv = function(data, outcome, covariates, nfolds=5) {
  set.seed(44) # ensures reproducibility of CV folds

  # Prepare response and predictor data
  y = data[[outcome]]
  X = data[, covariates]

  # Turn selected data into matrices
  X_mat = as.matrix(X)
  y_vec = as.numeric(as.character(y))

  # Perform cross-validated LASSO logistic regression
  cv_fit = cv.glmnet(
    X_mat, y_vec,
    family = "binomial",
    alpha = 0,              # L2 penalty
    nfolds = nfolds,
    type.measure = "class"  # min misclassification error
  )

  cat("Best lambda chosen by CV (min misclassification error):",
      cv_fit$lambda.min, "\n")

  # Fit final model using optimal lambda
  final_model = glmnet(X_mat, y_vec, family = "binomial", alpha = 0,
                        lambda = cv_fit$lambda.min)

  # Print selected coefficients
  cat("\nSelected coefficients at optimal lambda:\n")
  print(coef(final_model))

  return(final_model)
}

# Use function for our goal
outcome = "death"
covariates = names(data_clean)[!(names(data_clean) %in% outcome)]

# Apply optimal model for predictions
result_l2 = fit_logistic_l2_cv(train, outcome, covariates, nfolds=5)
```

```
## Best lambda chosen by CV (min misclassification error): 0.02450644
##
## Selected coefficients at optimal lambda:
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept) -5.4179733657
## age          0.0921088470
## sex         -0.3223164955
## BMI0         0.0303744339
## BMI10       -0.0276416993
## cigarettes   0.0234013847
## chol0        0.0028093443
## chol2        0.0010047304
## chol4       -0.0004730713
## chol6       -0.0027928526
## chol8       -0.0025978290
## chol10       0.0026128476
```

The optimal lambda chosen by 5-fold cross-validation was approximately 0.0245. At this level of regularization, Ridge logistic regression shrinks many coefficients toward zero, reducing variance and stabilizing the model, so variables with larger coefficient magnitudes have stronger influence on the outcome. Variables with larger coefficient magnitudes can thus be interpreted as having a stronger influence on the outcome. According to the model, the five most influential predictors (based on the magnitude of their coefficients) are:

1. Assigned sex at birth
2. Age of the individual
3. BMI initially
4. BMI 10 years into the study
5. Number of cigarettes initially smoked per day

The other covariates have a lessened impact due to the regularization and shrinkage imposed by Ridge, but none of them have been shrunk to exactly zero, highlighting a critical difference with the LASSO shrinkage done above.

Plain Logistic Regression

Next, we fit a plain logistic regression model, the standard form of classification in statistics that does not use any regularization. We will fit the model to our dataset and output the summary for interpretation.

```r
# Fit plain logistic regression model to the training dataset
plain_logit <- glm(death ~ ., data = train, family = "binomial")

# Print model summary for coefficient estimates and significance
summary(plain_logit)
```

```
##
## Call:
## glm(formula = death ~ ., family = "binomial", data = train)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.231358   0.780406  -7.985 1.41e-15 ***
## age          0.113477   0.010026  11.318  < 2e-16 ***
## sex         -0.384353   0.167102  -2.300   0.0214 *
## BMI0         0.066048   0.037309   1.770   0.0767 .
## BMI10       -0.064502   0.037263  -1.731   0.0835 .
## cigarettes   0.030759   0.006585   4.671 3.00e-06 ***
## chol0        0.003242   0.002785   1.164   0.2445
## chol2        0.001065   0.003054   0.349   0.7272
## chol4       -0.001710   0.003048  -0.561   0.5748
## chol6       -0.004993   0.003406  -1.466   0.1427
## chol8       -0.004508   0.003405  -1.324   0.1855
## chol10       0.006751   0.003297   2.048   0.0406 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1371.9  on 1377  degrees of freedom
## Residual deviance: 1178.8  on 1366  degrees of freedom
## AIC: 1202.8
##
## Number of Fisher Scoring iterations: 5
```

The summary above provides insight into covariate effects and their statistical significance. Age and the number of cigarettes smoked initially have the strongest impact on predicting death, with sex and serum cholesterol at 10 years also significant at the 10% level. BMI, both initially and after 10 years, shows moderate effect sizes but weaker statistical evidence. The model's overall fit, indicated by a residual deviance of 1178.8 and an AIC of 1202.8, suggests it captures key predictive relationships while remaining straightforward to interpret.

Stepwise (Subset) Selection Logistic Regression

We now pivot to stepwise feature selection to identify a subset of features for logistic regression. We choose forward stepwise selection because of the small number of predictors and our desire to keep the final model concise and interpretable. For simplicity, we allow the selection procedure to consider all covariates except death, the response variable. Since the regsubsets() function from the leaps package performs selection using linear regression, we use it solely to identify important predictors; the logistic regression coefficients are estimated separately with glm(). We use the BIC score as our selection criterion, as it penalizes model complexity more heavily than AIC, further supporting our goal of interpretability.

```
# Do forward stepwise variable selection on training data (linear regression)
regfit.fwd = regsubsets(death ~ ., data = train, nvmax = ncol(train) - 1,
                        method = "forward")
reg_summary = summary(regfit.fwd)

# Select model with lowest BIC score
best_size = which.min(reg_summary$bic)
cat("Best model size according to BIC:", best_size, "\n\n")
```

```
## Best model size according to BIC: 2
```

```
# Extract names of selected predictors
best_vars <- names(coef(regfit.fwd, best_size))[-1]  # remove intercept
cat("Selected predictors:", best_vars, "\n\n")
```

```
## Selected predictors: age cigarettes
```

```
# Build logistic regression using the selected predictors
subset_formula <- as.formula(paste("death ~", paste(best_vars, collapse = " + ")))
subset_logit <- glm(subset_formula, data = train, family = "binomial")

summary(subset_logit)
```

```
##
## Call:
## glm(formula = subset_formula, family = "binomial", data = train)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.920841   0.476626 -14.520  < 2e-16 ***
## age          0.114853   0.009552  12.024  < 2e-16 ***
## cigarettes   0.036734   0.006035   6.086 1.15e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1371.9  on 1377  degrees of freedom
## Residual deviance: 1195.2  on 1375  degrees of freedom
## AIC: 1201.2
##
## Number of Fisher Scoring iterations: 5
```

According to the BIC scoring criterion, the best model size utilizes two features: age and cigarettes. This is quite similar to what we have seen from the previous models as well, but the stepwise selection model interestingly does not include sex as one of its significant predictors. The model's overall fit, indicated by a residual deviance of 1195.2 and an AIC of 1201.2 suggests it performs quite similarly to plain logistic regression while significantly reducing the number of features considered.

Model Comparison on Testing Data

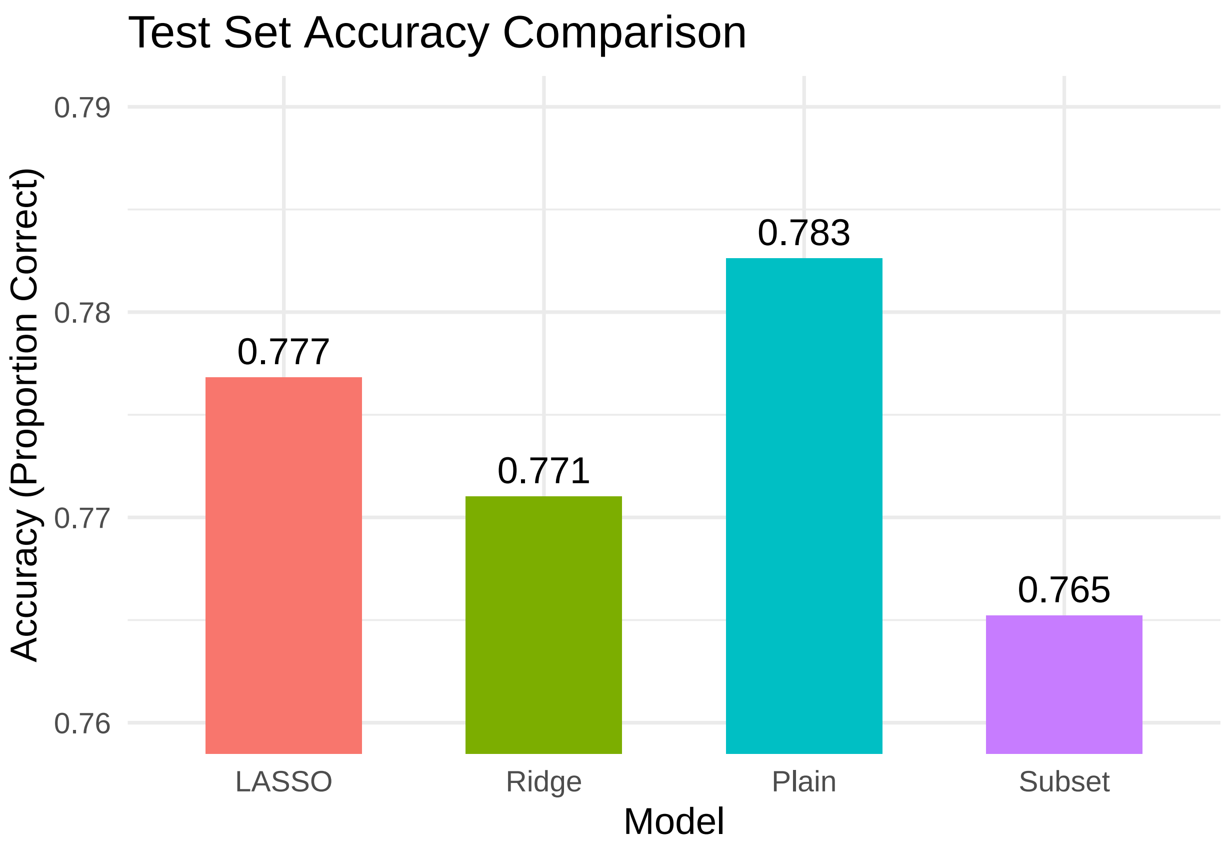
Our final task is to fit each of the four optimized models to the testing dataset and score them based on their accuracy. Accuracy measures the proportion of correct predictions on the testing dataset, allowing us to compare the predictive performance of the four models.

```r
# Predict test data using optimized LASSO logistic regression
lasso_prob = as.vector(predict(result_l1,
                               newx = as.matrix(test[, covariates]),
                               type = "response"))
lasso_class = ifelse(lasso_prob > 0.5, 1, 0)
lasso_accuracy <- mean(lasso_class == test$death)

# Predict test data using optimized Ridge logistic regression
ridge_prob = as.vector(predict(result_l2,
                               newx = as.matrix(test[, covariates]),
                               type = "response"))
ridge_class = ifelse(ridge_prob > 0.5, 1, 0)
ridge_accuracy <- mean(ridge_class == test$death)

# Predict test data using plain logistic regression
logit_prob <- predict(plain_logit, newdata = test, type = "response")
logit_class <- ifelse(logit_prob > 0.5, 1, 0)
logit_accuracy <- mean(logit_class == test$death)

# Predict test data using best subset logistic regression
subset_prob <- predict(subset_logit, newdata = test, type = "response")
subset_class <- ifelse(subset_prob > 0.5, 1, 0)
subset_accuracy <- mean(subset_class == test$death)

# Combine results into a data frame
models = c("LASSO", "Ridge", "Plain", "Subset")
results = data.frame(
  Model = models,
  Accuracy = c(lasso_accuracy, ridge_accuracy, logit_accuracy, subset_accuracy)
)
results
```

```
##    Model  Accuracy
## 1  LASSO 0.7768116
## 2  Ridge 0.7710145
## 3  Plain 0.7826087
## 4 Subset 0.7652174
```

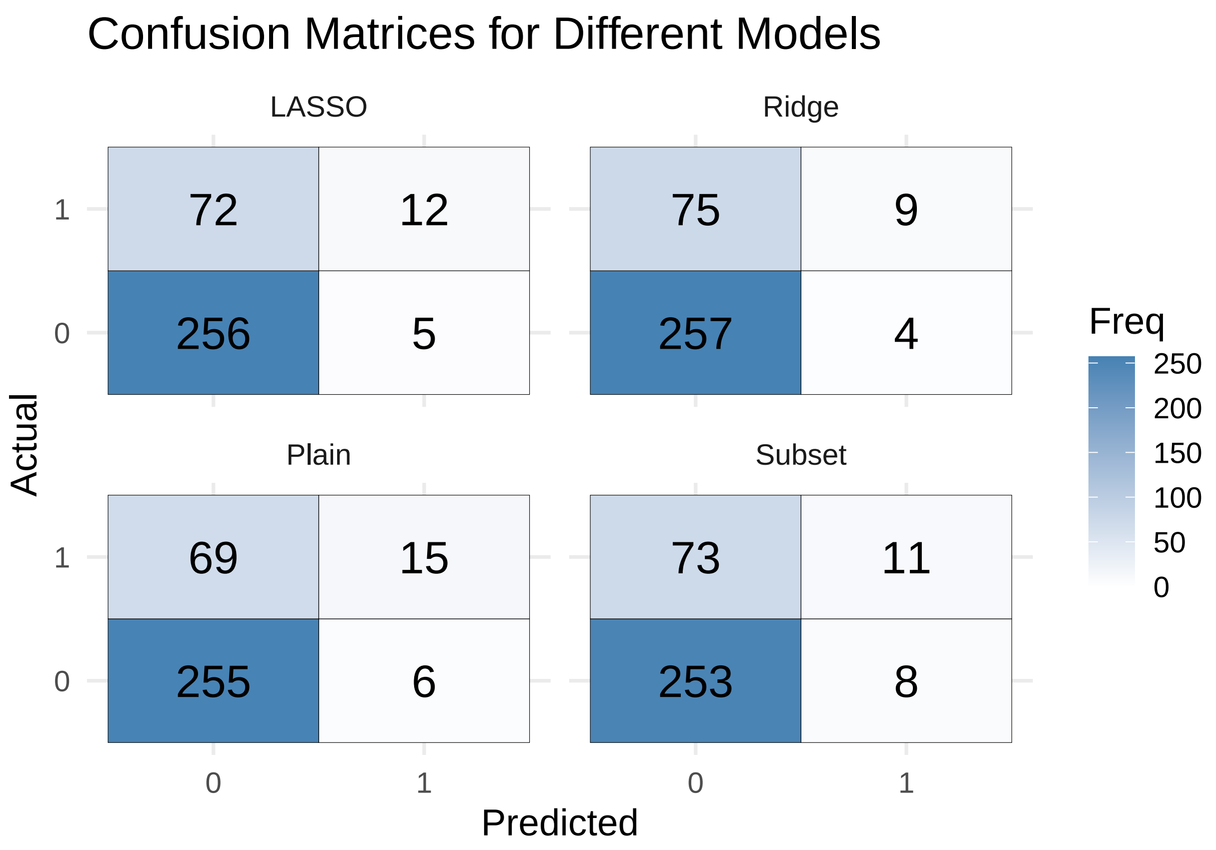
The accuracy scores of each model are summarized in the bar graph below. The y-axis has been adjusted to better emphasize differences in test set accuracy, given the fairly similar performances of the models.

```r
# Create custom ordering for bars based on work above
results$Model <- factor(results$Model, levels = c("LASSO", "Ridge", "Plain", "Subset"))

# Create bar graph with different colors and numerical values for each
# model's testing accuracy
ggplot(results, aes(x = Model, y = Accuracy, fill = Model)) +
  geom_bar(stat = "identity", width = 0.6) +
  geom_text(aes(label = round(Accuracy, 3)), vjust = -0.5, size = 5) +
  coord_cartesian(ylim = c(0.76, 0.79)) +  # Zoom without removing bars
  labs(
    title = "Test Set Accuracy Comparison",
    y = "Accuracy (Proportion Correct)",
    x = "Model"
  ) +
  theme_minimal(base_size = 14) +
  theme(legend.position = "none")
```



Another important metric we can compute is a confusion matrix to tell us the number of false negatives and false positives the models classified, which enhances the accuracy metric we calculated above. We now compute the confusion matrix for each of the four models and present heatmaps to visualize the quality of our classifications. In the confusion matrices, 1 indicates death and 0 indicates survival, consistent with our data encoding.

```r
# Compute confusion matrices
cm_lasso <- confusionMatrix(factor(lasso_class), factor(test$death))$table
cm_ridge <- confusionMatrix(factor(ridge_class), factor(test$death))$table
cm_logit <- confusionMatrix(factor(logit_class), factor(test$death))$table
cm_subset <- confusionMatrix(factor(subset_class), factor(test$death))$table

# Convert tables to data.frames for nicer plotting
cm_to_df <- function(cm, model_name) {
  df <- as.data.frame(cm)
  colnames(df) <- c("Predicted", "Actual", "Freq")
  df$Model <- model_name
  df
}

cm_lasso_df <- cm_to_df(cm_lasso, "LASSO")
cm_ridge_df <- cm_to_df(cm_ridge, "Ridge")
cm_logit_df <- cm_to_df(cm_logit, "Plain")
cm_subset_df <- cm_to_df(cm_subset, "Subset")

# Combine all into one data frame
cm_all <- rbind(cm_lasso_df, cm_ridge_df, cm_logit_df, cm_subset_df)
cm_all$Model <- factor(cm_all$Model, levels = c("LASSO", "Ridge", "Plain", "Subset"))

# Plot confusion matrices as separate facets
ggplot(cm_all, aes(x = Predicted, y = Actual, fill = Freq)) +
  geom_tile(color = "black") +
  geom_text(aes(label = Freq), size = 6) +
  scale_fill_gradient(low="white", high="steelblue", limits=c(0,max(cm_all$Freq))) +
  facet_wrap(~Model) +
  labs(title = "Confusion Matrices for Different Models") +
  theme_minimal(base_size = 14)
```



The accuracy scores were very close, with the plain logistic regression slightly outperforming the penalized models. The confusion matrices confirm that misclassifications are rare and distributed similarly across models.

Key Takeaways

Comparing LASSO, Ridge, plain, and best-subset logistic regression models highlights several insights. LASSO produced a sparse model by zeroing out less important coefficients, Ridge moderately shrank coefficients, plain logistic regression retained all coefficients, and best-subset logistic regression selected only two. Age and daily cigarette consumption consistently emerged as the key predictors.

Predictive performance was similar across models. Plain logistic regression had slightly higher test accuracy, while best-subset had slightly lower accuracy, as shown in the bar chart. Confusion matrices confirm that false positives and false negatives were comparable.

Overall, LASSO and Ridge simplify the model without sacrificing performance, improving interpretability and generalization. Best-subset reduced feature usage but at a minor cost in accuracy. The combined metrics, chart, and heatmaps provide a clear, visual summary of these findings.