ECN 142 Final Project

Jayden Kwong

2025-03-06

Import Libraries

```
library(leaps)
library(glmnet)
```

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

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

```
library(caret)
```

```
## Loading required package: ggplot2
```

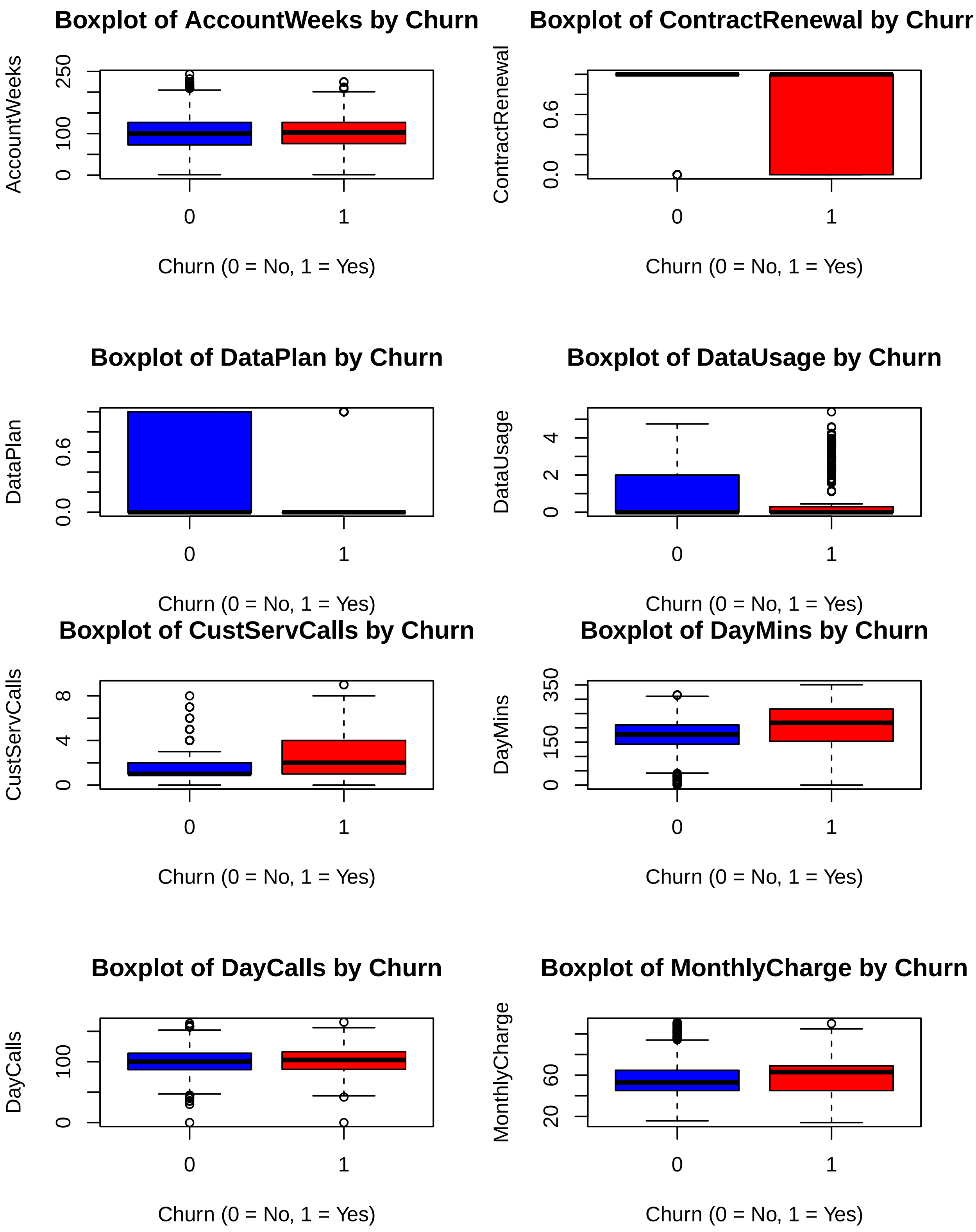
```
## Loading required package: lattice
```

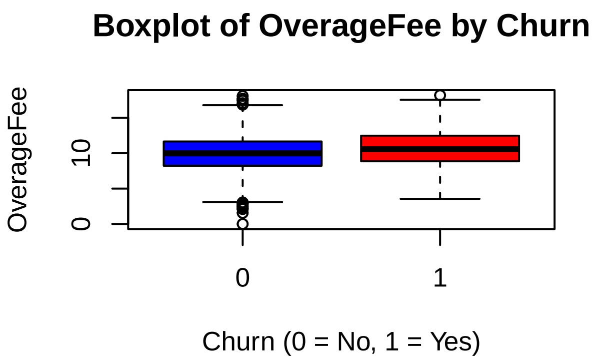
```
library(corrplot)
```

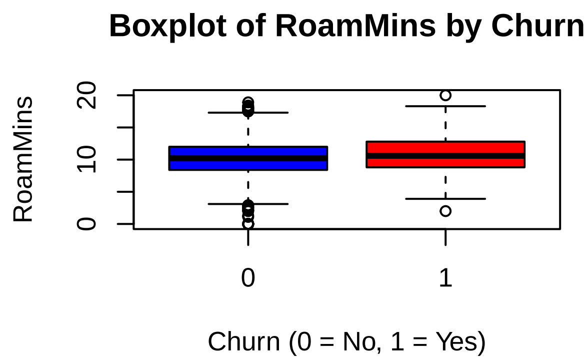
```
## corrplot 0.92 loaded
```

```
data <- read.csv("telecom_churn.csv")
data <- na.omit(data)
set.seed(123)
```

```
# Set graphical layout to display multiple plots
par(mfrow = c(2, 2))  # Adjust rows and columns based on the number of predictors

# List of predictors to plot
predictors <- c("AccountWeeks", "ContractRenewal", "DataPlan", "DataUsage",
                "CustServCalls", "DayMins", "DayCalls", "MonthlyCharge",
                "OverageFee", "RoamMins")

# Loop through each predictor and create a boxplot
for (var in predictors) {
  boxplot(data[[var]] ~ data$Churn,
          main = paste("Boxplot of", var, "by Churn"),
          xlab = "Churn (0 = No, 1 = Yes)",
          ylab = var,
          col = c("blue", "red"))
}
```


Boxplot of AccountWeeks by Churn
AccountWeeks
0
100
250
0
1
Churn (0 = No, 1 = Yes)
Boxplot of ContractRenewal by Churn
ContractRenewal
0.0
0.6
0
1
Churn (0 = No, 1 = Yes)
Boxplot of DataPlan by Churn
DataPlan
0.0
0.6
0
1
Churn (0 = No, 1 = Yes)
Boxplot of DataUsage by Churn
DataUsage
0
2
4
0
1
Churn (0 = No, 1 = Yes)
Boxplot of CustServCalls by Churn
CustServCalls
0
4
8
0
1
Churn (0 = No, 1 = Yes)
Boxplot of DayMins by Churn
DayMins
0
150
350
0
1
Churn (0 = No, 1 = Yes)
Boxplot of DayCalls by Churn
DayCalls
0
100
0
1
Churn (0 = No, 1 = Yes)
Boxplot of MonthlyCharge by Churn
MonthlyCharge
20
60
0
1
Churn (0 = No, 1 = Yes)





#
Logistic Regression with all Model

```
set.seed(123)

# 80% Train
trainIndex <- createDataPartition(data$Churn, p = 0.8, list = FALSE)
trainData <- data[trainIndex, ]
testData  <- data[-trainIndex, ]

log_model <- glm(Churn ~ ., data = trainData, family = "binomial")
summary(log_model)
```

```
##
## Call:
## glm(formula = Churn ~ ., family = "binomial", data = trainData)
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -5.6349292  0.6080743  -9.267  < 2e-16 ***
## AccountWeeks     0.0006289  0.0015459   0.407  0.68415
## ContractRenewal -1.9652691  0.1578396 -12.451  < 2e-16 ***
## DataPlan        -1.4420265  0.5977300  -2.413  0.01584 *
## DataUsage        0.9285898  2.1373236   0.434  0.66395
## CustServCalls    0.5118968  0.0423880  12.076  < 2e-16 ***
## DayMins          0.0247914  0.0360776   0.687  0.49198
## DayCalls         0.0026905  0.0030612   0.879  0.37946
## MonthlyCharge   -0.0755197  0.2119522  -0.356  0.72161
## OverageFee       0.2616866  0.3617678   0.723  0.46946
## RoamMins         0.0772774  0.0244615   3.159  0.00158 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2201.8  on 2666  degrees of freedom
## Residual deviance: 1748.0  on 2656  degrees of freedom
## AIC: 1770
##
## Number of Fisher Scoring iterations: 6
```

```
test_pred <- predict(log_model, newdata = testData, type = "response")
test_class <- ifelse(test_pred > 0.5, 1, 0)
```

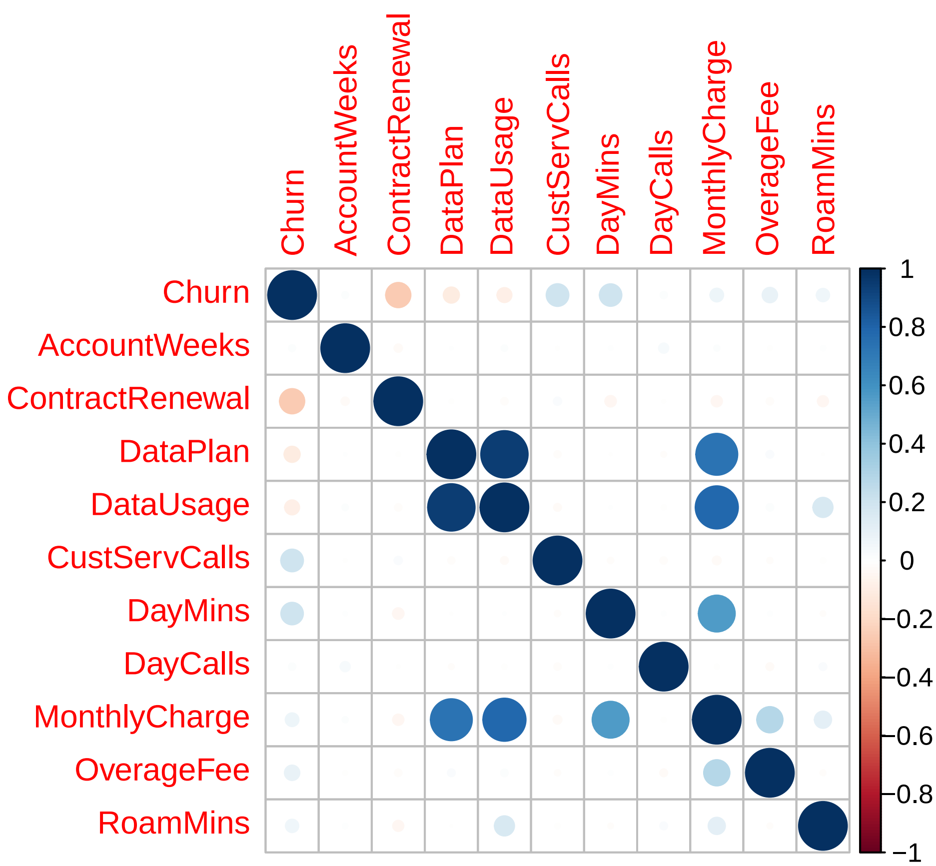
```r
accuracy <- mean(test_class == testData$Churn)
cat("Logistic Regression Accuracy:", accuracy, "\n")
```

```
## Logistic Regression Accuracy: 0.8618619
```

Look at Correlation Matrix

Polynomial Logistic Regression

```r
#cor(data)
corrplot(cor(data))
```




```r
# Load necessary libraries
library(splines)

# Fit a logistic regression with polynomial terms for selected variables
log_poly <- glm(Churn ~ poly(CustServCalls, 2) + poly(OverageFee, 2) + poly(MonthlyCharge, 2) +
                ContractRenewal + DataUsage,
                family = "binomial", data = trainData)

summary(log_poly)
```

```
##
## Call:
## glm(formula = Churn ~ poly(CustServCalls, 2) + poly(OverageFee,
##     2) + poly(MonthlyCharge, 2) + ContractRenewal + DataUsage,
##     family = "binomial", data = trainData)
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)                0.4849     0.1547   3.134  0.00172 **
## poly(CustServCalls, 2)1  30.9098     2.9200  10.586  < 2e-16 ***
## poly(CustServCalls, 2)2  20.9084     3.4608   6.042 1.53e-09 ***
## poly(OverageFee, 2)1      0.1209     3.6381   0.033  0.97349
## poly(OverageFee, 2)2      0.5283     3.2640   0.162  0.87142
## poly(MonthlyCharge, 2)1  66.6549     6.1890  10.770  < 2e-16 ***
## poly(MonthlyCharge, 2)2  12.6253     3.9133   3.226  0.00125 **
## ContractRenewal          -1.9564     0.1560 -12.540  < 2e-16 ***
## DataUsage                -1.1941     0.1140 -10.478  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2201.8  on 2666  degrees of freedom
## Residual deviance: 1724.0  on 2658  degrees of freedom
## AIC: 1742
##
## Number of Fisher Scoring iterations: 6
```

```r
# Predict probabilities on the test set
pred_probs <- predict(log_poly, newdata = testData, type = "response")

# Convert probabilities into binary predictions (Threshold = 0.5)
pred_labels <- ifelse(pred_probs > 0.5, 1, 0)

# Compute Accuracy
accuracy <- mean(pred_labels == testData$Churn)
print(paste("Model Accuracy:", round(accuracy, 4)))
```

```
## [1] "Model Accuracy: 0.8649"
```

Variable Selection (Best Subset Selecion)

```r
best_subset <- regsubsets(Churn ~., data= data, nvmax = 10)
subset_summary <- summary(best_subset)

best_adj <- which.max(subset_summary$adjr2)
best_cp <- which.min(subset_summary$cp)
best_bic <- which.min(subset_summary$bic)

best_model_adj <- coef(best_subset, best_adj)
best_model_cp <- coef(best_subset, best_cp)
```

```r
best_model_bic <- coef(best_subset, best_bic)

print(best_model_adj)
```

```
##     (Intercept) ContractRenewal       DataUsage   CustServCalls        DayCalls
##   -0.1470461398   -0.3001137523   -0.1029598144    0.0582735286    0.0003467457
##   MonthlyCharge        RoamMins
##    0.0074675218    0.0098361679
```

```r
print(best_model_cp)
```

```
##     (Intercept) ContractRenewal       DataUsage   CustServCalls   MonthlyCharge
##    -0.112451288    -0.300175125    -0.103006680     0.058173176     0.007465555
##        RoamMins
##     0.009894002
```

```r
print(best_model_bic) # Use BIC
```

```
##     (Intercept) ContractRenewal       DataUsage   CustServCalls   MonthlyCharge
##    -0.112451288    -0.300175125    -0.103006680     0.058173176     0.007465555
##        RoamMins
##     0.009894002
```

Variable Selection (Forward Step Selection)

```r
regfit.fwd <- regsubsets(Churn ~ ., data = data, nvmax = 10, method = "forward")
fwd_summary <- summary(regfit.fwd)

fwd_adj <- which.max(fwd_summary$adjr2)
fwd_cp <- which.min(fwd_summary$cp)
fwd_bic <- which.min(fwd_summary$bic)

fwd_model_adj <- coef(regfit.fwd, fwd_adj)
fwd_model_cp <- coef(regfit.fwd, fwd_cp)
fwd_model_bic <- coef(regfit.fwd, fwd_bic)

print(fwd_model_adj)
```

```
##     (Intercept) ContractRenewal        DataPlan   CustServCalls         DayMins
##   -0.1244205134   -0.2992595976   -0.0795901811    0.0583184330    0.0012624977
##        DayCalls      OverageFee        RoamMins
##    0.0003485952    0.0128758547    0.0077235678
```

```r
print(fwd_model_cp)
```

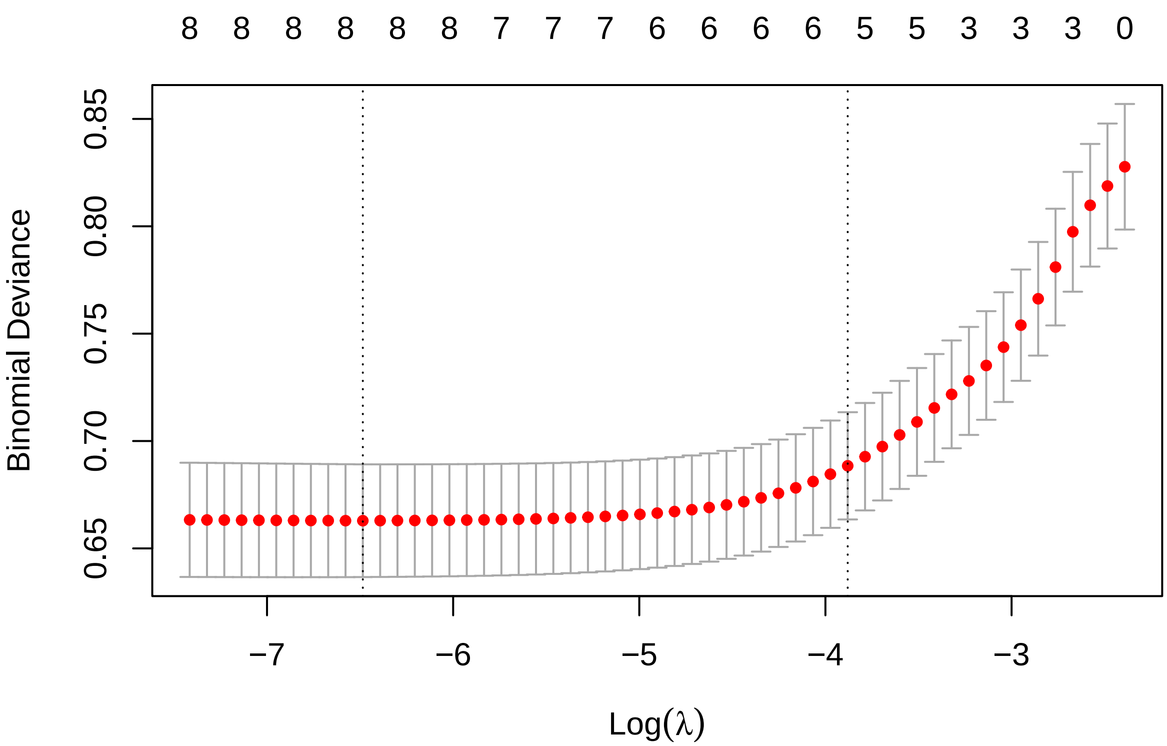
```
##     (Intercept) ContractRenewal        DataPlan   CustServCalls         DayMins
##    -0.089255626    -0.299317791    -0.079761287     0.058217071     0.001263361
##      OverageFee        RoamMins
##     0.012817028     0.007776409
```

```r
print(fwd_model_bic) # Use BIC
```

```
##     (Intercept) ContractRenewal        DataPlan   CustServCalls         DayMins
##    -0.089255626    -0.299317791    -0.079761287     0.058217071     0.001263361
##      OverageFee        RoamMins
##     0.012817028     0.007776409
```

Variable Selection (LASSO)

```r
library(glmnet)
x <- model.matrix(Churn ~ ., data = data)[,-1]
y <- data$Churn
lasso_cv <- cv.glmnet(x, y, alpha = 1, family = "binomial")
plot(lasso_cv)
```




```r
best_lambda <- lasso_cv$lambda.min
print(best_lambda)
```

```
## [1] 0.001525839
```

```r
lasso_coef <-coef(lasso_cv, s = "lambda.min")
print(lasso_coef)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                             s1
```

```
## (Intercept)     -5.6357616768
## AccountWeeks     0.0002844391
## ContractRenewal -1.9394322642
## DataPlan        -0.8764699438
## DataUsage        .
## CustServCalls    0.4907927477
## DayMins          0.0123204950
## DayCalls         0.0028228982
## MonthlyCharge    .
## OverageFee       0.1306506269
## RoamMins         0.0761396580
```

```
lasso_predictions <- predict(lasso_cv, newx = x, s = "lambda.min", type = "response")
pred_class <- ifelse(lasso_predictions > 0.5, 1, 0)
lasso_accuracy <- mean(pred_class == y)
lasso_accuracy
```

```
## [1] 0.860486
```

Comparing each Models

```
set.seed(123)

# Split into training (80%) and test (20%)
trainIndex <- createDataPartition(data$Churn, p = 0.8, list = FALSE)
trainData <- data[trainIndex, ]
testData  <- data[-trainIndex, ]

# Train Logistic Regression using Best Subset Features
selected_features_subset <- names(coef(best_subset, best_bic))[-1]
log_model_subset <- glm(Churn ~ ., data = trainData[, c("Churn", selected_features_subset)], family = "

# Train Logistic Regression using Forward Selection Features
selected_features_fwd <- names(coef(best_subset, fwd_bic))[-1]
log_model_fwd <- glm(Churn ~ ., data = trainData[, c("Churn", selected_features_fwd)], family = "binomi

# Train Logistic Regression using LASSO Features
selected_features_lasso <- rownames(lasso_coef)[lasso_coef[,1] != 0][-1]  # Remove Intercept
log_model_lasso <- glm(Churn ~ ., data = trainData[, c("Churn", selected_features_lasso)], family = "bi



# Make Predictions on Test Data
pred_subset <- predict(log_model_subset, newdata = testData, type = "response")
pred_fwd <- predict(log_model_fwd, newdata = testData, type = "response")
pred_lasso <- predict(log_model_lasso, newdata = testData, type = "response")

# Convert Probabilities to Binary (0 or 1)
pred_class_subset <- ifelse(pred_subset > 0.5, 1, 0)
pred_class_fwd <- ifelse(pred_fwd > 0.5, 1, 0)
pred_class_lasso <- ifelse(pred_lasso > 0.5, 1, 0)
```

```r
# Compute Accuracy
acc_subset <- mean(pred_class_subset == testData$Churn)
acc_fwd <- mean(pred_class_fwd == testData$Churn)
acc_lasso <- mean(pred_class_lasso == testData$Churn)

# Accuracy Results
cat("Best Subset Selection Accuracy:", acc_subset, "\n")
```

```
## Best Subset Selection Accuracy: 0.8618619
```

```r
cat("Forward Selection Accuracy:", acc_fwd, "\n")
```

```
## Forward Selection Accuracy: 0.8648649
```

```r
cat("LASSO Accuracy:", acc_lasso, "\n")
```

```
## LASSO Accuracy: 0.8618619
```

```r
regfit.fwd <- regsubsets(Churn ~ ., data = data, nvmax = 10, method = "forward")
fwd_summary <- summary(regfit.fwd)

best_model_size_fwd <- which.min(fwd_summary$bic)
selected_features_fwd <- names(coef(regfit.fwd, best_model_size_fwd))[-1]

# Selected Features
cat("Stepwise Selected Features:\n")
```

```
## Stepwise Selected Features:
```

```r
print(selected_features_fwd)
```

```
## [1] "ContractRenewal" "DataPlan"        "CustServCalls"   "DayMins"
## [5] "OverageFee"      "RoamMins"
```

```r
# Prepare predictor and response
x <- model.matrix(Churn ~ ., data = data[, c("Churn", selected_features_fwd)])[,-1]
y <- as.numeric(as.character(data$Churn))

# Perform LASSO with Cross-Validation
set.seed(123)
lasso_cv <- cv.glmnet(x, y, alpha = 1, family = "binomial")

# Best lambda
best_lambda <- lasso_cv$lambda.min
cat("Best Lambda for LASSO:", best_lambda, "\n")
```

```
## Best Lambda for LASSO: 0.0007248971
```

```r
# Coefficients after choosing lambda
lasso_coef <- coef(lasso_cv, s = "lambda.min")
print(lasso_coef)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)     -5.43959997
## ContractRenewal -1.96541674
## DataPlan        -0.90674248
## CustServCalls    0.49790446
## DayMins          0.01256298
## OverageFee       0.13438030
## RoamMins         0.08010583
```

```r
selected_features_lasso <- rownames(lasso_coef)[lasso_coef[,1] != 0][-1]

# Final Model
final_model_lasso <- glm(Churn ~ ., data = data[, c("Churn", selected_features_lasso)], family = "binomi

summary(final_model_lasso)
```

```
##
## Call:
## glm(formula = Churn ~ ., family = "binomial", data = data[, c("Churn",
##     selected_features_lasso)])
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -5.552897   0.432757 -12.831  < 2e-16 ***
## ContractRenewal -1.989219   0.143452 -13.867  < 2e-16 ***
## DataPlan        -0.934814   0.144015  -6.491 8.52e-11 ***
## CustServCalls    0.505651   0.038834  13.021  < 2e-16 ***
## DayMins          0.012774   0.001073  11.907  < 2e-16 ***
## OverageFee       0.138612   0.022648   6.120 9.34e-10 ***
## RoamMins         0.083476   0.020304   4.111 3.93e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2758.3  on 3332  degrees of freedom
## Residual deviance: 2190.6  on 3326  degrees of freedom
## AIC: 2204.6
##
## Number of Fisher Scoring iterations: 5
```

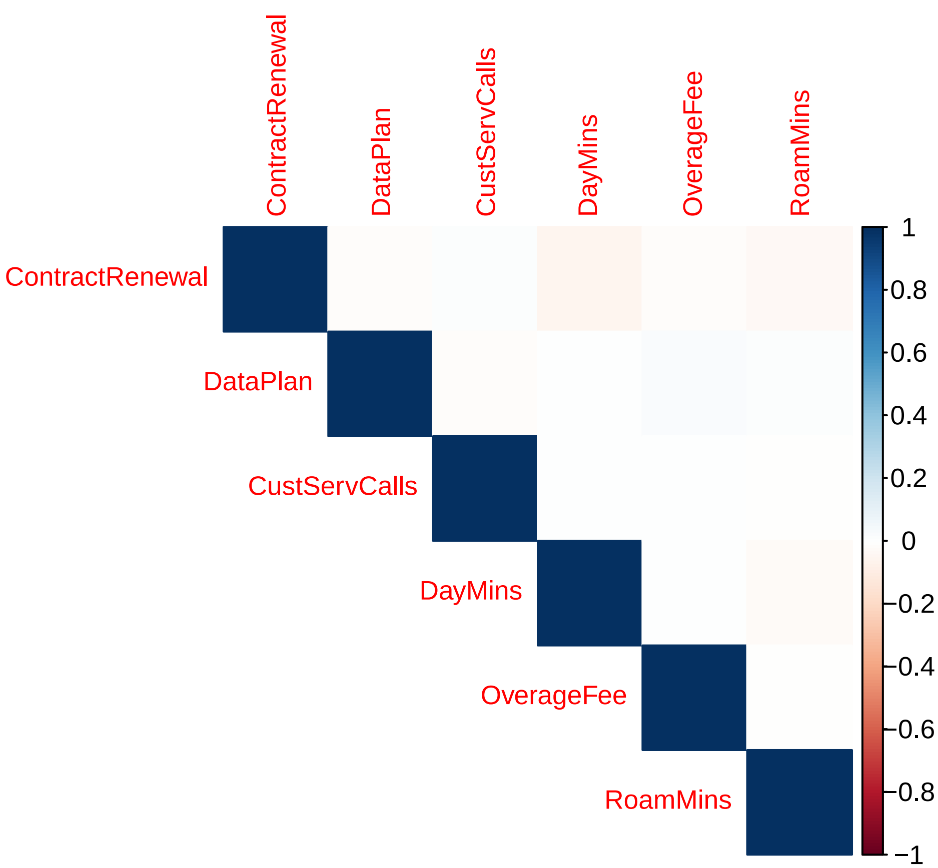
```r
# Evaluate Accuracy
# Split into training & test sets
set.seed(123)
trainIndex <- createDataPartition(data$Churn, p = 0.8, list = FALSE)
trainData <- data[trainIndex, ]
testData  <- data[-trainIndex, ]
```

```r
# Train model on training data
final_model_lasso <- glm(Churn ~ ., data = trainData[, c("Churn", selected_features_lasso)], family = "l

# Predict on test data
test_pred <- predict(final_model_lasso, newdata = testData, type = "response")

# Convert probabilities to binary (0 or 1)
test_class <- ifelse(test_pred > 0.5, 1, 0)

# Compute Accuracy
final_accuracy <- mean(test_class == testData$Churn)
cat("Final Model Accuracy After Stepwise + LASSO:", final_accuracy, "\n")
```

```
## Final Model Accuracy After Stepwise + LASSO: 0.8633634
```

Check for Multicollinearity

```r
library(corrplot)
numeric_features <- trainData[, selected_features_lasso]
numeric_features <- numeric_features[, sapply(numeric_features, is.numeric)]
corr_matrix <- cor(numeric_features, use = "complete.obs")
corrplot(corr_matrix, method = "color", type = "upper",
         tl.cex = 0.8, number.cex = 0.7)
```



PCA

```r
# Standardize features before PCA (important for variance scaling)
preProc <- preProcess(trainData[, selected_features_lasso], method = "pca", pcaComp = 5)

# Transform data using PCA
train_pca <- predict(preProc, trainData[, selected_features_lasso])
test_pca <- predict(preProc, testData[, selected_features_lasso])

# Add the target variable back
train_pca$Churn <- trainData$Churn
test_pca$Churn <- testData$Churn

# Fit logistic regression on PCA-transformed data
pca_log_model <- glm(Churn ~ ., data = train_pca, family = "binomial")

# Summary of PCA-based logistic model
summary(pca_log_model)
```

```
##
## Call:
## glm(formula = Churn ~ ., family = "binomial", data = train_pca)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.25277    0.07629 -29.529  < 2e-16 ***
## PC1         -0.57872    0.05423 -10.671  < 2e-16 ***
## PC2          0.54435    0.06445   8.446  < 2e-16 ***
## PC3          0.06816    0.06127   1.113    0.266
## PC4          0.86837    0.06090  14.258  < 2e-16 ***
## PC5          0.34574    0.06773   5.105 3.31e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2201.8  on 2666  degrees of freedom
## Residual deviance: 1753.2  on 2661  degrees of freedom
## AIC: 1765.2
##
## Number of Fisher Scoring iterations: 5
```

```r
# Predict probabilities
pca_pred_probs <- predict(pca_log_model, newdata = test_pca, type = "response")

# Convert to class predictions (Threshold = 0.5)
pca_pred_labels <- ifelse(pca_pred_probs > 0.5, 1, 0)

# Compute accuracy
pca_accuracy <- mean(pca_pred_labels == test_pca$Churn)
print(paste("PCA Logistic Regression Accuracy:", round(pca_accuracy, 4)))
```

```
## [1] "PCA Logistic Regression Accuracy: 0.8634"
```

Classification Tree

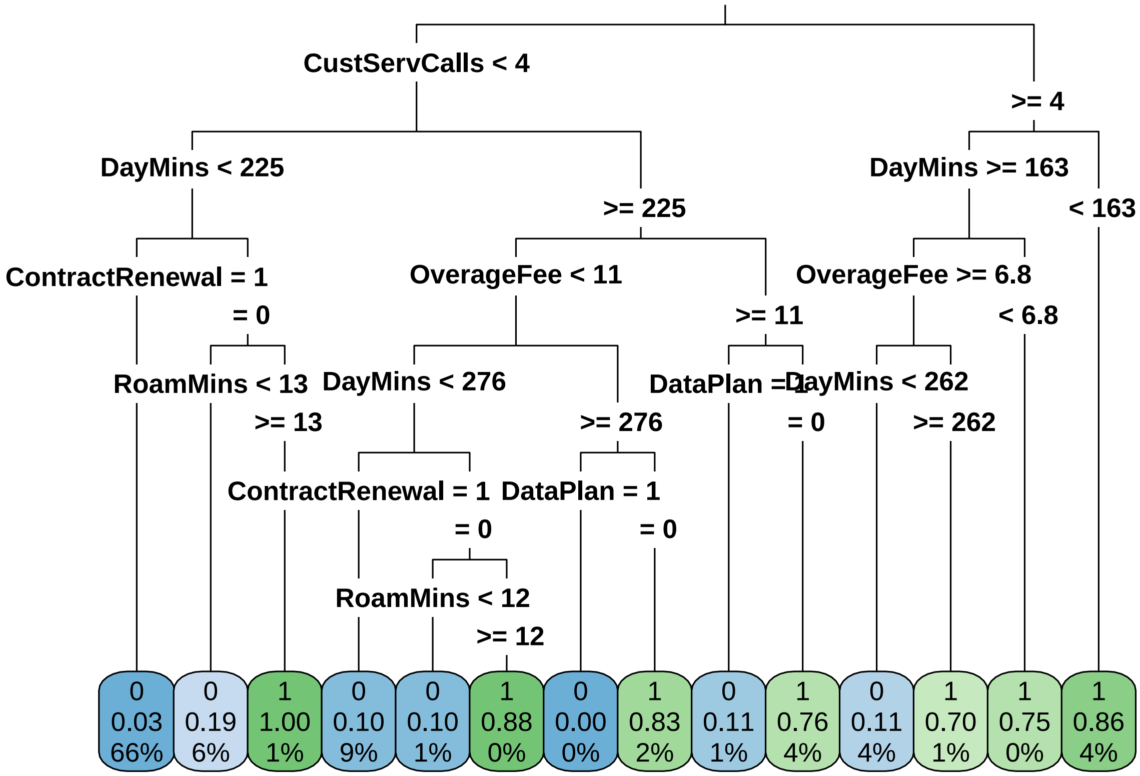
```r
library(tree)
library(rpart)
library(rpart.plot)
library(caret)  # For model evaluation

set.seed(42)
train_index <- sample(1:nrow(trainData), size = 0.7 * nrow(trainData))
valid_index <- setdiff(1:nrow(trainData), train_index)

# Define training and validation datasets
train <- trainData[train_index, c("Churn", selected_features_lasso)]
valid <- trainData[valid_index, c("Churn", selected_features_lasso)]

# Train the Classification Tree
tree_model <- rpart(Churn ~ ., data = train, method = "class")

printcp(tree_model)
```

```
##
## Classification tree:
## rpart(formula = Churn ~ ., data = train, method = "class")
##
## Variables actually used in tree construction:
## [1] ContractRenewal CustServCalls   DataPlan        DayMins
## [5] OverageFee      RoamMins
##
## Root node error: 271/1866 = 0.14523
##
## n= 1866
##
##         CP nsplit rel error  xerror     xstd
## 1 0.097786      0   1.00000 1.00000 0.056162
## 2 0.051661      2   0.80443 0.80443 0.051201
## 3 0.040590      5   0.64945 0.75646 0.049847
## 4 0.036900      7   0.56827 0.65683 0.046824
## 5 0.033210      8   0.53137 0.65683 0.046824
## 6 0.014760      9   0.49815 0.57196 0.043991
## 7 0.011070     11   0.46863 0.56458 0.043732
## 8 0.010000     13   0.44649 0.54244 0.042941
```

```r
optimal_cp <- tree_model$cptable[which.min(tree_model$cptable[, "xerror"]), "CP"]
print(paste("Optimal CP:", optimal_cp))
```

```
## [1] "Optimal CP: 0.01"
```

```r
pruned_tree <- prune(tree_model, cp = optimal_cp)
rpart.plot(pruned_tree, type = 3, fallen.leaves = TRUE, cex = 0.8)
```




```
conf_matrix <- confusionMatrix(predict(pruned_tree, newdata = testData, type = "class"), as.factor(test
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 555  37
##          1  13  61
##
##                Accuracy : 0.9249
##                  95% CI : (0.9022, 0.9438)
##     No Information Rate : 0.8529
##     P-Value [Acc > NIR] : 8.83e-09
##
##                   Kappa : 0.6672
##
##  Mcnemar's Test P-Value : 0.001143
##
##             Sensitivity : 0.9771
##             Specificity : 0.6224
##          Pos Pred Value : 0.9375
##          Neg Pred Value : 0.8243
##              Prevalence : 0.8529
##          Detection Rate : 0.8333
##    Detection Prevalence : 0.8889
##       Balanced Accuracy : 0.7998
##
```

```
##        'Positive' Class : 0
##
```