# INFS_SP5_2023
# Predictive Analytics
# PRACTICAL 4

Enna H

## Contents

### Task1. Fitting a logistic regression mode

```r
# Load libraries
pacman::p_load(pscl, ROCR)
```

```r
data <- read.csv(url("https://raw.githubusercontent.com/sreckojoksimovic/infs5100/main/wine-data.csv"))
```

Make sure quality_class is factor

```r
names(data) <- c("fixed_acidity", names(data)[2:12])
data$fixed_acidity <- scale(data$fixed_acidity, scale = TRUE, center = TRUE)
data$volatile_acidity <- scale(data$volatile_acidity, scale = TRUE, center =
TRUE)
data$citric_acid <- scale(data$citric_acid, scale = TRUE, center = TRUE)
data$residual_sugar <- scale(data$residual_sugar, scale = TRUE, center =
TRUE)
data$chlorides <- scale(data$chlorides, scale = TRUE, center = TRUE)
data$free_sulfur_dioxide <- scale(data$free_sulfur_dioxide, scale = TRUE,
center = TRUE)
data$total_sulfur_dioxide <- scale(data$total_sulfur_dioxide, scale = TRUE,
center = TRUE)
data$density <- scale(data$density, scale = TRUE, center = TRUE)
data$pH <- scale(data$pH, scale = TRUE, center = TRUE)
data$sulphates <- scale(data$sulphates, scale = TRUE, center = TRUE)
data$alcohol <- scale(data$alcohol, scale = TRUE, center = TRUE)
```

```r
# Ensure 'quality_class' is a factor
data$quality_class <- as.factor(data$quality_class)

# List of numeric variable names to be scaled
numeric_vars <- c(
  "fixed_acidity", "volatile_acidity", "citric_acid",
  "residual_sugar", "chlorides", "free_sulfur_dioxide",
  "total_sulfur_dioxide", "density", "pH", "sulphates", "alcohol"
)

# Scale numeric variables using vectorized operations
data[numeric_vars] <- scale(data[numeric_vars], center = TRUE, scale = TRUE)
```

```r
# Data input validation
head(data)
```

```
##   fixed_acidity volatile_acidity citric_acid residual_sugar   chlorides
## 1   -0.59560241      0.761527928 -1.42253170     -0.9547992 -0.48573993
## 2   -0.30870913      0.352212377 -1.31906260     -0.3816958 -0.31012582
## 3    0.09294147     -1.401997128  1.47460306     -0.5249717  0.10695768
## 4   -0.13657316     -0.817260626  0.02603568     -0.3100579 -0.46378816
## 5   -0.48084510      0.001370476 -0.59477891     -0.4533337 -0.04670466
## 6   -0.19395181      0.410686027 -0.59477891     -0.5249717 -0.48573993
##   free_sulfur_dioxide total_sulfur_dioxide     density          pH  sulphates
## 1         -0.09840439           -0.7855479 -1.13330106  0.53588028 -1.1468548
## 2         -0.67057779           -0.8763222  0.02686524  0.33954281 -0.5459443
## 3          1.80884027            1.6956150  0.07960007 -0.05313212  0.5356944
## 4         -0.28912885           -0.5132252  0.02686524 -0.51125288  0.4155124
## 5         -0.38449109           -0.3619348  0.02686524  0.47043446 -0.2454891
## 6         -1.24275118           -0.9368384 -0.28954375  0.73221775  1.5572422
##       alcohol quality_class
## 1 -0.40299122             1
## 2 -0.86998068             1
## 3  0.06399823             1
## 4 -0.68318490             1
## 5 -0.86998068             1
## 6  0.06399823             1
```

```r
summary(data)
```

```
##  fixed_acidity     volatile_acidity   citric_acid       residual_sugar
##  Min.   :-2.0874   Min.   :-2.3376   Min.   :-1.42253   Min.   :-1.16971
##  1st Qu.:-0.7104   1st Qu.:-0.7588   1st Qu.:-0.90519   1st Qu.:-0.45333
##  Median :-0.2513   Median :-0.0571   Median :-0.07743   Median :-0.23842
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000
##  3rd Qu.: 0.5520   3rd Qu.: 0.6446   3rd Qu.: 0.80205   3rd Qu.: 0.04813
##  Max.   : 4.3390   Max.   : 4.7377   Max.   : 2.66450   Max.   : 9.28942
##    chlorides         free_sulfur_dioxide total_sulfur_dioxide
##  Min.   :-1.64918   Min.   :-1.4335     Min.   :-1.2394
##  1st Qu.:-0.37598   1st Qu.:-0.7659     1st Qu.:-0.7553
##  Median :-0.17841   Median :-0.1938     Median :-0.2712
##  Mean   : 0.00000   Mean   : 0.0000     Mean   : 0.0000
##  3rd Qu.: 0.06305   3rd Qu.: 0.5691     3rd Qu.: 0.4853  
##  Max.   :11.49992   Max.   : 5.3372     Max.   : 7.3236  
##     density                pH             sulphates          alcohol       
##  Min.   :-3.522189   Min.   :-2.93275   Min.   :-1.7478   Min.   :-1.8974  
##  1st Qu.:-0.605953   1st Qu.:-0.64214   1st Qu.:-0.6661   1st Qu.:-0.8700  
##  Median : 0.003135   Median : 0.01231   Median :-0.2455   Median :-0.2162  
##  Mean   : 0.000000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.581899   3rd Qu.: 0.60133   3rd Qu.: 0.4155   3rd Qu.: 0.6244  
##  Max.   : 3.660295   Max.   : 4.59352   Max.   : 7.9269   Max.   : 4.1735  
##  quality_class
##  0:1319       
##  1: 217       
##               
##               
##               
##               
```

```
# Split data into training and test datasets. We will use 70%/30% split
# again.
set.seed(123)
dat.d <- sample(1:nrow(data),size=nrow(data)*0.7,replace = FALSE) #random selection of 70% data.
train.data <- data[dat.d,] # 70% training data
test.data <- data[-dat.d,] # remaining % test data
```

```
model <- glm(quality_class ~., family=binomial(link='logit'),
data=train.data)
```

## Task2. Interpreting a logistic regression model

```
summary(model)
```

```
## 
## Call:
## glm(formula = quality_class ~ ., family = binomial(link = "logit"), 
##     data = train.data)
## 
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)    
## (Intercept)          -2.70156    0.16058 -16.824  < 2e-16 ***
## fixed_acidity         0.50867    0.26622   1.911  0.05605 .  
## volatile_acidity     -0.24575    0.15766  -1.559  0.11906    
## citric_acid           0.26652    0.19489   1.368  0.17146    
## residual_sugar        0.33147    0.13617   2.434  0.01492 *  
## chlorides            -0.52590    0.21648  -2.429  0.01513 *  
## free_sulfur_dioxide   0.11237    0.15292   0.735  0.46245    
## total_sulfur_dioxide -0.45866    0.17664  -2.597  0.00941 ** 
## density              -0.65174    0.25497  -2.556  0.01059 *  
## pH                    0.02047    0.19031   0.108  0.91433    
## sulphates             0.60421    0.10715   5.639 1.71e-08 ***
## alcohol               0.68291    0.17293   3.949 7.84e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 857.87  on 1074  degrees of freedom
## Residual deviance: 609.09  on 1063  degrees of freedom
## AIC: 633.09
##
## Number of Fisher Scoring iterations: 6
```

residual_sugar, sulphates and alcohol are positively and significantly associated with the probability that wine is of high quality. On the other hand, chlorides, total_sulfur_dioxide, and density are also significantly, but negatively associated with the probability that wine is of high quality

Please remember that in the logit model the response variable is log odds: ln(odds) = ln(p/(1-p))= a*x1 + b*x2 + … + z*xn. This means that one unit increase in residual_sugar, increases the log odds by 0.33.

```
anova(model, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: quality_class
##
## Terms added sequentially (first to last)
##
##
##                      Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                  1074     857.87
## fixed_acidity         1   11.399      1073     846.47 0.0007348 ***
## volatile_acidity      1   67.045      1072     779.43 2.654e-16 ***
## citric_acid           1    4.386      1071     775.04 0.0362378 *
## residual_sugar        1    1.160      1070     773.88 0.2813884
## chlorides             1   19.750      1069     754.13 8.826e-06 ***
## free_sulfur_dioxide   1    3.187      1068     750.94 0.0742421 .
## total_sulfur_dioxide  1   14.265      1067     736.68 0.0001588 ***
## density               1   64.066      1066     672.61 1.203e-15 ***
## pH                    1    6.588      1065     666.02 0.0102648 *
## sulphates             1   41.208      1064     624.81 1.368e-10 ***
## alcohol               1   15.723      1063     609.09 7.333e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The difference between the NULL deviance and the residual deviance shows how our model is doing compare to a model with only the intercept. The wider this gap, the better model. Analyzing the output above, we can see the drop in deviance when adding each variable one at a time. All the variables, except for residual_sugar and free_sulfur_dioxide, significantly improve the model. While no exact equivalent to the R2 of linear regression exists, the McFadden R2 index can be used to assess the model fit.

```
pR2(model) # McFadden R2
```

```
## fitting null model for pseudo-r2
##          llh      llhNull           G2     McFadden         r2ML         r2CU
## -304.5458984 -428.9345978  248.7773989    0.2899946    0.2065945    0.3757770
```

Values between 0.2-0.4 for the McFadden R2 represent the excellent fit.

## Task 3. Evaluating the predictive power of the model

calculate various evaluation metrics, to assess the predictive ability of our model.

```
fitted.results <- predict(model, newdata=test.data, type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)
confusionMatrix(as.factor(fitted.results), as.factor(test.data[, 12]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 379  44
##          1  12  26
##
##                Accuracy : 0.8785
##                  95% CI : (0.8452, 0.9069)
##     No Information Rate : 0.8482
##     P-Value [Acc > NIR] : 0.03698
##
##                   Kappa : 0.4194
##
##  Mcnemar's Test P-Value : 3.435e-05
##
##             Sensitivity : 0.9693
##             Specificity : 0.3714
##          Pos Pred Value : 0.8960
##          Neg Pred Value : 0.6842
##              Prevalence : 0.8482
##          Detection Rate : 0.8221
##    Detection Prevalence : 0.9176
##       Balanced Accuracy : 0.6704
##
##        'Positive' Class : 0
##
```

- 379 instances where the model predicted class 0 when the actual class was also 0. There were 26 instances where the model predicted class 1 when the actual class was 1.
- the model achieved an accuracy of 0.8785, meaning it correctly classified approximately 87.85% of the test instances.
- The 95% confidence interval - the true accuracy is estimated to be within the range of 0.8452 to 0.9069.
- The no-information rate (NIR) is the accuracy that could be achieved by always predicting the most frequent class. In this case, the most frequent class is 0, and the no-information rate is 0.8482.

- The Cohen's Kappa statistic measures the agreement between the predicted and actual class labels, while accounting for the agreement that could occur by chance. A value closer to 1 indicates better agreement than would be expected by chance alone. Here, a Kappa of 0.4194 suggests moderate agreement.
- Sensitivity (also known as True Positive Rate) measures the proportion of actual positive instances that were correctly predicted. Specificity (also known as True Negative Rate) measures the proportion of actual negative instances that were correctly predicted. In this case, the model has high sensitivity (correctly predicting positives) but low specificity (correctly predicting negatives).

### Challenge 1. Can you add feature selection to the pipeline?

```
# Load libraries
pacman::p_load(glmnet)

# Load the data and preprocess (as before)
data <- read.csv(url("https://raw.githubusercontent.com/sreckojoksimovic/infs5100/main/wine-data.csv"))

# Ensure 'quality_class' is a factor
data$quality_class <- as.factor(data$quality_class)

# List of numeric variable names to be scaled
numeric_vars <- c(
  "fixed_acidity", "volatile_acidity", "citric_acid",
  "residual_sugar", "chlorides", "free_sulfur_dioxide",
  "total_sulfur_dioxide", "density", "pH", "sulphates", "alcohol"
)

# Scale numeric variables using vectorized operations
data[numeric_vars] <- scale(data[numeric_vars], center = TRUE, scale = TRUE)

# Split data into training and test datasets. We will use 70%/30% split
set.seed(123)
dat.d <- sample(1:nrow(data), size=nrow(data)*0.7, replace = FALSE)
train.data <- data[dat.d,]
test.data <- data[-dat.d,]

# Split into predictors (X) and response (Y) for train and test data
x.train <- as.matrix(train.data[, numeric_vars])
y.train <- train.data$quality_class
x.test <- as.matrix(test.data[, numeric_vars])
y.test <- test.data$quality_class

# Use LASSO (L1 regularization) for feature selection
cv.fit <- cv.glmnet(x.train, y.train, family="binomial", alpha=1)

# Get the coefficients of the selected features
coef(cv.fit, s=cv.fit$lambda.min)
```

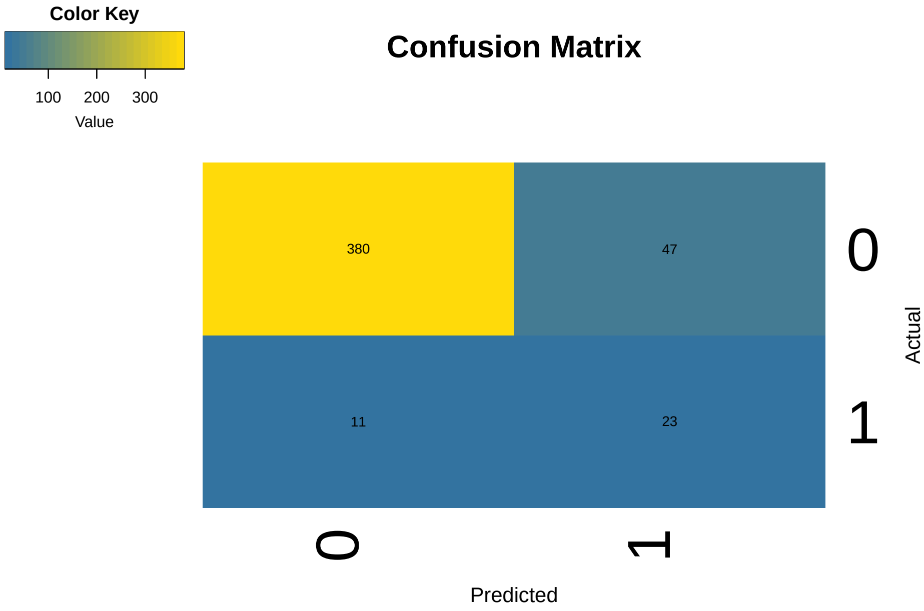
```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                 s1
## (Intercept)          -2.57852350
## fixed_acidity         0.33413646
## volatile_acidity     -0.29303019
## citric_acid           0.21304096
## residual_sugar        0.19384903
## chlorides            -0.41240296
## free_sulfur_dioxide   .
## total_sulfur_dioxide -0.29883855
## density              -0.41175645
## pH                   -0.01626956
## sulphates             0.52875584
## alcohol               0.76614711
```

```
# Build the final model with selected features
final.model <- glmnet(x.train, y.train, family="binomial", alpha=1, lambda=cv.fit$lambda.min)

# Predict using the test data
fitted.results <- predict(final.model, s=cv.fit$lambda.min, newx=x.test, type='response')
fitted.results <- ifelse(fitted.results > 0.5, 1, 0)

# Confusion matrix
confusionMatrix(as.factor(fitted.results), as.factor(y.test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 380  47
##          1  11  23
##
##                Accuracy : 0.8742
##                  95% CI : (0.8404, 0.9031)
##     No Information Rate : 0.8482
##     P-Value [Acc > NIR] : 0.06514
##
##                   Kappa : 0.3808
##
##  Mcnemar's Test P-Value : 4.312e-06
##
##             Sensitivity : 0.9719
##             Specificity : 0.3286
##          Pos Pred Value : 0.8899
##          Neg Pred Value : 0.6765
##              Prevalence : 0.8482
##          Detection Rate : 0.8243
##    Detection Prevalence : 0.9262
##       Balanced Accuracy : 0.6502
##
##        'Positive' Class : 0
##
```

```
cm <- confusionMatrix(as.factor(fitted.results), as.factor(y.test))
pacman::p_load(gplots)
```

```r
# Create the heatmap with numbers
heatmap.2(as.matrix(cm$table),
          xlab = "Predicted", ylab = "Actual",
          main = "Confusion Matrix",
          col = colorRampPalette(c("#3373a0", "#ffda0a"))(25),
          trace = "none", # removes the trace lines
          density.info = "none", # turns off density plot inside color legend
          dendrogram = "none", # suppresses row dendrogram
          Rowv = FALSE, Colv = FALSE, # suppresses column dendrogram
          margins = c(5,5), # sets margins
          symbreaks = FALSE, # ensures that breaks are at pretty intervals
          cellnote = cm$table, # same data set for cell labels
          notecol="black", # change font color of cell labels to black
          notecex=0.8) # change font size of cell labels
```



1. Accuracy: With Feature Selection: 87.42% Without Feature Selection: 87.85% Discussion: The accuracy is slightly higher without feature selection, but the difference is small. It may indicate that all or most features are relevant, and feature selection did not remove redundant information.
2. Sensitivity (True Positive Rate): With Feature Selection: 97.19% Without Feature Selection: 96.93% Discussion: Sensitivity is slightly higher with feature selection. This metric shows the ability of the model to correctly identify the positive class (0 in this case), and both models perform similarly in this aspect.
3. Specificity (True Negative Rate): With Feature Selection: 32.86% Without Feature Selection: 37.14% Discussion: Specificity is slightly higher without feature selection. This metric represents the ability to correctly identify the negative class (1 in this case), and the result without feature selection is slightly better in this respect.
4. Positive Predictive Value (Precision): With Feature Selection: 88.99% Without Feature Selection: 89.60% Discussion: Precision is slightly higher without feature selection, reflecting a marginally better ability to avoid false-positive predictions.
5. Negative Predictive Value: With Feature Selection: 67.65% Without Feature Selection: 68.42% Discussion: This metric reflects the ability to avoid false-negative predictions. The result without feature selection is slightly better here as well.

6. Balanced Accuracy: With Feature Selection: 65.02% Without Feature Selection: 67.04% Discussion: Balanced accuracy takes into account both sensitivity and specificity, and it’s higher without feature selection.
7. Kappa Statistic: With Feature Selection: 0.3808 Without Feature Selection: 0.4194 Discussion: The Kappa statistic measures the agreement between predicted and actual classifications, adjusted for what would be expected by chance. A higher Kappa indicates better agreement, so the result without feature selection is preferable. Overall Discussion: Both models perform similarly, with the one without feature selection having a slight edge in most metrics. The slight differences may not be substantial enough to make a strong conclusion about the relative merits of including or excluding feature selection. However, the model without feature selection has a higher Kappa statistic, which indicates better agreement between predicted and actual classifications. This suggests that the model without feature selection is preferable.

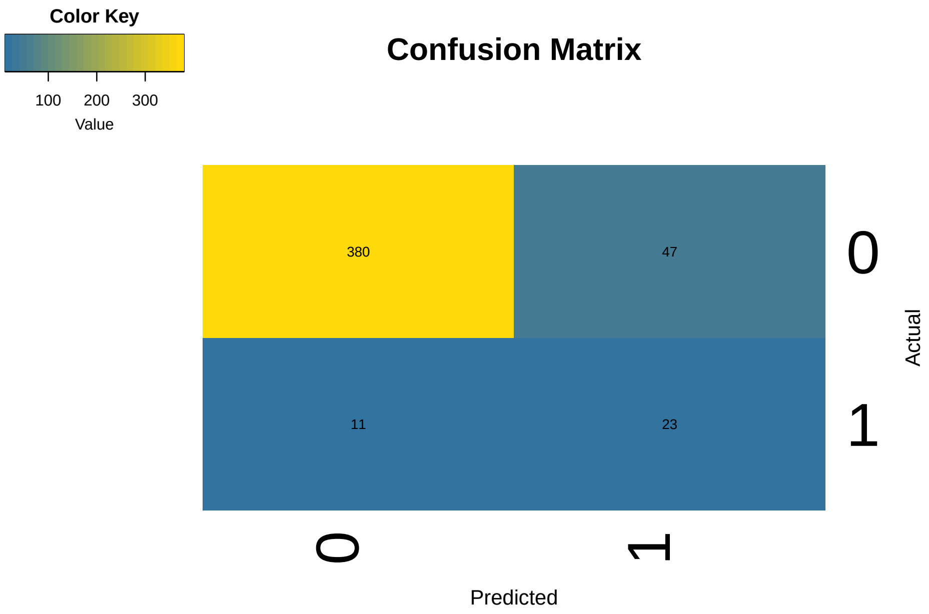
### Grid Search

```r
# Grid of hyperparameters
grid <- expand.grid(alpha = seq(0, 1, by = 0.1), lambda = seq(0.0001, 0.1, by = 0.001))

# Train model with grid search
cv.fit <- train(x.train, y.train, method = "glmnet",
                trControl = trainControl(method = "cv"),
                tuneGrid = grid)

# Get best hyperparameters
best_alpha <- cv.fit$bestTune$alpha
best_lambda <- cv.fit$bestTune$lambda

# Build the final model with selected hyperparameters
final.model <- glmnet(x.train, y.train, family="binomial", alpha = best_alpha, lambda = best_lambda)

# Predict using the test data
fitted.results <- predict(final.model, s=best_lambda, newx=x.test, type='response')
fitted.results <- ifelse(fitted.results > 0.5, 1, 0)

# Confusion matrix
confusionMatrix(as.factor(fitted.results), as.factor(y.test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 380  47
##          1  11  23
##
##                Accuracy : 0.8742
##                  95% CI : (0.8404, 0.9031)
##     No Information Rate : 0.8482
##     P-Value [Acc > NIR] : 0.06514
##
##                   Kappa : 0.3808
##
##  Mcnemar’s Test P-Value : 4.312e-06
##
```

```
##             Sensitivity : 0.9719
##             Specificity : 0.3286
##          Pos Pred Value : 0.8899
##          Neg Pred Value : 0.6765
##              Prevalence : 0.8482
##          Detection Rate : 0.8243
##    Detection Prevalence : 0.9262
##       Balanced Accuracy : 0.6502
##
##        'Positive' Class : 0
##
```

```
cm <- confusionMatrix(as.factor(fitted.results), as.factor(y.test))
# Create the heatmap with numbers
heatmap.2(as.matrix(cm$table),
          xlab = "Predicted", ylab = "Actual",
          main = "Confusion Matrix",
          col = colorRampPalette(c("#3373a0", "#ffda0a"))(25),
          trace = "none", # removes the trace lines
          density.info = "none", # turns off density plot inside color legend
          dendrogram = "none", # suppresses row dendrogram
          Rowv = FALSE, Colv = FALSE, # suppresses column dendrogram
          margins = c(5,5), # sets margins
          symbreaks = FALSE, # ensures that breaks are at pretty intervals
          cellnote = cm$table, # same data set for cell labels
          notecol="black", # change font color of cell labels to black
          notecex=0.8) # change font size of cell labels
```



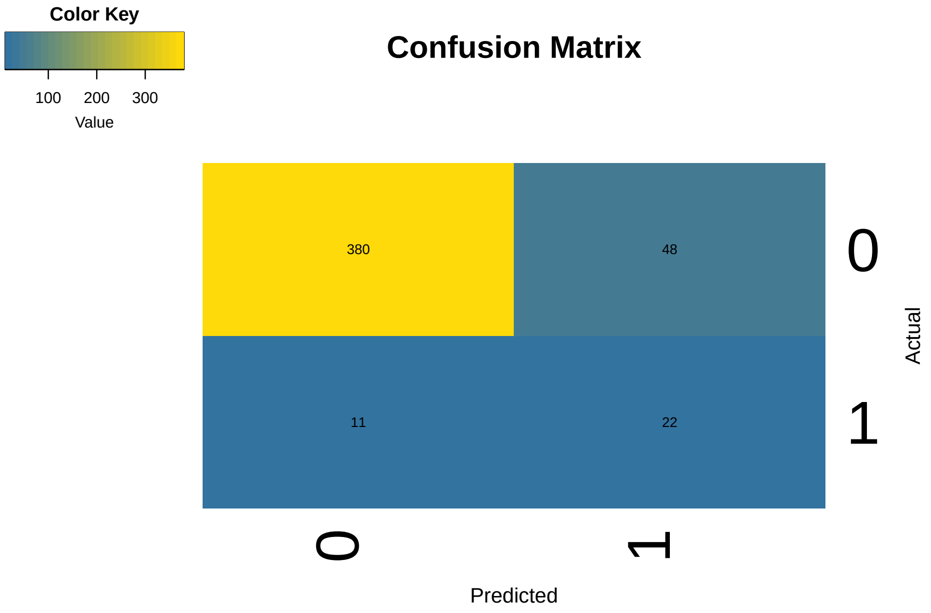
### Random Search

```
# Random hyperparameters
set.seed(123)
random_params <- data.frame(alpha = runif(100, 0, 1), lambda = runif(100, 0.0001, 0.1))
```

```r
# Train model with random search
cv.fit <- train(x.train, y.train, method = "glmnet",
                trControl = trainControl(method = "cv", search = "random"),
                tuneGrid = random_params)

# Get best hyperparameters
best_alpha <- cv.fit$bestTune$alpha
best_lambda <- cv.fit$bestTune$lambda

# Build the final model with selected hyperparameters
final.model <- glmnet(x.train, y.train, family="binomial", alpha = best_alpha, lambda = best_lambda)

# Predict using the test data
fitted.results <- predict(final.model, s=best_lambda, newx=x.test, type='response')
fitted.results <- ifelse(fitted.results > 0.5, 1, 0)

# Confusion matrix
confusionMatrix(as.factor(fitted.results), as.factor(y.test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 380  48
##          1  11  22
##
##                Accuracy : 0.872
##                  95% CI : (0.838, 0.9011)
##     No Information Rate : 0.8482
##     P-Value [Acc > NIR] : 0.08425
##
##                   Kappa : 0.3654
##
##  Mcnemar's Test P-Value : 2.775e-06
##
##             Sensitivity : 0.9719
##             Specificity : 0.3143
##          Pos Pred Value : 0.8879
##          Neg Pred Value : 0.6667
##              Prevalence : 0.8482
##          Detection Rate : 0.8243
##    Detection Prevalence : 0.9284
##       Balanced Accuracy : 0.6431
##
##        'Positive' Class : 0
##
```

```r
cm <- confusionMatrix(as.factor(fitted.results), as.factor(y.test))
# Create the heatmap with numbers
heatmap.2(as.matrix(cm$table),
          xlab = "Predicted", ylab = "Actual",
          main = "Confusion Matrix",
          col = colorRampPalette(c("#3373a0", "#ffda0a"))(25),
          trace = "none", # removes the trace lines
          density.info = "none", # turns off density plot inside color legend
          dendrogram = "none", # suppresses row dendrogram
          Rowv = FALSE, Colv = FALSE, # suppresses column dendrogram
          margins = c(5,5), # sets margins
          symbreaks = FALSE, # ensures that breaks are at pretty intervals
          cellnote = cm$table, # same data set for cell labels
          notecol="black", # change font color of cell labels to black
          notecex=0.8) # change font size of cell labels
```


Color Key
Value
Confusion Matrix
Actual
Predicted