Lab4

Guillermo Romero

2023-02-07

Lab 4: Fire and Tree Mortality

The database we'll be working with today includes 36066 observations of individual trees involved in prescribed fires and wildfires occurring over 35 years, from 1981 to 2016. It is a subset of a larger fire and tree mortality database from the US Forest Service (see data description for the full database here: link). Our goal today is to predict the likelihood of tree mortality after a fire.

Data Exploration

Outcome variable: *yr1status* = tree status (0=alive, 1=dead) assessed one year post-fire.

Predictors: *YrFireName, Species, Genus_species, DBH_cm, CVS_percent, BCHM_m, BTL* (Information on these variables available in the database metadata (link)).

```
trees_dat<- read_csv(file = "https://raw.githubusercontent.com/MaRo406/eds-232-machine-learning/main/da
```

```
## New names:
## Rows: 36066 Columns: 9
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (3): YrFireName, Species, Genus_species dbl (6): ...1, yr1status, DBH_cm,
## CVS_percent, BCHM_m, BTL
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
trees_dat <- trees_dat[,-1]
```

Question 1: Recode all the predictors to a zero_based integer form

```
trees_dat_0<- recipe(yr1status ~ ., data = trees_dat) %>%
  step_integer(all_predictors(), zero_based = TRUE) |>
  prep(trees_dat) |>
  bake(trees_dat)
trees_dat
```

```
## # A tibble: 36,066 x 8
##    yr1status YrFireName    Species DBH_cm Genus_species CVS_percent BCHM_m   BTL
##        <dbl> <chr>         <chr>    <dbl> <chr>               <dbl>  <dbl> <dbl>
##  1         1 2006 - Tripod 2TREE    26.4  Unknown               100   21.0      0
```

```
##   2          1 2006 - Tripod 2TREE     1.27 Unknown                100   0.61     0
##   3          1 2006 - Tripod 2TREE    25.4  Unknown                100  18.3      0
##   4          1 2006 - Tripod 2TREE     8.38 Unknown                100  12.8      0
##   5          1 2006 - Tripod 2TREE    21.8  Unknown                100  11.9      0
##   6          1 2006 - Tripod 2TREE    20.8  Unknown                100  14.3      0
##   7          1 2006 - Tripod 2TREE     8.64 Unknown                100   4.88     0
##   8          1 2006 - Tripod 2TREE     2.29 Unknown                100   1.83     0
##   9          1 2006 - Tripod 2TREE     6.10 Unknown                100   6.71     0
##  10          1 2006 - Tripod 2TREE    11.9  Unknown                100   5.18     0
## # ... with 36,056 more rows
```

Data Splitting

> Question 2: Create trees_training (70%) and trees_test (30%) splits for the modeling

```r
# Create training (70%) and test (30%) sets for the
set.seed(123)  # for reproducibility (random sample)
trees_split <- initial_split(trees_dat_0, prop = 0.70)
trees_train <- training(trees_split)
trees_test  <- testing(trees_split)
```

> Question 3: How many observations are we using for training with this split?

```r
trees_split
```

```
## <Training/Testing/Total>
## <25246/10820/36066>
```

We are using 25246 observations for this training split

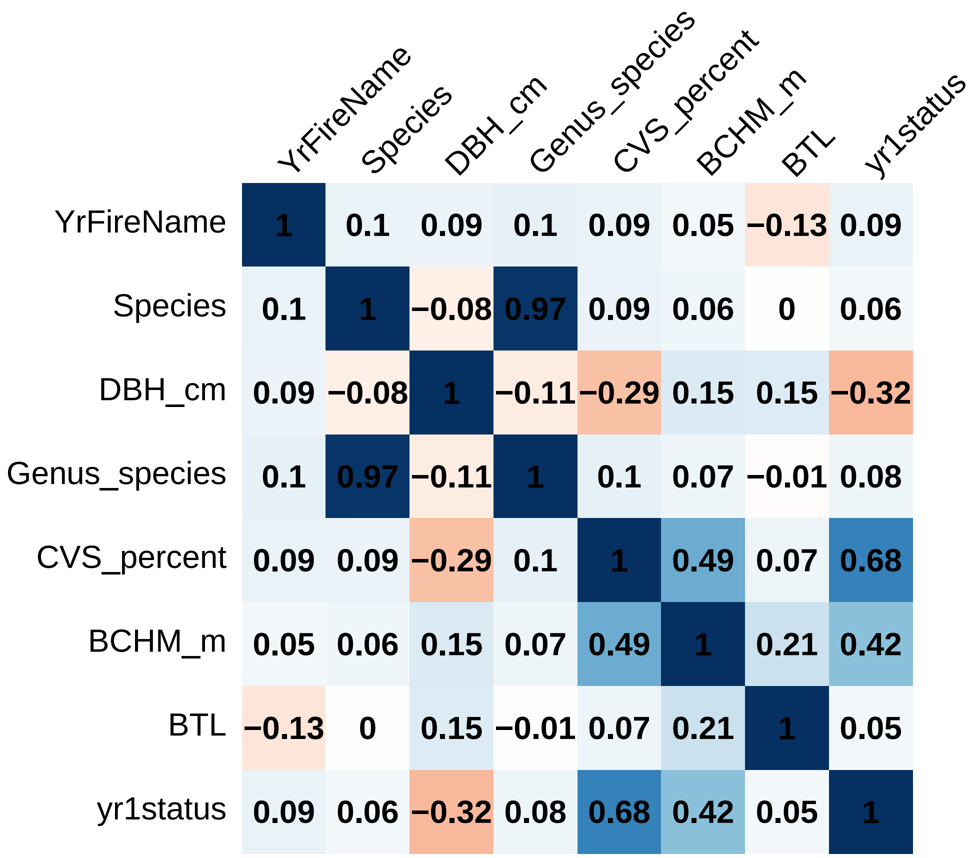
Simple Logistic Regression

Let's start our modeling effort with some simple models: one predictor and one outcome each.

> Question 4: Choose the three predictors that most highly correlate with our outcome variable for further investigation.

```r
cor_mat <- cor(trees_train)
```

```r
corrplot(cor_mat, method = "shade", shade.col = NA, tl.col = "black", tl.srt = 45, addCoef.col = "black
```



DBH_cm, BCHM_m, and CVS_percent are the most highly correlated with the outcome variable.

> Question 5: Use glm() to fit three simple logistic regression models, one for each of the predictors you identified.

```r
# simple logistic model for yr1status ~ DBH_cm
DBH_model <- glm(data = trees_train,
              yr1status ~ DBH_cm,
              family = "binomial")
# simple logistic model for yr1status ~ BCHM_m
BCHM_model <- glm(data = trees_train,
              yr1status ~ BCHM_m,
              family = "binomial")
# simple logistic model for yr1status ~ CVS_percent
CVS_model <- glm(data = trees_train,
              yr1status ~ CVS_percent,
              family = "binomial")
```

Interpret the Coefficients

We aren't always interested in or able to interpret the model coefficients in a machine learning task. Often predictive accuracy is all we care about.

Question 6: That said, take a stab at interpreting our model coefficients now.

```
exp(coef(DBH_model))
```

```
## (Intercept)      DBH_cm
##   1.4876101   0.9962419
```

```
exp(coef(BCHM_model))
```

```
## (Intercept)      BCHM_m
##   0.1387476   1.0061954
```

```
exp(coef(CVS_model))
```

```
## (Intercept) CVS_percent
## 0.001341998 1.079220197
```

Tree mortality in the DBH_model increases by 0.9962 for every 1 centimeter increase in diameter (DBH_cm).

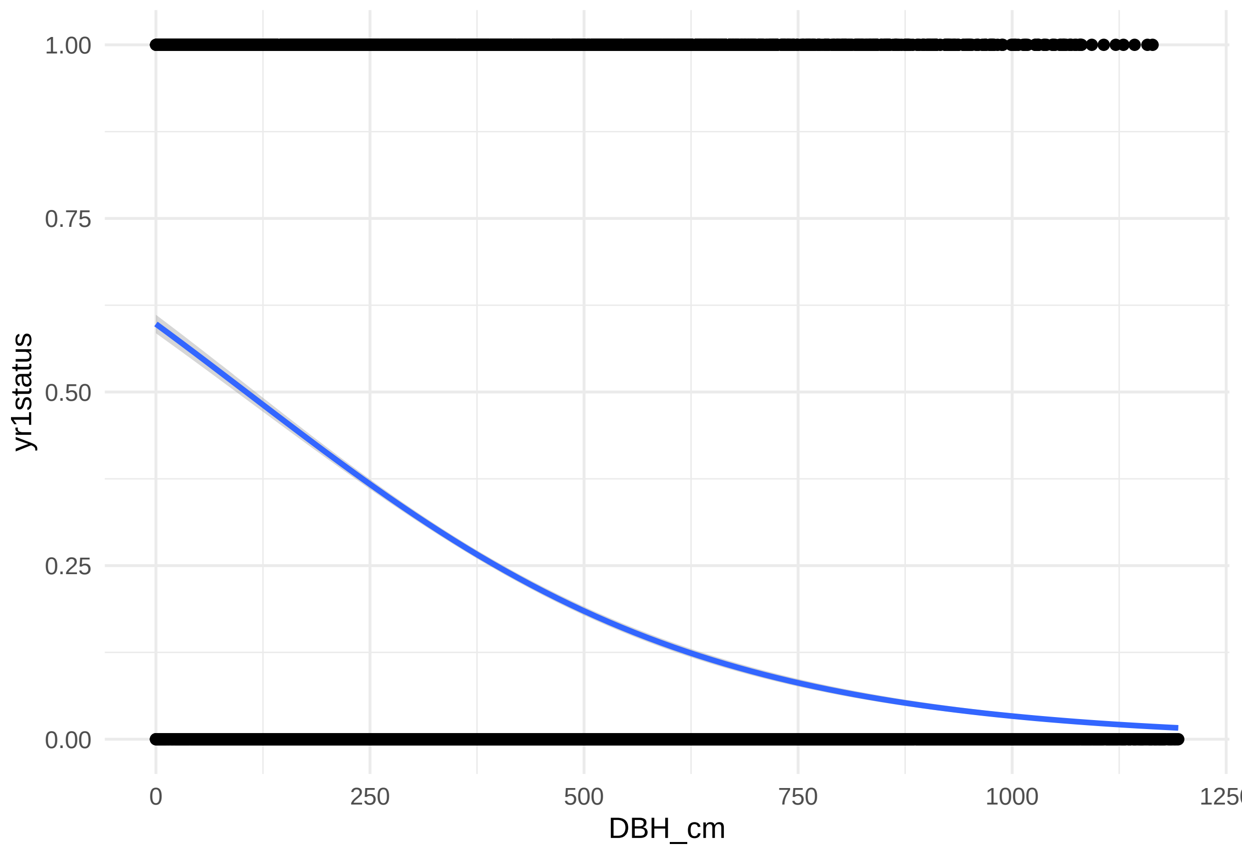
Tree mortality in the BCHM_model increases by 1.0062 for every 1 meter increase bark char (BCHM_m).

Tree mortality in the CVS_model increases by 1.0792 for every 1 percent of the pre-fire crown volume (CVS_percent).

Question 7: Now let's visualize the results from these models. Plot the fit to the training data of each model.

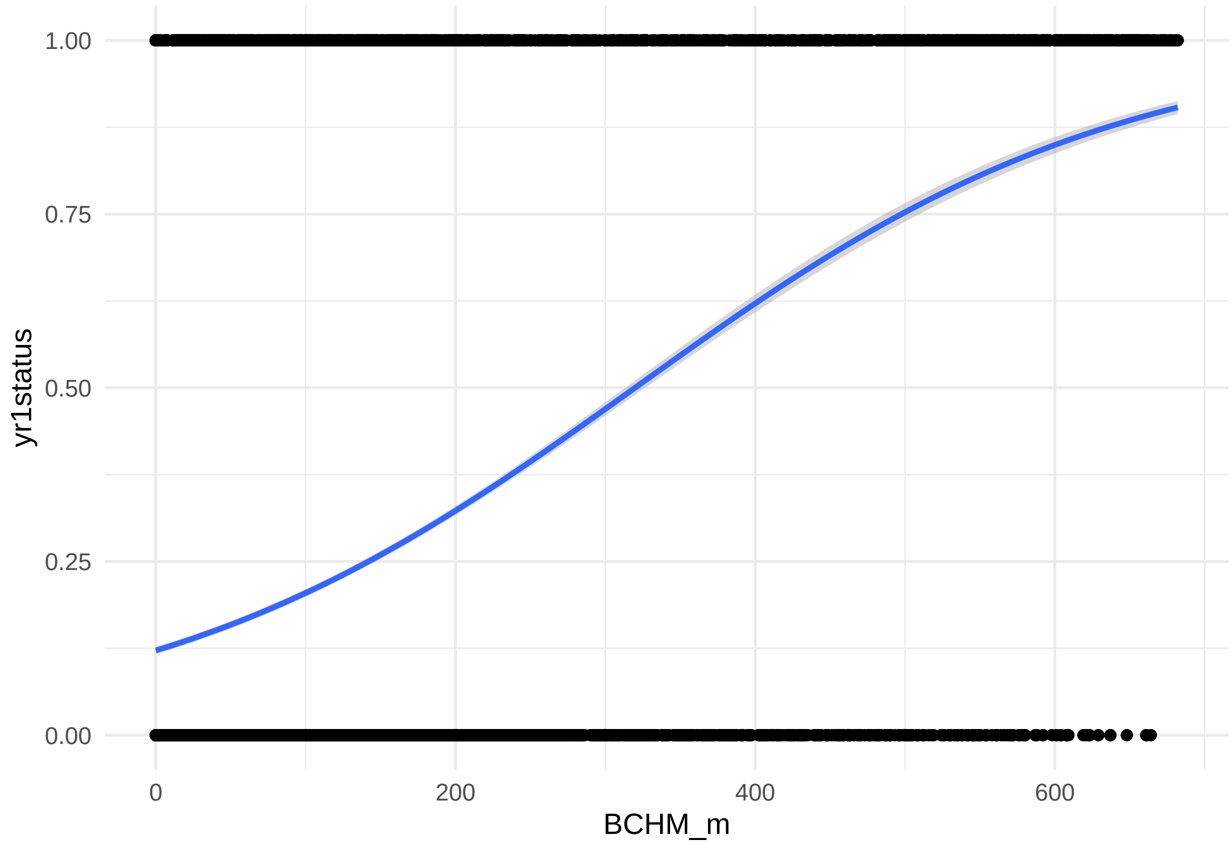
```
ggplot(trees_train,
       aes(x = DBH_cm,
           y = yr1status)) +
  geom_point() +
  stat_smooth(
    method = "glm",
    se = TRUE,
    method.args = list(family = binomial)
  ) +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



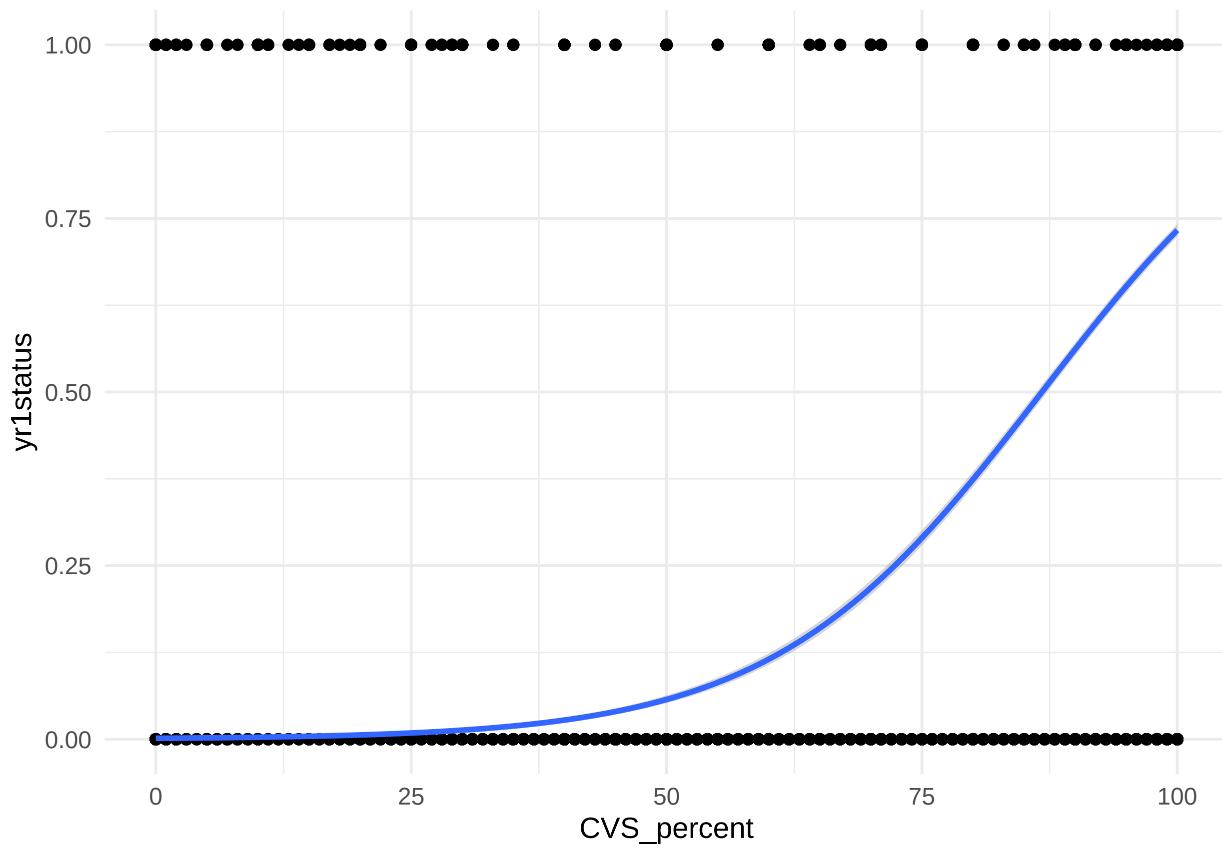

```
ggplot(trees_train,
       aes(x = BCHM_m,
           y = yr1status)) +
  geom_point() +
  stat_smooth(
    method = "glm",
    se = TRUE,
    method.args = list(family = binomial)
  ) +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(trees_train,
       aes(x = CVS_percent,
           y = yr1status)) +
  geom_point() +
  stat_smooth(
    method = "glm",
    se = TRUE,
    method.args = list(family = binomial)
  )  +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Multiple Logistic Regression

Let's not limit ourselves to a single-predictor model. More predictors might lead to better model performance.

> Question 8: Use glm() to fit a multiple logistic regression called "logistic_full", with all three of the predictors included. Which of these are significant in the resulting model?

```
logistic_full <- glm(yr1status ~ DBH_cm + BCHM_m + CVS_percent,
                     family = "binomial",
                     data = trees_train)
```

```
tidy(logistic_full)
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept) -5.09     0.114        -44.7 0
## 2 DBH_cm      -0.00371  0.000118     -31.4 2.39e-216
## 3 BCHM_m       0.00466  0.000161      28.9 6.05e-184
## 4 CVS_percent  0.0622   0.00119       52.4 0
```

All three are significant as they are all «< 0.5, relative to each other pre-fire crown volume (CVS_percent) is most signficant.

Estimate Model Accuracy

Now we want to estimate our model's generalizability using resampling.

> Question 9: Use cross validation to assess model accuracy. Use caret::train() to fit four 10-fold cross-validated models (cv_model1, cv_model2, cv_model3, cv_model4) that correspond to each of the four models we've fit so far: three simple logistic regression models corresponding to each of the three key predictors (CVS_percent, DBH_cm, BCHM_m) and a multiple logistic regression model that combines all three predictors.

```r
#Hint: resamples() wont give you what you need unless you convert the outcome variable to factor form
trees_train$yr1status <- as.factor(trees_train$yr1status)
```

```r
# 10-fold cross-validation on simple logistic regression model yr1status  ~ DBH_cm
set.seed(123)
cv_model1 <- train(
  yr1status ~ DBH_cm,
  data = trees_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)
```

```r
# 10-fold cross-validation on simple logistic regression model yr1status  ~ BCHM_m
set.seed(123)
cv_model2 <- train(
  yr1status ~ BCHM_m,
  data = trees_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)
```

```r
# 10-fold cross-validation on simple logistic regression model yr1status  ~ CVS_percent pre-fire crown
set.seed(123)
cv_model3 <- train(
  yr1status ~ CVS_percent,
  data = trees_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)
```

```r
#10-fold cross-validation on our multiple logistic regression model yr1status  ~ DBH_cm, BCHM_m, CVS_pe
set.seed(123)
cv_model4 <- train(
  yr1status ~ DBH_cm + BCHM_m + CVS_percent,
  data = trees_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)
```

> Question 10: Use caret::resamples() to extract then compare the classification accuracy for each model. (Hint: resamples() wont give you what you need unless you convert the outcome variable to factor form). Which model has the highest accuracy?

```r
#extract out of sample performance measures
summary(resamples(
  list(
    model1 = cv_model1,
    model2 = cv_model2,
    model3 = cv_model3,
    model4 = cv_model4
  )
))$statistics$Accuracy
```

```
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## model1 0.7437624 0.7475003 0.7534165 0.7522385 0.7555446 0.7603960    0
## model2 0.7588119 0.7658416 0.7717906 0.7714098 0.7758194 0.7837624    0
## model3 0.8899010 0.8923762 0.8962376 0.8975283 0.9006831 0.9080824    0
## model4 0.8902101 0.8969307 0.9037624 0.9031131 0.9093792 0.9144216    0
```

Model 4 has the highest accuracy with an average of 0.9031131

Let's move forward with this single most accurate model.

> Question 11: Compute the confusion matrix and overall fraction of correct predictions by the model.

```r
# predict class
pred_class <- predict(cv_model4, trees_train)

# create confusion matrix
confusionMatrix(
  data = relevel(pred_class, ref = "0"),
  reference = relevel(trees_train$yr1status, ref = "0")
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 16504   847
##          1  1595  6300
##
##                Accuracy : 0.9033
##                  95% CI : (0.8996, 0.9069)
##     No Information Rate : 0.7169
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.769
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9119
```

```
##             Specificity : 0.8815
##          Pos Pred Value : 0.9512
##          Neg Pred Value : 0.7980
##              Prevalence : 0.7169
##          Detection Rate : 0.6537
##    Detection Prevalence : 0.6873
##       Balanced Accuracy : 0.8967
##
##        'Positive' Class : 0
##
```

> Question 12: Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

The logistic regression mistakenly does not predict 847 occurrences of a tree being dead (yr1status = 1 = dead) when there was an actual event of a dead tree. The logistic regression also mistakenly predicted 1595 events of a dead tree when it was not.

> Question 13: What is the overall accuracy of the model? How is this calculated?

The overall accuracy is 0.9033, the overall accuracy is calculated by the sum of True Positives and True Negatives divided by the Total.

Test Final Model

Alright, now we'll take our most accurate model and make predictions on some unseen data (the test data).

> Question 14: Now that we have identified our best model, evaluate it by running a prediction on the test data, trees_test.

```r
trees_test$yr1status <- as.factor(trees_test$yr1status)

test_model_predict <- predict(cv_model4, trees_test)

confusionMatrix(
  data = relevel(test_model_predict, ref = "0"),
  reference = relevel(trees_test$yr1status, ref = "0")
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7013  362
##          1  721 2724
##
##                Accuracy : 0.8999
##                  95% CI : (0.8941, 0.9055)
##     No Information Rate : 0.7148
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.7628
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##            Sensitivity : 0.9068
##            Specificity : 0.8827
##         Pos Pred Value : 0.9509
##         Neg Pred Value : 0.7907
##             Prevalence : 0.7148
##         Detection Rate : 0.6482
##   Detection Prevalence : 0.6816
##      Balanced Accuracy : 0.8947
##
##       'Positive' Class : 0
##
```

Question 15: How does the accuracy of this final model on the test data compare to its cross validation accuracy? Do you find this to be surprising? Why or why not?

There is a difference of 0.0034 between the cross-validation accuracy (0.9033) and the final model(0.8999). This is not surprising as we used the predictors that are most highly correlated with the outcome variable, and the p-values for all three were very low indicating that they were all significant.