Lab4

Javier Patrón

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`
```

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

Data Splitting

```
trees <- trees_dat |>
  recipe(yr1status ~ .) |>
  step_integer(YrFireName, Species, Genus_species, zero_based = TRUE) |>
  prep() |>
  bake(new_data = NULL)
```

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

```
# Create a splitted data frame
trees_split <- trees |>
  initial_split(prop = 0.7, strata = "yr1status")

tree_train <- training(trees_split)
tree_test <- testing(trees_split)
```

Question 3: How many observations are we using for training with this split? *Response: For the training set we are using 25246 observations.*

```
dim(tree_train)[1]
```

```
## [1] 25246
```

Simple Logistic Regression

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

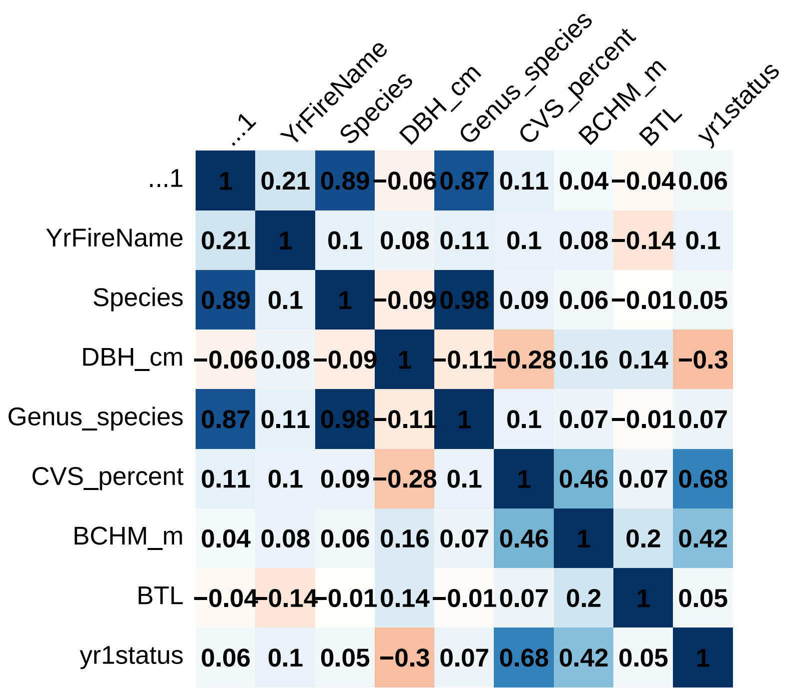
Question 4: Choose the three predictors that most highly correlate with our outcome variable for further investigation.

```
# Load the corrplot package
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
# Obtain correlation matrix
tree_matrix <- cor(trees)

# Make a correlation plot between the variables
corrplot(tree_matrix,
         method = "shade",
         shade.col = NA,
         tl.col = "black",
         tl.srt = 45,
         addCoef.col = "black",
         cl.pos = "n",
         order = "original")
```



Response: The three variables that have the highest correlation with our target variable (yr1status) are: CVS_Percent, BCHM_m, and DBH_cm. We will use this 3 variables as our predictors for the model. It is important to take into consideration the symbols as positive correlations will affect our target variable towards death (CVS, BCHM) and the negative correlation towards life (DBH).

Variables full names:

1. CVS_percent: Percent of the pre-fire crown volume that was scorched or consumed by fire (values 0 to 100). If measured, this is the CVS from field measurements. Otherwise it is the calculated CVS from crown length measurement.
2. BCHM_m: Maximum bark char (also called bole char, bole scorch in other publications) vertical height from ground on a tree bole, rounded to nearest 0.01 m.
3. DBH_cm: Diameter at breast height rounded to nearest 0.1 cm.

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

```r
model1_cvs <- glm(yr1status ~ CVS_percent,
              family = binomial,
              data= tree_train)

model2_bchm <- glm(yr1status ~ BCHM_m,
              data = tree_train,
              family = binomial)

model3_dbh <- glm(yr1status ~ DBH_cm,
              data = tree_train,
              family = binomial)
```

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

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.

```r
#Exponentiate the coefficients because they are log-transformed

exp(coef(model1_cvs))
```

```
## (Intercept) CVS_percent
## 0.001321742 1.079511396
```

```r
exp(coef(model2_bchm))
```

```
## (Intercept)      BCHM_m
##    0.152741    1.241098
```

```r
exp(coef(model3_dbh))
```

```
## (Intercept)      DBH_cm
##   1.6913593   0.9431536
```

Response: Predicting results

Model 1: The odds of a tree to be dead after 1 year of the fire (yr1status = 1), increases multiplicatively by 1.08 for each unit of `CVS_percent` (Crown Volume Scratch)

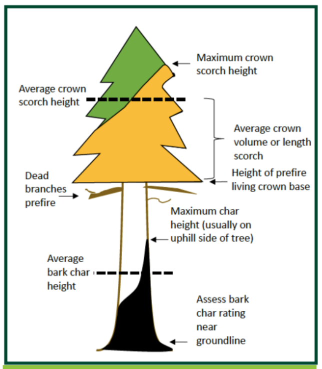
Model 2: The odds of a tree to be dead after 1 year of the fire (yr1status = 1), increases multiplicativly by 1.241 for each unit of `BCHM_m` (maximum Bark Char vertical Height from ground on a tree bole in meters).

Model 3: The odds of a tree to be dead after 1 year of the fire (yr1status = 1), increases multiplicativly by 0.943 for each unit of `DBH` (Diameter of Breast Height).

Just for context and understanding of the variables, here is a quick photo explaining the effects of some variables in tree fires.

How does fire kill trees?

Wildfire can directly kill trees by heating or burning the crown, trunk and/or roots, which damages photosynthetic and vascular tissues that support tree growth and defense. The ability for a tree to survive after a fire depends primarily on the magnitude of crown and cambium damage. Various conditions can influence these damage levels (e.g., fire intensity, duration, timing, heat conductivity across soil types, bark thickness, basal duff buildup, etc.).


Maximum crown scorch height
Average crown scorch height
Average crown volume or length scorch
Height of prefire living crown base
Dead branches prefire
Maximum char height (usually on uphill side of tree)
Average bark char height
Assess bark char rating near groundline

Crown scorch and bole char (USFS R6 FHP)

Crown damage occurs when needles are scorched or consumed by fire which disrupts photosynthesis. Buds and cones may also be damaged, which affects needle and seed production in the short term and tree survival and stand regeneration in the long term. Proportion of the live crown with crown scorch is the metric for crown damage.

Cambium damage occurs when the bole of a tree is charred or "cooked" enough to kill tissues. Cambium tissue includes both phloem and xylem, which are vascular tissues that transport water and nutrients throughout the tree. Trees with high levels of cambium damage but low crown scorch may take years to die from fire damage. For example, xylem tissues, which are deeper in the trunk, are unaffected by bole charring and continue to transport water to the crown for photosynthesis. While cambium and phloem tissues are closer to the surface and therefore more exposed to heat and bole char, which disrupts transport of nutrients from the crown to the roots. When the fine roots eventually start to die back, less water can be obtained and the tree starts to die. Proportions of the bole circumference with bole char or dead cambium is the metric for cambium damage.

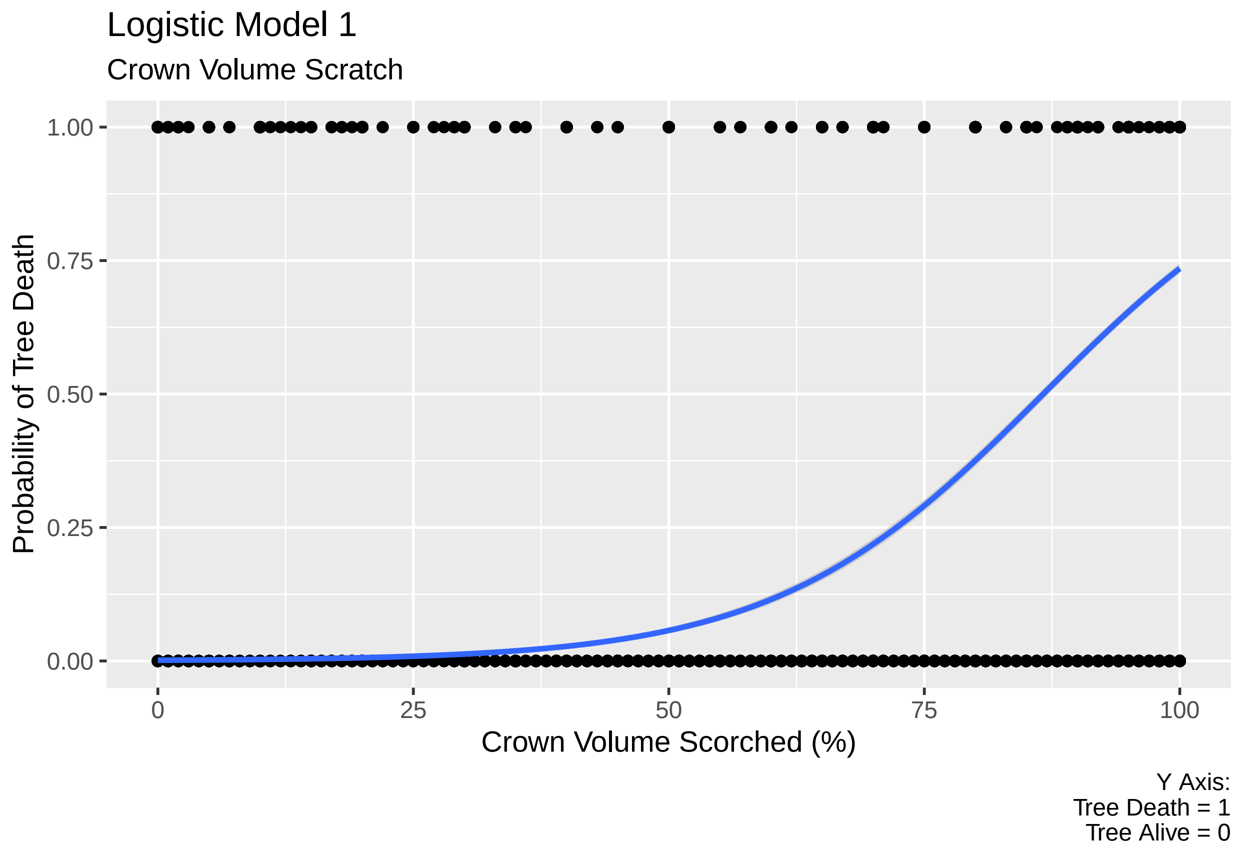
Image: How does fire kill trees? Oregon Department of Forestry (2020) Forest health fact sheet, Post-fire tree mortality. Available at: https://www.oregon.gov/odf/Documents/forestbenefits/post-fire-tree-mortality.pdf (Accessed: February 8, 2023).

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

Plotting the results:

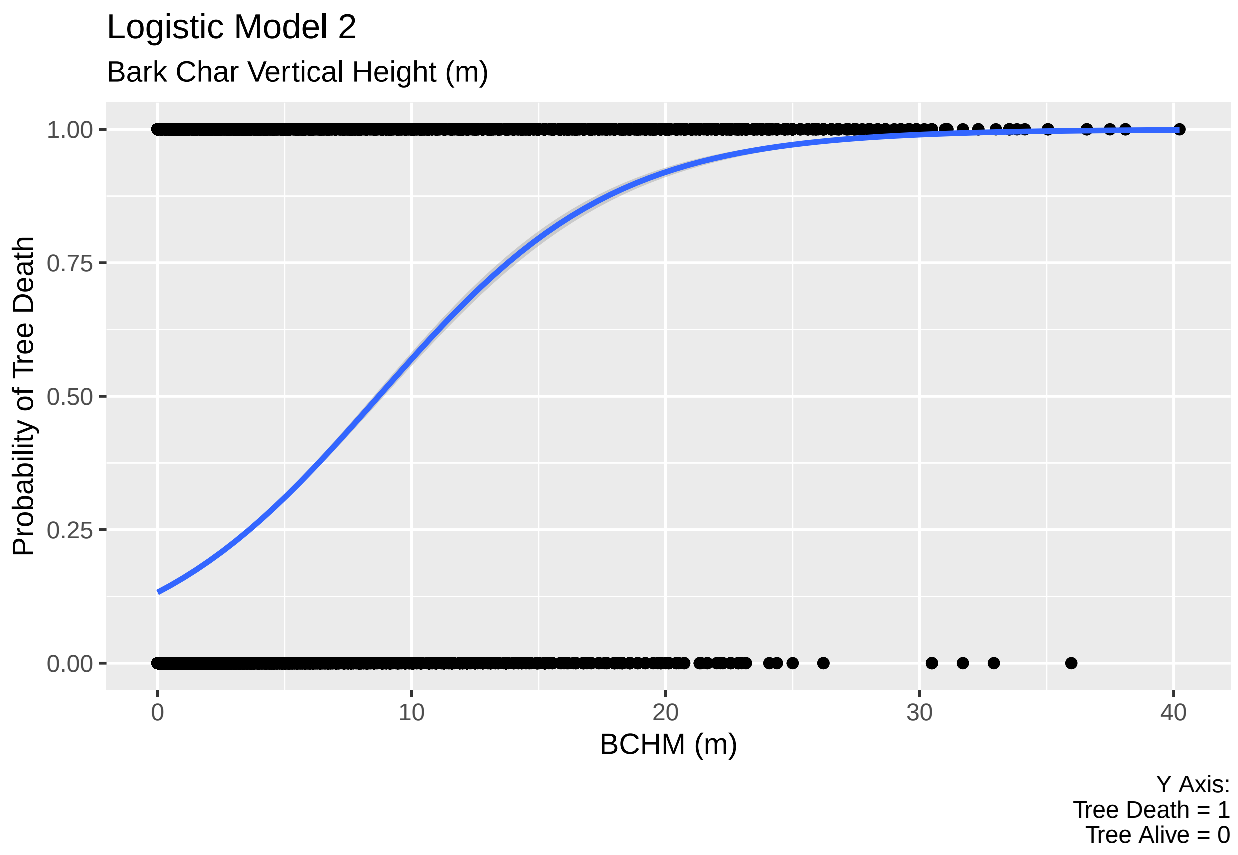
```
#Model 1
ggplot(tree_train, aes(x = CVS_percent, y = yr1status)) +
  geom_point() +
  labs(title = "Logistic Model 1",
       subtitle = "Crown Volume Scratch",
       y = "Probability of Tree Death",
       x = "Crown Volume Scorched (%)",
       caption = "Y Axis:\n Tree Death = 1\n Tree Alive = 0") +
 stat_smooth(method = "glm",
                  se = TRUE,
                  method.args = list(family = "binomial"))
```

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



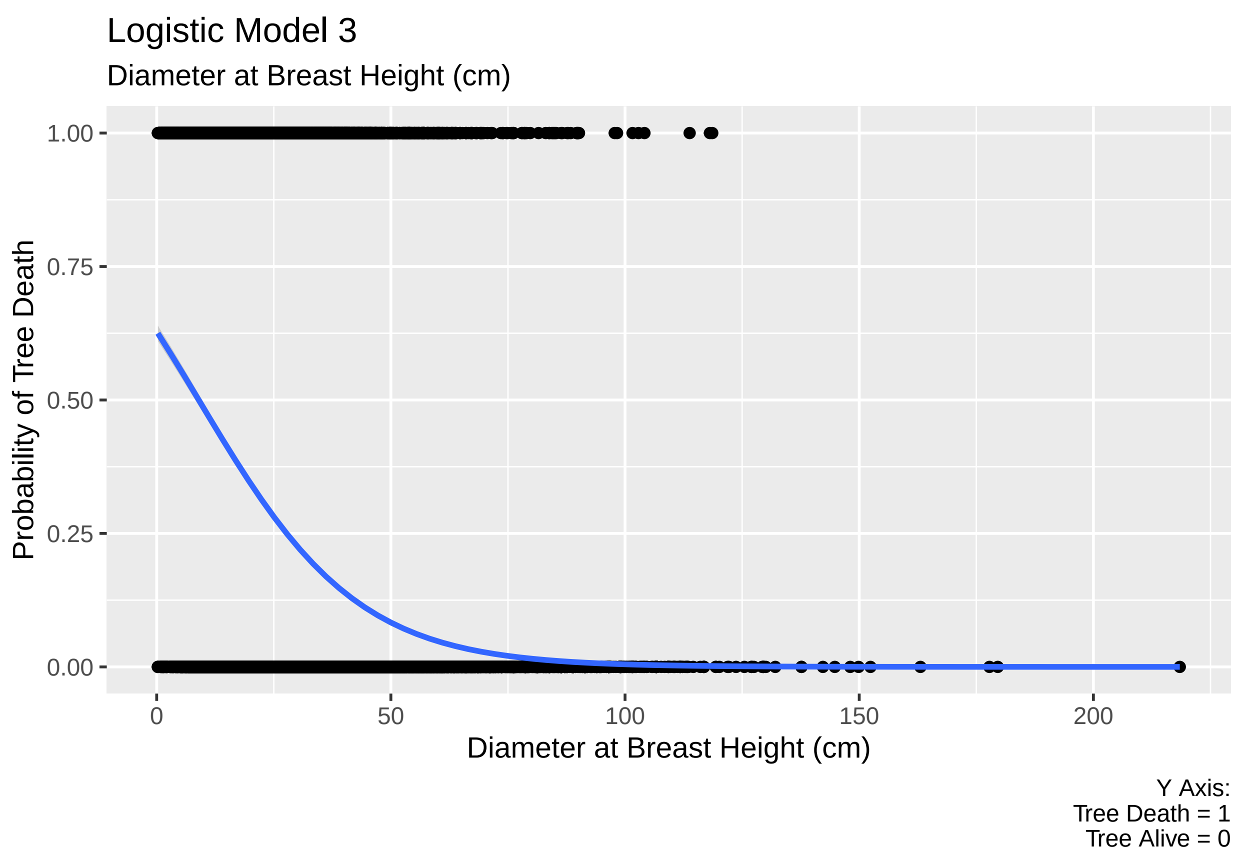
```
#Model 2
ggplot(tree_train, aes(x = BCHM_m, y = yr1status)) +
  geom_point() +
  labs(title = "Logistic Model 2",
       subtitle = "Bark Char Vertical Height (m)",
       y = "Probability of Tree Death",
       x = "BCHM (m)",
       caption = "Y Axis:\n Tree Death = 1\n Tree Alive = 0") +
 stat_smooth(method = "glm",
                  se = TRUE,
                  method.args = list(family = "binomial"))
```

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



```
#Model 3
ggplot(tree_train, aes(x = DBH_cm, y = yr1status)) +
  geom_point() +
  labs(title = "Logistic Model 3",
       subtitle = "Diameter at Breast Height (cm)",
       y = "Probability of Tree Death",
       x = "Diameter at Breast Height (cm)",
       caption = "Y Axis:\n Tree Death = 1\n Tree Alive = 0") +
 stat_smooth(method = "glm",
                  se = TRUE,
                  method.args = list(family = "binomial"))
```

```
## `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?

```
#Estimating the coefficients with a Multiple Logistic Regression

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

Response: In this case, the highest coefficient is now for the Bark Char Height with 0.158168123511389, but according to the p-value as they are all lower than 0.05 they are statistically significant.

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.

Check the class of my columns and factorize if needed

```
#Just informative
#skim(tree_train)
#levels(as.factor(tree_train$yr1status))
```

Model 1: Crown Volume Scratch in percentage

```
# First 10-fold cross-validation
# Model 1
set.seed(123)

cv_model1_cvs <- train(as.factor(yr1status) ~ CVS_percent,
                   data = tree_train,
                   method = "glm",
                   family = "binomial",
                   trControl = trainControl(method = "cv",
                                            number = 10))
```

Model 2: Bark Char Volume Height in Meters

```
# Second 10-fold cross-validation
# Model 2
set.seed(123)

cv_model2_bchm <- train(as.factor(yr1status) ~ BCHM_m,
                   data = tree_train,
                   method = "glm",
                   family = "binomial",
                   trControl = trainControl(method = "cv",
                                            number = 10))
```

Model 3: DBH_cm (Diameter at Breast Height in cm)

```
# Third 10-fold cross-validation
# Model 3
set.seed(123)

cv_model3_dbh <- train(as.factor(yr1status) ~ DBH_cm,
                   data = tree_train,
                   method = "glm",
                   family = "binomial",
                   trControl = trainControl(method = "cv",
                                            number = 10))
```

Model 4: Calculating the model with all 3 predicted variables

```
# Fourth 10-fold cross-validation
# Model 4
set.seed(123)

cv_model4_all <- train(as.factor(yr1status) ~ CVS_percent + BCHM_m + DBH_cm,
                   data = tree_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?

```
#Create a matrix that has a summary of the models

models_accuracy <- summary(
  resamples(
    list(
      mod1 = cv_model1_cvs,
      mod2 = cv_model2_bchm,
      mod3 = cv_model3_dbh,
      mod4 = cv_model4_all
    )
  )
)$statistics$Accuracy
```

Printing the model Accuracy

```
kable(models_accuracy,
      padding = 1,
      digits = round(3),
      align = "c",
      format = "pipe",
      caption = "Models Accurarcy Summary")
```

Table 1: Models Accurarcy Summary

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
mod1	0.889	0.894	0.900	0.899	0.904	0.906	0
mod2	0.766	0.772	0.773	0.773	0.775	0.780	0
mod3	0.740	0.744	0.750	0.748	0.752	0.752	0
mod4	0.891	0.901	0.903	0.904	0.908	0.915	0

Response: The highest accuracy model comes from the multiple logistic regression model (model4). It has an accuracy of 0.9038.

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.

```
#Overall predict class

predict_class <- predict(cv_model4_all, tree_train)
train_confusion_matrix <- confusionMatrix(data = predict_class,
                                          reference = as.factor(tree_train$yr1status))
```

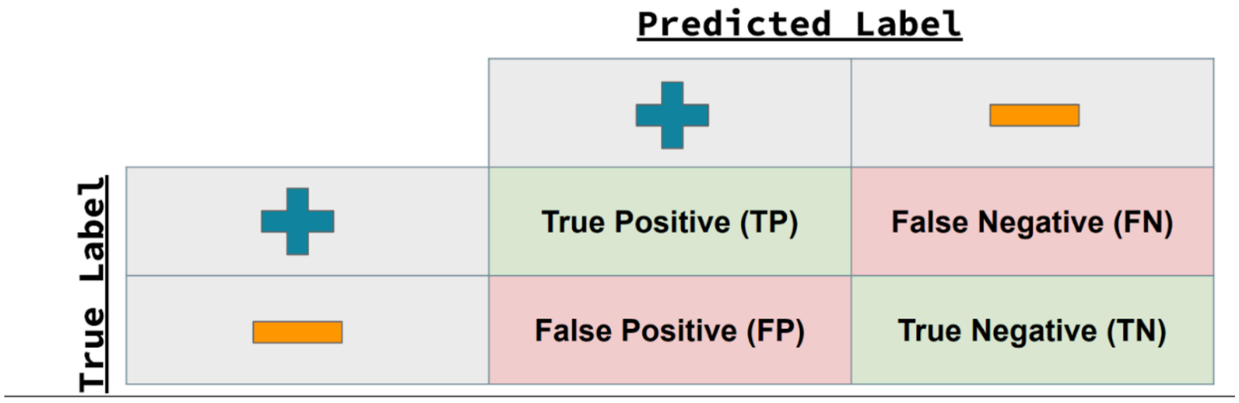
Question 12: Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

```r
knitr::kable(train_confusion_matrix$table,
             col.names = c("Predicted Alive", "Predicted Dead"),
             padding = 1,
             align = "c",
             format = "pipe",
             caption = "Training Confusion Matrix")
```

Table 2: Training Confusion Matrix

	Predicted Alive	Predicted Dead
0	16487	834
1	1596	6329

Response: The confusion Matrix is showing a high Accuracy with 0.9037471. (~ 90% accuracy). Going through the numbers in the matrix we have 16487as the True Positive (TP) observations and 6329 as True Negative (TN). On the other hand, there are 1596 within the False Positive (FP) observations saying that the observed trees are predicted alive, but their true value is that is dead. Following with 834 observations falling within the False Negative (FN) meaning that the predicted value is Dead Tree, but the true value is Alive.



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

```r
#Indexing
train_confusion_matrix$overall["Accuracy"]
```

```
##  Accuracy
## 0.9037471
```

Response: The overall accuracy of the model is ~ 90%. There is a formula to calculate the overall accuracy:

$$Overall\ Accuarcy = \frac{TP\ +\ TN}{\sum\ (Total\ Observations)}$$

```r
# Manual Calculation
(train_confusion_matrix$table[1] + train_confusion_matrix$table[4]) / dim(tree_train)[1]
```

```
## [1] 0.9037471
```

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
test_model4_all <- train(as.factor(yr1status) ~ CVS_percent + BCHM_m + DBH_cm,
                   data = tree_test,
                   method = "glm",
                   family = "binomial")

test_prediction <- predict(cv_model4_all, tree_test)

test_confusion_matrix <- confusionMatrix(data = test_prediction,
                 reference = as.factor(tree_test$yr1status))

knitr::kable(test_confusion_matrix$table,
             col.names = c("Predicted Alive", "Predicted Dead"),
             padding = 1,
             align = "c",
             format = "pipe",
             caption = "Test Confusion Matrix")
```

Table 3: Test Confusion Matrix

	Predicted Alive	Predicted Dead
0	7037	350
1	713	2720

> 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?

```r
train_confusion_matrix$overall["Accuracy"]
```

```
##  Accuracy
## 0.9037471
```

```r
test_confusion_matrix$overall["Accuracy"]
```

```
## Accuracy
## 0.901756
```

Response: As you can see in the results the Accuracy of the model in the test data is very similar to the training. This will confirm that the stratification data at the beginning of the split was done nicely and qualitatively. In other words, our testing model is very accurate on the true data giving us confidence to try it out in unknown data sets.

—————————————————————

EXTRA…..

Book Chapter 5: Logistic Regression

https://bradleyboehmke.github.io/HOML/logistic-regression.html

PLS Logistic Regression.- Assess if reducing the dimension of our numeric predictors helps to improve accuracy.

```r
#Creating the two pls models (training and testing)

set.seed(123)
cv_model_pls <- train(
  as.factor(yr1status) ~ .,
  data = tree_train,
  method = "pls",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10),
  preProcess = c("zv", "center", "scale"),
  tuneLength = 16)

test_model_pls <- train(
  as.factor(yr1status) ~ .,
  data = tree_test,
  method = "pls",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10),
  preProcess = c("zv", "center", "scale"),
  tuneLength = 16)
```

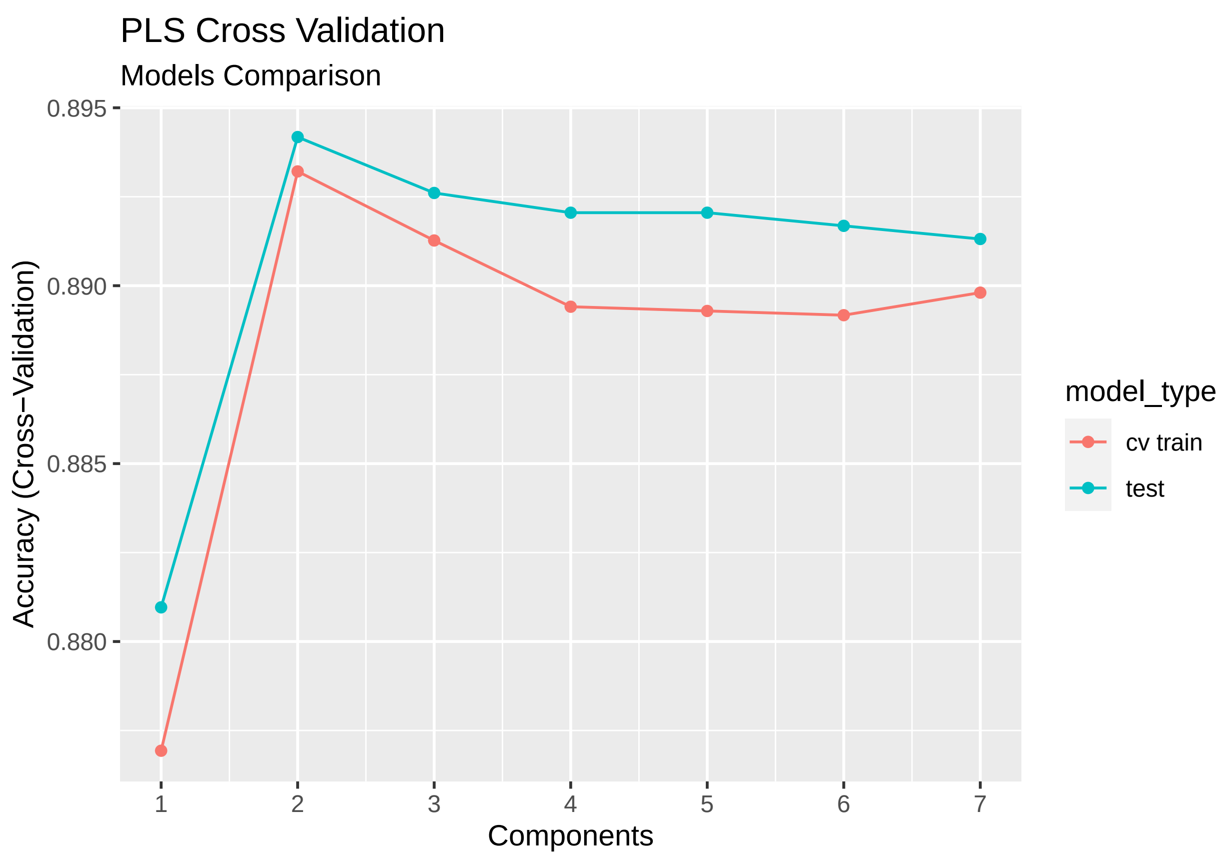
Graph the results

```r
df_test <- data.frame(test_model_pls$results[,1:2]) |>
  mutate(model_type = "test")
df_cv <- data.frame(cv_model_pls$results[,1:2]) |>
  mutate(model_type = "cv train")

model_merge <- rbind(df_test, df_cv)

x <- 1:7

ggplot(model_merge, aes(x = ncomp, y = Accuracy, col = model_type)) +
  geom_line() +
  geom_point() +
  labs(title = "PLS Cross Validation",
       subtitle = "Models Comparison",
       x = "Components",
       y = "Accuracy (Cross-Validation)") +
  scale_x_continuous(labels= as.character(x),breaks=x)
```



PLS Cross Validation
Models Comparison
Accuracy (Cross−Validation)
0.895
0.890
0.885
0.880
1
2
3
4
5
6
7
Components
model_type
cv train
test