Lab 4 Assignment

Lewis White

2023-02-09

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 (https://www.nature.com/articles/s41597-020-0522-7#Sec10)). 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 (https://www.fs.usda.gov/rds/archive/products/RDS-2020-0001-2/_metadata_RDS-2020-0001-2.html))).

```
#Reading in the data from github
trees_dat<- read_csv(file = "https://raw.githubusercontent.com/MaRo406/eds-232-machine-l
earning/main/data/trees-dat.csv")[,-1]
```

```
## 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

```
#recoding the prdictors

trees_encode <- recipe(yr1status ~ ., data = trees_dat) %>% #creates a recipe for the da
ta that defines the target variable "yr1status" and all other columns as predictors.
  step_integer(all_predictors(), zero_based = TRUE) %>% # applies integer encoding to al
l predictors and starts with 0
  prep(trees_dat) %>% #preprocessor to the "trees_dat" data
  bake(trees_dat) #creates a new dataset by applying the preprocessor to the "trees_dat"
data.
```

Data Splitting

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

```
set.seed(123) #set seed for reproducibility
tree_split <- initial_split(data = trees_encode, prop = 0.7) #create split proportion
tree_train <- training(tree_split) #create training data
tree_test <- testing(tree_split) #create testing data
```

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

```
paste("We are using", nrow(tree_train), "obersvations with this split.")
```

```
## [1] "We are using 25246 obersvations with this split."
```

Simple Logistic Regression

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

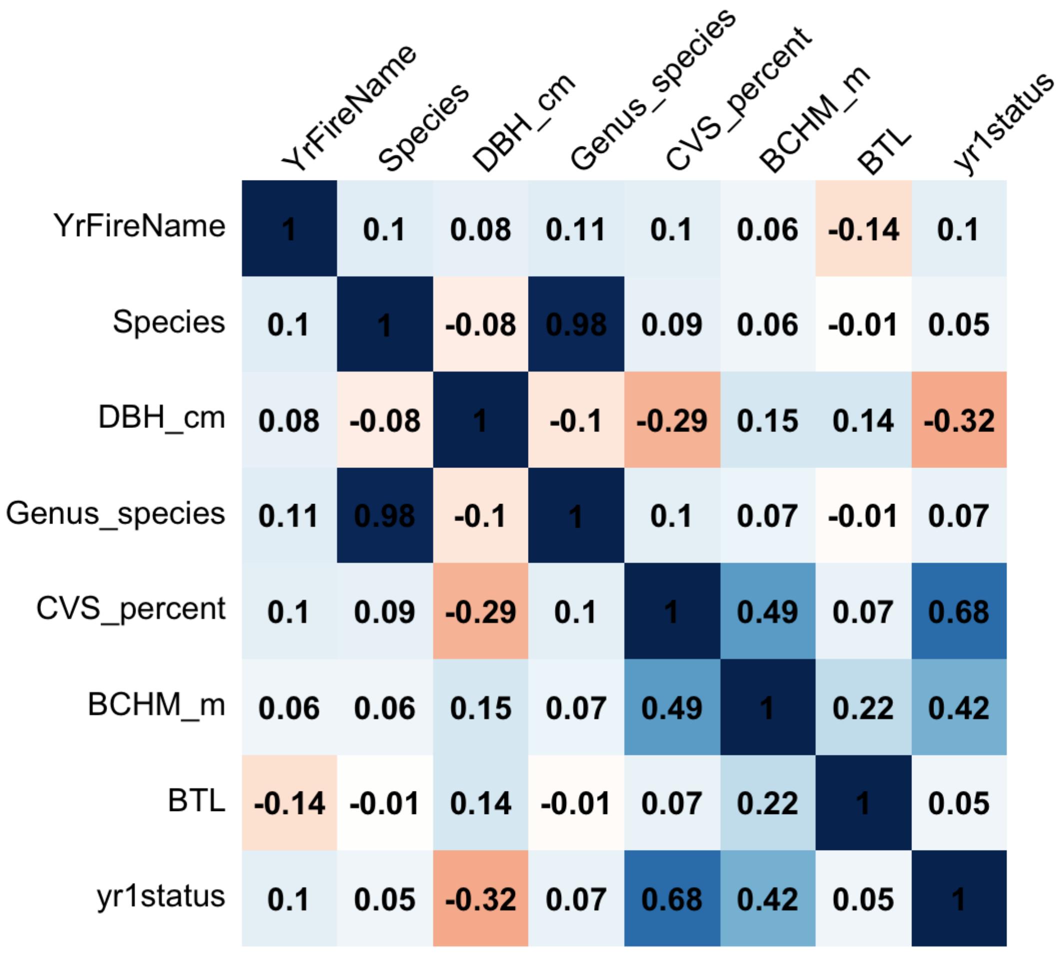
> Question 4: Choose the three predictors that most highly correlate with our outcome variable for further investigation.

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
# Obtain correlation matrix
corr_mat <- cor(trees_encode)

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



CVS_percent (crown volume scorched), BCHM_m (maximum burn char in meters), and DBH_cm (diameter at breast height in cm) have the strongest correlation with yr1status.

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

```r
#fitting a model on % crown volume scorched
cvs_model <- glm(data = tree_train, yr1status ~ CVS_percent, family = "binomial")

#fitting a model on maximum burn char (in meters)
bchm_model <- glm(data = tree_train, yr1status ~ BCHM_m, family = "binomial")

#fitting a model on diameter of the tree at breast height (in cm)
dbh_model <- glm(data = tree_train, yr1status ~ DBH_cm, 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.

```
paste0("A one percent increase in crown volume scorched increases the probability that a
tree will have died multiplicatively by ", exp(cvs_model$coefficients)[2])
```

```
## [1] "A one percent increase in crown volume scorched increases the probability that a
tree will have died multiplicatively by 1.07922019741702"
```

```
paste0("A one meter increase in the maximum burn char increase the probability that a tr
ee will have died multiplicatively by ", exp(bchm_model$coefficients)[2])
```

```
## [1] "A one meter increase in the maximum burn char increase the probability that a tr
ee will have died multiplicatively by 1.00619542136237"
```

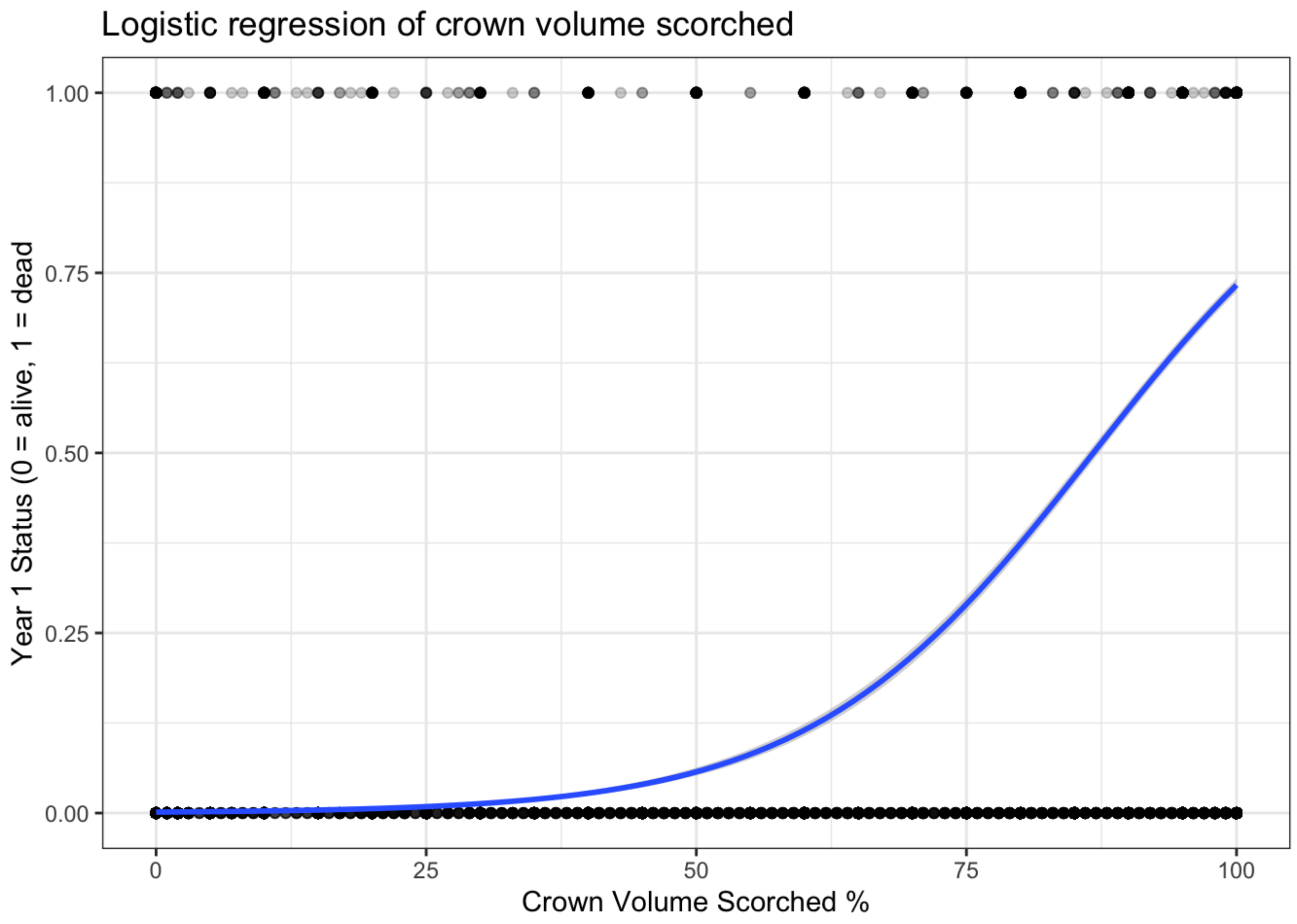
```
paste0("A one cm increase in the diameter of the tree at the breast height decreases the
probability that a tree will have died multiplicatively by ", 1 - exp(dbh_model$coeffici
ents)[2], " (e.g. p(dead) - p(dead) * ", 1 - exp(dbh_model$coefficients)[2], ")")
```

```
## [1] "A one cm increase in the diameter of the tree at the breast height decreases the
probability that a tree will have died multiplicatively by 0.00375813723556395 (e.g. p(d
ead) - p(dead) * 0.00375813723556395)"
```

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

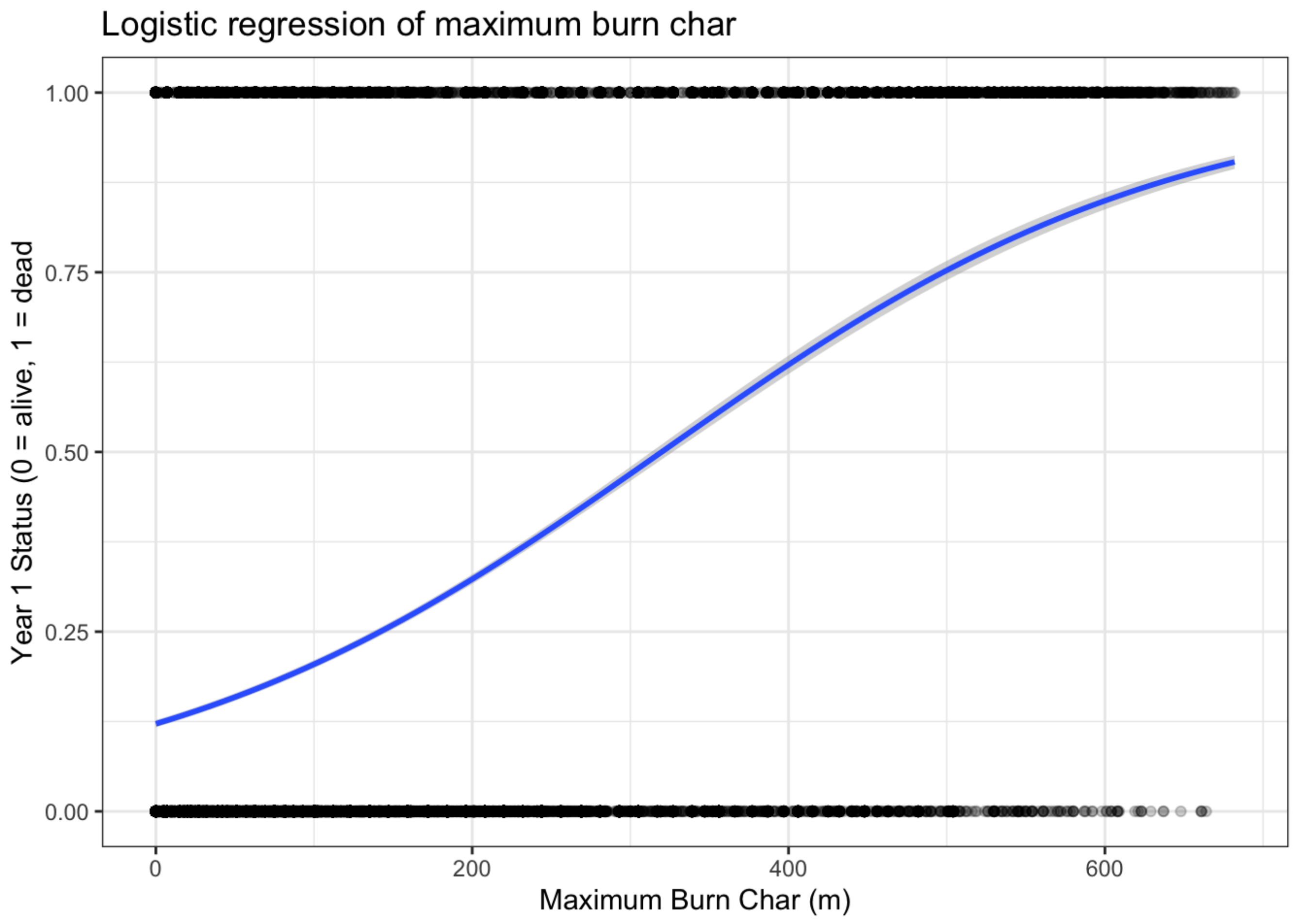
```
#PLOT OF CVS
 ggplot(tree_train, aes(x=CVS_percent, y=yr1status)) + geom_point(alpha = 0.2) +
      stat_smooth(method="glm",  se=TRUE,
                  method.args = list(family=binomial)) +
  theme_bw() +
  labs(x = "Crown Volume Scorched %",
       y = "Year 1 Status (0 = alive, 1 = dead",
       title = "Logistic regression of crown volume scorched")
```

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



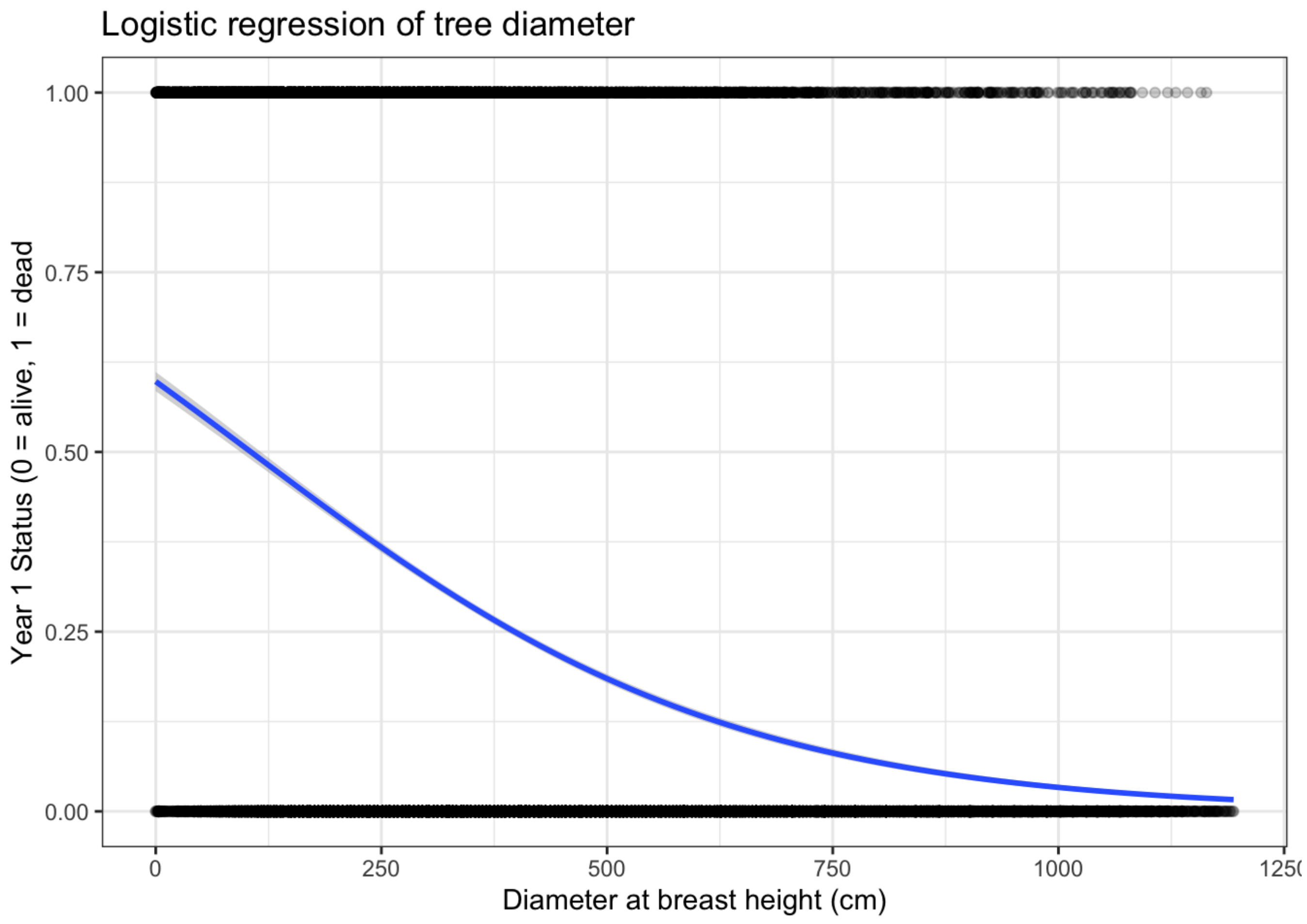
```
#PLOT OF MAXIMUM BURN CHAR AREA
 ggplot(tree_train, aes(x=BCHM_m, y=yr1status)) + geom_point(alpha = 0.2) +
      stat_smooth(method="glm",  se=TRUE,
                method.args = list(family=binomial)) +
  theme_bw() +
  labs(x = "Maximum Burn Char (m)",
       y = "Year 1 Status (0 = alive, 1 = dead",
       title = "Logistic regression of maximum burn char")
```

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



```
#PLOT OF DIAMETER AT BREAST HEIGHT
 ggplot(tree_train, aes(x=DBH_cm, y=yr1status)) + geom_point(alpha = 0.2) +
     stat_smooth(method="glm",  se=TRUE,
               method.args = list(family=binomial)) +
 theme_bw() +
 labs(x = "Diameter at breast height (cm)",
      y = "Year 1 Status (0 = alive, 1 = dead",
      title = "Logistic regression of tree diameter")
```

```
## `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(data = tree_train, yr1status ~ CVS_percent + BCHM_m + DBH_cm, famil
y = "binomial")

summary(logistic_full)
```

```
##
## Call:
## glm(formula = yr1status ~ CVS_percent + BCHM_m + DBH_cm, family = "binomial",
##     data = tree_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3843  -0.2419  -0.0674   0.4442   4.1464
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.0899300  0.1139418  -44.67   <2e-16 ***
## CVS_percent  0.0621854  0.0011878   52.35   <2e-16 ***
## BCHM_m       0.0046560  0.0001610   28.92   <2e-16 ***
## DBH_cm      -0.0037087  0.0001181  -31.39   <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: 30086  on 25245  degrees of freedom
## Residual deviance: 13386  on 25242  degrees of freedom
## AIC: 13394
##
## Number of Fisher Scoring iterations: 7
```

In the resulting model, all three variables appear to be significant at the 0.001 significance level.

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.

```
#convert yr1status to a factor
tree_train <- tree_train %>%
  mutate(yr1status = as.factor(yr1status)) # I converted the outcome variable to a facto
r after receiving the following warning message when running the code below: "In train.d
efault(x, y, weights = w, ...) :You are trying to do regression and your outcome only ha
s two possible values Are you trying to do classification? If so, use a 2 level factor a
s your outcome column."

#CVS model
set.seed(123)
cv_model1 <- train(
  yr1status ~ CVS_percent,
  data = tree_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

#BCHM model
set.seed(123)
cv_model2 <- train(
  yr1status ~ BCHM_m,
  data = tree_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10))

#DBH model
set.seed(123)
cv_model3 <- train(
  yr1status ~ DBH_cm,
  data = tree_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10))

#Full Model
set.seed(123)
cv_model4 <- train(
  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? Let's move forward with this single most accurate model.

```
#comparing the classification accuracy of the models
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 0.8899010 0.8923762 0.8962376 0.8975283 0.9006831 0.9080824    0
## model2 0.7588119 0.7658416 0.7717906 0.7714098 0.7758194 0.7837624    0
## model3 0.7437624 0.7475003 0.7534165 0.7522385 0.7555446 0.7603960    0
## model4 0.8902101 0.8969307 0.9037624 0.9031131 0.9093792 0.9144216    0
```

Model 4, the one that inlcudes all three predictors of interest, appears to have the highest accuracy across the board. We can proceed with this model.

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

```
#making predictions on the training data
pred_trees <- predict(cv_model4, data = tree_train)

#confusion matrix using the model predictions
confusionMatrix(
  data = factor(pred_trees),
  reference = factor(tree_train$yr1status)
)
```

```
## 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
##
```

```
##           Reference
## Prediction     0     1
##          0 16504   847
##          1  1595  6300



# original method I tried
# confusionMatrix(
#   data = relevel(pred_trees, ref = "1"),
#   reference = relevel(tree_train$yr1status, ref = "1")
```

The overall fraction of correct predictions is (16504 + 6300)/(16504 + 6300 + 847 + 1595), which is equal to the 0.903. The “Accuracy” of the model is calculated by the number of true positive and true negative predictions over all of the predictions made.

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

The confusion matrix is telling you the number of true negatives (16504), false negatives (847), false positives (1595) and true positives (6300).

You can use this information to show the accuracy (described in Q11), the specificity (the number of correct negative predictions divided by the total number of actual negatives), and the sensitivity (the number of correct positive predictions over the total number of positives).

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

The accuracy of the model is .903 and is calculated by the total number of correct predictions divided by the total number of predictions (correct and incorrect predictions).

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.

```
#making the yr1ststus a factor on the test data as well
tree_test <- tree_test %>%
  mutate(yr1status = as.factor(yr1status))

#using the model to make predictions on the test data
set.seed(123)
pred_test_trees <- predict(cv_model4, newdata = tree_test)

#confusion matrix for the test data predictions
confusionMatrix(
  data = factor(pred_test_trees),
  reference = factor(tree_test$yr1status)
)
```

```
## 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
##
```

```
##           Reference
## Prediction    0    1
##          0 7013  362
##          1  721 2724
```

> 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?

Accuracy : 0.8999

The accuracy of this final model is very similar to that of the cross validation accuracy on the test data. I don't find this particularly surprising, as the variables included in the model were fairly correlated with tree death. Additionally, there is a class imbalance in the response variable, with 71.6% of the trees living. If the model just used that statistic for prediction it would be right roughly 71% of the time. Using other variables to improve the model to 90% seems reasonable.

```
#code for above paragraph
trees_encode %>%
  filter(yr1status == 0) %>%
  count()
```

```
## # A tibble: 1 × 1
##       n
##   <int>
## 1 25833
```

```
25833/360666
```

```
## [1] 0.07162583
```