Lab4

Sofia Ingersoll

2024-02-07

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)).

```r
url <- "https://raw.githubusercontent.com/MaRo406/eds-232-machine-learning/main/data/trees-dat.csv"

trees_dat<- read_csv(file = url) %>%
  select(-...1)
```

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

Data Splitting

```r
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----                      Recipe Prep & Bake Data                       ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# specify and prep recipe to convert entire df
# turn all categorical predictors to a binary variable using 0 or 1
trees_rec <- recipe(yr1status~., data = trees_dat) %>%
  step_integer(c(YrFireName, Species, Genus_species), zero_based = T) %>%
  prep(trees_dat)

# to view the training status of this object, call it in the console.
# the binary variable will be converted when we bake our recipe

# bake recipe with zero_based trees df
trees_baked <- bake(trees_rec, new_data =  trees_dat)
```

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

```r
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----                        Split Baked Data                                ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Create training (70%) and test (30%) sets for the
#set.seed(123)        # for reproducibility
trees_baked_split <- initial_split(trees_baked, prop = .7)
trees_baked_train <- training(trees_baked_split)
trees_baked_test  <- testing(trees_baked_split)
```

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

> `trees_baked_train` contains 25246 observations.

Simple Logistic Regression

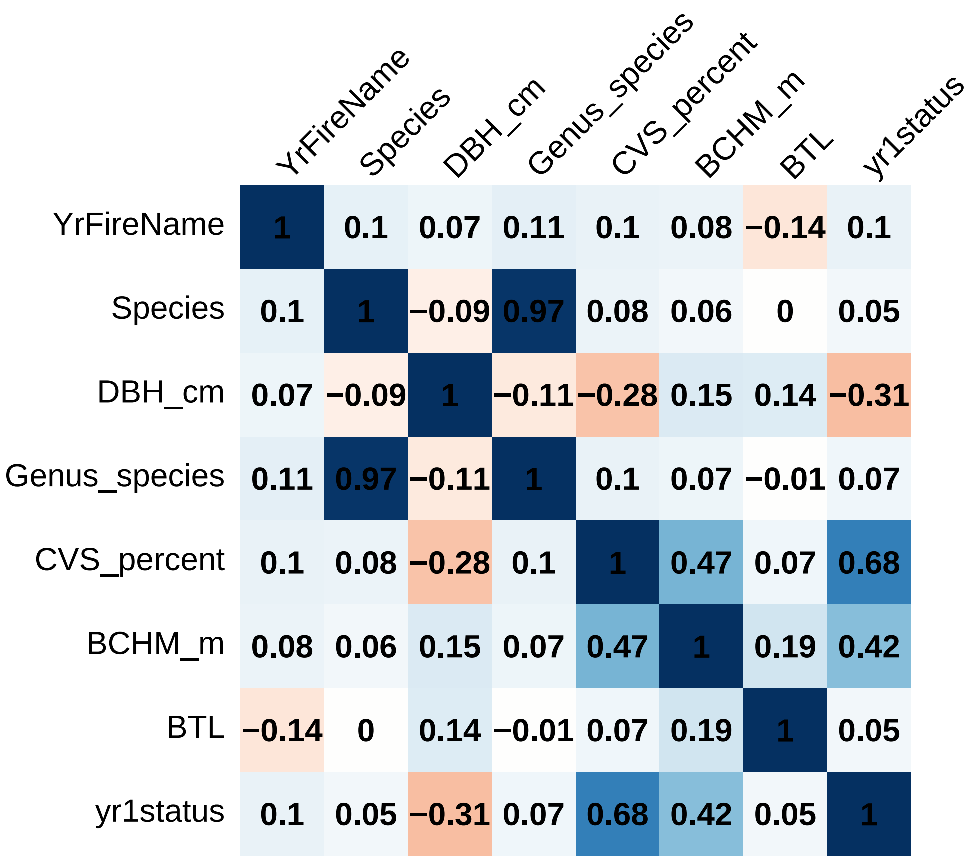
Let's start our modeling effort with some simple models: one predictor and one outcome each.

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

Highest Correlated Predictors in descending order: - CVS_percent (0.68) - BCHM_m (0.42) - DBH_cm (-0.3)

```r
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----                     Correlation Matrix                                 ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Obtain correlation matrix
# Drop columns that are not really informative
# Convert chr to integers
corr_mat <- trees_baked_train %>%
  mutate(Species = as.integer(Species, zero_based = TRUE),
         Genus_species = as.integer(Genus_species, zero_based = TRUE)) %>%
  cor()

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



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

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
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----                     Split Unbaked Data                               ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Create training (70%) and test (30%) sets for the
set.seed(123)  # for reproducibility
trees_split <- initial_split(trees_dat, prop = .7)
trees_train <- training(trees_split)
trees_test  <- testing(trees_split)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----                  Making 1 v 1 unbaked models                         ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# ---- Model Tree status & DBH_cm ----
model_DBH_cm <- glm(data = trees_train,
                    yr1status ~ DBH_cm,
```

```
                    family = "binomial")
broom::tidy(model_DBH_cm)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   0.546    0.0318       17.2 4.55e-66
## 2 DBH_cm       -0.0598   0.00128     -46.7 0
```

```
# --- Model Tree status & CVS_percent ----
model_CVS_percent <- glm(data = trees_train,
                         yr1status ~ CVS_percent,
                         family = "binomial")
broom::tidy(model_CVS_percent)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)  -6.61     0.111       -59.4       0
## 2 CVS_percent   0.0762   0.00122      62.7       0
```

```
# ---- Model Tree status & BCHM_m ----
model_BCHM_m <- glm(data = trees_train,
                    yr1status ~ BCHM_m,
                    family = "binomial")
broom::tidy(model_BCHM_m)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   -1.86    0.0227      -81.9       0
## 2 BCHM_m         0.211   0.00379      55.8       0
```

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

Understanding these variables we need to exponentiate the model coefficients. This will give us the log odds of experiencing the outcome variable for a single unit increase in our beta 1 variable. This gives us the multiplicative of the outcome.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----           Interpreting 1 v 1 unbaked models                          ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# exponentiate the coefficients from model objects for interpretation.
# Gives us changes in odds of tree status

# ---- Model Tree status & DBH_cm ----
# the odds the tree status is living decreases by
# a multiplicative 0.06 for a unit increase in DBH_cm
exp(coef(model_DBH_cm)) %>%
  broom::tidy()
```

```
## # A tibble: 2 x 2
##   names           x
##   <chr>       <dbl>
## 1 (Intercept) 1.73
## 2 DBH_cm      0.942
```

```
# --- Model Tree status & CVS_percent ----
# the odds the tree status is living increases by
# a multiplicative 0.08 for a unit increase in CVS percent
exp(coef(model_CVS_percent)) %>%
  broom::tidy()
```

```
## # A tibble: 2 x 2
##   names             x
##   <chr>         <dbl>
## 1 (Intercept) 0.00134
## 2 CVS_percent 1.08
```

```
# ---- Model Tree status & BCHM_m ----
# the odds the tree status is living increases by
# a multiplicative 0.24 for a unit increase in BCHM m
exp(coef(model_BCHM_m)) %>%
  broom::tidy()
```

```
## # A tibble: 2 x 2
##   names           x
##   <chr>       <dbl>
## 1 (Intercept) 0.155
## 2 BCHM_m      1.24
```

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

Multiple Logistic Regression

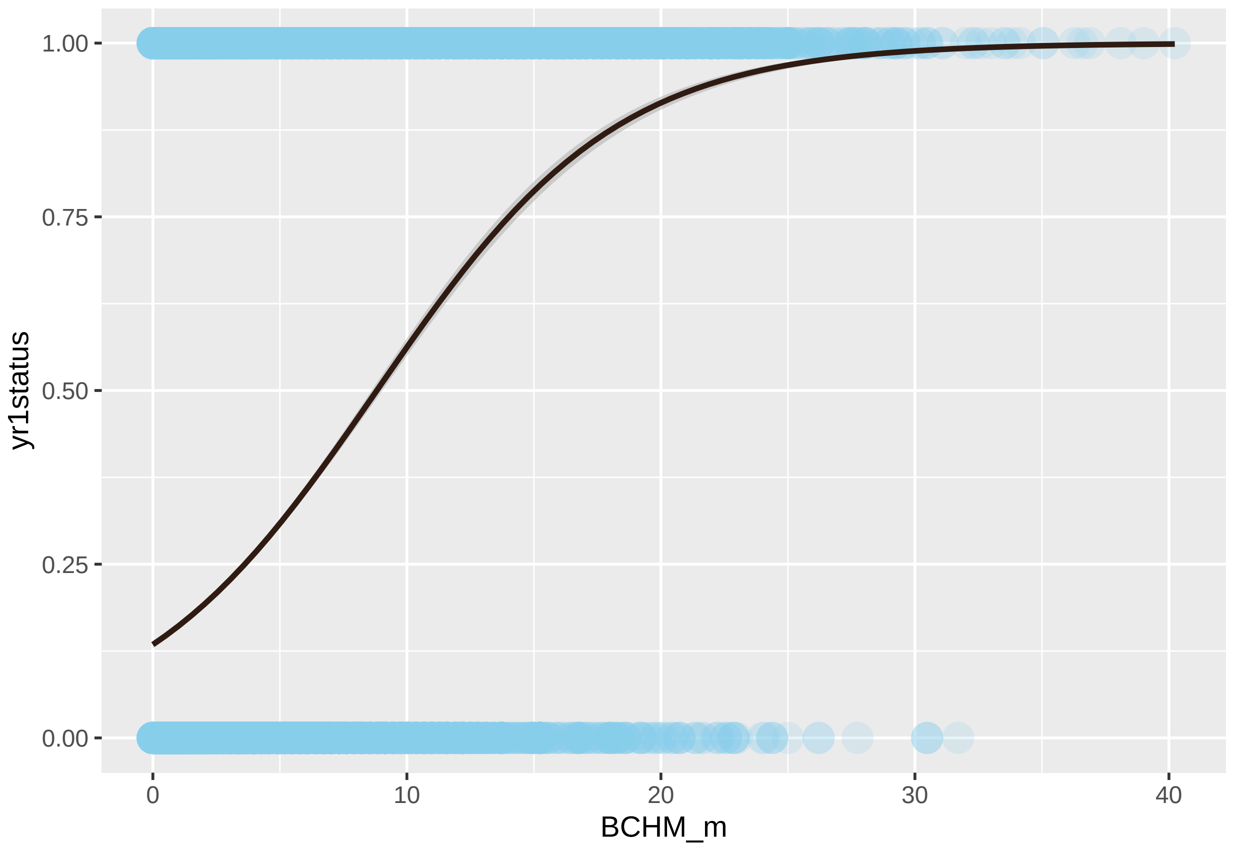
Let's not limit ourselves to a single-predictor model. More predictors might lead to better model performance.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----                  Visualize Models                         ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

#---- BCHM_m vs yr1status ----
BCHM_m_reg_plot <- ggplot(trees_train,
                          aes(x = BCHM_m,
                              y = yr1status,
                          )
) +
  geom_point(alpha = 0.2,
             col = 'skyblue',
             size = 5
  ) +
  stat_smooth(method = "glm",
```

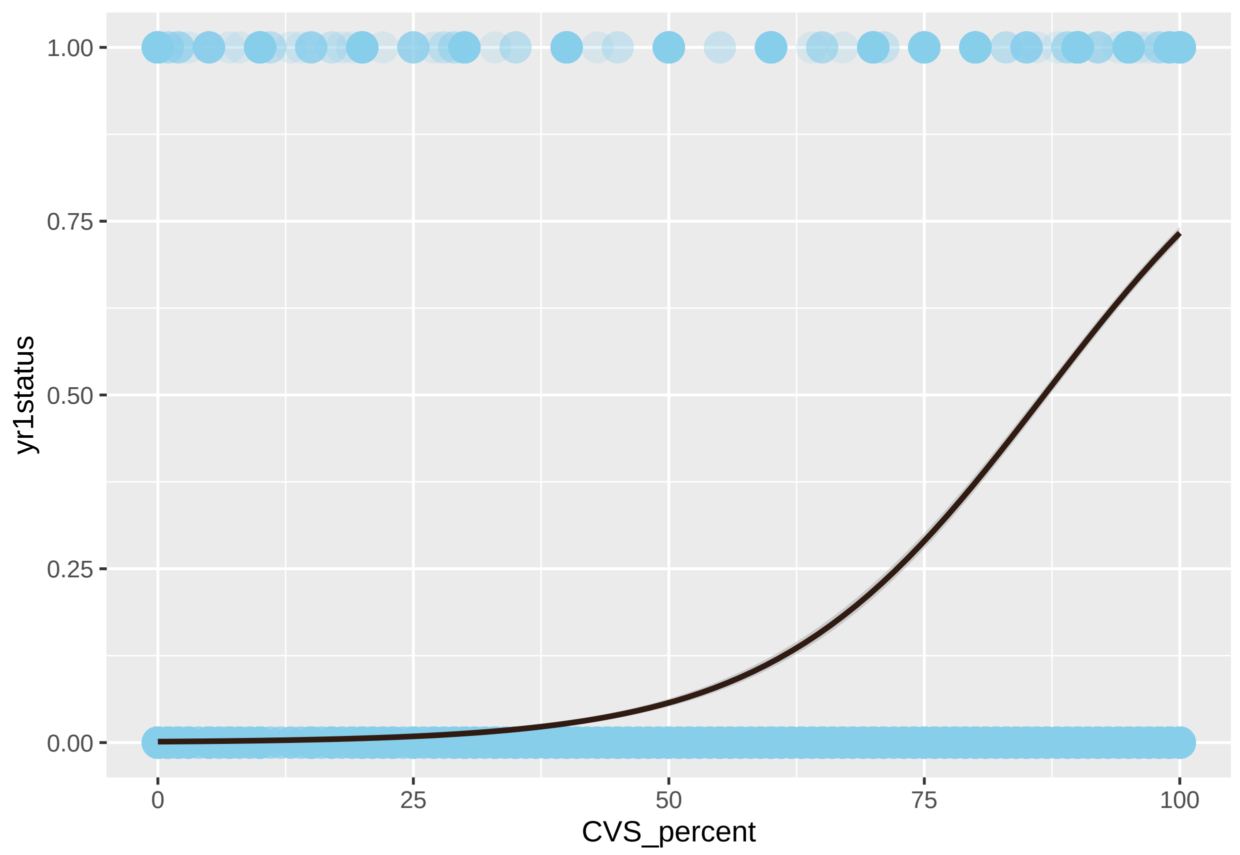
```
              se = TRUE,
              method.args = list(family = 'binomial'),
              col = '#2F1B12')

BCHM_m_reg_plot
```



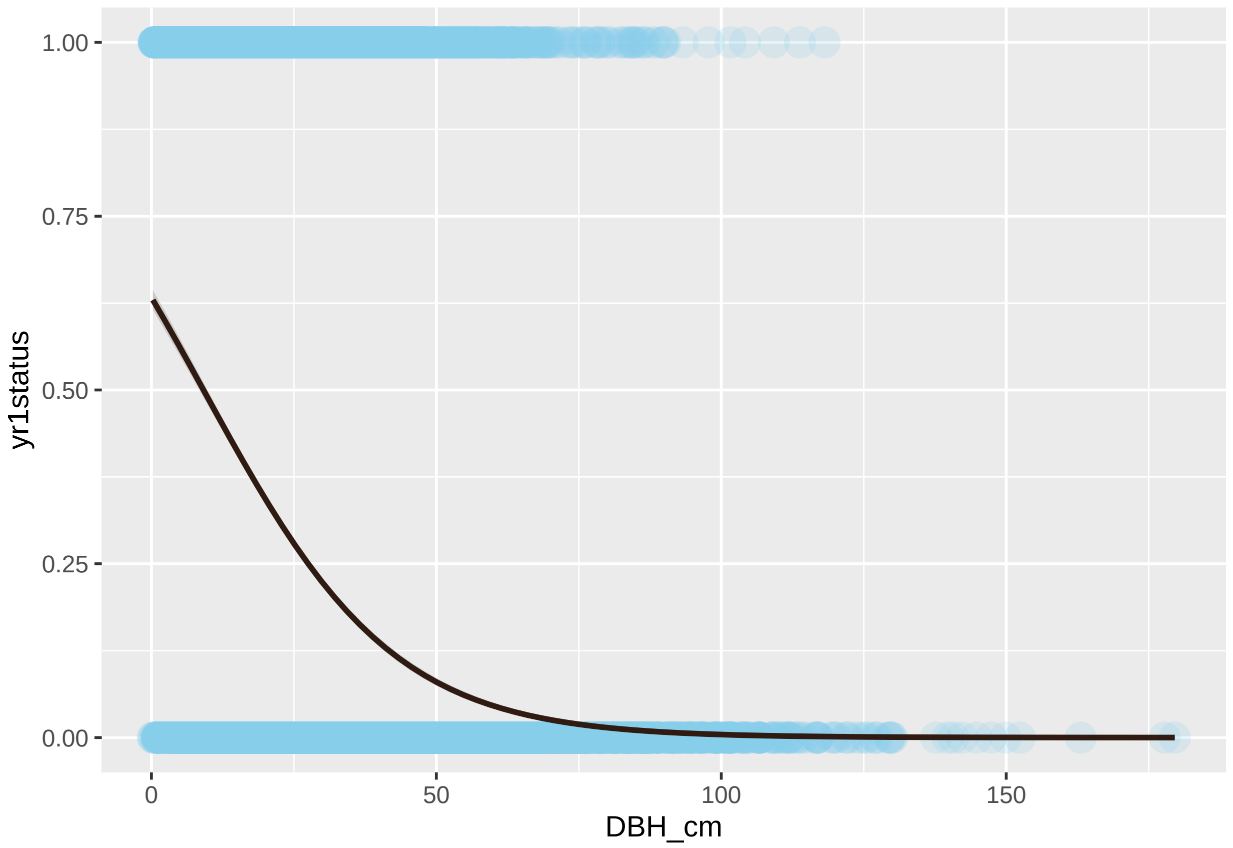
```
#---- CVS percent vs yr1status ----
CVS_perc_reg_plot <- ggplot(trees_train,
                            aes(x = CVS_percent,
                                y = yr1status,
                            )
) +
  geom_point(alpha = 0.2,
             col = 'skyblue',
             size = 5) +
  stat_smooth(method = 'glm',
              se = TRUE,
              method.args = list(family = 'binomial'),
              col = '#2F1B12')

CVS_perc_reg_plot
```



```r
#---- DBH_cm vs yr1status ----
DBH_cm_reg_plot <- ggplot(trees_train,
                          aes(x = DBH_cm,
                              y = yr1status,
                          )
) +
  geom_point(alpha = 0.2,
             col = 'skyblue',
             size = 5) +
  stat_smooth(method = 'glm',
              se = TRUE,
              method.args = list(family = 'binomial'),
              col = '#2F1B12')

DBH_cm_reg_plot
```



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?

All three of the predictors are considered significant in the resulting logistic regression. The p-values associated with CVS_percent, BCHM_m, and DBH_cm are less than the critical value 0.01 [respectively: 0.00, 7.54e-189, 6.84e-214].

Estimate Model Accuracy

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

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----                  Multivariate model                         ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ---- Full logistic regression ----
logistic_full <- glm(yr1status ~ CVS_percent + BCHM_m + DBH_cm,
                     family = 'binomial',
                     data = trees_train)

# now clean our model and give me the stats
broom::tidy(logistic_full)
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic   p.value
```

```
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)  -4.76     0.115       -41.3 0
## 2 CVS_percent   0.0615   0.00118      52.2 0
## 3 BCHM_m        0.159    0.00544      29.3 7.54e-189
## 4 DBH_cm       -0.0607   0.00195     -31.2 6.84e-214
```

```
# ---- combined model coefficients interpretation ----
# the odds the tree status is living increases by
# a multiplicative 0.24 for a unit increase in BCHM m
exp(coef(logistic_full)) %>%
  broom::tidy()
```

```
## # A tibble: 4 x 2
##   names             x
##   <chr>         <dbl>
## 1 (Intercept) 0.00854
## 2 CVS_percent 1.06
## 3 BCHM_m      1.17
## 4 DBH_cm      0.941
```

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.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----                  Cross Validation                         ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

trees_train_fact <- trees_train %>%
  mutate(yr1status = as.factor(yr1status))


# ---- k-fold cv method settings ----
cv <- trainControl(method = "cv",
                   number = 10
)

# ---- cv model for DBH_cm ----
cv_DBH_cm_model <- caret::train(
  yr1status ~ DBH_cm,
  data = trees_train_fact,
  method = 'glm',
  family = 'binomial',
  trControl =  cv)

# ---- cv model for CVS percent ----
cv_CVS_perc_model <- caret::train(
  yr1status ~ CVS_percent,
  data = trees_train_fact,
  method = 'glm',
  family = 'binomial',
```

```r
  trControl = cv)

# ---- cv model for BCHM_m ----
cv_BCHM_m_model <- caret::train(
  yr1status ~ BCHM_m,
  data = trees_train_fact,
  method = 'glm',
  family = 'binomial',
  trControl = cv)

# ---- cv model for multivarite predictors ----
cv_combined_model <- caret::train(
  yr1status ~ DBH_cm + CVS_percent + BCHM_m,
  data = trees_train_fact,
  method = 'glm',
  family = 'binomial',
  trControl = cv)
```

> 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?
>
> The combined multivariate model has the highest mean accuracy of ~91%.

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

```r
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----                    Resample + Accuracy                          ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# extract out of sample performance measures
summary(
  resamples(
    list(
      cv_DBH_cm_model,
      cv_CVS_perc_model,
      cv_BCHM_m_model,
      cv_combined_model
    )
  )
)$statistics$Accuracy
```

```
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.7437624 0.7490095 0.7502477 0.7513667 0.7521782 0.7606973    0
## Model2 0.8902970 0.8924431 0.8980198 0.8975283 0.9014560 0.9064976    0
## Model3 0.7580198 0.7716832 0.7744109 0.7722419 0.7764683 0.7785261    0
## Model4 0.8918812 0.9009901 0.9057052 0.9041038 0.9086139 0.9104950    0
```

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

```r
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----                 Confusion Matrix                         ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# predict class
pred_class <- predict(cv_combined_model, trees_train_fact)

# summarize results
confusionMatrix(data = relevel(pred_class, ref = '1'),
                reference = relevel(trees_train_fact$yr1status,
                                    ref = '1'))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     0
##          1  6295  1572
##          0   852 16527
##
##                Accuracy : 0.904
##                  95% CI : (0.9003, 0.9076)
##     No Information Rate : 0.7169
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7705
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8808
##             Specificity : 0.9131
##          Pos Pred Value : 0.8002
##          Neg Pred Value : 0.9510
##              Prevalence : 0.2831
##          Detection Rate : 0.2493
##    Detection Prevalence : 0.3116
##       Balanced Accuracy : 0.8970
##
##        'Positive' Class : 1
##
```

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

The confusion matrix is articulating the count of True (+) [‘Reference 1’] and False (+) [‘Prediction 1’] for the model predictions. Additionally, it includes the count of True (-) [‘Reference 0’] and False (-) [‘Prediction 0’].

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

The overall accuracy of the combined multivariate model is 90.4%.

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.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----           Predictions on best model                          ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

trees_test_fact <- trees_test %>%
  mutate(yr1status = as.factor(yr1status))

# get prediction probabilities for test
test_predict <- predict(cv_combined_model, trees_test_fact) #%>%
 # bind_cols(trees_test)          # bind to testing df
```

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?

The accuracy of the test model is 90.1%, this is very close to the training model accuracy and is what we would expect for this model.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ----          Model Accuracy                          ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# get accuracy of testing prediction
# summarize results
confusion_matrix_test <- confusionMatrix(data = relevel(test_predict, ref = '1'),
                                         reference = relevel(trees_test_fact$yr1status,
                                                             ref = '1'))

confusion_matrix_test$overall[1]
```

```
##  Accuracy
## 0.9012939
```