Lab 4

Elke Windschitll

2023-02-01

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
  select(-...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`
```

```
#trees_dat <- trees_dat %>% mutate_if(is.ordered, factor, ordered = FALSE)
```

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

```
# Encode 0 and 1 for predictors
trees_encoded <- recipe(yr1status ~ ., data = trees_dat) %>%
  step_integer(all_predictors(), zero_based = TRUE) %>%
  prep(trees_dat) %>%
  bake(trees_dat)
```

Data Splitting

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

```
# Create training (70%) and test (30%) sets
set.seed(123)  # for reproducibility (random sample)
trees_split <- initial_split(trees_encoded, 0.7)
trees_train <- training(trees_split)
trees_test  <- testing(trees_split)
```

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

```
print(paste("We are using", nrow(trees_train), "observations in the training set after the split"))
```

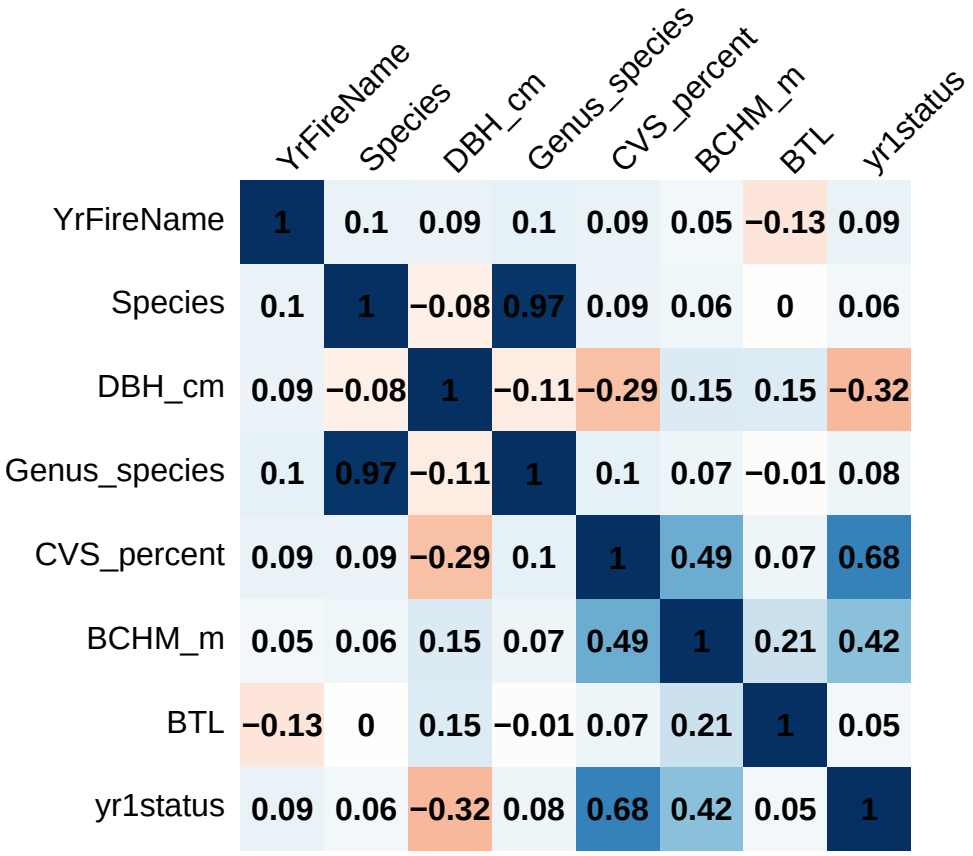
```
## [1] "We are using 25246 observations in the training set after the split"
```

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.

```
corr_mat <- cor(trees_train)
# Make a correlation plot between the variables
corrplot(corr_mat, method = "shade", shade.col = NA, tl.col = "black", tl.srt = 45, addCoef.col = "black
```




```
print(paste("Based on this correlation matrix we see that CVS_percent (crown volume scorched), BCHM_m (
```

```
## [1] "Based on this correlation matrix we see that CVS_percent (crown volume scorched), BCHM_m (burn
```

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

```
cvs_model <- glm(data = trees_train, yr1status ~ CVS_percent, family = "binomial")


bchm_model <- glm(data = trees_train, yr1status ~ BCHM_m, family = "binomial")
```

```
dbh_model <- glm(data = trees_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.

```
print(paste0("For every one percent increase in crown volume scorched, the odds of a tree being dead in
```

```
## [1] "For every one percent increase in crown volume scorched, the odds of a tree being dead increase
```

```
print(paste0("For every one meter increase in burn char height, the odds of a tree being dead increase
```

```
## [1] "For every one meter increase in burn char height, the odds of a tree being dead increase multipl
```

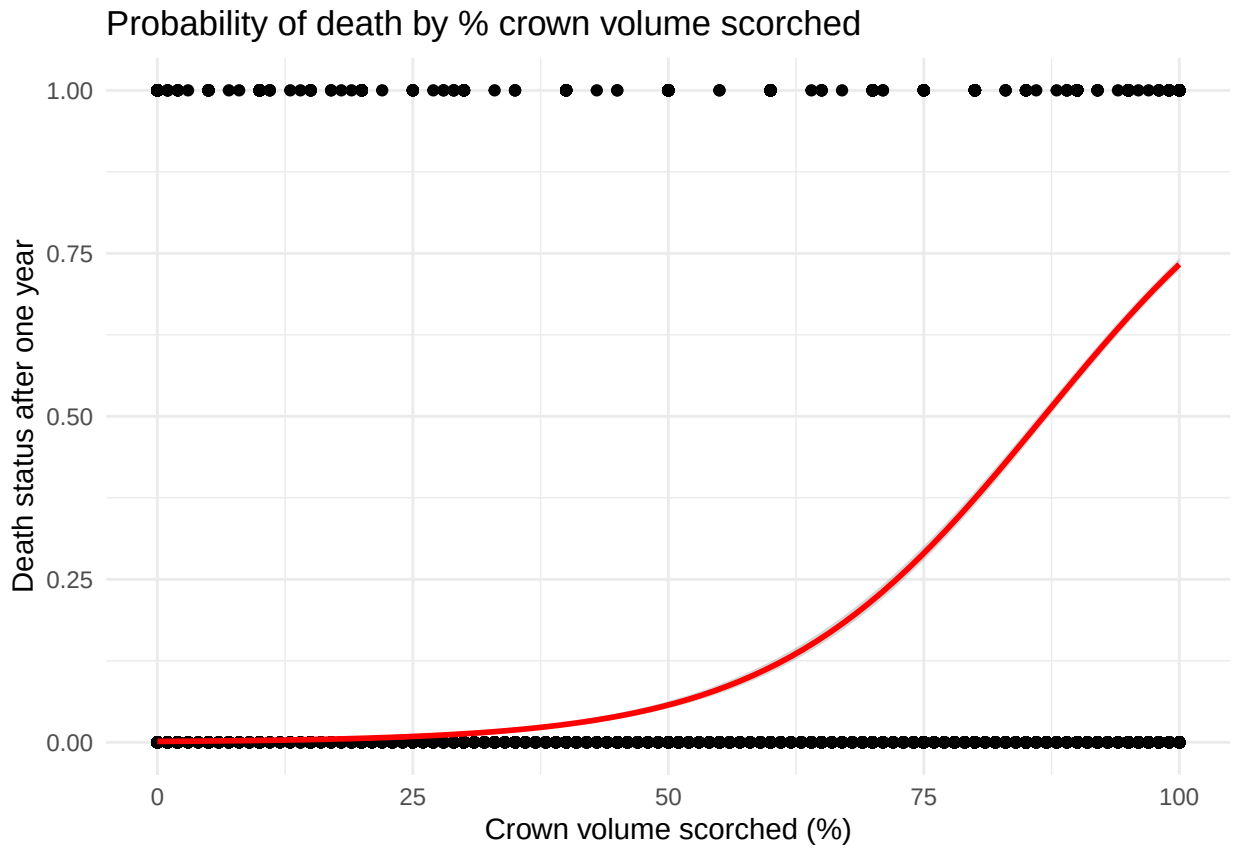
```
print(paste0("For every one cm increase in diameter breast height, the odds of a tree being dead decrea
```

```
## [1] "For every one cm increase in diameter breast height, the odds of a tree being dead decrease mul
```

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

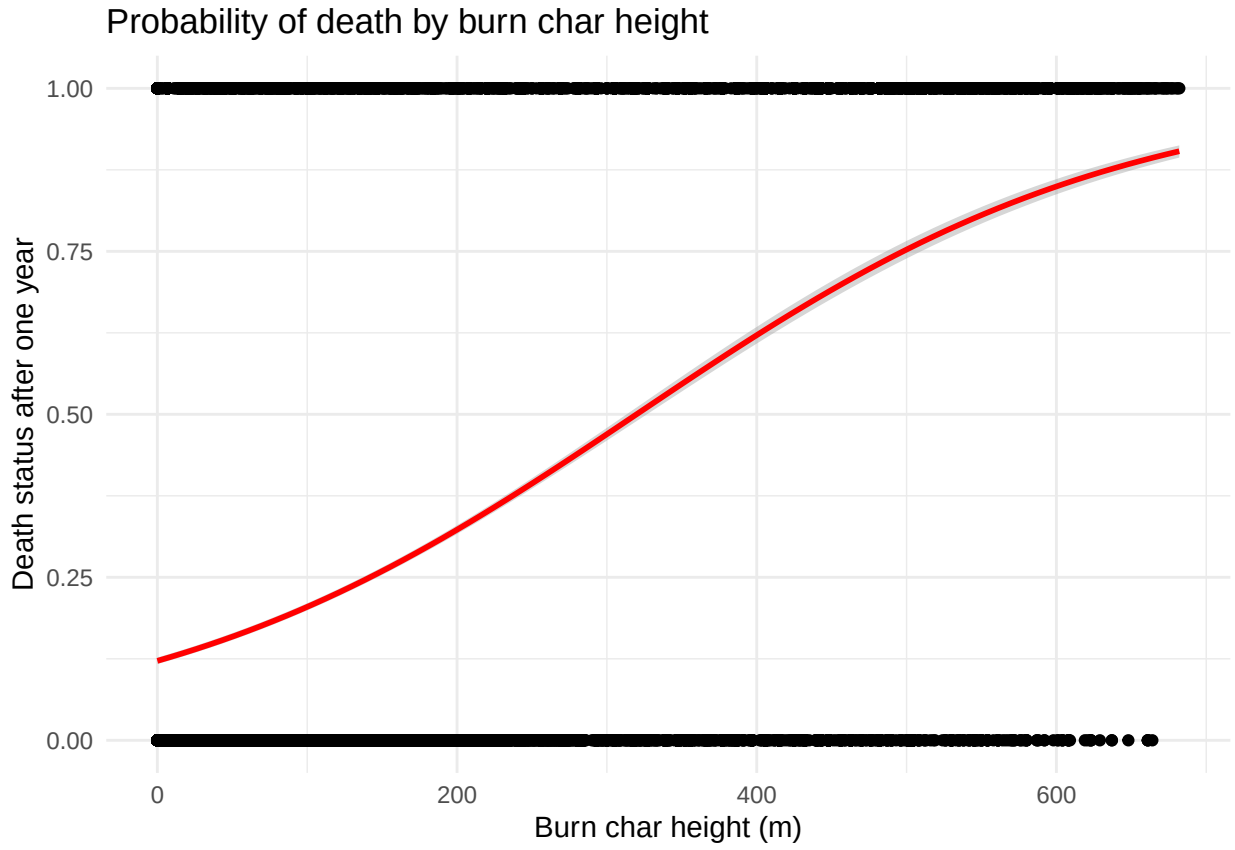
```
# CVS plot
ggplot(trees_train, aes(x = CVS_percent, y = yr1status)) +
  geom_point() +
  stat_smooth(method="glm",  se=TRUE,
                method.args = list(family=binomial),
              color = "red") +
  xlab("Crown volume scorched (%)") +
  ylab("Death status after one year") +
  ggtitle("Probability of death by % crown volume scorched") +
  theme_minimal()
```

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




```
# BCHM plot
ggplot(trees_train, aes(x = BCHM_m, y = yr1status)) +
  geom_point() +
  stat_smooth(method="glm",  se=TRUE,
                method.args = list(family=binomial),
              color = "red") +
  xlab("Burn char height (m)") +
  ylab("Death status after one year") +
  ggtitle("Probability of death by burn char height") +
  theme_minimal()
```

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




```
# DBH
ggplot(trees_train, aes(x = DBH_cm, y = yr1status)) +
  geom_point() +
  stat_smooth(method="glm",  se=TRUE,
                method.args = list(family=binomial),
              color = "red") +
  xlab("Diameter at breast height (cm)") +
  ylab("Death status after one year") +
  ggtitle("Probability of death by diameter") +
  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(data = trees_train, yr1status ~ CVS_percent + BCHM_m + DBH_cm, family = "binomial"

log_full_table <- broom::tidy(logistic_full)
summary(logistic_full)
```

```
##
## Call:
## glm(formula = yr1status ~ CVS_percent + BCHM_m + DBH_cm, family = "binomial",
##     data = trees_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
```

```
print(paste0("It appears all three predictors are significant as we can see from their very small p-valu
```

```
## [1] "It appears all three predictors are significant as we can see from their very small p-values res
```

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 to factor
trees_train$yr1status <- as_factor(trees_train$yr1status)
trees_test$yr1status <- as_factor(trees_test$yr1status)

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

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

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


set.seed(123)
# full cv model
cv_model4 <- train(yr1status ~ CVS_percent + BCHM_m + DBH_cm,
```

```
                          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?

```
# create a summary table of the accuracy of each model
acc_table <- summary(
  resamples(
    list(
      model1 = cv_model1,
      model2 = cv_model2,
      model3 = cv_model3,
      model4 = cv_model4
    )
  )
)$statistics$Accuracy

acc_table
```

```
##             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
```

```
print(paste("The model with the hightest accuracy is model4 (the model with all three factors) with a m
```

```
## [1] "The model with the hightest accuracy is model4 (the model with all three factors) with a mean a
```

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.

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

# create confusion matrix
matrix <- confusionMatrix(
  data = relevel(pred_class, ref = 1),
  reference = relevel(trees_train$yr1status, ref = 1)
)

# print table
matrix$table
```

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

```
# find fraction correct
frac_correct <- (matrix$table[1,1] + matrix$table[2,2])/nrow(trees_train)

print(paste("The overall proportion of correct predictions is", frac_correct))
```

```
## [1] "The overall proportion of correct predictions is 0.903271805434524"
```

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

The confusion matrix shows us that there are 16,504 trees that are predicted to survive and do actually survive. There are 6300 trees predicted to die and do actually die. There are 847 trees that are predicted to survive under this model that actually die (this is the count of false negatives). There are 1595 trees that are predicted to die under this model that actually survive (this is the count of false positives).

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

```
print(paste0("The accuracy of the model is ", round(matrix$overall[1],3), ". This is calculated by addi
```

```
## [1] "The accuracy of the model is 0.903. This is calculated by adding the true positive and true nega
```

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.

```
# run model on train data
pred_class_test <- predict(cv_model4, trees_test)

# create confusion matrix
matrix2 <- confusionMatrix(
  data = relevel(pred_class_test, ref = 1),
  reference = relevel(trees_test$yr1status, ref = 1)
)

# print table
matrix2$table
```

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

```
# find fraction correct
frac_correct2 <- (matrix2$table[1,1] + matrix2$table[2,2])/nrow(trees_test)

print(paste("The overall proportion of correct predictions is", frac_correct2))
```

```
## [1] "The overall proportion of correct predictions is 0.899907578558226"
```

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 final model is ever so slightly lower than the cross validation accuracy, but it is extremely close. I don't necessarily find this too surprising, as this was the goal of performing the cross validation – to get as close to the true model accuracy as we could without touching the test data. In a way, though, it is surprising to me personally, as this is all new to me and super cool :) ~ 90% accuracy seems pretty good to me given that the dummy classifier would be 71% ((7013 + 721)/(7013 + 721 + 362 + 2724))