Lab4

Hope Hahn

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)).

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

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

```
# create recipe to recode data
trees_recipe <- recipe(yr1status ~ ., data = trees_dat) %>%
    step_integer(all_predictors(), zero_based = TRUE) %>%
    prep(trees_dat)

# apply recipe to data
trees_baked <- bake(trees_recipe, new_data = trees_dat)
```

Data Splitting

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

```
# all split data are already baked
set.seed(123)  # for reproducibility

# split tree data with 70/30 proportion
trees_split <- initial_split(trees_baked, prop = 0.7)

# save training data
trees_train <- training(trees_split)

# save test data
trees_test <- testing(trees_split)
```

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

```
train_obs <- nrow(trees_train)
```

- **We are using 25246 observations.**

Simple Logistic Regression

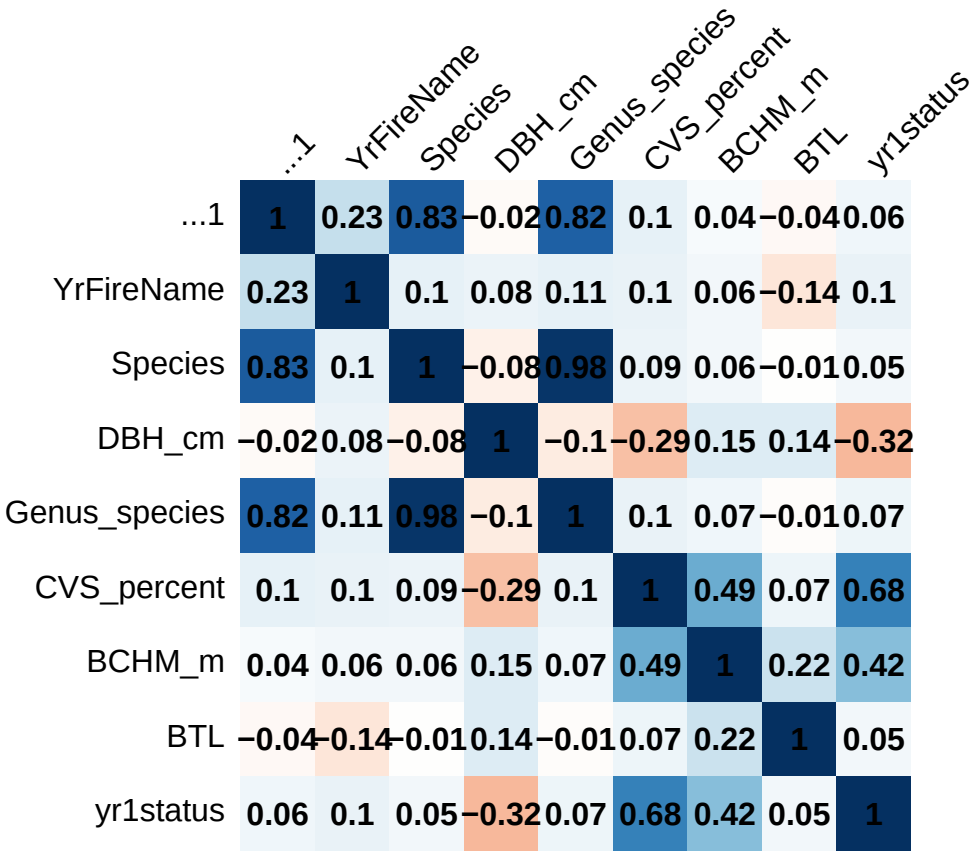
Let's start our modeling effort with some simple models: one predictor and one outcome each.

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

- **The three predictors that most highly correlate with the outcome variable are CVS_percent, BCHM_m, and DBH_cm.**

```
# Obtain correlation matrix of tree data
corr_mat <- cor(trees_baked)

# Make a correlation plot to see which are most highly
# correlated
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.

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

# BCHM_m glm
bchm_glm <- glm(data = trees_train, yr1status ~ BCHM_m, family = "binomial")

# DBH_cm
dbh_glm <- 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.

- **The odds of a tree dying changes multipicatively by 1.079 with a one unit increase in CVS_percent, 1.006 with a one unit increase in BCHM_m, and 0.996 with a one unit increase in DBH_cm. This means that with an increase in CVS_percent and BCHM_m the odds of tree death increases, while an increase in DBH_cm leads to a decrease of the odds of a tree dying.**

```
# exponentiate coefficients for interpretation for each
# variable
exp(coef(cvs_glm))
```

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

```
exp(coef(bchm_glm))
```

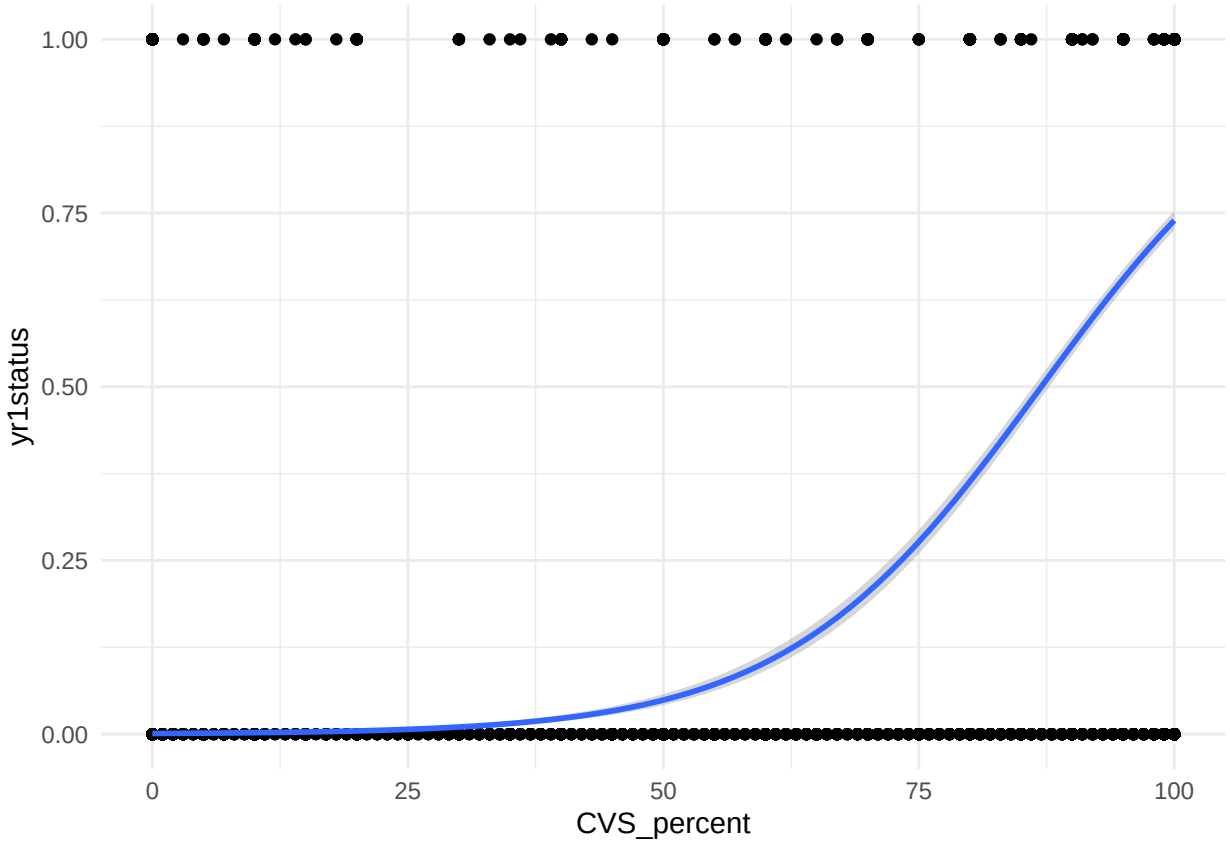
```
## (Intercept)      BCHM_m
##   0.1387476   1.0061954
```

```
exp(coef(dbh_glm))
```

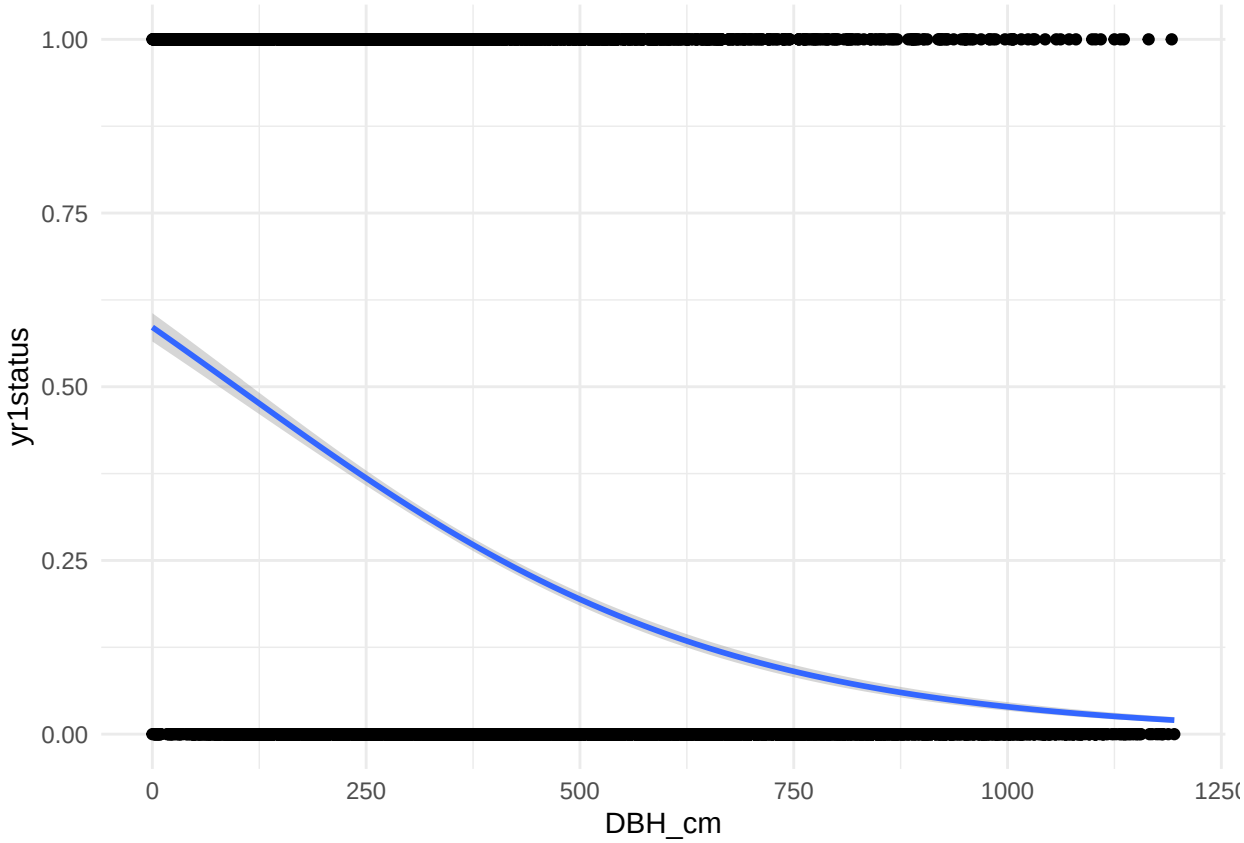
```
## (Intercept)      DBH_cm
##   1.4876101   0.9962419
```

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

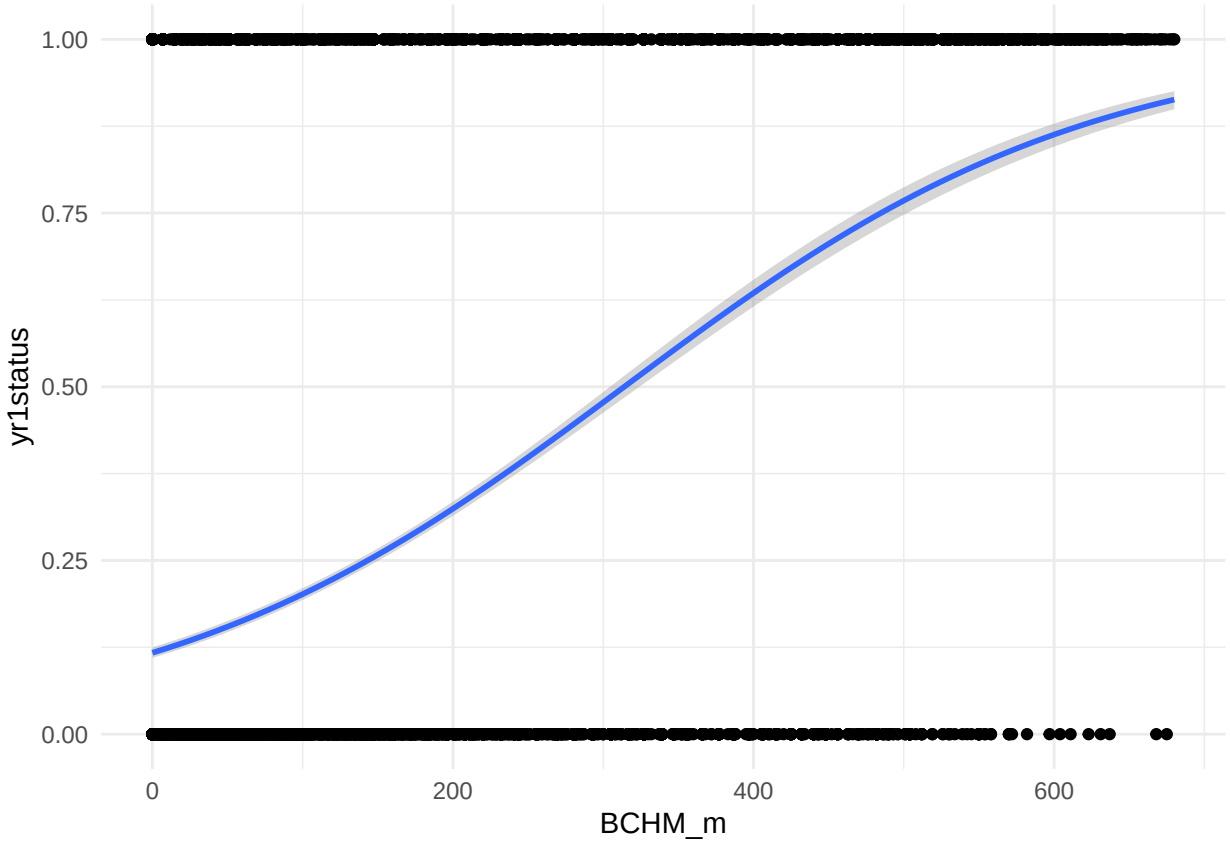
```
# cvs percent plot
ggplot(trees_test, aes(x = CVS_percent, y = yr1status)) + geom_point() +
    stat_smooth(method = "glm", se = TRUE, method.args = list(family = "binomial")) +
    theme_minimal()
```




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




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



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?

- ***Using a significance level of alpha = 0.05, all three of these variables are significant in the resulting model.***

```
# multiple logistic regression
logistic_full <- glm(yr1status ~ CVS_percent + DBH_cm + BCHM_m,
    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 CVS_percent  0.0622   0.00119       52.4 0
## 3 DBH_cm      -0.00371  0.000118     -31.4 2.39e-216
## 4 BCHM_m       0.00466  0.000161      28.9 6.05e-184
```

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.

```
# change outcome to factor
trees_train <- trees_train %>%
    mutate(yr1status = as.factor(yr1status))

# cross validation CVS_percent
cv_model1 <- train(yr1status ~ CVS_percent, data = trees_train,
    method = "glm", family = "binomial", trControl = trainControl(method = "cv",
        number = 10))

# cross validation DBH_cm
cv_model2 <- train(yr1status ~ DBH_cm, data = trees_train, method = "glm",
    family = "binomial", trControl = trainControl(method = "cv",
        number = 10))

# cross validation BCHM_m
cv_model3 <- train(yr1status ~ BCHM_m, data = trees_train, method = "glm",
    family = "binomial", trControl = trainControl(method = "cv",
        number = 10))

# cross validation multiple regression
cv_model4 <- train(yr1status ~ CVS_percent + DBH_cm + BCHM_m,
    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?

- **Model 4 has the highest accuracy with a mean accuracy rate of 90.3%.**

```
# Test Accuracy
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 0.8887129 0.8918708 0.8976238 0.8975291 0.9004556 0.9152139    0
## model2 0.7469307 0.7487129 0.7509905 0.7520005 0.7558681 0.7580198    0
## model3 0.7588119 0.7690099 0.7728713 0.7715290 0.7763471 0.7777338    0
## model4 0.8887129 0.8998711 0.9047147 0.9029551 0.9069215 0.9093069    0
```

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.

- **The overall fraction of correct predictions is 0.9033.**

```
# predict train data
pred_train <- predict(cv_model4, trees_train)

# create confusion matrix on train data
confusionMatrix(data = relevel(pred_train, ref = "1"), reference = relevel(trees_train$yr1status,
    ref = "1"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     0
##          1  6300  1595
##          0   847 16504
##
##                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 : 0.8815
##             Specificity : 0.9119
##          Pos Pred Value : 0.7980
##          Neg Pred Value : 0.9512
##              Prevalence : 0.2831
##          Detection Rate : 0.2495
##    Detection Prevalence : 0.3127
##       Balanced Accuracy : 0.8967
##
##        '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 telling us that false positives (predicting a tree will die when it does not) are about twice as likely as false negatives (predicting a tree will stay alive when it dies).**

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

- **The overall accuracy of the model is 90.3%. This is calculated by dividing correct predictions by total 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.

```
# convert output to factor
trees_test <- trees_test %>%
    mutate(yr1status = as.factor(yr1status))

# predict class on test data
pred_test <- predict(cv_model4, trees_test)

# create confusion matrix
confusionMatrix(data = relevel(pred_test, ref = "1"), reference = relevel(trees_test$yr1status,
    ref = "1"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    0
##          1 2724  721
##          0  362 7013
##
##                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 : 0.8827
##             Specificity : 0.9068
##          Pos Pred Value : 0.7907
##          Neg Pred Value : 0.9509
##              Prevalence : 0.2852
##          Detection Rate : 0.2518
##    Detection Prevalence : 0.3184
##       Balanced Accuracy : 0.8947
##
##        'Positive' Class : 1
##
```

> 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 on the test data is pretty close to the cross validation accuracy, as the test data accuracy is 89.99% while the cv accuracy is 90.3%. I do not find this to be surprising because when we do cross validation, we are treating part of the training data like "test" data and testing the model accuracy on random "test" data. Because of this, the accuracy should be relatively similar.**