Lab4

Amanda Herbst

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

```
# set up and prep recipe to turn all predictors to
# zero-based integers
tree_recipe <- recipe(yr1status ~ ., data = trees_dat) %>%
    step_integer(all_predictors(), zero_based = TRUE) %>%
    prep()

# bake recipe so all predictors are zero-based integers
trees_baked <- bake(tree_recipe, new_data = NULL)
```

Data Splitting

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

```
set.seed(123)
trees_split <- initial_split(trees_baked, prop = 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?

```
nrow(trees_train)
```

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

We are using 25,246 observations for training

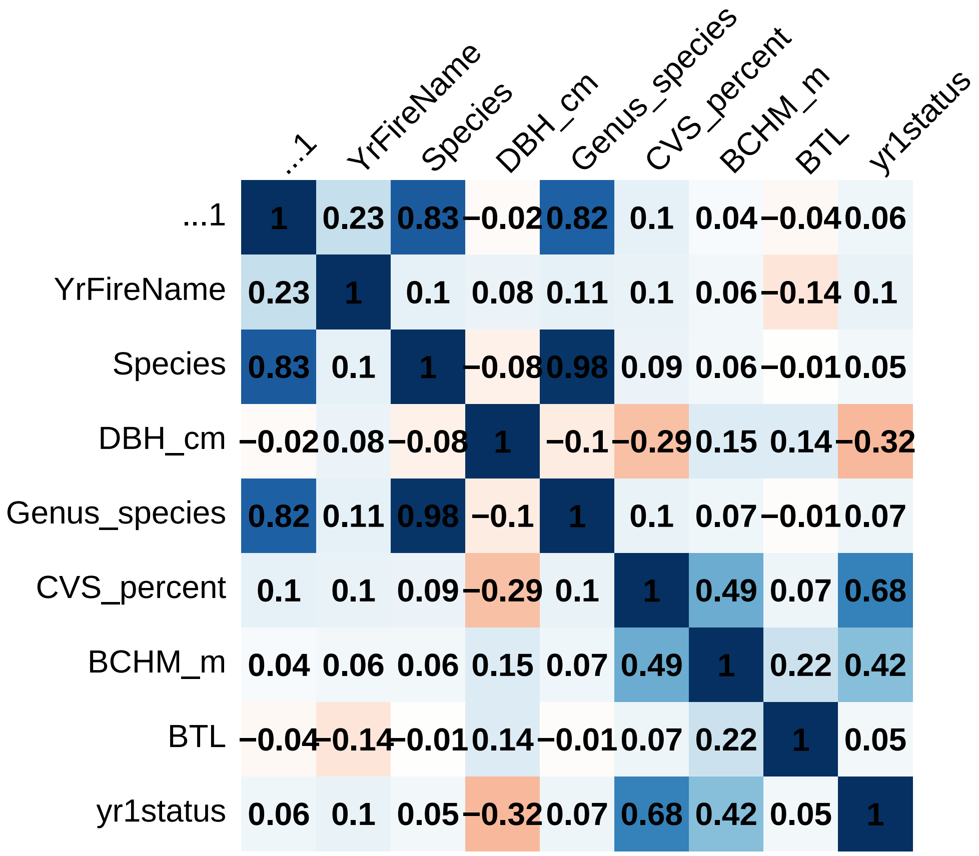
Simple Logistic Regression

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.

```r
# Obtain correlation matrix
corr_mat <- cor(trees_baked)

# 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")
```



Highest correlation with yr1status = DBH_cm, CVS_percent, BCHM_m. These will be the three predictors to further investigate.

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

```r
# DBH_cm model
model_dbh <- glm(data = trees_train, yr1status ~ DBH_cm, family = "binomial")

# CVS_percent model
model_cvs <- glm(data = trees_train, yr1status ~ CVS_percent,
    family = "binomial")

# BCHM_m model
model_bchm <- glm(data = trees_train, yr1status ~ BCHM_m, 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.

```
# DBH_cm coefficients
exp(coef(model_dbh))
```

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

```
# CVS_percent coefficients
exp(coef(model_cvs))
```

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

```
# BCHM_m coefficients
exp(coef(model_bchm))
```

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

The odds of a tree being dead one year after a fire increases multiplicatively by 0.996 by every one additional cm of diameter at breast height.

The odds of a tree being dead one year after a fire increases multiplicatively by 1.08 by every one additional percentage point of tree crown volume being scorched or consumed by fire.

The odds of a tree being dead one year after a fire increases multiplicatively by 1.01 by every one additional meter of maximum bark char.

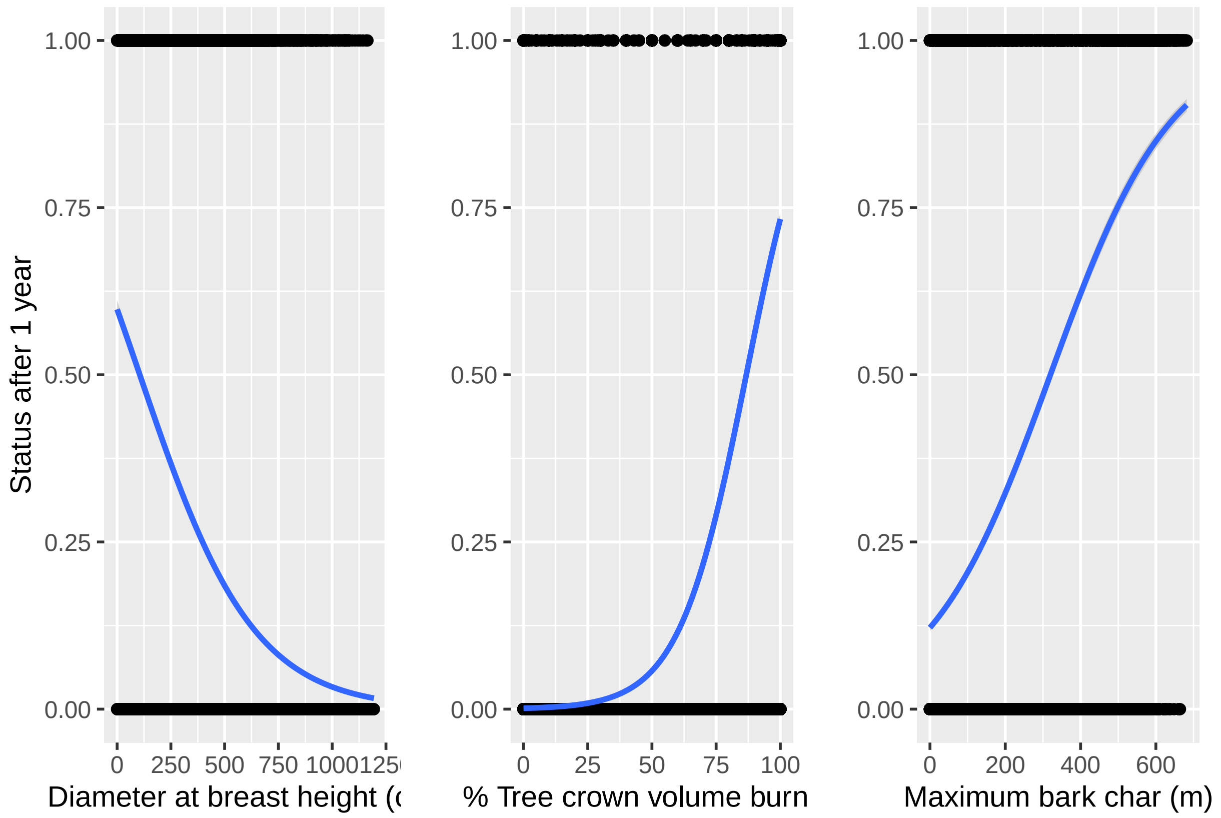
> Question 7: Now let's visualize the results from these models. Plot the fit to the training data of each model.

```
dbh_train_plot <- ggplot(data = trees_train, aes(x = DBH_cm,
    y = yr1status)) + geom_point() + stat_smooth(method = "glm",
    method.args = list(family = binomial)) + labs(y = "Status after 1 year",
    x = "Diameter at breast height (cm)")

cvs_train_plot <- ggplot(data = trees_train, aes(x = CVS_percent,
    y = yr1status)) + geom_point() + stat_smooth(method = "glm",
    method.args = list(family = binomial)) + labs(y = "", x = "% Tree crown volume burned")

bchm_train_plot <- ggplot(data = trees_train, aes(x = BCHM_m,
    y = yr1status)) + geom_point() + stat_smooth(method = "glm",
    method.args = list(family = binomial)) + labs(y = "", x = "Maximum bark char (m)")

dbh_train_plot + cvs_train_plot + bchm_train_plot
```



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?

```r
logistic_full <- glm(yr1status ~ DBH_cm + CVS_percent + BCHM_m,
    data = trees_train, family = "binomial")
broom::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 DBH_cm      -0.00371  0.000118      -31.4 2.39e-216
## 3 CVS_percent  0.0622   0.00119        52.4 0
## 4 BCHM_m       0.00466  0.000161       28.9 6.05e-184
```

All three parameters are significant in this model (p < 0.01).

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.

```r
# DBH_m
cv_model1 <- train(as.factor(yr1status) ~ DBH_cm, data = trees_train,
    method = "glm", family = "binomial", trControl = trainControl(method = "cv",
        number = 10))

# CVS_percent
cv_model2 <- train(as.factor(yr1status) ~ CVS_percent, data = trees_train,
    method = "glm", family = "binomial", trControl = trainControl(method = "cv",
        number = 10))

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

# All three predictors
cv_model4 <- train(as.factor(yr1status) ~ DBH_cm + CVS_percent +
    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?

```r
# extract out of sample performance measures
summary(resamples(list(cv_model1, cv_model2, cv_model3, cv_model4)))$statistics$Accuracy
```

```
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.7476228 0.7488363 0.7506931 0.7525153 0.7535399 0.7619802    0
## Model2 0.8902970 0.8924431 0.8980198 0.8975283 0.9014560 0.9064976    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
```

cv_model4, with all predictor variables, has the highest average accuracy of 90.3%

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.

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

# convert yr1status to factor
trees_train <- trees_train %>%
    mutate(yr1status = as.factor(yr1status))

# create confusion matrix
confusionMatrix(data = pred_class, reference = trees_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
##
```

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

The confusion matrix shows that the model predicts more false positives (predicted alive but died) than false negatives (predicted dead but alive). However, there appears to be more observations of 0s than 1s so the proportions of mistakes will tell us more: 5% of the 0s were predicted wrong, and 20% of the 1s were predicted wrong. Therefore, this model, potentially due to the class imbalance, is predicting false negatives much more than false positives.

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

Overall accuracy is 90% which is calculated by the sum of the True Positives and True Negatives divided by the total number of 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.

```r
# predict with test data
pred_test <- predict(cv_model4, trees_test)

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

# create confusion matrix
confusionMatrix(data = pred_test, reference = trees_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
##
```

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 this final model and the accuracy of the cross validation are extremely similar and essentially the same at approximately 90%. This is not surprising because the cross validation tests for generalizability and since there was a high accuracy, it makes sense that the model would predict another, unseen, set of data well.