ML Lab 4

Andrew Bartnik

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 (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))).

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

```
# This recodes all the predictors to a zero_based integer form
trees_data <- 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

```
# See if there is a class imbalance, if so, we'll stratify on the outcome. I checked the final models accuracy with and without stratifying on yr1status, and it looks like the final model performs slightly better with the stratification, so we'll keep it.
table(trees_dat$yr1status)
```

```
##
##     0     1
## 25833 10233
```

```
#Creating our initial split. there are almost twice as many alive trees as dead trees, so we'll stratify on the outcome
trees_data <- initial_split(trees_data, prop = 0.7, strata = yr1status)
trees_training <- training(trees_data)
trees_testing <- testing(trees_data)
```

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

```
nrow(trees_training)
```

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

We're using 25246 observations for training with this split.

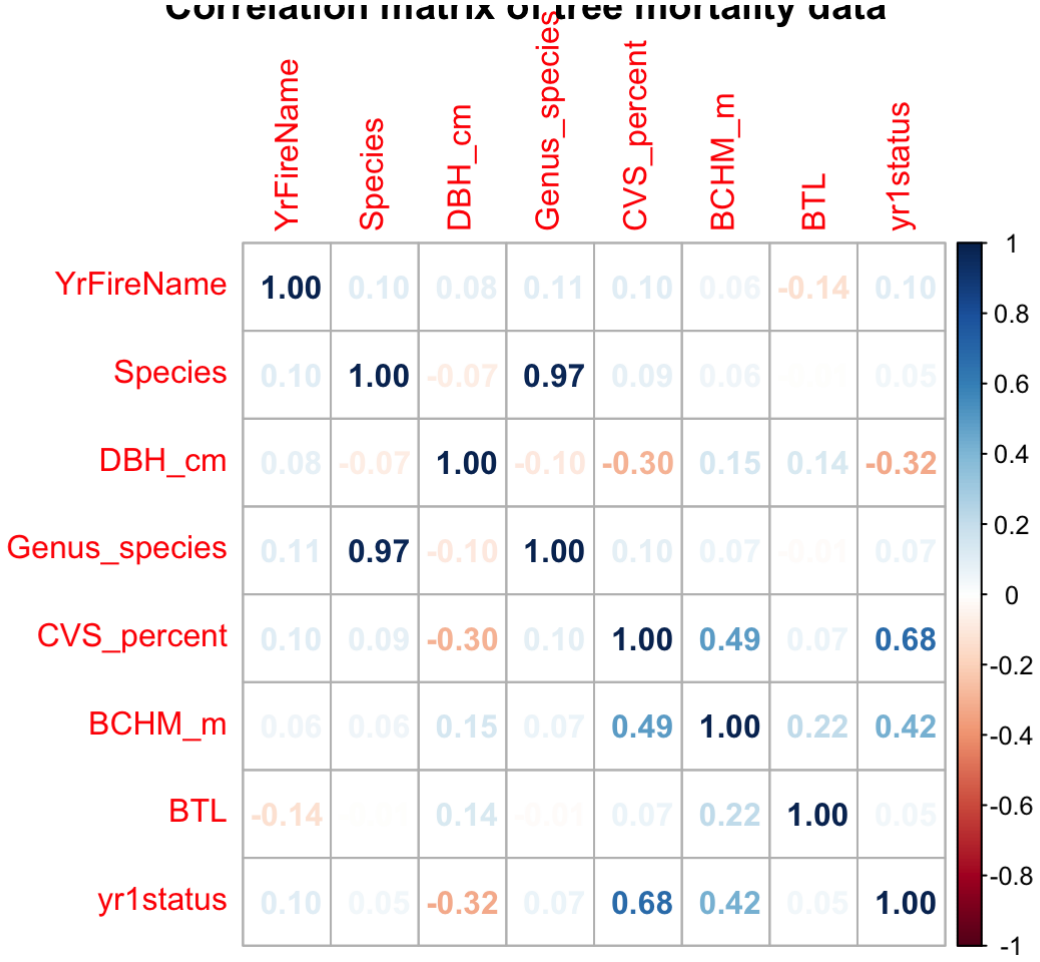
Simple Logistic Regression

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.

```
# Calculate the correlation matrix
correlation_matrix <- cor(trees_training)

# Plot the correlation matrix
corrplot(correlation_matrix, method = "number", title = "Correlation matrix of tree mortality data")
```



The three variables that most highly correlate with yr1status are: Percentage Crown-Volume Scorched `CVS_percent`, Maximum bark char height from the ground on a tree bole `BCHM_m`, and Diameter at Breast Height `DBH_cm`

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

```
# CVS_percent was most highly correlated with our outcome variable, so we'll make this m
odel first
mod_cvs <- glm(data = trees_training, yr1status ~ CVS_percent, family = 'binomial')

# Now for BCHM
mod_bchm <- glm(data = trees_training, yr1status ~ BCHM_m, family = 'binomial')

# And for DBH
mod_dbh <- glm(data = trees_training, 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.

```
library(broom)
# Using Broom to interpret our coeffs
tidy(mod_cvs)
```

```
## # A tibble: 2 × 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)  -6.79     0.116       -58.7       0
## 2 CVS_percent   0.0781   0.00126      62.1       0
```

```
tidy(mod_bchm)
```

```
## # A tibble: 2 × 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept) -1.99     0.0242       -82.2       0
## 2 BCHM_m       0.00626  0.000105      59.8       0
```

```
tidy(mod_dbh)
```

```
## # A tibble: 2 × 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  0.407   0.0283         14.4 9.59e-47
## 2 DBH_cm      -0.00378 0.0000779     -48.5 0
```

```
# Now to exponentiate our coefficients since they're log-transformed
exp(coef(mod_cvs))
```

```
## (Intercept) CVS_percent
## 0.001127365 1.081265308
```

```
exp(coef(mod_bchm))
```

```
## (Intercept)      BCHM_m
##   0.1369449   1.0062790
```

```
exp(coef(mod_dbh))
```

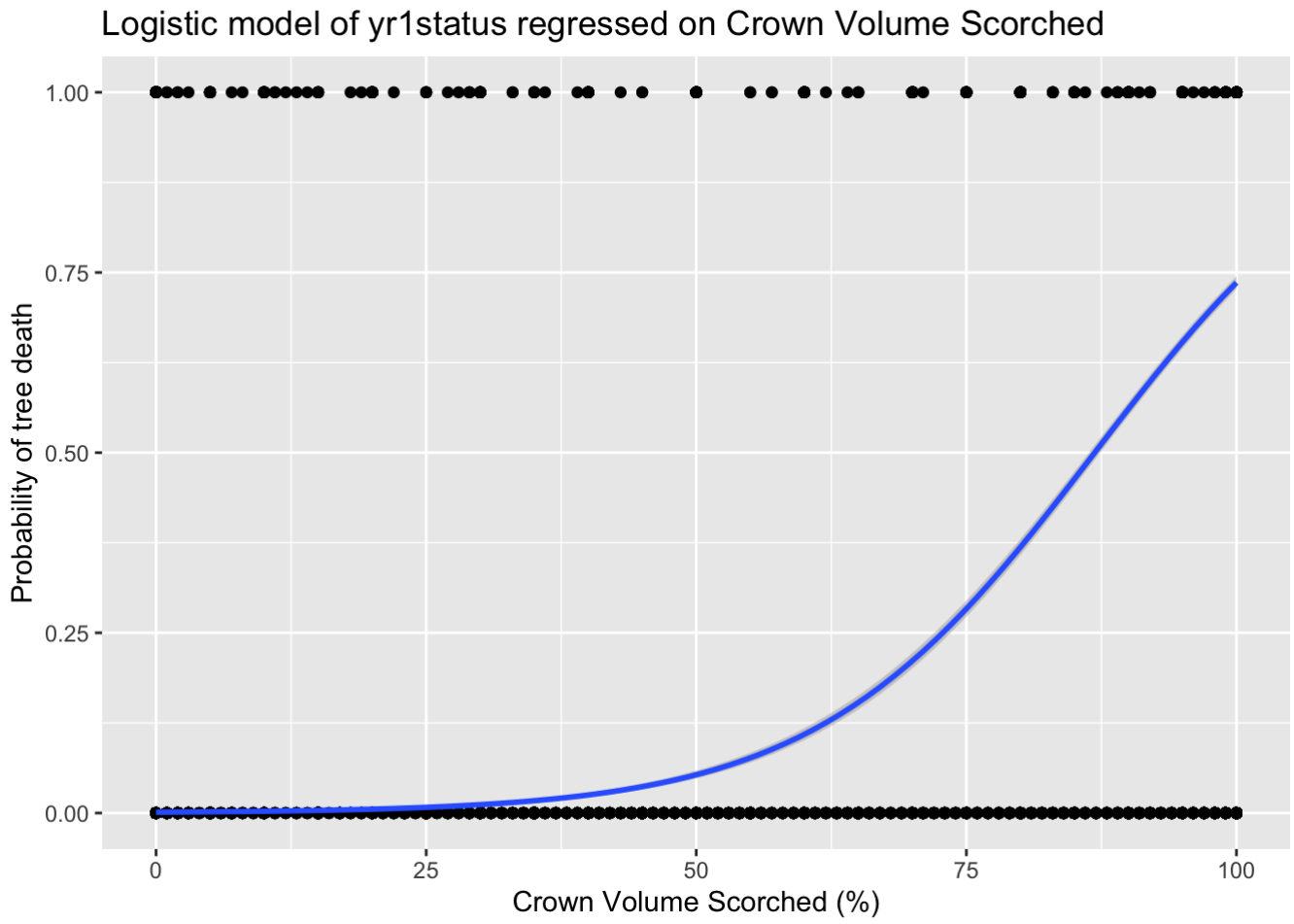
```
## (Intercept)      DBH_cm
##    1.501677    0.996225
```

For each 1 unit increase in CVS_percent, a tree's odds of death increases multiplicatively by 1.08. For each 1 unit increase in BCHM_m, a tree's odds of death increases multiplicatively by 1.006. For each 1 unit increase in DBH_cm, a trees odds of death increases multiplicatively by 0.996. In this last case, a 1 unit increase in DBH_cm actually corresponds to a decrease in a trees odds of death, since our multiplier is less than 1.

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

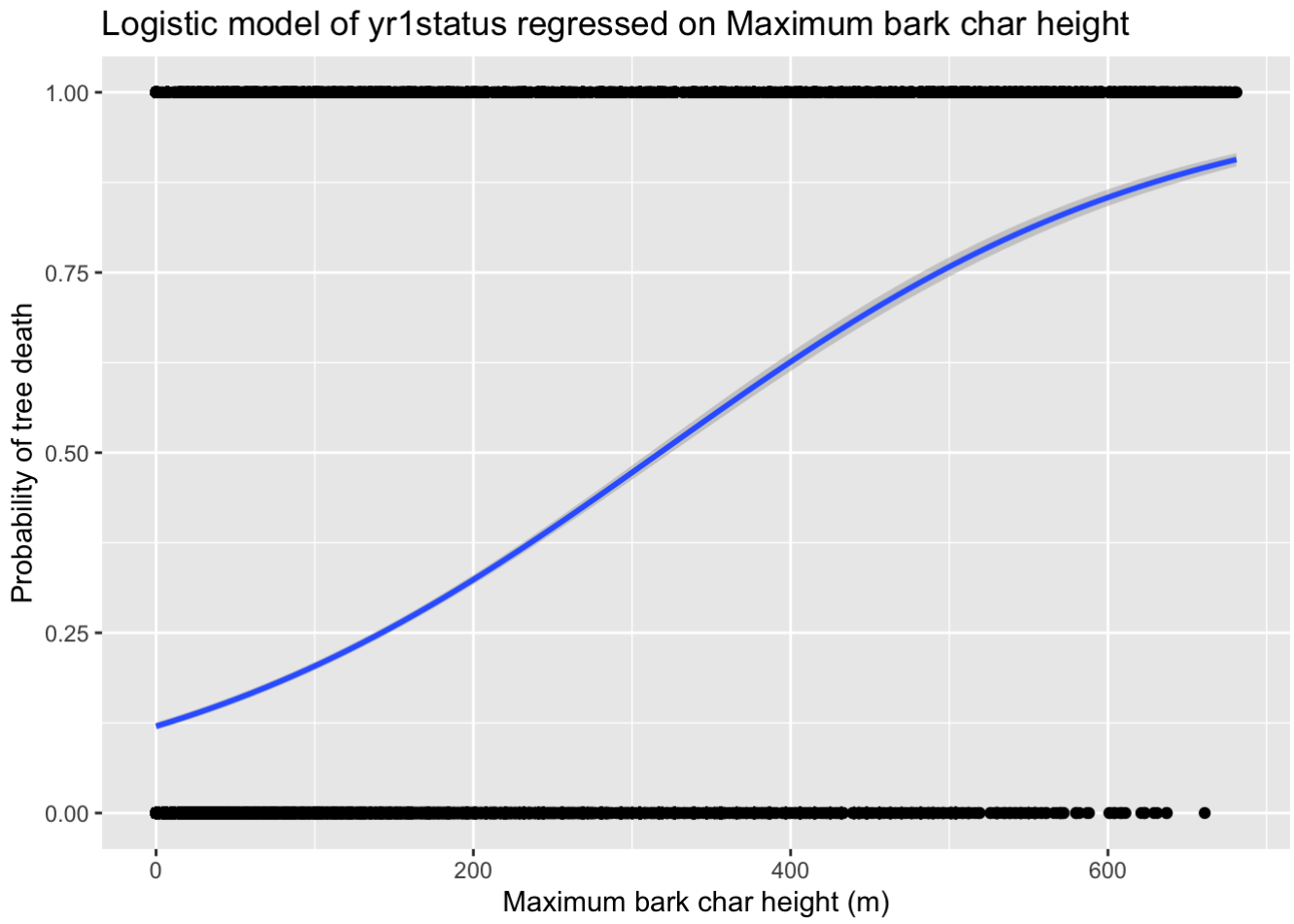
```
# Visualizing our CVS_percent model
ggplot(trees_training, aes(x = CVS_percent, y = yr1status)) + geom_point() +
  stat_smooth(
    method = "glm",
    se = TRUE,
    method.args = list(family = binomial)
  ) +
labs(title = "Logistic model of yr1status regressed on Crown Volume Scorched", x = "Crow
n Volume Scorched (%)", y = "Probability of tree death")
```

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



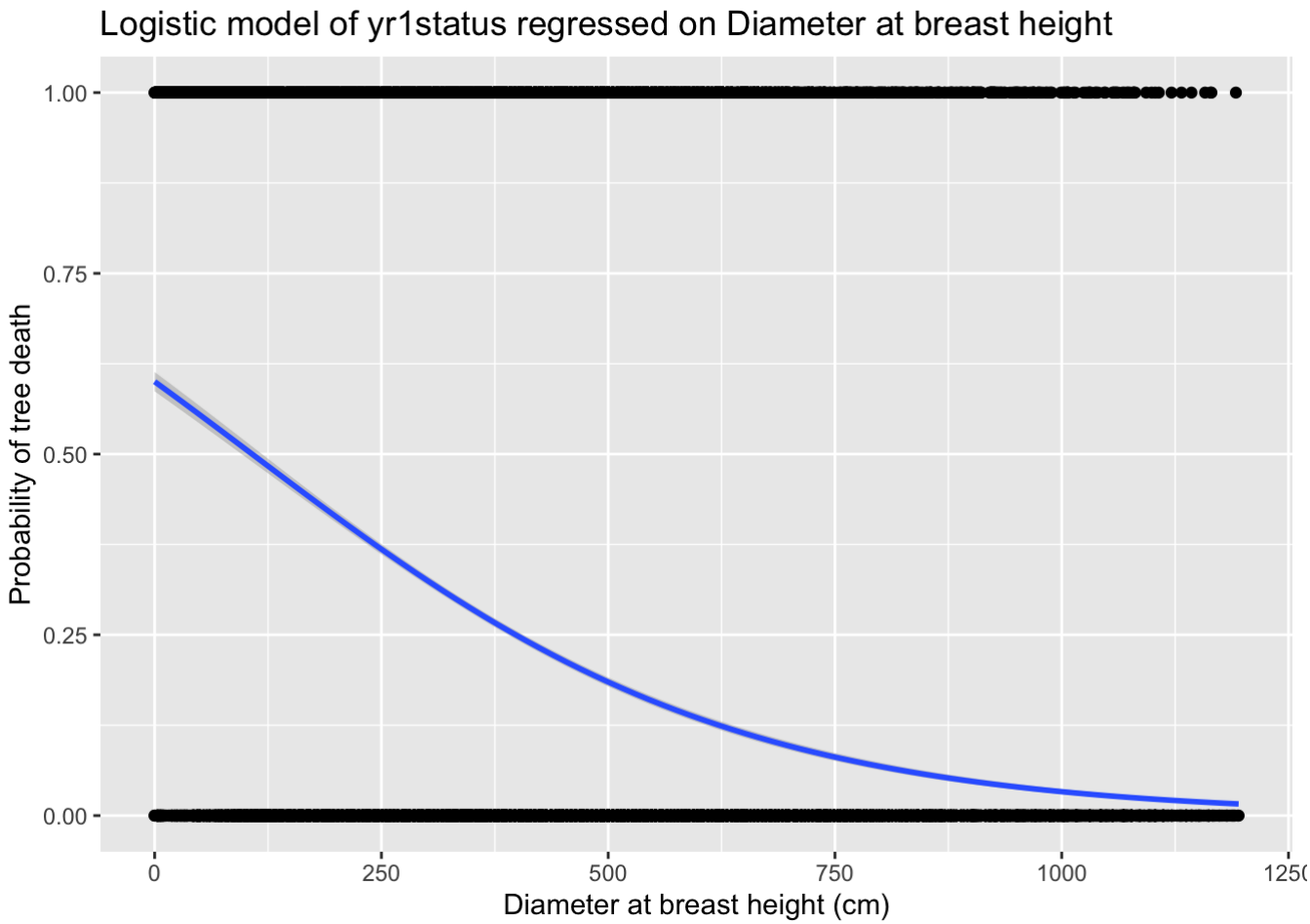

```
# Then for our BCHM_m model
ggplot(trees_training, aes(x = BCHM_m, y = yr1status)) + geom_point() +
  stat_smooth(
    method = "glm",
    se = TRUE,
    method.args = list(family = binomial)
  ) +
  labs(title = "Logistic model of yr1status regressed on Maximum bark char height", x =
"Maximum bark char height (m)", y = "Probability of tree death")
```

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




```
# And for our DBH_cm model
ggplot(trees_training, aes(x = DBH_cm, y = yr1status)) + geom_point() +
  stat_smooth(
    method = "glm",
    se = TRUE,
    method.args = list(family = binomial)
  ) +
    labs(title = "Logistic model of yr1status regressed on Diameter at breast height", x
= "Diameter at breast height (cm)", y = "Probability of tree death")
```

```
## `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_training, yr1status ~ CVS_percent + DBH_cm + BCHM_m, f
amily = "binomial")

tidy(logistic_full)
```

```
## # A tibble: 4 × 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept) -5.25     0.118        -44.5 0
## 2 CVS_percent  0.0638   0.00123       51.9 0
## 3 DBH_cm      -0.00363  0.000119     -30.6 3.60e-205
## 4 BCHM_m       0.00460  0.000162      28.5 3.13e-178
```

Each of these variables are significant in the resulting model, since the p-values for each of them are essentially 0.

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.

```
trees_training$yr1status <- as.factor(trees_training$yr1status)
levels(trees_training$yr1status) <- make.names(levels(trees_training$yr1status))


train_control <- trainControl(method = "cv", number = 10, classProbs = TRUE)

# First model based on CVS_percent
cv_model1 <- train(as.factor(yr1status) ~ CVS_percent, data = trees_training, method =
"glm", family = 'binomial', trControl = train_control)

# Second model based on DBH_cm
cv_model2 <- train(as.factor(yr1status) ~ DBH_cm, data = trees_training, method = "glm",
family = 'binomial', trControl = train_control)

# Third model based on BCHM_m
cv_model3 <- train(as.factor(yr1status) ~ BCHM_m, data = trees_training, method = "glm",
family = 'binomial', trControl = train_control)

# Fourth model based on all variables
cv_model4 <- train(as.factor(yr1status) ~ CVS_percent + DBH_cm + BCHM_m, data = trees_tr
aining, method = "glm", family = 'binomial', trControl = train_control)
```

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?

```
# Storing each model to a list
results <- resamples(list(Model1 = cv_model1, Model2 = cv_model2, Model3 = cv_model3, Mo
del4 = cv_model4))

# Looking at the results
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: Model1, Model2, Model3, Model4
## Number of resamples: 10
##
## Accuracy
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.8930269 0.8979208 0.8992083 0.8989149 0.9015550 0.9025357    0
## Model2 0.7461386 0.7496783 0.7526245 0.7535452 0.7546535 0.7631683    0
## Model3 0.7592079 0.7670297 0.7694591 0.7700238 0.7724078 0.7805071    0
## Model4 0.8961965 0.9004258 0.9035451 0.9042628 0.9048232 0.9219493    0
##
## Kappa
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.7536190 0.7643070 0.7678182 0.7671607 0.7736514 0.7753421    0
## Model2 0.2064245 0.2197520 0.2311940 0.2343148 0.2393445 0.2725019    0
## Model3 0.2993742 0.3271987 0.3372984 0.3361756 0.3418179 0.3701256    0
## Model4 0.7527185 0.7638486 0.7699581 0.7722260 0.7726730 0.8144458    0
```

It looks like model 4 has the highest overall accuracy

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.

```
pred_class <- predict(cv_model4, trees_training)

confusionMatrix(data = pred_class,
                reference = factor(trees_training$yr1status))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    X0    X1
##         X0 16466   804
##         X1  1617  6359
##
##                Accuracy : 0.9041
##                  95% CI : (0.9004, 0.9077)
##     No Information Rate : 0.7163
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7719
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9106
##             Specificity : 0.8878
##          Pos Pred Value : 0.9534
##          Neg Pred Value : 0.7973
##              Prevalence : 0.7163
##          Detection Rate : 0.6522
##    Detection Prevalence : 0.6841
##       Balanced Accuracy : 0.8992
##
##        'Positive' Class : X0
##
```

The overall fraction of correct predictions by the model is ~90/100

> 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 the model is more likely to predict false positives than false negatives. That is, it is more likely to predict that a tree is dead when it is actually alive than predicting a tree is alive when it is actually dead

> Question 13: What is the overall accuracy of the model? How is this calculated?
>
> **The overall accuracy of this model is ~90%. It is calculated by dividing the sum of observed true positives and true negatives with the sum of observed true positives, true negatives, false positives, and false negatives.**

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.

```
trees_testing$yr1status <- as.factor(trees_testing$yr1status)
levels(trees_testing$yr1status) <- make.names(levels(trees_testing$yr1status), unique =
TRUE)
final_model <- train(as.factor(yr1status) ~ CVS_percent + DBH_cm + BCHM_m, data = trees_
training,family = 'binomial', method = 'glm')

pred_final <- predict(final_model, trees_testing)
```

Question 15: How does the accuracy of this final model on the test data compare to its cross validation accuracy?

```
confusionMatrix(data = pred_final, reference = factor(trees_testing$yr1status))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   X0   X1
##         X0 7056  390
##         X1  694 2680
##
##                Accuracy : 0.8998
##                  95% CI : (0.894, 0.9054)
##     No Information Rate : 0.7163
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7607
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9105
##             Specificity : 0.8730
##          Pos Pred Value : 0.9476
##          Neg Pred Value : 0.7943
##              Prevalence : 0.7163
##          Detection Rate : 0.6521
##    Detection Prevalence : 0.6882
##       Balanced Accuracy : 0.8917
##
##        'Positive' Class : X0
##
```

Do you find this to be surprising? Why or why not?

The final model had an accuracy of ~90%, which is almost identical to its performance on the training data. This is not surprising, since we stratified on our outcome variable to address any class imbalances, we should achieve a similar score on our out of sample data.