131 Project Task 1

Classification trees

David Brackbill

Import, Tidy

We want winners of each county and we also want to drop variables that interfere with the classification tree.

```
# Get vote-winner of each county
winner_merged_data <- merged_data %>%
  group_by(fips) %>%
  top_n(1, pct) %>%
  rename(winner = candidate) %>%
  mutate(winner = factor(winner)) %>%
  ungroup %>%
  select(-c(fips, votes, pct))            #Drop problematic variables

#winner_merged_data[0:3,0:8] %>% pander()
```

Considerations

```
## # A tibble: 2 x 2
## # Groups:   winner [2]
##   winner              n
##   <fct>           <int>
## 1 Donald Trump     2606
## 2 Hillary Clinton   464
```

There are almost 6 times as many Trump-voting counties as Clinton-voting counties.

As a result, un-weighted partitioning might not make sense for this problem.

Classification: Winner prediction by county

Lab pre-sets

Our initial goal is to predictively classify the winner in each county using a decision tree. We'll start with the vanilla classification model from lab.

Features

We'll be classifying the winner (factor `winner`) based on the rest of the features. What does the feature space look like?

```
##  [1] "county"       "state"        "total"        "Women"        "White"
##  [6] "Citizen"      "IncomePerCap" "Poverty"      "ChildPoverty" "Professional"
## [11] "Service"      "Office"       "Production"   "Drive"        "Carpool"
## [16] "Transit"      "OtherTransp"  "WorkAtHome"   "MeanCommute"  "Employed"
```

```
## [21] "PrivateWork"  "SelfEmployed" "FamilyWork"   "Unemployment" "Minority"
## [1] p =  25
```

Partition

We'll reuse the options from Lab 4 to partition the data set, grow the tree, and prune the tree.

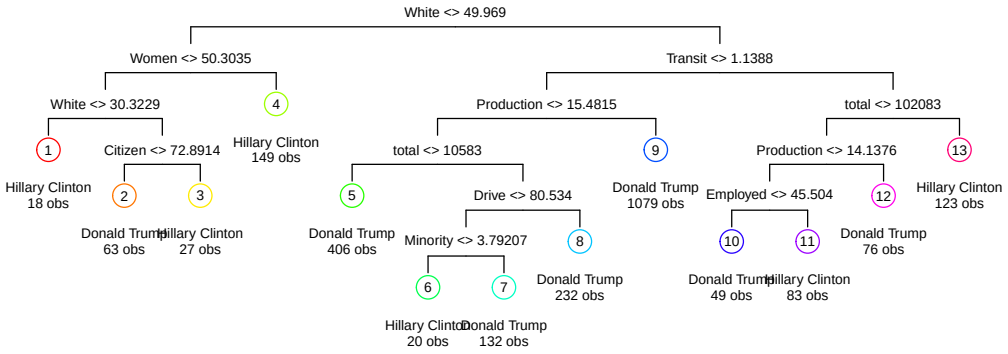
```
# hold out 20% of data as a test set
set.seed(12521)
winner_part <- resample_partition(winner_merged_data, c(test = 0.2, train = 0.8))
train <- as_tibble(winner_part$train)
test <- as_tibble(winner_part$test)
```

Prune large tree

```
# grow a large tree to prune
nmin <- 2
tree_opts <- tree.control(nobs = nrow(train),
                          minsize = nmin,
                          mindev = exp(-8))
t_0 <- tree(winner ~ ., data = train,
                control = tree_opts, split = 'deviance')

# cost-complexity pruning
nfolds <- 8
cv_out <- cv.tree(t_0, K = nfolds)

# convert to tibble
cv_df <- tibble(alpha = cv_out$k,
                impurity = cv_out$dev,
                size = cv_out$size)

# choose optimal alpha
best_alpha <- slice_min(cv_df, impurity) %>%
  slice_min(size)

# select final tree
t_opt <- prune.tree(t_0, k = best_alpha$alpha)
summary(t_opt)

# Save this tree and partition for later
t_12521 <- t_opt
winner_part_12521 <- winner_part
train_12521 <- train
test_12521 <- test
```

```
##
## Classification tree:
## snip.tree(tree = t_0, nodes = c(8L, 51L, 56L, 100L, 19L, 24L,
## 5L, 13L, 15L, 101L, 57L, 18L))
## Variables actually used in tree construction:
## [1] "White"      "Women"      "Citizen"    "Transit"    "Production"
## [6] "total"      "Drive"      "Minority"   "Employed"
## Number of terminal nodes:  13
## Residual mean deviance:  0.3372 = 824 / 2444
```

```
## Misclassification error rate: 0.06309 = 155 / 2457
```




```
## [1] Training Errors:
##                  pred
## class             Donald Trump Hillary Clinton
##   Donald Trump      0.95023923      0.04976077
##   Hillary Clinton   0.13896458      0.86103542
## [1]
## [1] Testing Errors:
##                  pred
## class             Donald Trump Hillary Clinton
##   Donald Trump      0.93410853      0.06589147
##   Hillary Clinton   0.20618557      0.79381443
```

Results

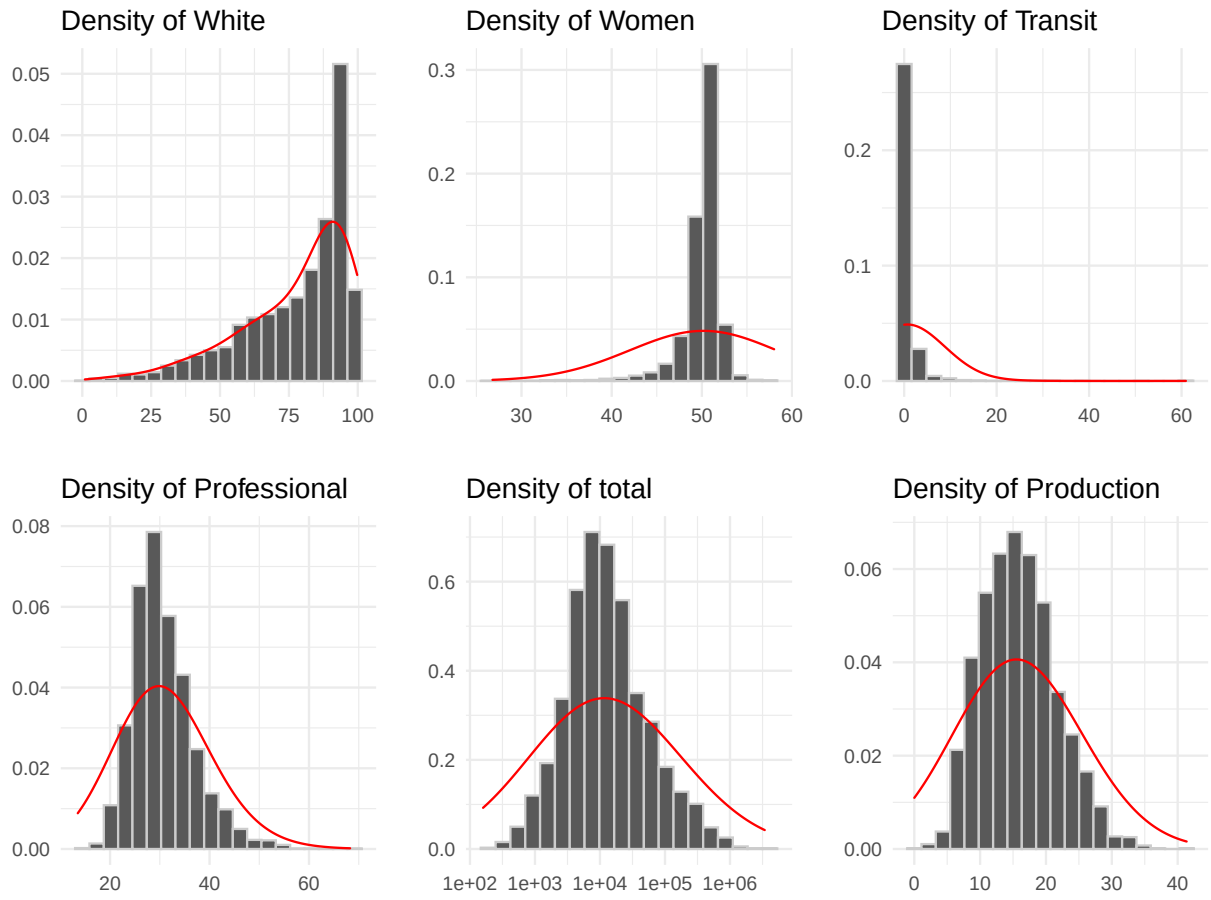
We see a very strong classification rate for Trump but an extremely weak classification rate for Clinton. Further iterations should bring their classification errors closer together.

EDA

When set.seed(12521), the pruned model used 6 variables:

`White`, `Women`, `Transit`, `Professional`, `total`, and `Production`

How are these variables distributed?



Something seems wrong with `transit`, which should have a more normal distribution.

```
# Transit range vs mean
winner_merged_data$Transit %>% range()
winner_merged_data$Transit %>% mean()
```

```
## [1]  0.00000 60.93217
## [1] 0.9377312
```

Upon further comparison of census_ref and census_tidy, our `transit` variable is correct. It seems that transit participation is just very low in the US.

Examining seed effect

How do our results vary based on the randomized seed?

We'll try out 30 more iterations with different seeds.

Re-run 30 times

Results

We see that, regardless of seed, classification of red counties far outperforms classification of blue counties.

Notably, our highest values of blue classification come with lowered values of red classification, signalling that finding and implementing Youden's statistic could improve forecasts for blue counties.

We also see that `Transit`, `White` and `total` were used in all 30 trees.

Because seed variability strongly affects the tree performance, in order to say we have improved our decision tree we have to get quite **strong** results that are **robust** against seed-setting.

Youden's statistic

```
# Get initial tree and partition
t_opt <- t_12521
winner_part <- winner_part_12521
train <- train_12521
test <- test_12521

# Calculate Youden's statistic on first pruned tree
probs_train <- predict(t_opt, newdata = train, type = 'vector')

topt_prediction <- prediction(predictions = probs_train[, 2],
 labels = classes_train)

topt_perf <- performance(topt_prediction, 'tpr', 'fpr')

rate_df <- tibble(fpr = topt_perf@x.values[[1]],
                  tpr = topt_perf@y.values[[1]],
                  thresh = topt_perf@alpha.values[[1]]) %>%
  mutate(youden = tpr - fpr)

# rate_df <- tibble(tpr = slot(topt_perf, 'y.values'),
#                   fpr = slot(topt_perf, 'x.values'),
#                   alpha = slot(topt_perf, 'alpha.values')) %>%
#   unnest(everything()) %>%
#   mutate(youden = tpr - fpr)

# Store new threshold
optimal_thresh <- slice_max(rate_df, youden) %>% select(youden)
```

```
# class probabilities on test partition
probs_test <- predict(t_opt, newdata = test, type = 'vector')

# predicted class labels
preds_test <- factor(probs_test[,2]
                     >= optimal_thresh$youden,
                     labels = colnames(probs_test))

noquote('Testing errors with optimal threshold:')
classes_test <- as.data.frame(test) %>% pull(winner)
test_errors <- table(class = classes_test, pred = preds_test)
test_errors / rowSums(test_errors)

noquote('Testing errors with automatic threshold:')
vanilla_test_errors
```

```
## [1] Testing errors with optimal threshold:
##                  pred
## class             Donald Trump Hillary Clinton
##   Donald Trump      0.97674419      0.02325581
##   Hillary Clinton   0.65979381      0.34020619
```

```
## [1] Testing errors with automatic threshold:
##                   pred
## class              Donald Trump Hillary Clinton
##   Donald Trump      0.93410853      0.06589147
##   Hillary Clinton   0.20618557      0.79381443
```

Why are results the same?

Gini loss

We'll prune a large tree again, this time replacing the deviance loss function with the Gini loss function.

Missingness

The Gini loss function mandates that there are no missing values. Do we have any missingness?

```
sum(is.na(winner_merged_data))
```

```
## [1] 0
```

No.

Prune large tree

We need to use the `rpart` and `rattle` libraries for this step because `tree` was giving us missing value errors when trying to set split = 'gini'.

```
## [1] Unpruned Training Errors:
##                   pred
## class              Donald Trump Hillary Clinton
##   Donald Trump      0.97942584      0.02057416
##   Hillary Clinton   0.18801090      0.81198910
## [1]
## [1] Unpruned Testing Errors:
##                   pred
## class              Donald Trump Hillary Clinton
##   Donald Trump      0.95542636      0.04457364
##   Hillary Clinton   0.25773196      0.74226804
## [1]
## [1] Pruned Training Errors:
##                   pred
## class              Donald Trump Hillary Clinton
##   Donald Trump      0.98133971      0.01866029
##   Hillary Clinton   0.24250681      0.75749319
## [1]
## [1] Pruned Testing Errors:
##                   pred
## class              Donald Trump Hillary Clinton
##   Donald Trump      0.96317829      0.03682171
##   Hillary Clinton   0.30927835      0.69072165
```

Results

We see better classification accuracy from the unpruned tree using Gini loss formula.

Comparing the unpruned Gini tree to the pruned 'deviance' tree, we see ___.

Overall, we don't see significant benefits from using the Gini loss formula.

Hyperparameters

Can we obtain better prediction by tuning the hyperparameters of the original pruned model we created?

Ideas:

- https://www.guru99.com/r-decision-trees.html#7
- Make a set of 5 values for each parameter and try all combinations?

K-Folds

Which k-fold validation gives us the best result for the seed?

Aside Is there a way to easily test model performance over a set of seeds??

```
## [[1]]
##    Donald Trump Hillary Clinton
##       0.9545455       0.8119891
##
## [[2]]
##    Donald Trump Hillary Clinton
##       0.9617225       0.8746594
##
## [[3]]
##    Donald Trump Hillary Clinton
##       0.9617225       0.8746594
##
## [[4]]
##    Donald Trump Hillary Clinton
##       0.9617225       0.8746594
##
## [[5]]
##    Donald Trump Hillary Clinton
##       0.9545455       0.8119891
##
## [[6]]
##    Donald Trump Hillary Clinton
##       0.9545455       0.8119891
##
## [[7]]
##    Donald Trump Hillary Clinton
##       0.9502392       0.8610354
##
## [[8]]
##    Donald Trump Hillary Clinton
##        0.977512        0.613079
##
## [[9]]
##    Donald Trump Hillary Clinton
##       0.9545455       0.8119891
##
## [[10]]
##    Donald Trump Hillary Clinton
##       0.9617225       0.8746594
```

Doesn't seem like any one obvious choice, although 3/4/5 seem to be the best.

nMin

```
## [[1]]
##    Donald Trump Hillary Clinton
##       0.9617225       0.8746594
##
## [[2]]
##    Donald Trump Hillary Clinton
##       0.9617225       0.8746594
##
## [[3]]
##    Donald Trump Hillary Clinton
##       0.9617225       0.8746594
```

Has no effect.

Grid search

Source: https://www.jeremyjordan.me/hyperparameter-tuning/#:~:text=Grid%20search%20is%20arguably%20the,which%

Further optimizations

Classifying on unbalanced data is hard. How can we improve this decision tree model?

- General ideas
- Transform feature space
  - Scale, normalize (White, Transit are skewed and total can be normalized)
    - This does nothing
  - Feature creation/selection

Normalize features

Pruned tree

```
## [1] Training Errors:
##                  pred
## class             Donald Trump Hillary Clinton
##   Donald Trump      0.95023923      0.04976077
##   Hillary Clinton   0.13896458      0.86103542
## [1]
## [1] Testing Errors:
##                  pred
## class             Donald Trump Hillary Clinton
##   Donald Trump      0.93410853      0.06589147
##   Hillary Clinton   0.20618557      0.79381443
```

Results This does nothing lol.