# Tweet Decision Tree

Ada Lazuli

2022-07-17

```
library(tidymodels)
## -- Attaching packages -------------------------------------- tidymodels 1.0.0 --
## v broom        1.0.0     v recipes      1.0.1
## v dials        1.0.0     v rsample      1.0.0
## v dplyr        1.0.9     v tibble       3.1.8
## v ggplot2      3.3.6     v tidyr        1.2.0
## v infer        1.0.2     v tune         1.0.0
## v modeldata    1.0.0     v workflows    1.0.0
## v parsnip      1.0.0     v workflowsets 1.0.0
## v purrr        0.3.4     v yardstick    1.0.0
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Search for functions across packages at https://www.tidymodels.org/find/
library(rattle)
## Loading required package: bitops
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
library("ggfortify")
## Registered S3 method overwritten by 'ggfortify':
##   method          from
##   autoplot.glmnet parsnip
set.seed(101011)
```

## Data Loading

```
df <- read.csv("../data/enriched_tweet_data.csv")
# convert the classes to factors
df$X <- NULL
df$tweet <- NULL
df$tweet_orig <- NULL
df$hashtag <-NULL
df$link <- NULL
df$label <- as.factor(df$label)
data_partitioned <- initial_split(df, prop = 0.75, strata = label)
```

```
train <- training(data_partitioned)
test <-  testing(data_partitioned)
```

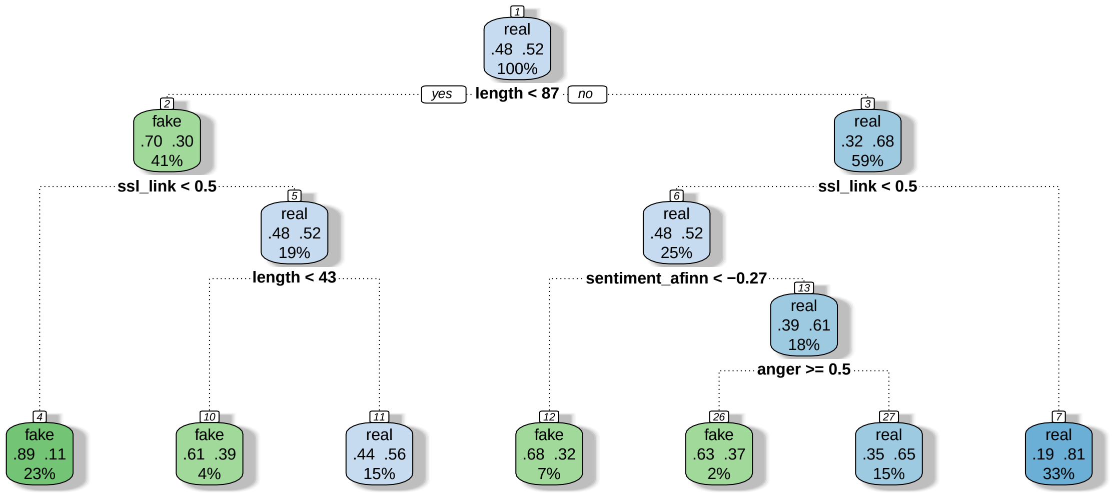
## Tree Creation

### First Attempt

#### Model Definition

The first attempt involved using all of the available features and not specifying any limitations to the model. The tree is created using the `parsnip` package in `tidymodels`, with the *rpart* engine and set for classification (Kuhn). The tree was fit on the data, using all of the available columns

```
tree_template <- decision_tree() %>%
  set_engine("rpart") %>%
  set_mode("classification")
tree_model <- tree_template %>%
  fit(formula = label ~ ., data =  train)
fancyRpartPlot(tree_model$fit, caption = "First Decision Tree Attempt")
```



First Decision Tree Attempt

### Model Results

To assess the performance of the tree, the accuracy, confusion matrix, ROC Curve, and AUC are all captured (Han et al, 2011, p. 49).

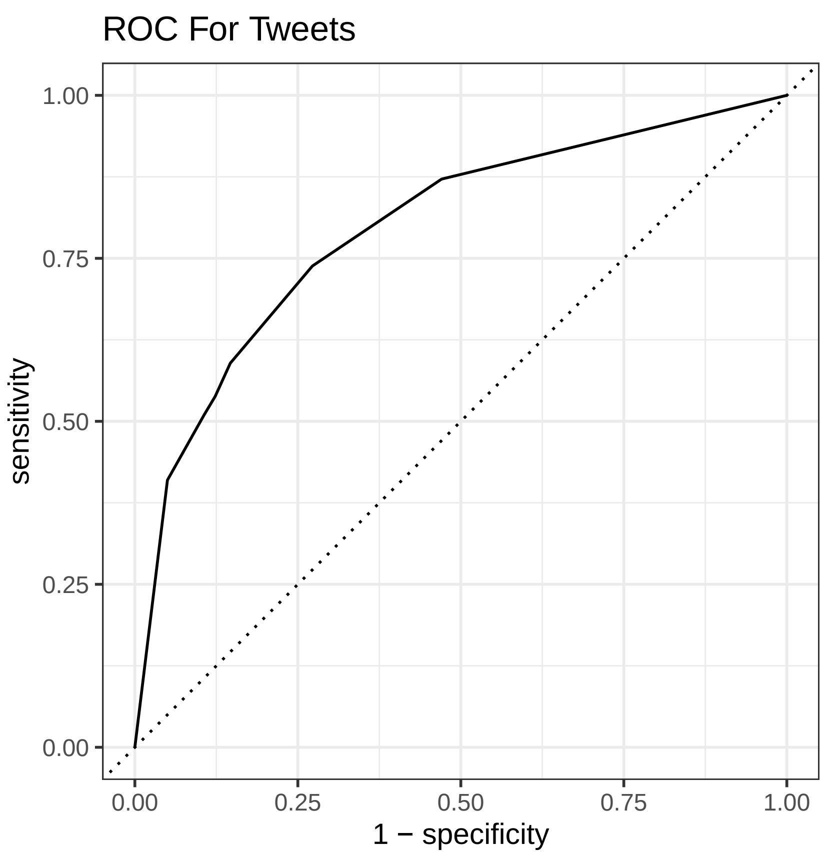
```r
# For the confusion Matrix
predictions <- predict(tree_model, test) %>% mutate(true = test$label)
# For the plot of the ROC Curve
predictions_prob <- predict(tree_model, test, type = "prob") %>% bind_cols(test)
accuracy(data = predictions, estimate = .pred_class, truth = true)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.728
```

```r
conf_mat(data = predictions, estimate = .pred_class, truth = true)
```

```
##           Truth
## Prediction fake real
##       fake  601  164
##       real  419  956
```

```r
autoplot(roc_curve(data = predictions_prob, estimate = .pred_fake, truth = label)) +
  ggtitle("ROC For Tweets")
```



```
print(roc_auc(data = predictions_prob, estimate = .pred_fake, truth = label))
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.797
```

**Note**: The article written by Brendan Cullen (2021) here helped a bit with using features available in the collection packages found in tidymodels.

## Second Attempt

### Optimization

The second attempt is to use a grid search to find the optimal combination of *min_n, tree depth, and cost complexity* for the model using `tune_grid` from the `tidymodels` set of packages (Kuhn).

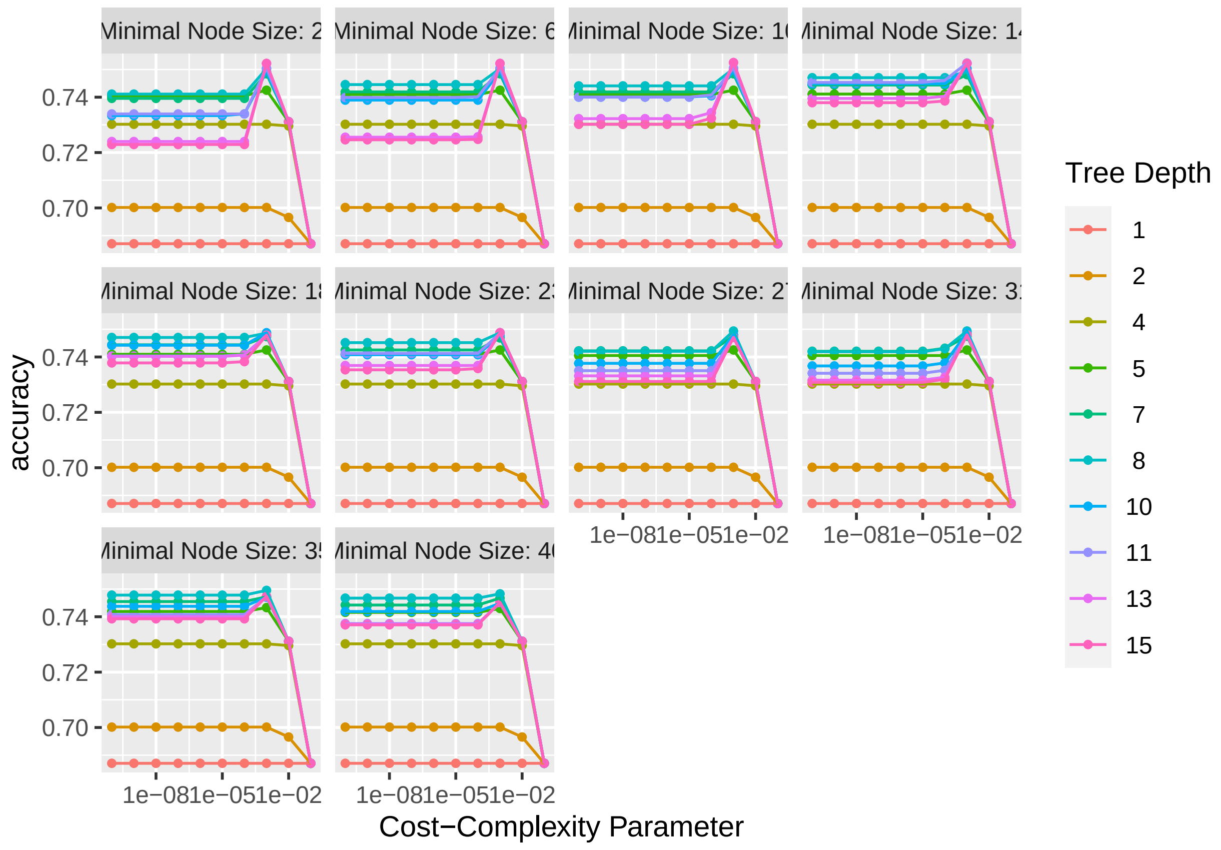
```
tune_specification <- decision_tree(tree_depth = tune(), min_n = tune(), cost_complexity = tune()) %>% s

grid_search <- grid_regular(parameters(tune_specification), levels = 10)
```

```
## Warning: ‘parameters.model_spec()‘ was deprecated in tune 0.1.6.9003.
## Please use ‘hardhat::extract_parameter_set_dials()‘ instead.
```

```
tuned <- tune_grid(tune_specification, label ~ ., resample = vfold_cv(train, v = 3), grid = grid_search

autoplot(tuned)
```



### Using Best Parameters

Following the grid search, the best performing set of parameters were saved and used to create a second model.

```
optimal_parameters <- select_best(tuned)
print(optimal_parameters)
```

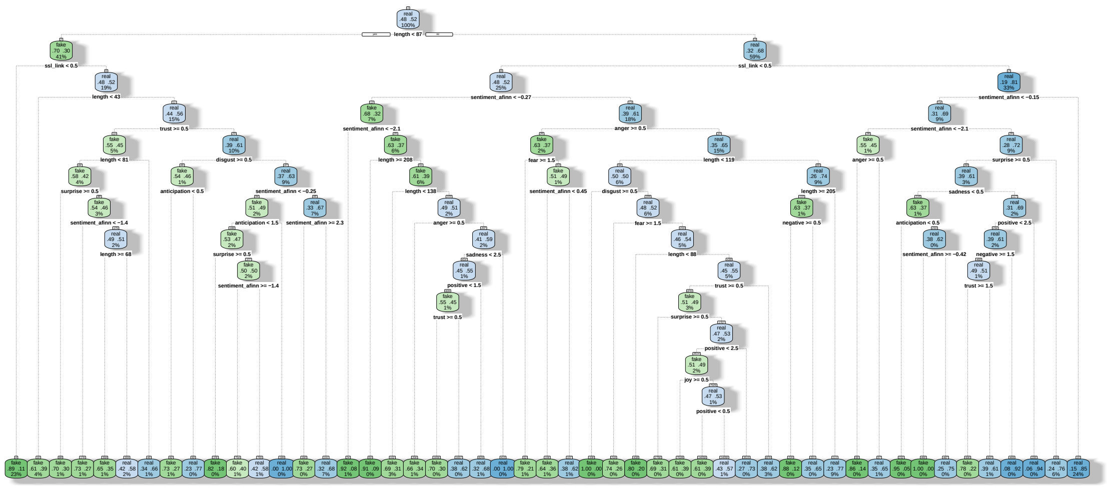
```
## # A tibble: 1 x 4
##   cost_complexity tree_depth min_n .config
##             <dbl>      <int> <int> <chr>
## 1           0.001         13    10 Preprocessor1_Model0288
```

```
optimal_tree_specification <- finalize_model(tune_specification, optimal_parameters)

optimal_model <- fit(optimal_tree_specification,
                     label ~ .,
                     train)

fancyRpartPlot(optimal_model$fit, caption = "Final Decision Tree Attempt")
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Final Decision Tree Attempt

### Model Performance

To assess the performance of the tree, the accuracy, confusion matrix, ROC Curve, and AUC are all captured (Han et al, 2011, p. 49).

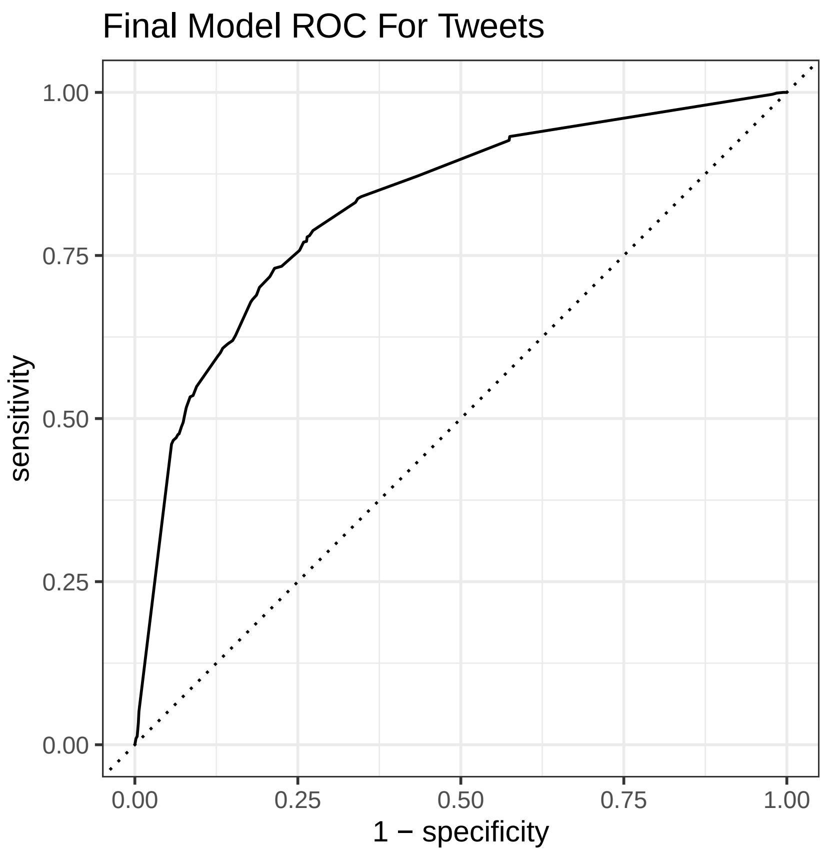
```
predictions <- predict(optimal_model, test) %>% mutate(true = test$label)
predictions_prob <- predict(optimal_model, test, type = "prob") %>% bind_cols(test)
accuracy(data = predictions, estimate = .pred_class, truth = true)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.754
```

```
conf_mat(data = predictions, estimate = .pred_class, truth = true)
```

```
##           Truth
## Prediction fake real
##       fake  703  209
##       real  317  911
```

```
autoplot(roc_curve(data = predictions_prob, estimate = .pred_fake, truth = label)) + ggtitle("Final Mode
```



```r
print(roc_auc(data = predictions_prob, estimate = .pred_fake, truth = label))
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.825
```

## References

Han, Kamber, & Pei. (2011). Chapter 8. Classification: Basic Concepts. Elsevier Science. Kuhn, M. (n.d.) *Model Tuning Via Grid Search* Retrieved from: https://tune.tidymodels.org/reference/tune_grid.html Kuhn, M. (n.d.) *Decision Trees*. Retrieved from: https://parsnip.tidymodels.org/reference/decision_tree.html Silge, J. (n.d). *Simple Training/Test Set Splitting*. Retrieved from: https://rsample.tidymodels.org/reference/initial_split.html