Classificaiton Trees on Cumulative Risk

Lyndsey Umsted

2022-11-18

Loading neccessary libraries:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.0.0 --
## v broom        1.0.1     v rsample      1.1.0
## v dials        1.0.0     v tune         1.0.1
## v infer        1.0.3     v workflows    1.1.0
## v modeldata    1.0.1     v workflowsets 1.0.0
## v parsnip      1.0.2     v yardstick    1.1.0
## v recipes      1.0.1
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```
library(ISLR)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.2.2
```

```
## Loading required package: rpart
##
## Attaching package: 'rpart'
```

```
##
## The following object is masked from 'package:dials':
##
##     prune
```

```
library(vip)
```

```
## Warning: package 'vip' was built under R version 4.2.2
```

```
##
## Attaching package: 'vip'
##
## The following object is masked from 'package:utils':
##
##     vi
```

```
library(janitor)
```

```
## Warning: package 'janitor' was built under R version 4.2.2
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

Splitting into training and testing

```
setwd("C:/Users/18586/Desktop/PSTAT 131/PSTAT-131-final-project/models")
load("pandemic_cum.rda")

set.seed(2002)

pandemic_cum <- pandemic_cum %>%
  select(-c("prop", "confirmed_cases", "population", "ID_co", "proplog"))

pandemic_cum_split <- initial_split(pandemic_cum, prop = 0.80, strata = risk)

pandemic_cum_train <- training(pandemic_cum_split)
pandemic_cum_test <- testing(pandemic_cum_split)
```

Fitting classification tree:

```
tree_spec <- decision_tree() %>%
  set_engine("rpart")

class_tree_spec <- tree_spec %>%
  set_mode("classification")
```

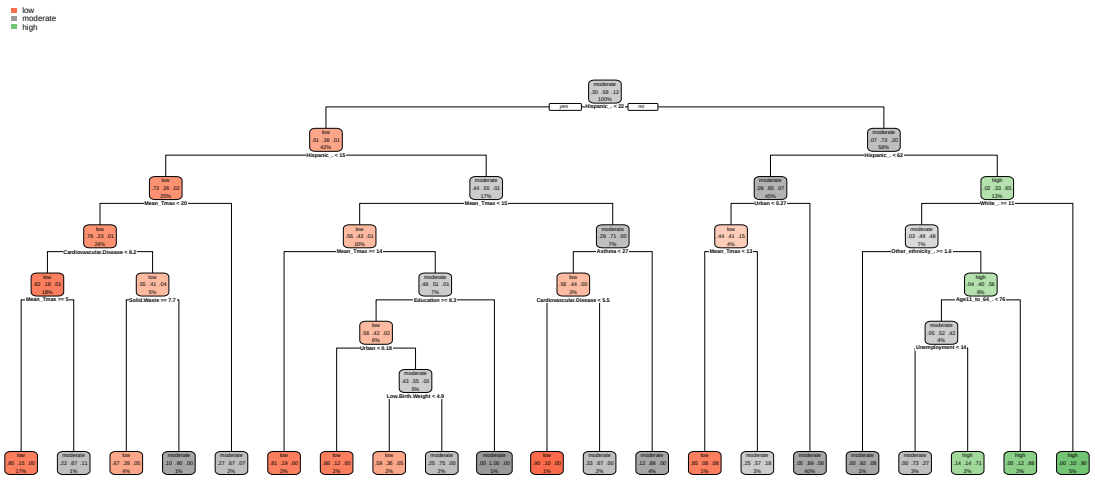
Fit the model:

```
class_tree_fit <- class_tree_spec %>%
  fit(risk ~., data = pandemic_cum_train)
```

Visualize the decision tree:

```
class_tree_fit %>%
  extract_fit_engine() %>%
  rpart.plot()
```

```
## Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and is.binary
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```



Training set accuracy:

```
augment(class_tree_fit, new_data = pandemic_cum_train) %>%
  accuracy(truth = risk, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.837
```

Confusion Matrix:

```
augment(class_tree_fit, new_data = pandemic_cum_train) %>%
  conf_mat(truth = risk, estimate = .pred_class)
```

```
##           Truth
## Prediction low moderate high
##   low      227       51    4
##   moderate  47      475   37
##   high       2        9   69
```

Accuracy on testing set:

```
augment(class_tree_fit, new_data = pandemic_cum_test) %>%
  accuracy(truth = risk, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.703
```

Confusion Matrix:

```
augment(class_tree_fit, new_data = pandemic_cum_test) %>%
  conf_mat(truth = risk, estimate = .pred_class)
```

```
##           Truth
## Prediction low moderate high
##   low       42       20    2
##   moderate  28      111   16
##   high       0        3   10
```

Let us try to tune the cost_complexity of the decision tree, or the pruning penalty, to find a more optimal complexity. We use the class_tree_spec object and use the set_args() function to specify that we want to tune cost_complexity. This is then passed directly into the workflow object to avoid creating an intermediate object.

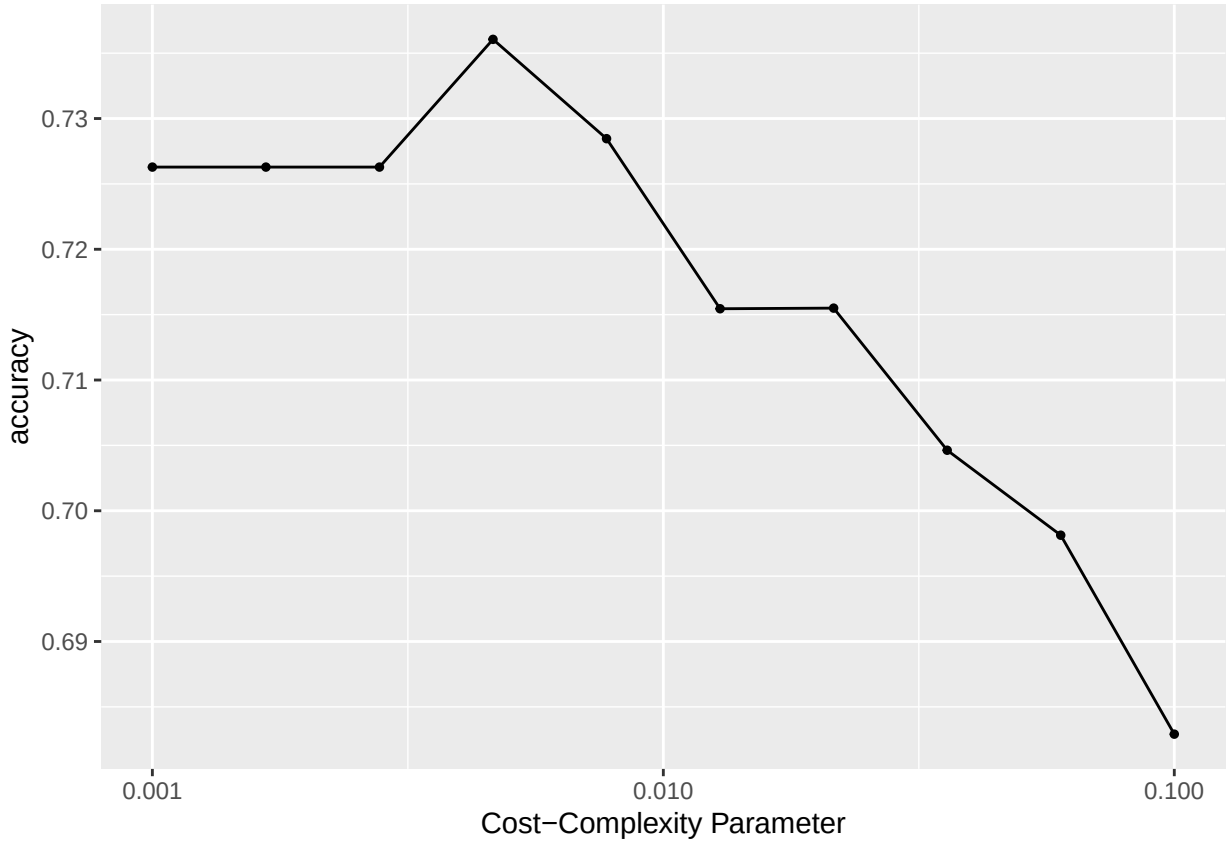
```
class_tree_wf <- workflow() %>%
  add_model(class_tree_spec %>% set_args(cost_complexity = tune())) %>%
  add_formula(risk ~ .)
```

To be able to tune the hyperparameter, we need 2 more objects – a resamples object (we will use a k-fold cross-validation data set), and a grid of values to try. Since we are only tuning one hyperparameter, a regular grid is probably simplest.

```
set.seed(3435)
pandemic_fold <- vfold_cv(pandemic_cum_train)

param_grid <- grid_regular(cost_complexity(range = c(-3, -1)), levels = 10)

tune_res <- tune_grid(
  class_tree_wf,
  resamples = pandemic_fold,
  grid = param_grid,
  metrics = metric_set(accuracy)
)
```

Using autoplot() shows which values of cost_complexity appear to produce the highest accuracy:

```
autoplot(tune_res)
```



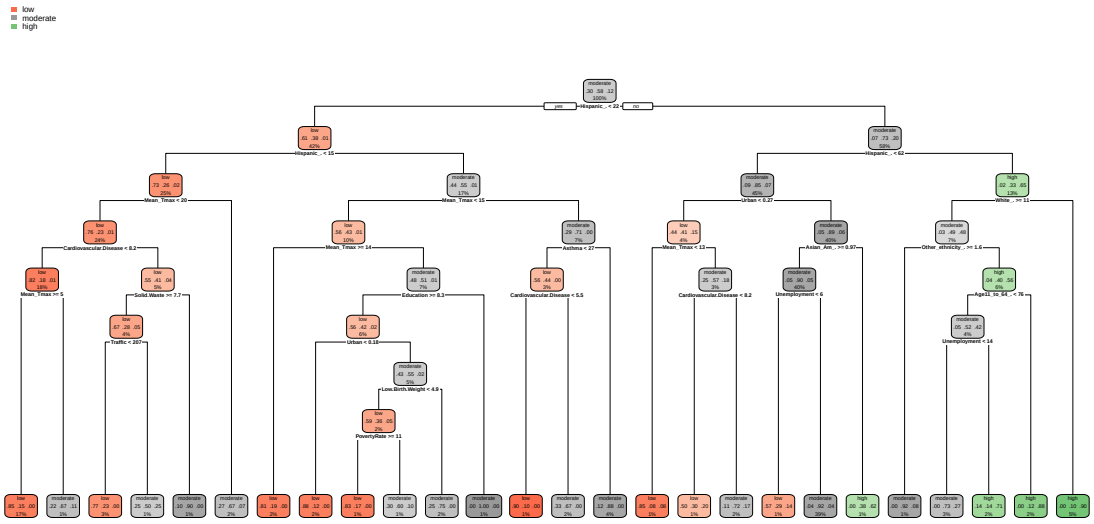
We can now select the best performing value with select_best(), finalize the workflow by updating the value of cost_complexity, and fit the model on the full training data set.

```
best_complexity <- select_best(tune_res)

class_tree_final <- finalize_workflow(class_tree_wf, best_complexity)

class_tree_final_fit <- fit(class_tree_final, data = pandemic_cum_train)
```

At last we can visualize the model, and we see that the better-performing model is much less complex than the original model we fit.

```
class_tree_final_fit %>%
  extract_fit_engine() %>%
  rpart.plot()
```

```
## Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and is.binary
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```



Accuracy on testing set:

```
augment(class_tree_final_fit, new_data = pandemic_cum_test) %>%
  accuracy(truth = risk, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.694
```

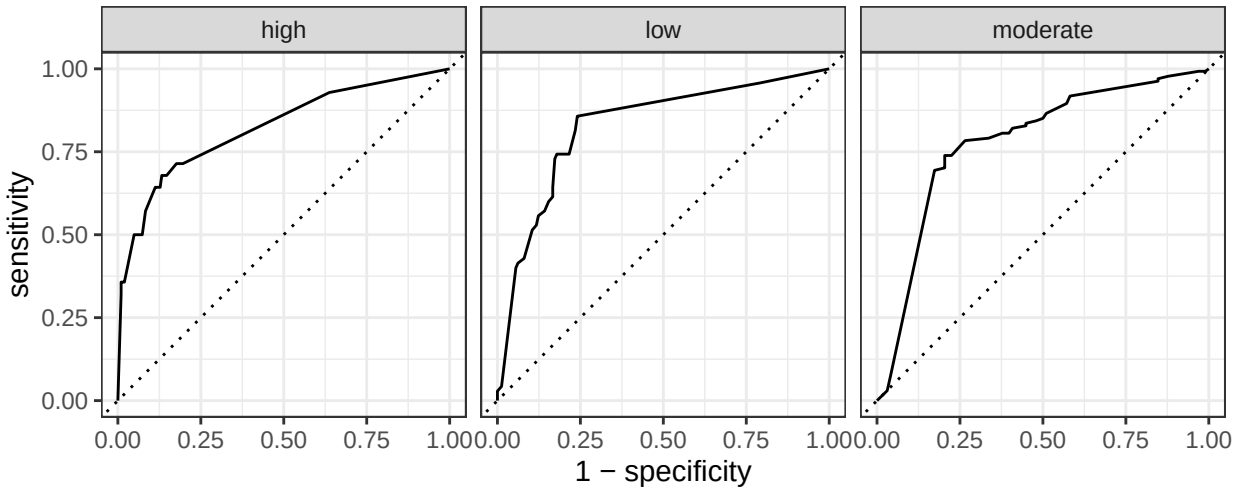
Confusion Matrix:

```
augment(class_tree_final_fit, new_data = pandemic_cum_test) %>%
  conf_mat(truth = risk, estimate = .pred_class)
```

```
##           Truth
## Prediction low moderate high
##   low       40       19    4
##   moderate  30      111   14
##   high       0        4   10
```

ROC Curve:

```
roc <- augment(class_tree_final_fit, pandemic_cum_test)

roc %>%
  roc_curve(risk, c(.pred_low, .pred_moderate, .pred_high)) %>%
  autoplot()
```



AUC:

```
roc %>%
  roc_auc(risk, c(.pred_low, .pred_moderate, .pred_high))
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.814
```

Bagging and Random Forest:

```
bagging_spec <- rand_forest(mtry = .cols()) %>%
  set_engine("randomForest", importance = TRUE) %>%
  set_mode("classification")
```

fit the model:

```
bagging_fit <- fit(bagging_spec, risk ~ .,
                   data = pandemic_cum_train)
```

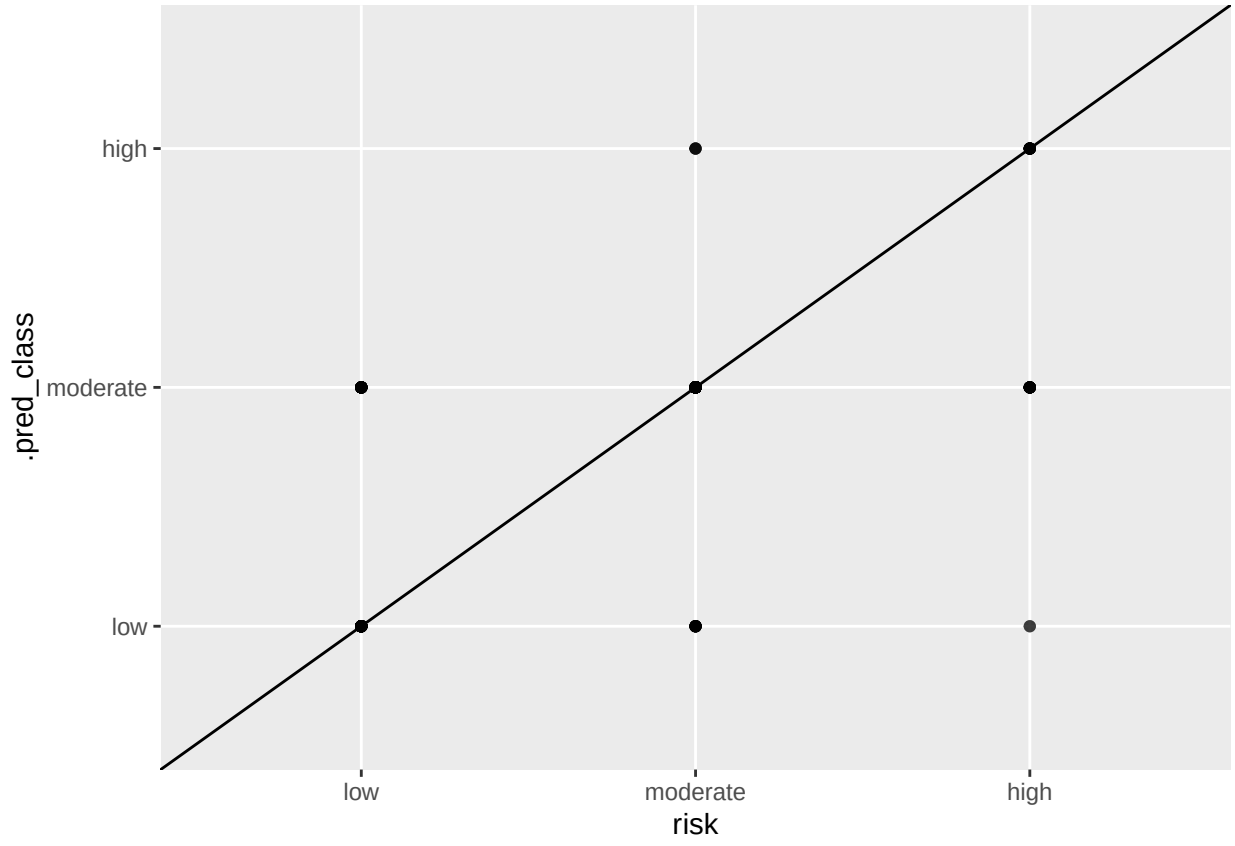
testing performance:

```
augment(bagging_fit, new_data = pandemic_cum_test) %>%
  accuracy(truth = risk, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.746
```

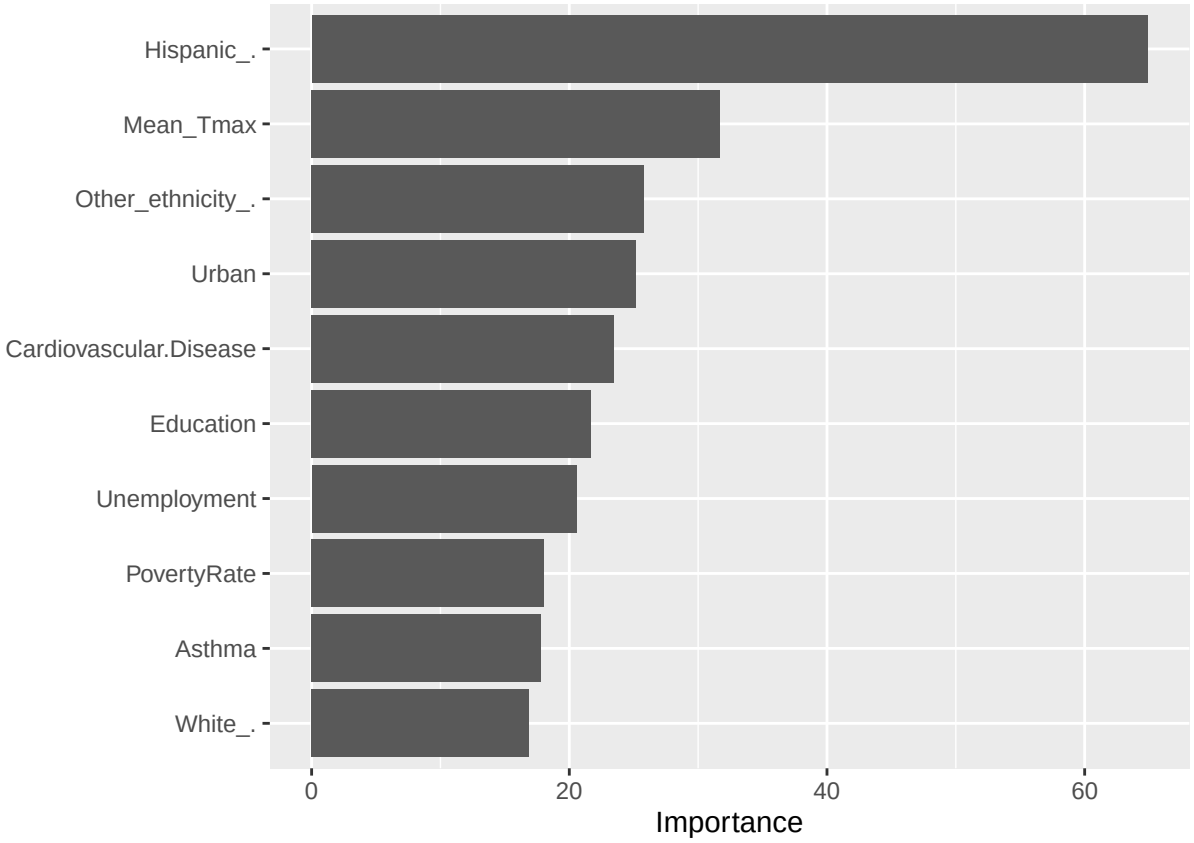
We can also create a quick scatterplot between the true and predicted values:

```
augment(bagging_fit, new_data = pandemic_cum_test) %>%
  ggplot(aes(risk, .pred_class)) +
  geom_abline() +
  geom_point(alpha = 0.5)
```



Variable Importance:

```
vip(bagging_fit)
```



looking at the random forest:

```
rf_spec <- rand_forest(mtry = 6) %>%
  set_engine("randomForest", importance = TRUE) %>%
  set_mode("classification")
```

and fitting the model like normal:

```
rf_fit <- fit(rf_spec, risk ~ ., data = pandemic_cum_train)
```

accuracy on training:

```
augment(rf_fit, new_data = pandemic_cum_train) %>%
  accuracy(truth = risk, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.999
```

accuracy on testing:

```
augment(rf_fit, new_data = pandemic_cum_test) %>%
  accuracy(truth = risk, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.746
```

Variable importance:

```
vip(rf_fit)
```