Classification

Sar, North, Henry, Quinn

3/29/2022

```
library(ISLR)
library(dplyr)
```

```
## Warning: replacing previous import 'lifecycle::last_warnings' by
## 'rlang::last_warnings' when loading 'pillar'
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)
```

```
## Warning: replacing previous import 'lifecycle::last_warnings' by
## 'rlang::last_warnings' when loading 'hms'
```

```
library(broom)
```

```
## Warning: package 'broom' was built under R version 4.1.2
```

```
library(ggplot2)
library(splines)
library(tidymodels)
```

```
## Registered S3 method overwritten by 'tune':
##   method                   from
##   required_pkgs.model_spec parsnip
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.1.4 --
```

```
## v dials        0.0.10     v tibble       3.1.6
## v infer        1.0.0      v tidyr        1.1.4
## v modeldata    0.1.1      v tune         0.1.6
## v parsnip      0.1.7      v workflows    0.2.4
## v purrr        0.3.4      v workflowsets 0.1.0
## v recipes      0.1.17     v yardstick    0.0.9
## v rsample      0.1.1

## -- Conflicts ---------------------------------------- tidymodels_conflicts() --
## x purrr::discard()  masks scales::discard()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Search for functions across packages at https://www.tidymodels.org/find/
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(maps)
```

```
##
## Attaching package: 'maps'

## The following object is masked from 'package:purrr':
##
##     map
```

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(vip)
```

```
##
## Attaching package: 'vip'
```

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

```
library(probably)
```

```
##
## Attaching package: 'probably'
```

```
## The following objects are masked from 'package:base':
##
##     as.factor, as.ordered
```

```
tidymodels_prefer()
```

```
COVID_State <- read.csv("COVID - State - Daily.csv", na.strings = ".")

Employment_State <- read.csv("Employment - State - Daily.csv", na.strings = ".")

Mobility_State <- read.csv("Google Mobility - State - Daily.csv", na.strings = ".")

Spending_State <- read.csv("Affinity - State - Daily.csv", na.strings = ".")

regions <- read.csv("regions.csv")
fips <- state.fips
```

```
COVID_State$Date<-as.Date(with(COVID_State,paste(year,month,day,sep="-")),"%Y-%m-%d")

Employment_State$Date<-as.Date(with(Employment_State,paste(year,month,day,sep="-")),"%Y-%m-%d")

Mobility_State$Date<-as.Date(with(Mobility_State,paste(year,month,day,sep="-")),"%Y-%m-%d")

Spending_State$Date<-as.Date(with(Spending_State,paste(year,month,day,sep="-")),"%Y-%m-%d")

full_data <- merge(merge(merge(COVID_State, Employment_State, by=c("Date","statefips")), Mobility_State
```

```
## Warning in merge.data.frame(merge(merge(COVID_State, Employment_State, by =
## c("Date", : column names 'year.x', 'month.x', 'day.x', 'year.y', 'month.y',
## 'day.y' are duplicated in the result
```

```
head(full_data)
```

```
##         Date statefips year.x month.x day.x new_case_count new_death_count
## 1 2020-02-24         1   2020       2    24             NA              NA
```

```
## 2 2020-02-24        10   2020       2    24             NA              NA
## 3 2020-02-24        11   2020       2    24             NA              NA
## 4 2020-02-24        12   2020       2    24             NA              NA
## 5 2020-02-24        13   2020       2    24             NA              NA
## 6 2020-02-24        15   2020       2    24             NA              NA
##   case_count death_count vaccine_count fullvaccine_count booster_first_count
## 1         NA          NA            NA                NA                  NA
## 2         NA          NA            NA                NA                  NA
## 3         NA          NA            NA                NA                  NA
## 4         NA          NA            NA                NA                  NA
## 5         NA          NA            NA                NA                  NA
## 6         NA          NA            NA                NA                  NA
##   new_vaccine_count new_fullvaccine_count new_booster_first_count
## 1                NA                   NA                     NA
## 2                NA                   NA                     NA
## 3                NA                   NA                     NA
## 4                NA                   NA                     NA
## 5                NA                   NA                     NA
## 6                NA                   NA                     NA
##   new_test_count test_count hospitalized_count new_case_rate case_rate
## 1             NA         NA                NA            NA        NA
## 2             NA         NA                NA            NA        NA
## 3             NA         NA                NA            NA        NA
## 4             NA         NA                NA            NA        NA
## 5             NA         NA                NA            NA        NA
## 6             NA         NA                 0            NA        NA
##   new_death_rate death_rate new_test_rate test_rate new_vaccine_rate
## 1             NA         NA            NA        NA               NA
## 2             NA         NA            NA        NA               NA
## 3             NA         NA            NA        NA               NA
## 4             NA         NA            NA        NA               NA
## 5             NA         NA            NA        NA               NA
## 6             NA         NA            NA        NA               NA
##   vaccine_rate new_fullvaccine_rate fullvaccine_rate new_booster_first_rate
## 1           NA                   NA                NA                    NA
## 2           NA                   NA                NA                    NA
## 3           NA                   NA                NA                    NA
## 4           NA                   NA                NA                    NA
## 5           NA                   NA                NA                    NA
## 6           NA                   NA                NA                    NA
##   booster_first_rate hospitalized_rate year.y month.y day.y      emp emp_incq1
## 1                 NA                NA   2020       2    24  0.01580   0.00751
## 2                 NA                NA   2020       2    24  0.00537  -0.02670
## 3                 NA                NA   2020       2    24       NA        NA
## 4                 NA                NA   2020       2    24  0.00448  -0.00263
## 5                 NA                NA   2020       2    24  0.00532  -0.00537
## 6                 NA                 0   2020       2    24 -0.03530  -0.07190
##   emp_incq2 emp_incq3 emp_incq4 emp_incmiddle emp_incbelowmed emp_incabovemed
## 1   0.02320   0.01680        NA       0.01960        0.013600          0.0183
## 2   0.00570   0.01680    0.0242       0.01170       -0.011400          0.0206
## 3        NA        NA        NA            NA              NA              NA
## 4  -0.00458   0.01070    0.0164       0.00324       -0.003550          0.0133
## 5   0.00520   0.00873    0.0140       0.00710       -0.000838          0.0114
## 6  -0.04920  -0.00520        NA      -0.02980       -0.058300         -0.0112
```

```
##    emp_ss40 emp_ss60 emp_ss65 emp_ss70 year.x month.x day.x
## 1  0.001540 -0.00399  0.05300 -0.01620   2020       2    24
## 2  0.015400  0.01340  0.01030 -0.05550   2020       2    24
## 3        NA       NA       NA       NA   2020       2    24
## 4 -0.002320  0.00134  0.00576  0.01620   2020       2    24
## 5 -0.000237  0.00168  0.00889  0.00964   2020       2    24
## 6  0.054800       NA       NA -0.01530   2020       2    24
##   gps_retail_and_recreation gps_grocery_and_pharmacy gps_parks
## 1                   0.00286                -0.00714    0.0557
## 2                   0.03710                 0.01290    0.2340
## 3                  -0.01140                -0.03290    0.1400
## 4                   0.02710                 0.00714    0.0943
## 5                  -0.00571                -0.02290    0.0186
## 6                   0.01140                -0.00571    0.0814
##   gps_transit_stations gps_workplaces gps_residential gps_away_from_home year.y
## 1              0.06000        0.01290         0.00857           -0.00798   2020
## 2              0.07000        0.02860        -0.00571            0.00850   2020
## 3              0.00571       -0.01430         0.00714           -0.00492   2020
## 4              0.03430        0.01000         0.00143           -0.00138   2020
## 5              0.01710       -0.01140         0.01000           -0.00781   2020
## 6              0.02570        0.00714         0.00143           -0.00049   2020
##   month.y day.y freq spend_all spend_aap spend_acf spend_aer spend_apg
## 1       2    24    d   -0.0198   -0.1320   -0.0220   -0.1000   -0.0810
## 2       2    24    d   -0.0461    0.1130   -0.0279   -0.6280    0.4140
## 3       2    24    d    0.0192   -0.1280   -0.0113    0.0740   -0.0855
## 4       2    24    d   -0.0452   -0.0847   -0.0493   -0.1020   -0.0675
## 5       2    24    d   -0.0163   -0.0321   -0.0334    0.0287   -0.0308
## 6       2    24    d   -0.0504   -0.1210   -0.0447   -0.1650   -0.0851
##   spend_durables spend_nondurables spend_grf spend_gen spend_hic spend_hcs
## 1        -0.0317          -0.04750   -0.0223  -0.01050  -0.06180  -0.07310
## 2         0.0208           0.13400   -0.0284   0.63600   0.13400  -0.01060
## 3         0.0311          -0.00364    0.0294   0.00856   0.59500   0.02630
## 4        -0.0492          -0.04720   -0.0468  -0.03810  -0.08320   0.00175
## 5        -0.0164          -0.02450   -0.0110  -0.03000  -0.00361  -0.02010
## 6        -0.0118          -0.04380   -0.0173  -0.04770   0.16600  -0.08730
##   spend_inpersonmisc spend_remoteservices spend_sgh spend_tws
## 1             0.0062              0.02110   -0.0453   -0.1020
## 2            -0.1380             -0.15500   -0.1540   -0.0929
## 3             0.2100             -0.03610   -0.1230   -0.1360
## 4            -0.0815             -0.04600   -0.0426   -0.1030
## 5            -0.0658             -0.00774    0.0940   -0.1060
## 6            -0.0645             -0.04000   -0.2270   -0.0909
##   spend_retail_w_grocery spend_retail_no_grocery spend_all_incmiddle
## 1               -0.03910                -0.0459           -0.02970
## 2                0.10200                 0.1560           -0.06480
## 3               -0.00169                -0.0124           -0.06430
## 4               -0.04390                -0.0421           -0.03880
## 5               -0.01640                -0.0176           -0.01870
## 6               -0.03610                -0.0498            0.00268
##   spend_all_q1 spend_all_q2 spend_all_q3 spend_all_q4 provisional
## 1      -0.0158      -0.0717     0.036100     0.009840           0
## 2       0.2240      -0.0565    -0.068700    -0.016000           0
## 3      -0.0265      -0.5850    -0.047300     0.039400           0
## 4      -0.0677      -0.0420    -0.035100    -0.035700           0
```

```
## 5      -0.0386      -0.0234    -0.015600    -0.000937           0
## 6           NA       0.0134     0.000257    -0.076700           0
```

```
full_data1 <- full_data %>%
  select(-year.x, -month.x, -day.x, - year.y, -month.y, -day.y, -year.x )

regions <- regions%>%
  inner_join(fips, by=c("State.Code"="abb"))

# Created dataset with the fips code
full_cut <- full_data1 %>%
  filter(Date > "2020-04-13")%>%
  select(statefips, Date, gps_away_from_home, case_rate, hospitalized_rate, spend_remoteservices, spend
  left_join(regions, by=c("statefips"="fips"))

# Final Data Set
full_cut <- full_cut %>%
  select(statefips, Date, gps_away_from_home, case_rate, hospitalized_rate, spend_remoteservices, spend

# Splitting the Full cut so it has regions for CV
random_forest_data <- full_cut %>% na.omit()
random_forest_data <- random_forest_data %>%
  mutate(Region = factor(Region)) %>%
  mutate(across(where(is.character), as.factor))
random_forest_data <- random_forest_data %>% select(gps_away_from_home, case_rate, hospitalized_rate, s
```

Bagging

```
# Re-sampleing
spade_rec <- recipe(Region ~ ., data = random_forest_data) %>%
  step_nzv(all_predictors()) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors())

# Find out the Total Number of Predictors
(spade_rec %>% prep(random_forest_data) %>% juice() %>% ncol()) - 1
```

```
## [1] 6
```

```
# Bagging Model Spec
bag_spec <- rand_forest() %>%
  set_engine(engine = 'ranger') %>%
  set_args(mtry = 5,
           trees = 500,
           min_n = 20,
           probability = FALSE) %>%
  set_mode('classification')

region_bag_wf <- workflow() %>%
  add_model(bag_spec) %>%
  add_recipe(spade_rec)
```

Fit Models

```
set.seed(123) # Randomness in the bootstrap samples

region_bag_fit <- region_bag_wf %>%
  fit(data = random_forest_data)

region_bag_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
## * step_nzv()
## * step_novel()
## * step_dummy()
##
## -- Model -----------------------------------------------------------------------
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~5,      x), num.trees = ~500, min.n
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      29518
## Number of independent variables:  6
## Mtry:                             5
## Target node size:                 20
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error:             3.48 %
```

Evaluate Models

To calculate OOB metrics, we need to get the OOB predictions from the fit model.

```
region_bag_OOB_output <- tibble(
  .pred_class = region_bag_fit %>% extract_fit_engine() %>% pluck('predictions'),
  Region = random_forest_data %>% pull(Region))

bag_metrics <- metric_set(sens, yardstick::spec, accuracy)

region_bag_OOB_output %>%
  bag_metrics(truth = Region, estimate = .pred_class)
```

```
## # A tibble: 3 x 3
##   .metric  .estimator .estimate
```

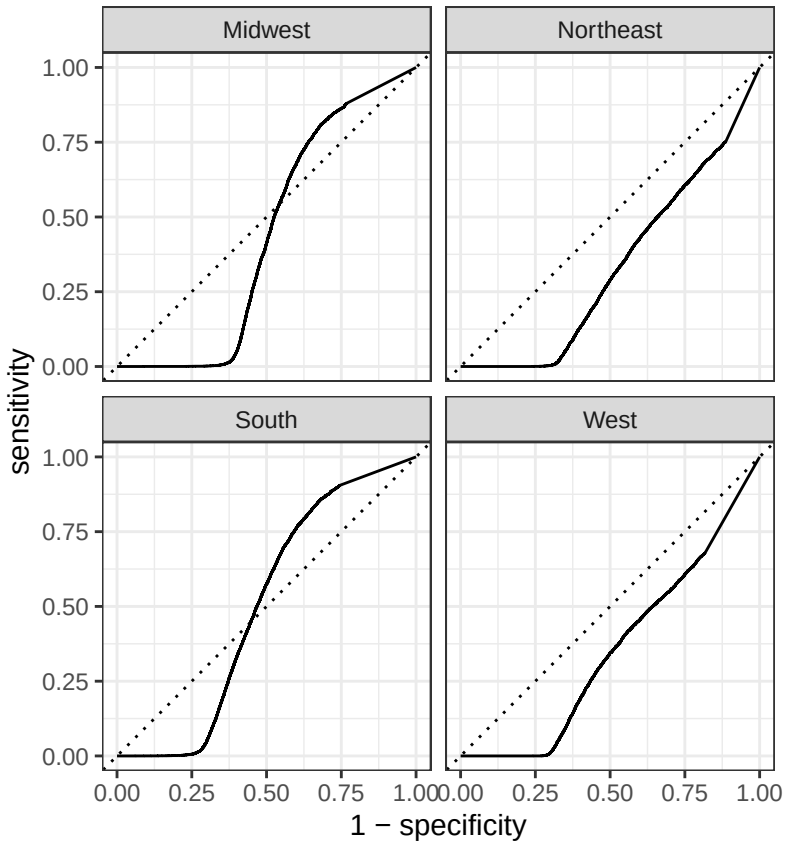
```
##   <chr>    <chr>          <dbl>
## 1 sens     macro          0.962
## 2 spec     macro          0.988
## 3 accuracy multiclass     0.965
```

To estimate AUC of ROC curve based on OOB predictions, we'll need to refit the model to get the predicted probabilities.

```
set.seed(123) # to get the same bootstrap samples, use same seed

region_bag_fit2 <- region_bag_wf %>%
  update_model(bag_spec %>% set_args(probability = TRUE)) %>% # Now, we want soft (probability) predict
  fit(data = random_forest_data)

region_bag_fit2
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
## * step_nzv()
## * step_novel()
## * step_dummy()
##
## -- Model -----------------------------------------------------------------------
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~5,      x), num.trees = ~500, min.n
##
## Type:                             Probability estimation
## Number of trees:                  500
## Sample size:                      29518
## Number of independent variables:  6
## Mtry:                             5
## Target node size:                 20
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.0448471
```

```
region_bag_OOB_output2 <- bind_cols(
  region_bag_fit2 %>% extract_fit_engine() %>% pluck('predictions') %>% as_tibble(),
  random_forest_data %>% select(Region))

region_bag_OOB_output2 %>%
  roc_curve(Region, c(South, West, Midwest, Northeast), event_level = "second") %>% autoplot()
```




```
region_bag_OOB_output2 %>%
  roc_auc(Region, c(South, West, Midwest, Northeast), event_level = "second") #Area under Curve
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.404
```

Random Forest

Model Specification

```
rf_spec <- rand_forest() %>%
  set_engine(engine = 'ranger') %>%
  set_args(mtry = NULL, # size of random subset of variables; default is floor(sqrt(ncol(x)))
           trees = 500, # Number of bags
           min_n = 20,
           probability = FALSE, # want hard predictions first
           importance = 'impurity') %>%
  set_mode('classification') # change this for regression tree

rf_spec
```

```
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   trees = 500
##   min_n = 20
##
## Engine-Specific Arguments:
##   probability = FALSE
##   importance = impurity
##
## Computational engine: ranger
```

```
region_rf_wf <- workflow() %>%
  add_model(rf_spec) %>%
  add_recipe(spade_rec)
```

Fit Models

```
set.seed(123)
region_rf_fit <- region_rf_wf %>%
  fit(data = random_forest_data)

region_rf_fit # check out OOB prediction error (accuracy = 1 - OOB prediction error)
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
## * step_nzv()
## * step_novel()
## * step_dummy()
##
## -- Model -----------------------------------------------------------------------
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, num.trees = ~500,      min.node.size = min_rows(~20,
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      29518
## Number of independent variables:  6
## Mtry:                             2
## Target node size:                 20
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error:             4.02 %
```

Evaluate Models

To calculate OOB metrics, we need to get the OOB predictions from the fit model.

```
#region_rf_OOB_output <- tibble(
#  .pred_class = region_rf_fit %>% extract_fit_engine() %>% pluck('predictions'),
#  Region = random_forest_data %>% pull(Region))

#bag_metrics <- metric_set(sens, yardstick::spec, accuracy)


region_rf_OOB_output <- function(fit_model, model_label, truth){
    tibble(
          .pred_class = region_rf_fit %>% extract_fit_engine() %>% pluck('predictions'), #OOB predictio
          Region = truth,
          label = model_label
      )
}

#check out the function output
region_rf_OOB_output(region_rf_fit2,NULL, random_forest_data %>% pull(Region))
```

```
## # A tibble: 29,518 x 2
##    .pred_class Region
##    <fct>       <fct>
##  1 West        South
##  2 South       South
##  3 South       South
##  4 West        South
##  5 West        West
##  6 West        Midwest
##  7 Midwest     Midwest
##  8 West        Midwest
##  9 West        Midwest
## 10 South       South
## # ... with 29,508 more rows
```

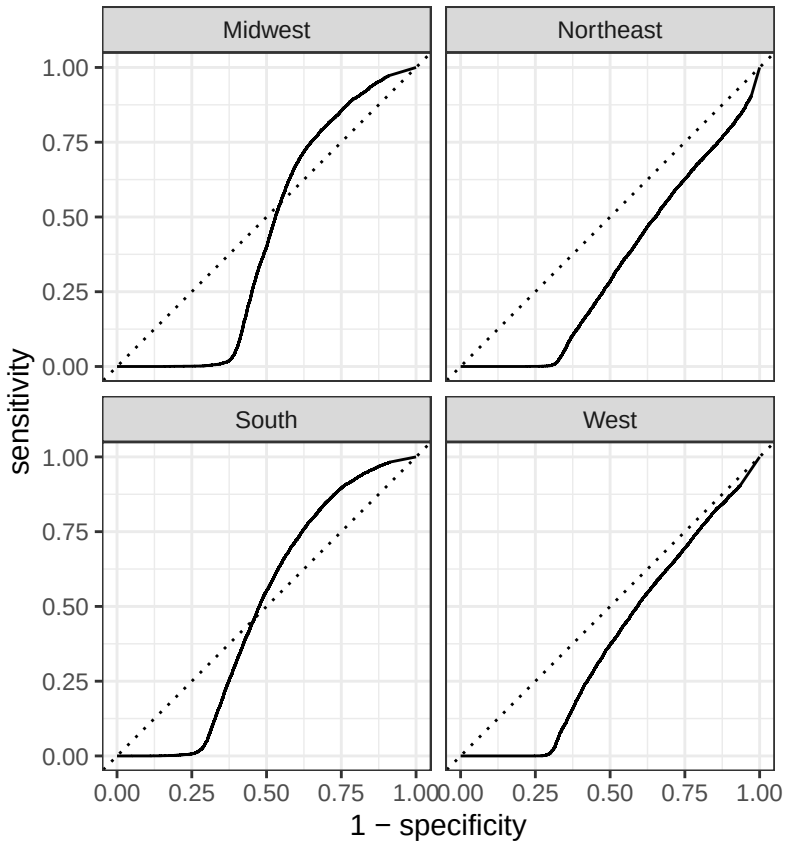
To estimate AUC of ROC curve using OOB predictions, we'll need to refit the model to get the predicted probabilities.

```
set.seed(123) #to get the same bootstrap samples, use same seed
region_rf_fit2 <- region_rf_wf %>%
  update_model(rf_spec %>% set_args(probability = TRUE)) %>%
  fit(data = random_forest_data)

region_rf_fit2
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
```

```
##
## * step_nzv()
## * step_novel()
## * step_dummy()
##
## -- Model ---------------------------------------------------------------------
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, num.trees = ~500,      min.node.size = min_rows(~20,
##
## Type:                             Probability estimation
## Number of trees:                  500
## Sample size:                      29518
## Number of independent variables:  6
## Mtry:                             2
## Target node size:                 20
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.05958307
```

```
region_rf_OOB_output2 <- bind_cols(
  region_rf_fit2 %>% extract_fit_engine() %>% pluck('predictions') %>% as_tibble(),
  random_forest_data%>% select(Region))

region_rf_OOB_output3 <- bind_cols(
  .pred_class = region_rf_fit2 %>% extract_fit_engine() %>% pluck('predictions'),
  Region = random_forest_data %>% pull(Region))

region_rf_OOB_output2 %>%
  roc_curve(Region, c(South, West, Midwest, Northeast), event_level = "second") %>% autoplot()
```

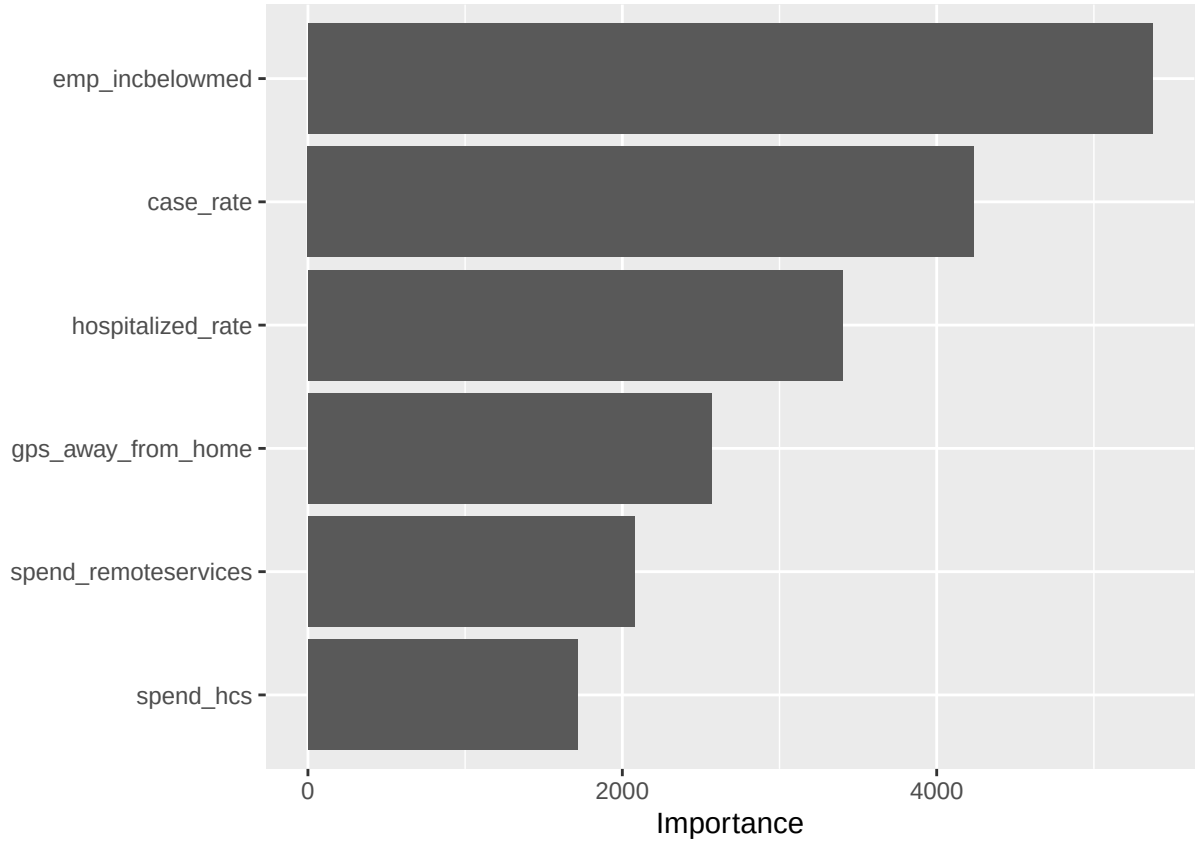



```
region_rf_OOB_output2 %>%
  roc_auc(Region, c(South, West, Midwest, Northeast), event_level = "second") #Area under Curve
```

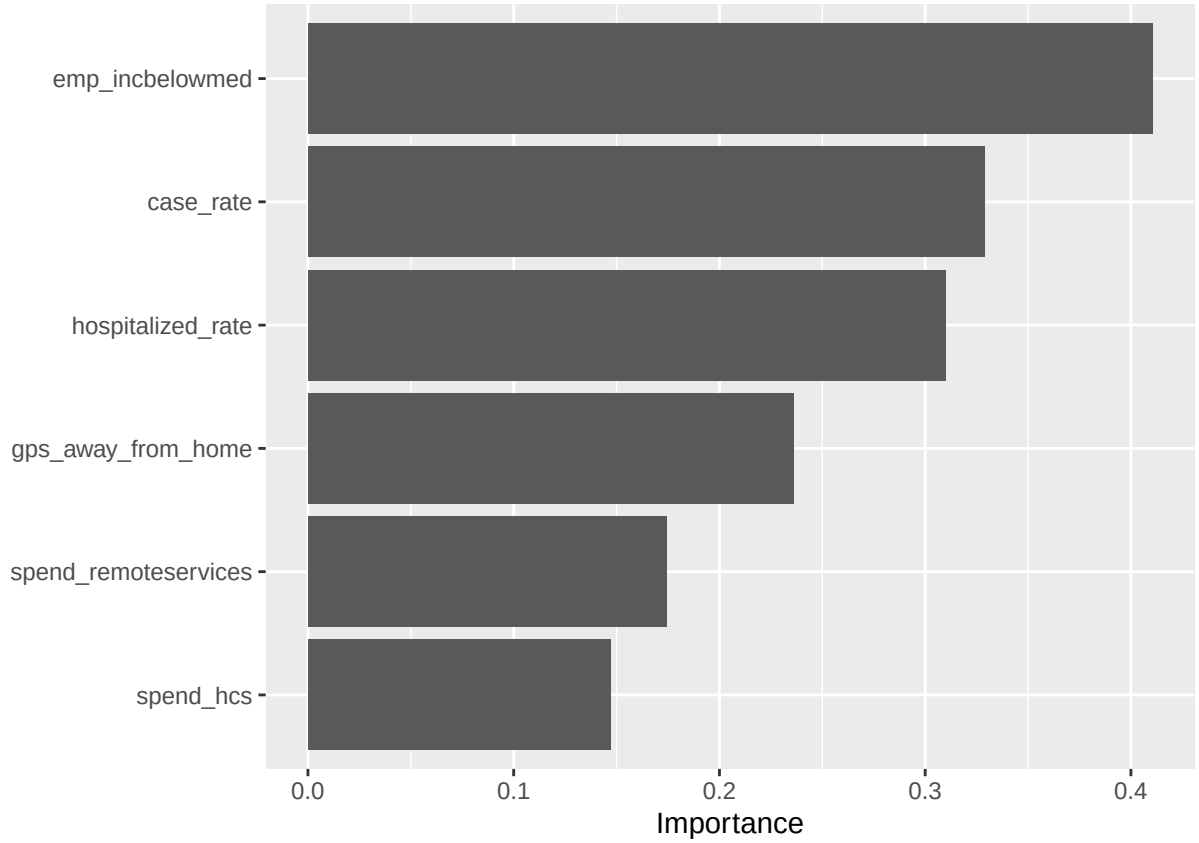
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.411
```

Variable Importance

```
region_rf_fit %>% extract_fit_engine() %>% vip() #based on impurity
```




```
region_rf_wf %>% #based on permutation
  update_model(rf_spec %>% set_args(importance = "permutation")) %>%
  fit(data = random_forest_data) %>% extract_fit_engine() %>% vip()
```




```
#rf_OOB_output(data_fit_mtry12,12, land %>% pull(class)) %>%
#    conf_mat(truth = class, estimate= .pred_class)

 region_rf_OOB_output(region_rf_fit2, NULL, random_forest_data %>% pull(Region)) %>%
   conf_mat(truth = Region, estimate = .pred_class)
```

```
##            Truth
## Prediction  Midwest Northeast South West
##   Midwest      5266        36   123   48
##   Northeast      56      6563    26   44
##   South         427       115  9471  131
##   West           59        62    60 7031
```

```
# log_metrics <- metric_set(sens, yardstick::spec, accuracy) # these metrics are based on hard predicti

#sens: sensitivity = chance of correctly predicting second level, given second level (Yes)
#spec: specificity = chance of correctly predicting first level, given first level (No)
#accuracy: accuracy = chance of correctly predicting outcome

# region_rf_OOB_output3 %>%
#   log_metrics(estimate = .pred_class, truth = c(South, West, Midwest, Northeast), event_level = "seco
```

Logistic Regression

To build logistic regression models in `tidymodels`, first load the package and set the seed for the random number generator to ensure reproducible results:

```
policy <- read.csv("policy.csv", na.strings = "")
set.seed(253)
```

```
policy$date_restrictions_start<-as.Date(policy$date_restrictions_start,"%Y-%m-%d")
policy$date_restrictions_end<-as.Date(policy$date_restrictions_end,"%Y-%m-%d")

policy_cut <- policy %>%
  select(statename, statefips, all_restrictions, date_restrictions_start, date_restrictions_end)%>%
  filter(statename != "")

full_cut2 <- full_cut%>%
  mutate(isSouth = if_else(Region == "South", 1, 0))%>%
  inner_join(policy_cut, by=c("statefips"="statefips"))

# Creating a Dummy Variable that is yes for each day in a state where both nonessential buisness closer
full_cut2 <- mutate(full_cut2, day_with_allCloser = if_else(Date >= date_restrictions_start & Date <= da
  filter(Date <= as.Date("2020-06-09","%Y-%m-%d"))
```

```
# Log Model and data Cutting
full_cut2$isSouth <- as.factor(full_cut2$isSouth)
full_cut2$day_with_allCloser <- as.factor(full_cut2$day_with_allCloser)

full_cut_sub3 <- full_cut2 %>%
  select(gps_away_from_home, case_rate, hospitalized_rate, spend_remoteservices,spend_hcs, emp_incbelow

data_cv10 <- vfold_cv(full_cut_sub3, v = 10)

# Logistic Regression Model Spec
logistic_spec <- logistic_reg() %>%
    set_engine('glm') %>%
    set_mode('classification')

# Recipe
logistic_rec <- recipe(day_with_allCloser ~ ., data = full_cut_sub3) %>%
    step_normalize(all_numeric_predictors()) %>%
    step_dummy(all_nominal_predictors())

log_wf <- workflow() %>%
    add_recipe(logistic_rec) %>%
    add_model(logistic_spec)

# Fit Model to Training Data
log_fit <- fit(log_wf, data = full_cut_sub3)
```

Examining the logistic model

```
# Print out Coefficients
log_fit %>% tidy()
```

```
## # A tibble: 7 x 5
##   term                 estimate std.error statistic  p.value
##   <chr>                   <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)            0.985     0.0824    12.0   6.09e-33
## 2 gps_away_from_home    -4.64      0.245    -19.0   3.42e-80
## 3 case_rate             -1.11      0.0918   -12.1   1.63e-33
## 4 hospitalized_rate      0.274     0.149      1.84  6.60e- 2
## 5 spend_remoteservices  -0.0313    0.0964    -0.325 7.45e- 1
## 6 spend_hcs             -1.12      0.108    -10.4   2.15e-25
## 7 emp_incbelowmed        1.58      0.167      9.45  3.53e-21
```

```
# Get Exponentiated coefficients + CI
log_fit %>% tidy() %>%
  mutate(OR.conf.low = exp(estimate - 1.96*std.error), OR.conf.high = exp(estimate + 1.96*std.error)) %
  mutate(OR = exp(estimate))
```

```
## # A tibble: 7 x 8
##   term    estimate std.error statistic  p.value OR.conf.low OR.conf.high      OR
##   <chr>      <dbl>     <dbl>     <dbl>    <dbl>       <dbl>        <dbl>   <dbl>
## 1 (Inter~   0.985     0.0824    12.0   6.09e-33     2.28          3.15   2.68
## 2 gps_aw~  -4.64      0.245    -19.0   3.42e-80     0.00596       0.0156 0.00963
## 3 case_r~  -1.11      0.0918   -12.1   1.63e-33     0.276         0.396  0.330
## 4 hospit~   0.274     0.149      1.84  6.60e- 2     0.982         1.76   1.32
## 5 spend_~  -0.0313    0.0964    -0.325 7.45e- 1     0.802         1.17   0.969
## 6 spend_~  -1.12      0.108    -10.4   2.15e-25     0.263         0.402  0.325
## 7 emp_in~   1.58      0.167      9.45  3.53e-21     3.49          6.72   4.84
```

Making predictions from the logistic model

```
# Make soft (probability) predictions
predict(log_fit, new_data = full_cut_sub3, type = "prob")
```

```
## # A tibble: 3,705 x 2
##       .pred_0 .pred_1
##         <dbl>   <dbl>
##  1  0.00528     0.995
##  2  0.000193    1.00
##  3 NA          NA
##  4  0.000407    1.00
##  5  0.000722    0.999
##  6  0.0000576   1.00
##  7  0.0527      0.947
##  8  0.000240    1.00
##  9  0.00188     0.998
## 10  0.00424     0.996
## # ... with 3,695 more rows
```

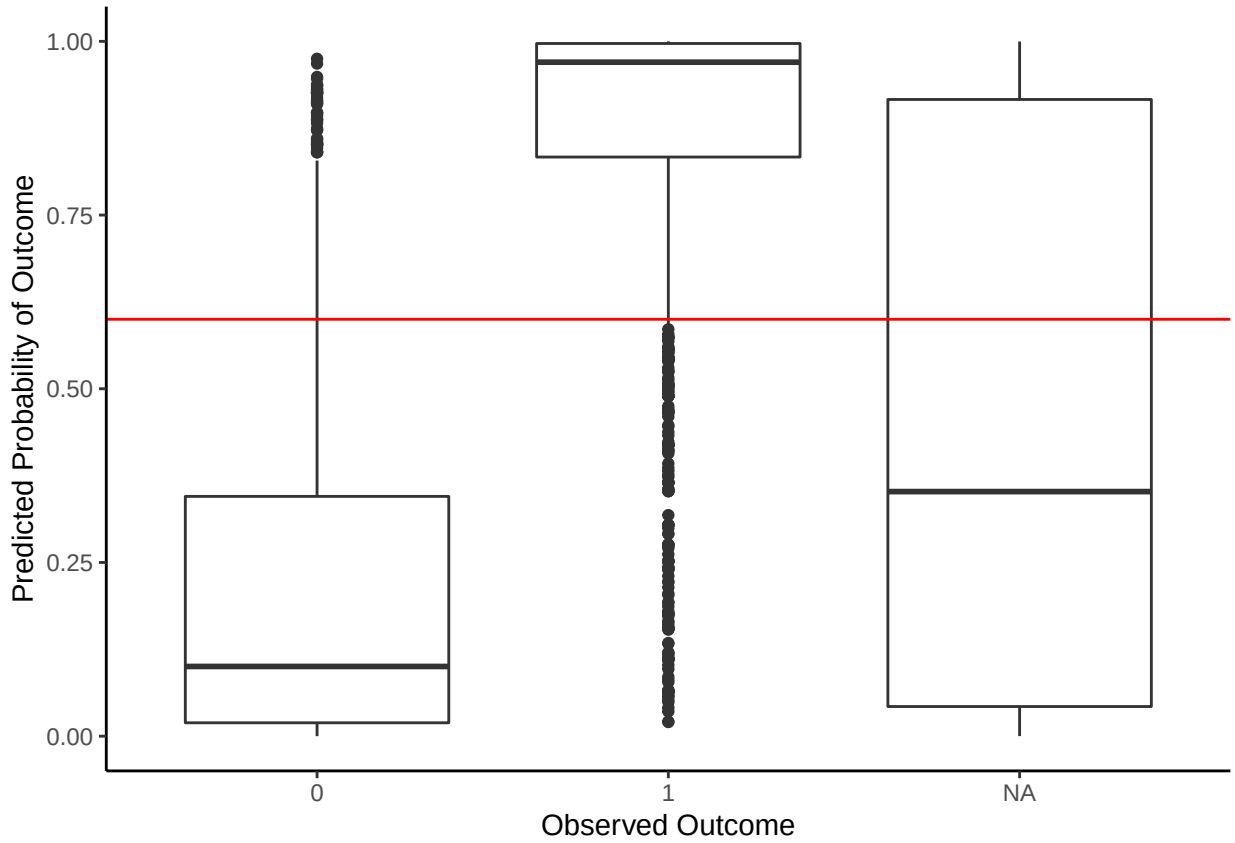
```r
# Make hard (class) predictions (using a default 0.5 probability threshold)
predict(log_fit, new_data = full_cut_sub3, type = "class")
```

```
## # A tibble: 3,705 x 1
##    .pred_class
##    <fct>
##  1 1
##  2 1
##  3 <NA>
##  4 1
##  5 1
##  6 1
##  7 1
##  8 1
##  9 1
## 10 1
## # ... with 3,695 more rows
```

Evaluating the logistic model on training data

```r
# Soft predictions
logistic_output <-  full_cut_sub3 %>%
  bind_cols(predict(log_fit, new_data = full_cut_sub3, type = 'prob'))

# Hard predictions (you pick threshold)
logistic_output <- logistic_output %>%
  mutate(.pred_class = make_two_class_pred(.pred_0, levels(day_with_allCloser), threshold = .6))

# Visualize Soft Predictions
logistic_output %>%
  ggplot(aes(x = day_with_allCloser, y = .pred_1)) +
  geom_boxplot() +
  geom_hline(yintercept = 0.6, color='red') +
  labs(y = 'Predicted Probability of Outcome', x = 'Observed Outcome') +
  theme_classic()
```

```
## Warning: Removed 114 rows containing non-finite values (stat_boxplot).
```



Calculate evaluation metrics of the logistic model on training data

```
# Confusion Matrix
logistic_output %>%
  conf_mat(truth = day_with_allCloser, estimate = .pred_class)
```

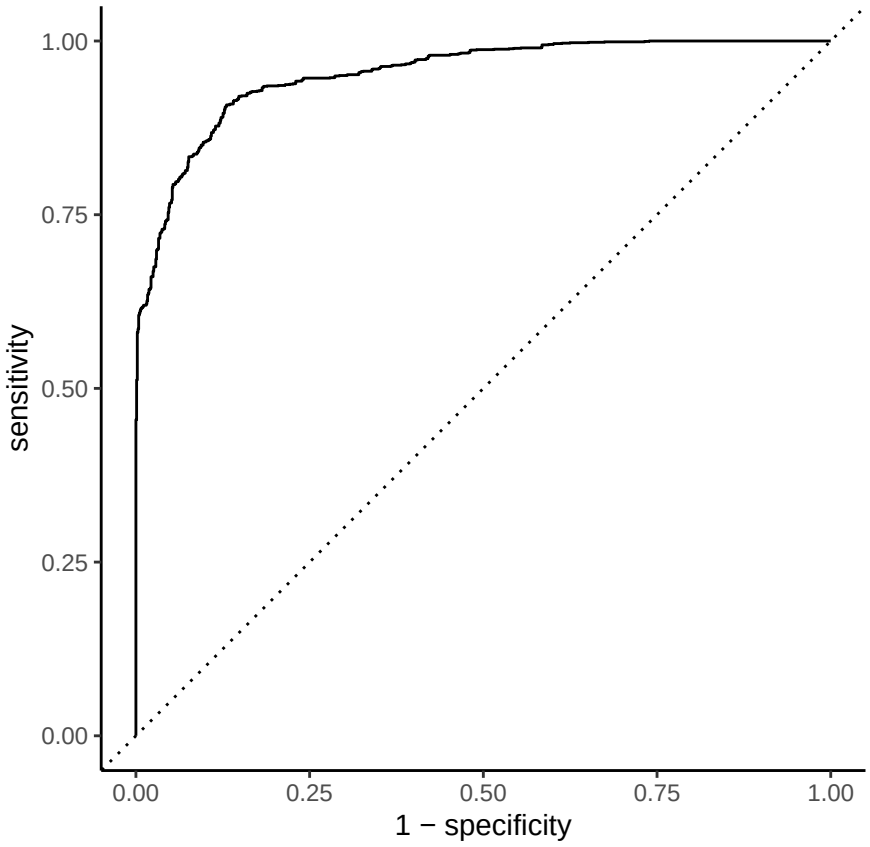
```
##           Truth
## Prediction    0    1
##          0  819  104
##          1  194 1505
```

```
log_metrics <- metric_set(sens, yardstick::spec, accuracy) # these metrics are based on hard prediction

#sens: sensitivity = chance of correctly predicting second level, given second level (Yes)
#spec: specificity = chance of correctly predicting first level, given first level (No)
#accuracy: accuracy = chance of correctly predicting outcome

logistic_output %>%
  log_metrics(estimate = .pred_class, truth = day_with_allCloser, event_level = "second") # set second
```

```
## # A tibble: 3 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 sens     binary         0.935
## 2 spec     binary         0.808
## 3 accuracy binary         0.886
```

ROC Curve: evaluating logistic model using soft predictions

```
logistic_roc <- logistic_output %>%
    roc_curve(day_with_allCloser, .pred_1, event_level = "second")

autoplot(logistic_roc) + theme_classic()
```



Calculate evaluation metrics of the logistic model using CV

```
log_cv_fit <- fit_resamples(
    log_wf,
    resamples = data_cv10,
    metrics = metric_set(sens, yardstick::spec, accuracy, roc_auc),
    control = control_resamples(save_pred = TRUE, event_level = 'second'))
```

```
## Warning: package 'rlang' was built under R version 4.1.2
```

```
collect_metrics(log_cv_fit) #default threshold is 0.5
```

```
## # A tibble: 4 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.891    10 0.00610 Preprocessor1_Model1
```

```
## 2 roc_auc  binary     0.951    10 0.00445 Preprocessor1_Model1
## 3 sens     binary     0.912    10 0.00290 Preprocessor1_Model1
## 4 spec     binary     0.856    10 0.0150  Preprocessor1_Model1
```