Lecture 12 Notes

2024-07-23

```
require(tidyverse)
require(tidymodels)
fn <- read_rds("https://github.com/jbisbee1/ISP_Data_Science_2024/raw/main/data/fn_clean
ed_final.rds")
```

Step 1: Calculate probability of Y

```
require(ranger)
m_rf <- ranger(formula = won ~ damage_to_players,
               data = fn)

rf_preds <- predict(m_rf,data = fn)
rf_preds
```

```
## Ranger prediction
##
## Type:                             Regression
## Sample size:                      957
## Number of independent variables:  1
```

```
forAUC <- fn %>%
  mutate(prob_win_rf = rf_preds$predictions) %>%
  select(won,prob_win_rf)

forAUC <- forAUC %>%
  mutate(won = factor(won,levels = c('1','0')))

roc_auc(forAUC,won,prob_win_rf)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.953
```

Cross Validation for RF

```
set.seed(123)
cvRes <- NULL
for(i in 1:100) {
  # Step 1: Split the data (use 60-40)
  train <- fn %>%
    select(won,damage_to_players) %>%
    drop_na() %>%
    sample_n(size = round(nrow(.)*.6),
             replace = F)

  test <- fn %>%
    select(won,damage_to_players) %>%
    drop_na() %>%
    anti_join(train)

  # Step 2: train model on training data
  tmp_rf <- ranger(formula = won ~ damage_to_players,
                   data = train,
                   seed = 123)

  # Step 3: evaluate the model
  tmp_pred_rf <- predict(tmp_rf,data = test)

  forAUC <- test %>%
    mutate(prob_win_rf = tmp_pred_rf$predictions) %>%
    mutate(won = factor(won,levels = c('1','0')))

  answer <- roc_auc(forAUC,won,prob_win_rf) %>%
    rename(metric = .metric,
           type = .estimator,
           auc = .estimate) %>%
    mutate(cv_number = i)

  # Save the answer to our cvRes object
  cvRes <- cvRes %>%
    bind_rows(answer)
}
```

Summarize the evaluation

```
cvRes %>%
  summarise(avg_auc = mean(auc))
```

```
## # A tibble: 1 × 1
##   avg_auc
##     <dbl>
## 1   0.540
```

Comparing multiple models

```
set.seed(123)
cvRes <- NULL
for(i in 1:100) {
  # Step 1: Split the data (use 60-40)
  train <- fn %>%
    select(won,damage_to_players) %>%
    drop_na() %>%
    sample_n(size = round(nrow(.)*.6),
             replace = F)

  test <- fn %>%
    select(won,damage_to_players) %>%
    drop_na() %>%
    anti_join(train)

  # Step 2: train model on training data
  tmp_rf <- ranger(formula = won ~ damage_to_players,
                   data = train,
                   seed = 123)

  tmp_lm <- lm(formula = won ~ damage_to_players,
               data = train)

  tmp_glm <- glm(formula = won ~ damage_to_players,
                 data = train,
                 family = binomial(link = 'logit'))

  # Step 3: evaluate the model
  tmp_pred_rf <- predict(tmp_rf,data = test)

  forAUC <- test %>%
    mutate(prob_win_rf = tmp_pred_rf$predictions,
           prob_win_lm = predict(tmp_lm,
                                 newdata = test),
           prob_win_glm = predict(tmp_glm,
                                  newdata = test,
                                  type = 'response')) %>%
    mutate(won = factor(won,levels = c('1','0')))

  answer_rf <- roc_auc(forAUC,won,prob_win_rf) %>%
    rename(metric = .metric,
           type = .estimator,
           auc = .estimate) %>%
    mutate(cv_number = i,
           model = 'random forest')

  answer_lm <- roc_auc(forAUC,won,prob_win_lm) %>%
    rename(metric = .metric,
           type = .estimator,
           auc = .estimate) %>%
    mutate(cv_number = i,
           model = 'linear model')
```

```
  answer_glm <- roc_auc(forAUC,won,prob_win_glm) %>%
    rename(metric = .metric,
           type = .estimator,
           auc = .estimate) %>%
    mutate(cv_number = i,
           model = 'logit')

  # Save the answer to our cvRes object
  cvRes <- cvRes %>%
    bind_rows(answer_rf) %>%
    bind_rows(answer_lm) %>%
    bind_rows(answer_glm)
}
```

Look at the results

```
cvRes %>%
  group_by(model) %>%
  summarise(avg_auc = mean(auc))
```

```
## # A tibble: 3 × 2
##   model         avg_auc
##   <chr>           <dbl>
## 1 linear model    0.689
## 2 logit           0.689
## 3 random forest   0.540
```

Comparing specifications

- Specification 1: simplest, `won ~ damage_to_players`
- Specification 2: slightly more complicated: `won ~ damage_to_players + mental_state + accuracy`
- Specification 3: most complicated: `won ~ .` (everything that is not Y)

```
set.seed(123)
cvRes <- NULL
for(i in 1:100) {
  # Step 1: Split the data (use 60-40)
  train <- fn %>%
    drop_na() %>%
    sample_n(size = round(nrow(.)*.6),
             replace = F)

  test <- fn %>%
    drop_na() %>%
    anti_join(train)

  # Step 2: train model on training data
  tmp_rf1 <- ranger(formula = won ~ damage_to_players,
                    data = train,
                    seed = 123)

  tmp_rf2 <- ranger(formula = won ~ damage_to_players + mental_state + accuracy,
                    data = train,
                    seed = 123)

    tmp_rf3 <- ranger(formula = won ~ .,
                    data = train,
                    seed = 123)

  # Step 3: evaluate the model
  tmp_pred_rf1 <- predict(tmp_rf1,data = test)
  tmp_pred_rf2 <- predict(tmp_rf2,data = test)
  tmp_pred_rf3 <- predict(tmp_rf3,data = test)

  forAUC <- test %>%
    mutate(prob_win_rf1 = tmp_pred_rf1$predictions,
           prob_win_rf2 = tmp_pred_rf2$predictions,
           prob_win_rf3 = tmp_pred_rf3$predictions) %>%
    mutate(won = factor(won,levels = c('1','0')))

  answer_rf1 <- roc_auc(forAUC,won,prob_win_rf1) %>%
    rename(metric = .metric,
           type = .estimator,
           auc = .estimate) %>%
    mutate(cv_number = i,
           specification = 'simplest')

  answer_rf2 <- roc_auc(forAUC,won,prob_win_rf2) %>%
    rename(metric = .metric,
           type = .estimator,
           auc = .estimate) %>%
    mutate(cv_number = i,
           specification = 'three variables')

  answer_rf3 <- roc_auc(forAUC,won,prob_win_rf3) %>%
```

```
    rename(metric = .metric,
           type = .estimator,
           auc = .estimate) %>%
    mutate(cv_number = i,
           specification = 'kitchen sink')

  # Save the answer to our cvRes object
  cvRes <- cvRes %>%
    bind_rows(answer_rf1) %>%
    bind_rows(answer_rf2) %>%
    bind_rows(answer_rf3)
}
```

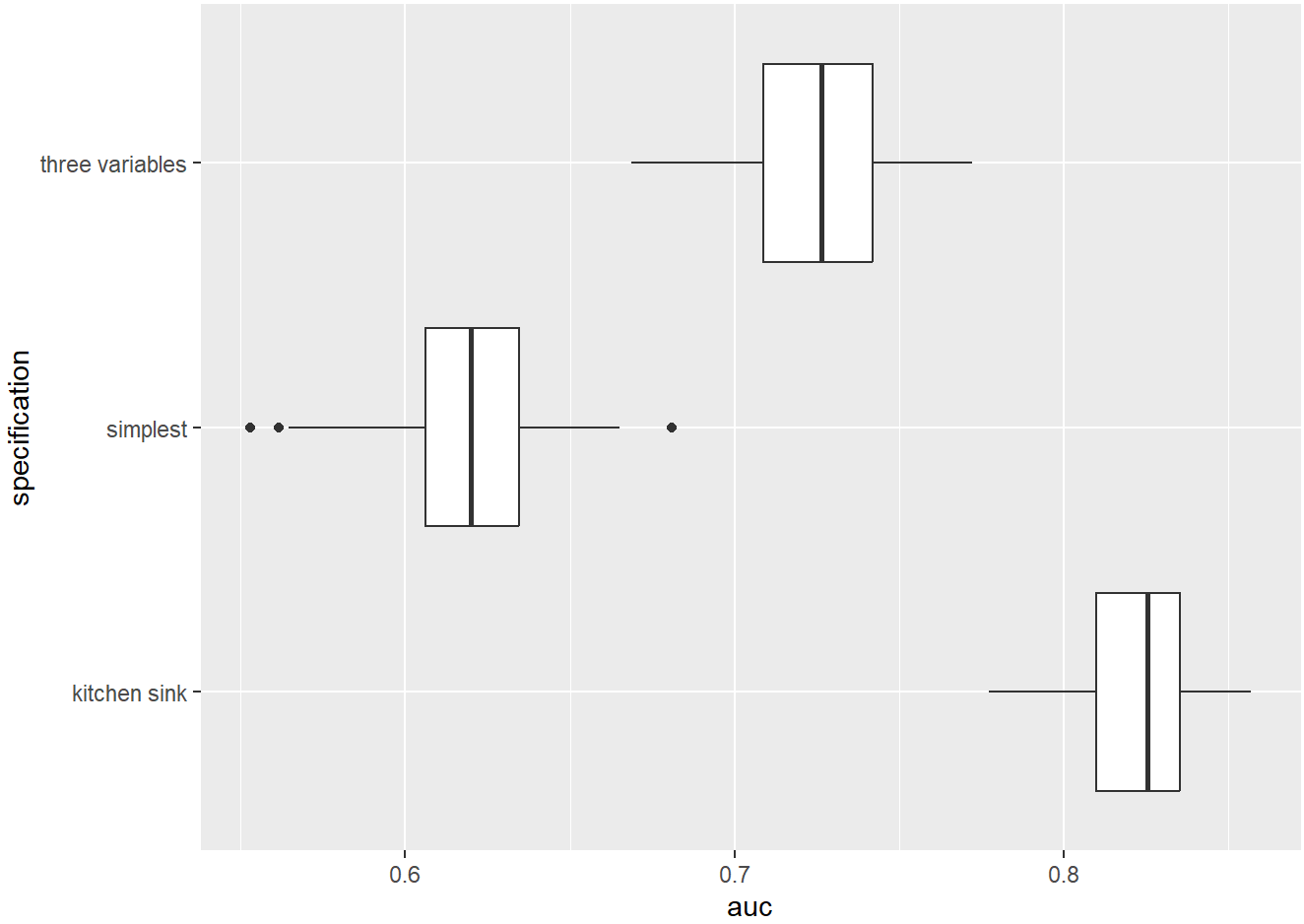
Looking at result

```
cvRes %>%
  group_by(specification) %>%
  summarise(avg_auc = mean(auc))
```

```
## # A tibble: 3 × 2
##   specification   avg_auc
##   <chr>             <dbl>
## 1 kitchen sink      0.823
## 2 simplest          0.620
## 3 three variables   0.725
```

Visualize the cross validation result

```
cvRes %>%
  ggplot(aes(x = auc,
             y = specification)) +
  geom_boxplot()
```



Advanced: nested for() loops

```
formulas <- c('won ~ damage_to_players',
              'won ~ damage_to_players + mental_state + accuracy',
              'won ~ .')

set.seed(123)
cvRes_master <- NULL
for(j in 1:3) {
  cvRes <- NULL
  for(i in 1:10) {
  # Step 1: Split the data (use 60-40)
  train <- fn %>%
    drop_na() %>%
    sample_n(size = round(nrow(.)*.6),
             replace = F)

  test <- fn %>%
    drop_na() %>%
    anti_join(train)

  # Step 2: train model on training data
  tmp_rf <- ranger(formula = as.formula(formulas[j]),
                   data = train,
                   seed = 123)

  tmp_lm <- lm(formula = as.formula(formulas[j]),
               data = train)

  tmp_glm <- glm(formula = as.formula(formulas[j]),
                 data = train,
                 family = binomial(link = 'logit'))

  # Step 3: evaluate the model
  tmp_pred_rf <- predict(tmp_rf,data = test)

  forAUC <- test %>%
    mutate(prob_win_rf = tmp_pred_rf$predictions,
           prob_win_lm = predict(tmp_lm,
                                 newdata = test),
           prob_win_glm = predict(tmp_glm,
                                  newdata = test,
                                  type = 'response')) %>%
    mutate(won = factor(won,levels = c('1','0')))

  answer_rf <- roc_auc(forAUC,won,prob_win_rf) %>%
    rename(metric = .metric,
           type = .estimator,
           auc = .estimate) %>%
    mutate(cv_number = i,
           model = 'random forest')

  answer_lm <- roc_auc(forAUC,won,prob_win_lm) %>%
    rename(metric = .metric,
```

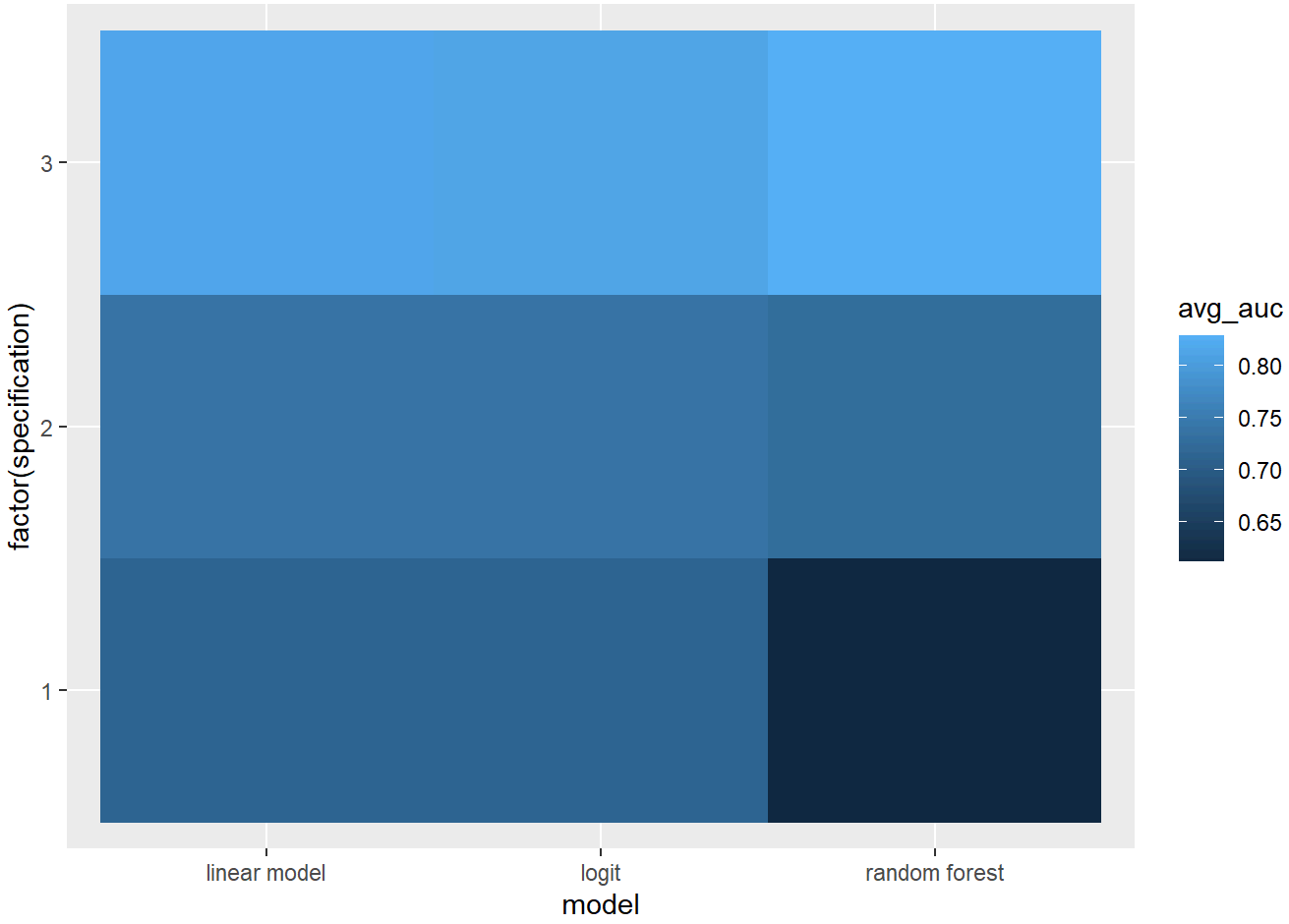
```
            type = .estimator,
            auc = .estimate) %>%
    mutate(cv_number = i,
            model = 'linear model')

  answer_glm <- roc_auc(forAUC,won,prob_win_glm) %>%
    rename(metric = .metric,
            type = .estimator,
            auc = .estimate) %>%
    mutate(cv_number = i,
            model = 'logit')

  # Save the answer to our cvRes object
  cvRes <- cvRes %>%
    bind_rows(answer_rf) %>%
    bind_rows(answer_lm) %>%
    bind_rows(answer_glm) %>%
    mutate(specification = j)
  }
  cvRes_master <- cvRes_master %>%
    bind_rows(cvRes)
}
```

Look at the result

```
cvRes_master %>%
  group_by(model,specification) %>%
  summarise(avg_auc = mean(auc)) %>%
  ggplot(aes(x = model,
             y = factor(specification),
             fill = avg_auc)) +
  geom_tile()
```



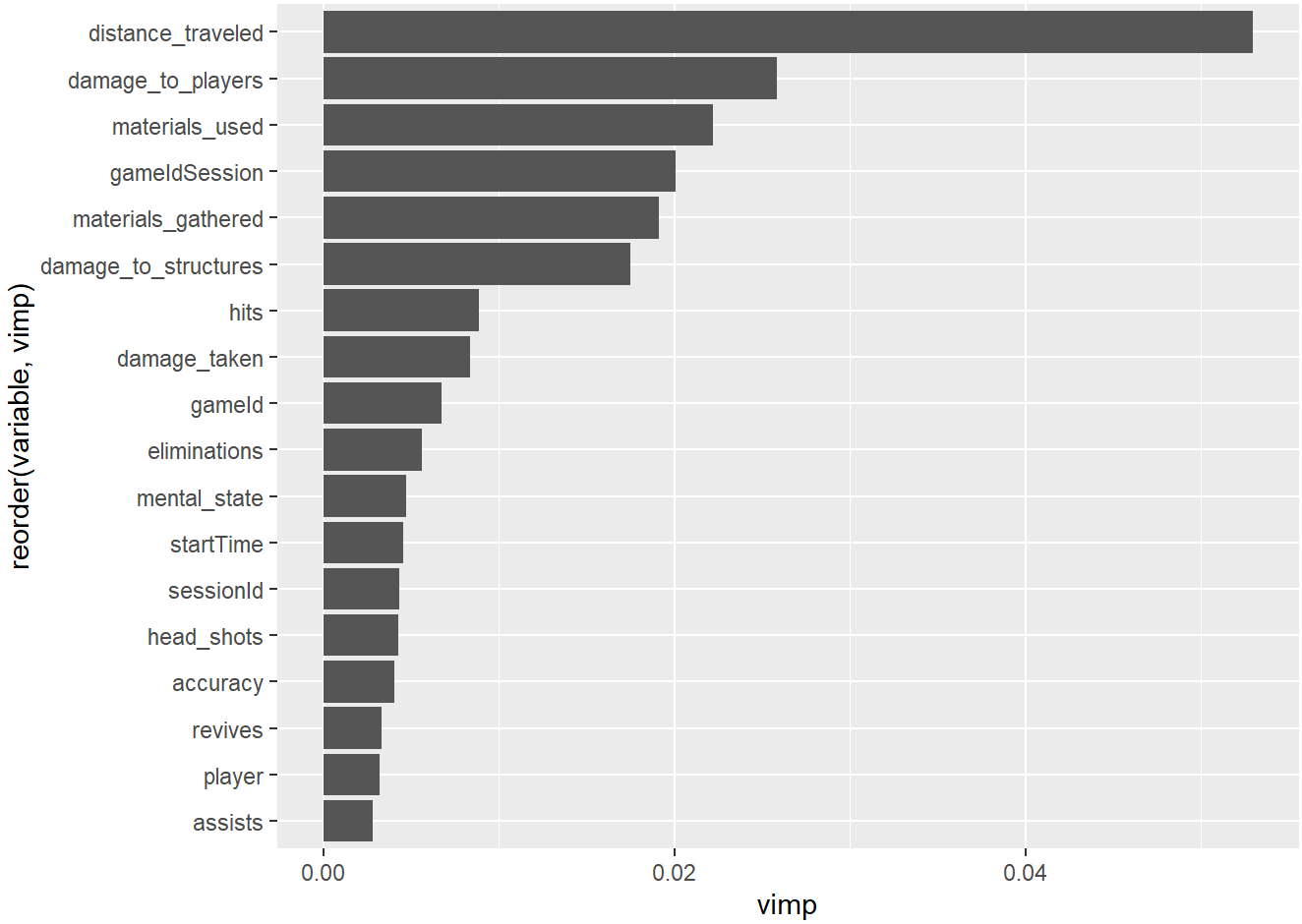
Variable importance

```
m_rf <- ranger(formula = as.formula(formulas[3]),
               data = fn,importance = "permutation")

vimpRaw <- m_rf$variable.importance

# Create a barplot
toplot <- data.frame(variable = names(vimpRaw),
                     vimp = as.numeric(vimpRaw))

toplot %>%
  ggplot(aes(x = vimp,
             y = reorder(variable,vimp))) +
  geom_bar(stat = 'identity')
```




```
beta <- .7

(exp(beta)-1)*100
```

```
## [1] 101.3753
```