target_creation

Juraj Szitás

11/1/2021

Load all data

```r
qs::qread("accuracies_1_2021-11-04_1.qs") -> accuracies1
qs::qread("accuracies_2_2021-11-05_1.qs") -> accuracies2
qs::qread("../data/M_all_features.qs") -> feature_df
```

```r
library(magrittr)
accuracies <- dplyr::bind_rows(accuracies1, accuracies2) %>%
  as.data.frame() %>%
  dplyr::select_if(~ !all(is.na(.x))) %>%
  dplyr::select_if(~ !all(.x == .x[1])) %>%
  tidyr::drop_na() %>%
  dplyr::filter( .model !=  "<environment>")
```

```r
accuracy_ranks <- accuracies %>%
  dplyr::mutate(dplyr::across(where(is.numeric), abs)) %>%
  dplyr::group_by( key ) %>%
  dplyr::mutate(dplyr::across(where(is.numeric), dplyr::min_rank)) %>%
  dplyr::ungroup() %>%
  dplyr::mutate( avg_rank = rowMeans( dplyr::across(where(is.numeric))) )
```

Winner takes all: (best method is taken as a label, and no probabilistic voodoo is done)

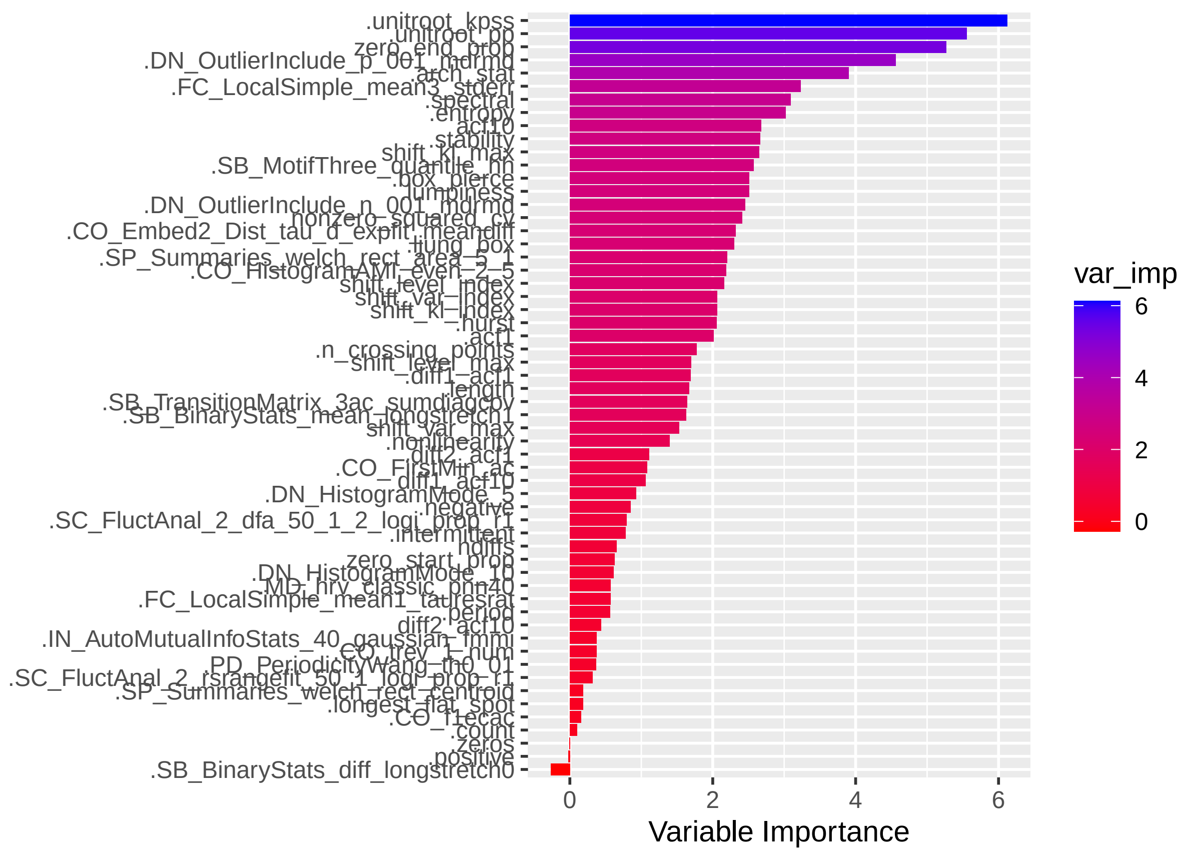
```r
winner_takes_all <- accuracy_ranks %>%
  dplyr::group_by(key) %>%
  dplyr::filter(avg_rank == min(avg_rank)) %>%
  dplyr::select( .model, key) %>%
  dplyr::ungroup()
```

Fit model on this data:

```r
soothsayer_data <- winner_takes_all %>%
  dplyr::full_join(feature_df, by = c("key" = "keys")) %>%
  dplyr::select_if(~ !all(is.na(.x))) %>%
  dplyr::select_if(~ !all(.x == .x[1])) %>%
  tidyr::drop_na() %>%
  dplyr::mutate(sample_type = sample(c("train", "test"),
    length(key),
    replace = TRUE,
    prob = c(0.7, 0.3)
  ))

soothsayer_train <- soothsayer_data %>%
  dplyr::filter(sample_type == "train") %>%
  dplyr::select(-sample_type) %>%
  dplyr::select(-key)
soothsayer_test <- soothsayer_data %>%
  dplyr::filter(sample_type == "test") %>%
  dplyr::select(-sample_type)

ranger_x <- soothsayer_train %>%
  dplyr::select(-.model) %>%
  as.matrix()
ranger_y <- soothsayer_train %>%
  dplyr::select(.model) %>%
  unlist() %>%
  as.factor()

soothsayer_model <- ranger::ranger(
  x = ranger_x,
  y = ranger_y,
  num.trees = 2000,
  probability = TRUE
)

soothsayer_model_importance <- ranger::ranger(
  x = ranger_x,
  y = ranger_y,
  num.trees = 2000,
  importance = 'impurity_corrected',
  probability = TRUE
)
```

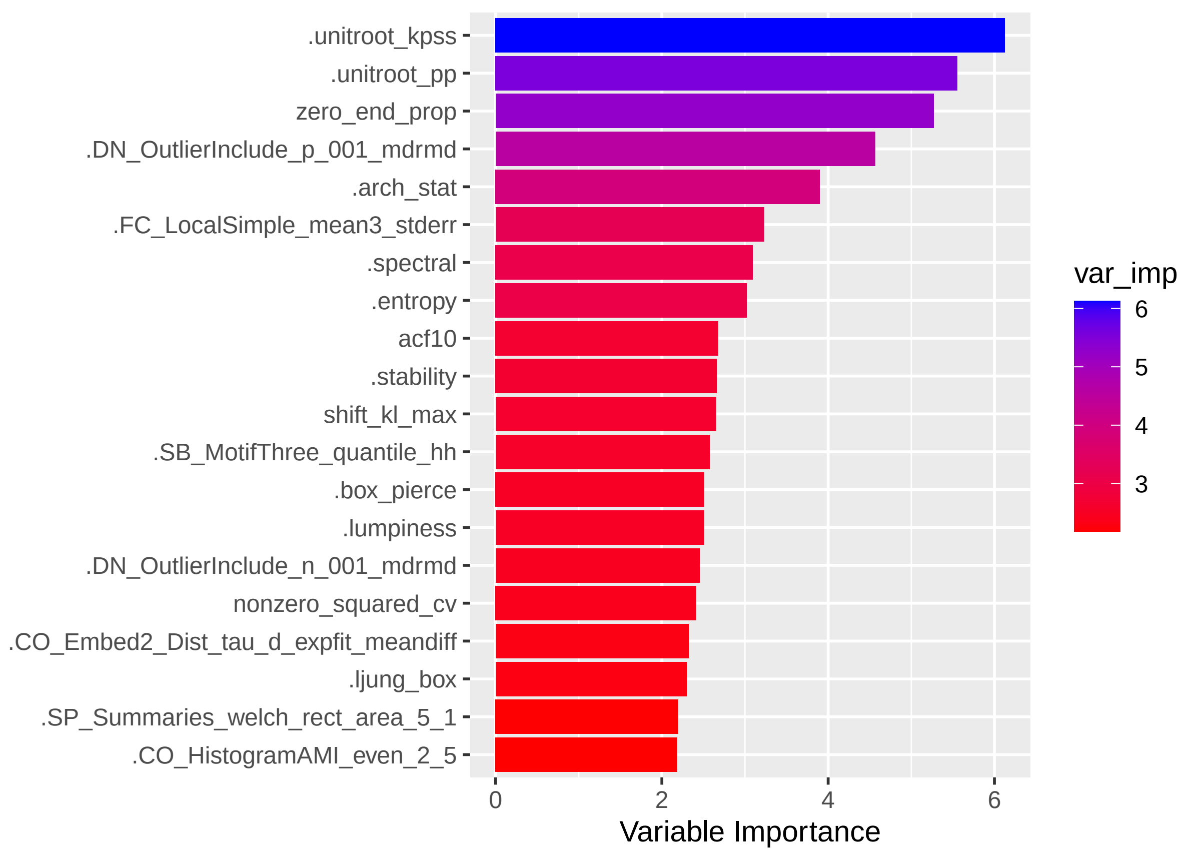
Before we go on, lets look at which variables the model actually uses (pruning could prove useful):

```r
variable_importances <- soothsayer_model_importance[["variable.importance"]]
# rescale so they sum to 100
variable_importances <- (variable_importances/sum(variable_importances)) * 100
df <- data.frame( var_imp = variable_importances, var = names(variable_importances) )

ggplot2::ggplot(df, ggplot2::aes( x = reorder(var,var_imp),
                                  y = var_imp,
                                  fill = var_imp )
                ) +
      ggplot2::geom_bar(stat="identity", position="dodge") +
      ggplot2::coord_flip()+
      ggplot2::ylab("Variable Importance")+
      ggplot2::xlab("")+
      ggplot2::scale_fill_gradient(low="red", high="blue")
```



Pretty, but not super readable - what about top 20?

```
df %>%
  dplyr::arrange(dplyr::desc(var_imp)) %>%
  .[1:20,] %>%
  ggplot2::ggplot( ggplot2::aes( x = reorder(var,var_imp),
                                 y = var_imp,
                                 fill = var_imp )
               ) +
      ggplot2::geom_bar(stat="identity", position="dodge") +
      ggplot2::coord_flip()+
      ggplot2::ylab("Variable Importance")+
      ggplot2::xlab("")+
      ggplot2::scale_fill_gradient(low="red", high="blue")
```



Interesting. TODO: Elaborate.

```
preds <- predict(soothsayer_model, soothsayer_test %>%
  dplyr::select(-c(".model", "key")) %>%
  as.matrix(), type = "response")

selected_models <- preds[["predictions"]] %>%
  as.data.frame() %>% dplyr::rowwise() %>%
  dplyr::mutate( best = which.max(dplyr::across())) %>%
  dplyr::ungroup() %>%
  dplyr::mutate( selected_model = colnames(preds[["predictions"]])[best] ) %>%
  dplyr::select( selected_model ) %>%
  dplyr::bind_cols( soothsayer_test ) %>%
  dplyr::select( c("selected_model", "key") )

matches <- accuracies %>%
  dplyr::right_join( selected_models, by = "key" )
```

Are we better than just fitting any individual model to all series?

```
check_against_model <- function( acc_data, model, metric = "RMSE" ) {

  model_acc <- acc_data %>%
    dplyr::filter( .model == model  ) %>%
    dplyr::select( tidyselect::all_of(c("key", metric)))

  selected_acc <- acc_data %>%
    dplyr::filter( .model == selected_model ) %>%
```

```r
    dplyr::select(  tidyselect::all_of(c("key", metric)) )

  colnames(selected_acc)[ colnames(selected_acc) == metric ] <- paste0("selected_",metric)

  dplyr::full_join(model_acc, selected_acc, key = "key")
}

performance_vs_single_model <- purrr::map( c("ar", "arima", "croston", "ets", "nnetar", "theta"),
            function( model_name ){
              check_against_model(matches, model_name) %>%
                dplyr::mutate( ratio = RMSE/selected_RMSE) %>%
                # deliberately remove high ratios
                dplyr::filter( ratio < quantile(ratio, probs = 0.99, na.rm = TRUE) &
                               ratio < 10) %>%
                dplyr::summarise( mean_ratio = mean(ratio, na.rm = TRUE)) %>%
                dplyr::mutate( model_name = model_name )
            }) %>%
  dplyr::bind_rows()
```

```
## Joining, by = "key"
## Joining, by = "key"
## Joining, by = "key"
## Joining, by = "key"
## Joining, by = "key"
## Joining, by = "key"
```

We get the performance against each individual model: The ratio is the RMSE of a given model, vs our model (averaged). Thus, numbers > 1 are good, numbers < 1 are bad, etc.

```r
knitr::kable(performance_vs_single_model)
```

mean_ratio	model_name
1.333228	ar
1.145757	arima
1.630552	croston
1.178058	ets
1.371265	nnetar
1.269242	theta

So we pretty much beat any single model individually (cool!). Maybe that changes with different metrics, or we can do even better.

How often do we do better, and how often do we do worse, though?

```r
better_or_worse_than_single_model <- purrr::map( c("ar", "arima", "croston", "ets", "nnetar", "theta"),
            function( model_name ){
              check_against_model(matches, model_name) %>%
                dplyr::mutate( ratio = RMSE/selected_RMSE) %>%
                dplyr::filter( !is.na(ratio) ) %>%
                dplyr::summarise( worse = sum(ratio < 1)/length(ratio),
                                  better_or_equal = sum(ratio >= 1)/length(ratio)) %>%
                dplyr::mutate( model_name = model_name )
            }) %>%
  dplyr::bind_rows()
```

```
## Joining, by = "key"
## Joining, by = "key"
## Joining, by = "key"
## Joining, by = "key"
## Joining, by = "key"
## Joining, by = "key"
```

```
knitr::kable(better_or_worse_than_single_model, digits = 2)
```

worse	better_or_equal	model_name
0.26	0.74	ar
0.32	0.68	arima
0.28	0.72	croston
0.37	0.63	ets
0.26	0.74	nnetar
0.27	0.73	theta

So even against **fable::ETS** we are still better about 74.037461, 67.6477684, 72.0528136, 63.3704062, 73.5348408, 73.0326737% of the time. But how do we measure up to the **best** model for any particular time series, if we take this as a ratio of RMSE again?

```
check_against_best_model <- function( acc_data, metric = "RMSE" ) {

  model_acc <- acc_data %>%
    dplyr::select( tidyselect::all_of(c(metric,"key"))) %>%
    dplyr::group_by(key) %>%
    dplyr::mutate(ranking = dplyr::across(where(is.numeric), dplyr::min_rank)) %>%
    dplyr::filter(ranking == 1) %>%
    dplyr::ungroup() %>%
    dplyr::select( tidyselect::all_of(c(metric, "key")))

  selected_acc <- acc_data %>%
    dplyr::filter( .model == selected_model ) %>%
    dplyr::select(  tidyselect::all_of(c("key", metric)) )

  colnames(selected_acc)[ colnames(selected_acc) == metric ] <- paste0("selected_",metric)

  dplyr::full_join(model_acc, selected_acc, key = "key")
}

performance_vs_best_model <-
              check_against_best_model(matches) %>%
                dplyr::mutate( ratio = RMSE/selected_RMSE)
```

```
## Joining, by = "key"
```

```
knitr::kable(dplyr::summarise( performance_vs_best_model, mean_ratio = mean(ratio, na.rm = TRUE)) )
```

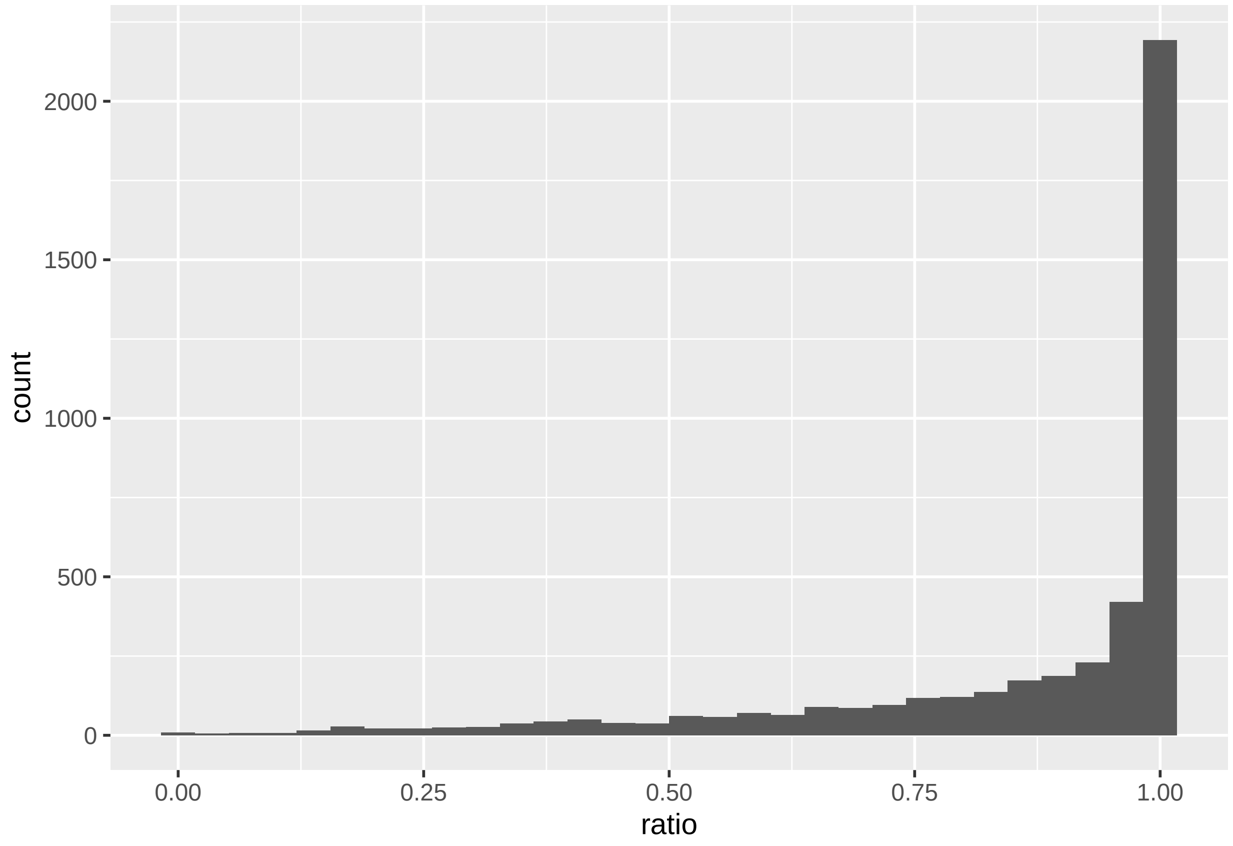
mean_ratio
0.8668714

Here we see that we are some 0.866871407155055% worse (on average) than the absolute best. What is the rough distribution here? (Note that if we always picked the best model, we would get a ratio of 1 - the lower we are, the worse.)

```
performance_vs_best_model %>%
  ggplot2::ggplot( ggplot2::aes(x = ratio)) +
  ggplot2::geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 37 rows containing non-finite values (stat_bin).
```



That looks like we pick the best model (or close to it) quite often - lets check:

```
performance_vs_best_model %>%
  dplyr::summarise( picked_almost_best = sum(ratio > 0.95, na.rm = TRUE)/length(na.omit(ratio)),
                    picked_best = sum(ratio == 1, na.rm = TRUE)/length(na.omit(ratio))) %>%
  knitr::kable()
```

picked_almost_best	picked_best
0.5797718	0.3604833

So we pick close to the best model about % of the time, and the absolute best model about 38% of the time.

How often do we fail (i.e. how often do we pick a model with a RMSE more than 20% worse than the best one) ?

```
bad <- performance_vs_best_model %>%
  dplyr::summarise( picked_bad = sum(ratio < 0.8, na.rm = TRUE)/length(na.omit(ratio)))
```

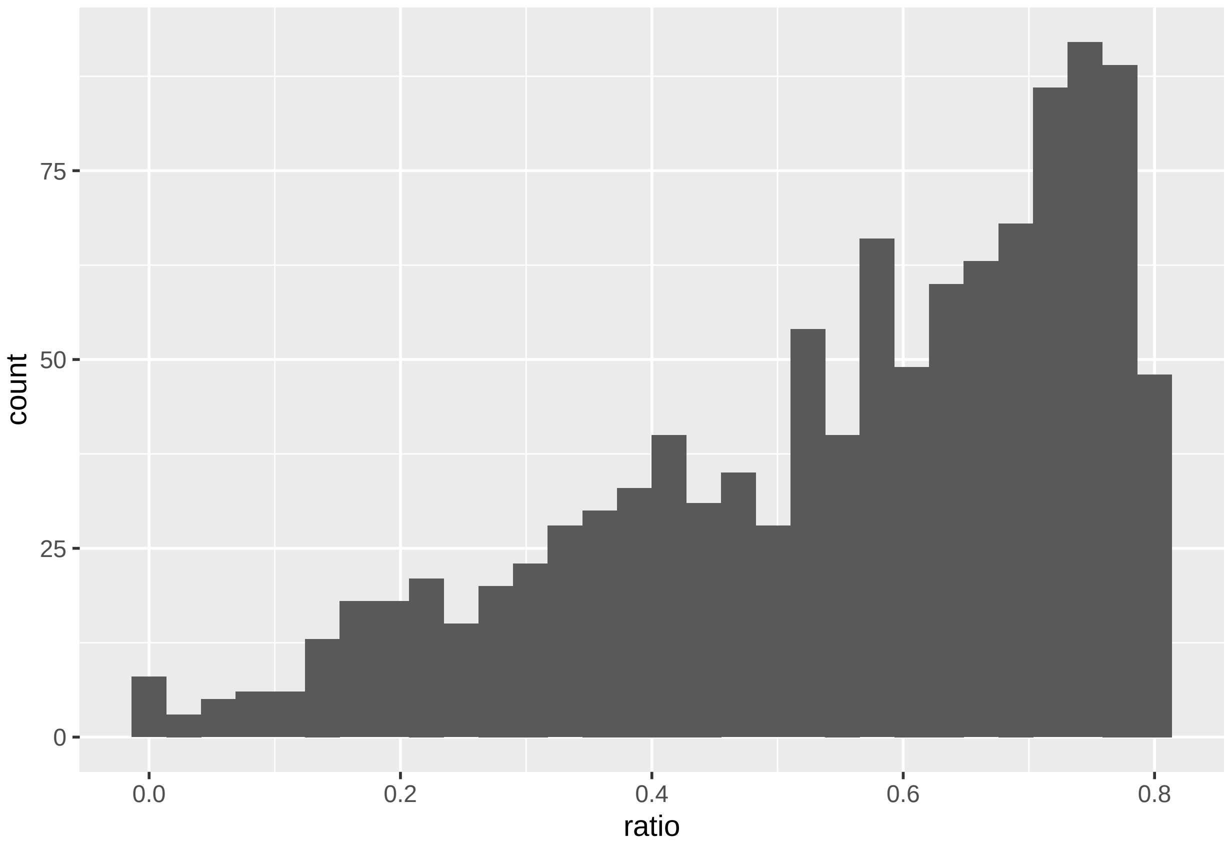
```
knitr::kable(bad)
```

picked_bad
0.245245

About 24.5245021% of the time - unfortunate. Just how bad are these picks, and where do they tend to cluster?

```
performance_vs_best_model %>%
  dplyr::filter( ratio < 0.8 ) %>%
  ggplot2::ggplot( ggplot2::aes(x = ratio)) +
  ggplot2::geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



TODO: We need to look a bit deeper into the cases where we fail.

TODO:

Podium counts: (3 best methods get re-sampled probabilistically)

```
podium_counts <- accuracy_ranks %>%
  dplyr::group_by(key) %>%
  dplyr::arrange( avg_rank ) %>%
  dplyr::slice_head(n = 3) %>%
  dplyr::select(.model, key) %>%
  dplyr::ungroup()
```