1 Intro

The aim of this series of blog is to predict monthly admissions to Singapore public acute adult hospitals. EDA for the dataset was explored in past posts (part 1 ; part 2).

```
library(tidyverse)
library(tidymodels)
library(timetk)
library(modeltime)
library(modeltime.ensemble)

# dataset and dataset for future forecast dats and pre-processing
recipes were done in the past post. The output was uploaded onto my
github.
url_datasets<-url("https://github.com/notast/hierarchical-forecasting/blob/
main/4Dataset_ML.rds?raw=true")
load(url_datasets)
close(url_datasets)

head(to_train,10)
## # A tibble: 10 x 47
##     row_id Level     Name  Date       Admission Admission_lag10
Admission_lag10_r~
##      <int> <chr>     <chr> <date>         <dbl>           <dbl>
<dbl>
##  1       1 Cluster~ NHG   2016-01-01      8035           8184.
7840.
##  2       2 Cluster~ NHG   2016-02-01      7526           7801.
8185.
##  3       3 Cluster~ NHG   2016-03-01      8419           8283.
8072.
##  4       4 Cluster~ NHG   2016-04-01      7934           7926.
8188.
##  5       5 Cluster~ NHG   2016-05-01      8048           8168.
```

```
8169.
## 6       6 Cluster~ NHG   2016-06-01      8199            8229.
8097.
## 7       7 Cluster~ NHG   2016-07-01      8230            7608.
7982.
## 8       8 Cluster~ NHG   2016-08-01      8496            7969.
7844.
## 9       9 Cluster~ NHG   2016-09-01      7991            7792.
8009.
## 10     10 Cluster~ NHG   2016-10-01      8284            8041.
8035
## # ... with 40 more variables: Admission_lag10_roll_6 <dbl>,
## #   Admission_lag10_roll_12 <dbl>, Covid <chr>, F_trend_strength
<dbl>,
## #   F_seasonal_strength_year <dbl>, F_seasonal_peak_year <dbl>,
## #   F_seasonal_trough_year <dbl>, F_spikiness <dbl>, F_linearity
<dbl>,
## #   F_curvature <dbl>, F_stl_e_acf1 <dbl>, F_stl_e_acf10 <dbl>,
F_acf1 <dbl>,
## #   F_acf10 <dbl>, F_diff1_acf1 <dbl>, F_diff1_acf10 <dbl>,
F_diff2_acf1 <dbl>,
## #   F_diff2_acf10 <dbl>, F_season_acf1 <dbl>, F_kpss_stat <dbl>,
## #   F_kpss_pvalue <dbl>, F_pp_stat <dbl>, F_pp_pvalue <dbl>,
F_ndiffs <int>,
## #   F_bp_stat <dbl>, F_bp_pvalue <dbl>, F_lb_stat <dbl>, F_lb_pvalue
<dbl>,
## #   F_var_tiled_var <dbl>, F_var_tiled_mean <dbl>, F_shift_level_max
<dbl>,
## #   F_shift_level_index <dbl>, F_shift_var_max <dbl>,
F_shift_var_index <dbl>,
## #   F_shift_kl_max <dbl>, F_shift_kl_index <dbl>, F_spectral_entropy
<dbl>,
## #   F_n_crossing_points <int>, F_longest_flat_spot <int>,
F_stat_arch_lm <dbl>
```

The admissions were treated as a hierarchical time series as every country has a hierarchical order to its public hospitals including Singapore. The levels are

National level
 |– Cluster level (Clusters are a network of hospitals based on geographical regions. There are 3 health clusters in Singapore.)
  |– Hospital level (There are 8 public acute adult hospitals.)

Admissions were forecasted at each level. Previously, classical approach using traditional techniques e.g. bottoms up and newer techniques e.g. reconciliation were employed. In this blog, machine learning approaches were experimented.

2 Cross validation

The dataset starts from Jan 2016 and ends in Feb 2021 (`to_train`). This dataset was split into training and testing set. The training set was from Jan 16 to Apr 20 (3 years, 4months) and the test set was from May 20 to Feb 21 (10 months). Cross validation was applied to the training set as the forecasting was a machine learning problem.

```
# train/test
splits<- to_train %>%
    time_series_split(Date, assess= "10 months", cumulative = T)
## Data is not ordered by the 'date_var'. Resamples will be arranged by
`Date`.
## Overlapping Timestamps Detected. Processing overlapping time series
together using sliding windows.
# training set
splits_train<- training(splits)

# vfold from training set
set.seed(69)
folds<-vfold_cv(splits_train, strata = Admission, v=5)
```

Metrics

```
metrics_custom= metric_set(rmse, mae)
```

3. Pre-processing

The base recipe, `rec_PC`, was crafted in the previous post where different combination of predictors and features engineered for machine learning were screened The predictors are:

1. Lags (`Admission_lag10`)
2. Rolling lags (`Admission_lag_roll_???`)
3. Covid peak period (`Covid`)
4. The PCA of time series features and statistics (`F_???`)
5. The hierarchical level (`Level`) and its members (`Name`)

```
rec_PC %>% summary() %>%  filter(role=="predictor") %>% pull(variable)
##  [1] "Level"                     "Name"
##  [3] "Covid"                     "Admission_lag10"
##  [5] "Admission_lag10_roll_3"    "Admission_lag10_roll_6"
##  [7] "Admission_lag10_roll_12"   "F_trend_strength"
##  [9] "F_seasonal_strength_year"  "F_seasonal_peak_year"
## [11] "F_seasonal_trough_year"    "F_spikiness"
## [13] "F_linearity"               "F_curvature"
## [15] "F_stl_e_acf1"              "F_stl_e_acf10"
## [17] "F_acf1"                    "F_acf10"
## [19] "F_diff1_acf1"              "F_diff1_acf10"
## [21] "F_diff2_acf1"              "F_diff2_acf10"
## [23] "F_season_acf1"             "F_kpss_stat"
## [25] "F_kpss_pvalue"             "F_pp_stat"
## [27] "F_pp_pvalue"               "F_ndiffs"
## [29] "F_bp_stat"                 "F_bp_pvalue"
## [31] "F_lb_stat"                 "F_lb_pvalue"
## [33] "F_var_tiled_var"           "F_var_tiled_mean"
## [35] "F_shift_level_max"         "F_shift_level_index"
## [37] "F_shift_var_max"           "F_shift_var_index"
## [39] "F_shift_kl_max"            "F_shift_kl_index"
```

```
## [41] "F_spectral_entropy"       "F_n_crossing_points"
## [43] "F_longest_flat_spot"      "F_stat_arch_lm"
```

The pre-processing steps are:

```
tidy(rec_PC)
## # A tibble: 8 x 6
##   number operation type                  trained skip  id
##    <int> <chr>     <chr>                 <lgl>   <lgl> <chr>
## 1      1 step      timeseries_signature FALSE   FALSE
timeseries_signature_ExurK
## 2      2 step      rm                    FALSE   FALSE rm_OHKgv
## 3      3 step      rm                    FALSE   FALSE rm_GnkOP
## 4      4 step      rm                    FALSE   FALSE rm_CeaGH
## 5      5 step      rm                    FALSE   FALSE rm_BZcN8
## 6      6 step      dummy                 FALSE   FALSE dummy_xLyW5
## 7      7 step      normalize             FALSE   FALSE
normalize_pAZbw
## 8      8 step      pca                   FALSE   FALSE pca_aRkvg
```

The following machine learning models were trial thus the base recipes had to be revised to compliment some of the models.

1. Elastic net regression `GLM` (Linear model)
2. Multivariate adaptive regression spline `MARS` (Non-linear model)
3. Random forest `RF` (Tree model)
4. Extreme gradient boost `XGB` (Tree-model)
5. Boosted PROPHET `PB` (Classical approach + Tree-model) ~~6. LightGBM (Tree-model, Light GBM has seen success with hierarchical time series in the M5 competition but fatal errors were encountered when running it in `R`)~~

3.1 Base recipe

The base recipe was extended to remove non-zero variance.

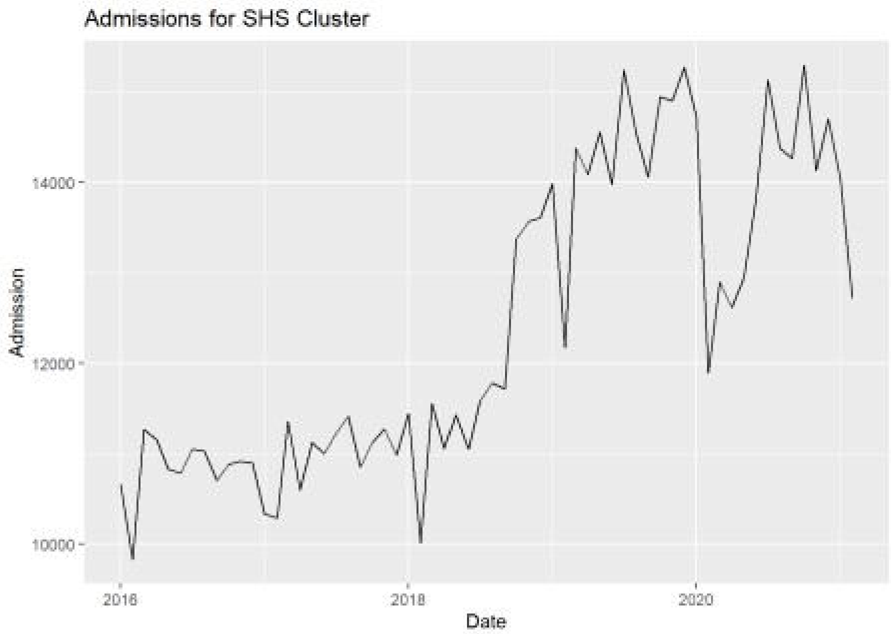
```
rec_PC<- rec_PC %>% step_nzv(all_numeric_predictors())

rec_PC %>% tidy()
## # A tibble: 9 x 6
##   number operation type                  trained skip  id
##    <int> <chr>     <chr>                 <lgl>   <lgl> <chr>
## 1      1 step      timeseries_signature FALSE   FALSE
timeseries_signature_ExurK
## 2      2 step      rm                    FALSE   FALSE rm_OHKgv
## 3      3 step      rm                    FALSE   FALSE rm_GnkOP
## 4      4 step      rm                    FALSE   FALSE rm_CeaGH
## 5      5 step      rm                    FALSE   FALSE rm_BZcN8
## 6      6 step      dummy                 FALSE   FALSE dummy_xLyW5
## 7      7 step      normalize             FALSE   FALSE
normalize_pAZbw
## 8      8 step      pca                   FALSE   FALSE pca_aRkvg
## 9      9 step      nzv                   FALSE   FALSE nzv_bU0rc
```

3.2 Spline recipe

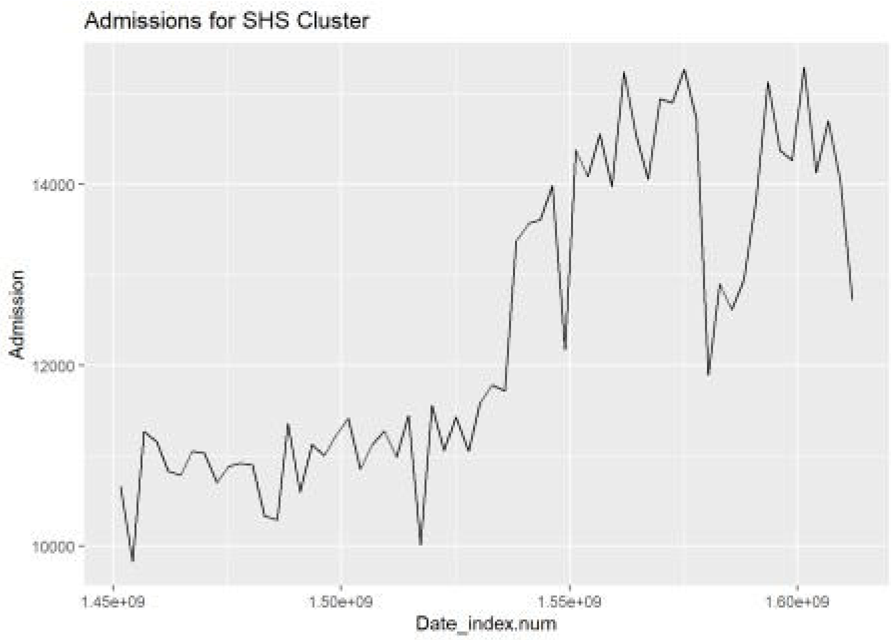
To assist the linear model `GLM` to capture the wiggles of the time series, splines were included.

```
to_train %>% filter(Name=="SHS") %>% ggplot(aes(Date, Admission)) +
geom_line() + labs(title = "Admissions for SHS Cluster")
```



However `glmnet` does not tolerate non-numeric variables thus `Date` cannot be used as the variable for creating the splines. `step_timeseries_signature` derives various features from `Date` including a numeric equivalent, `Date_index.num`.

```
to_train %>% filter(Name=="SHS") %>%
  recipe(~.) %>% step_timeseries_signature(Date) %>% prep() %>% juice()
%>%
  ggplot(aes(Date_index.num, Admission)) + geom_line() + labs(title =
"Admissions for SHS Cluster")
```



B-splines were created as there are more flexible than natural splines; hopefully, capturing the

wiggles of the time series. The degrees of freedom was `tune`d.
All numeric predictors with zero variance were removed and then all numeric predictors were normalized to prepare it for elastic net regression.

```
rec_spline<- rec_PC %>%
    step_bs(Date_index.num, deg_free = tune("bs_df")) %>%
    step_zv(all_numeric_predictors())%>%
    step_normalize(all_numeric_predictors())

rec_spline %>% tidy() %>% pull(type)
##  [1] "timeseries_signature" "rm"                   "rm"
##  [4] "rm"                   "rm"                   "dummy"
##  [7] "normalize"            "pca"                  "nzv"
## [10] "bs"                   "zv"                   "normalize"
```

3.3 Prophet boost recipe

When using `modeltime` for forecasting, the date column is treated differently for classical and machine learning approach. Previously, the date column was not treated as a predictor as a machine learning approach was adopted. However, Prophet Boost requires the date to be a predictor for the Prophet part of the model as prophet is a classical approach.

```
rec_Date<- rec_PC %>% update_role(Date, new_role = "predictor")
```

4 Modelling

Modelling steps are easy with `tidymodels`.

i. Set up the model
ii. Add the recipe and model into a workflow
iii. Tune the workflow which in turn tunes parameters of the recipe and/or the model inside the workflow
iv. Finalize the workflow with the best tuned parameters
v. Fit the finalized workflow with its best tuned recipe and best tuned model onto the whole training data.

4.1 GLM

While it is possible to increase the speed of `glmnet` modelling with `blueprint = hardhat::default_recipe_blueprint(composition = "dgCMatrix")`, this sparse matrix is not tolerated further downstream when `modeltime` functions are used for forecasting.

```
glm_m<- linear_reg(penalty = tune(), mixture = tune()) %>%
    set_engine("glmnet") %>% set_mode("regression")


glm_wf<- workflow() %>%
    add_recipe(rec_spline) %>%
    add_model(glm_m)
```

The current degree of freedom is:

```
parameters(glm_wf)$object[[3]]
## Piecewise Polynomial Degree (quantitative)
```

```
## Range: [1, 15]
```

Splines have been recommended to use a different range of degrees of freedom.

```
param_spline<-glm_wf %>% parameters() %>% update(bs_df=spline_degree())

param_spline$object[[3]]
## Piecewise Polynomial Degree (quantitative)
## Range: [1, 10]
set.seed(69)
glm_t<- tune_grid(
    object= glm_wf,
    resamples= folds,
    param_info= param_spline,
    grid= 20,
    metrics= metrics_custom)

fun_fwf<- function(wf, t_wf){
    finalize_workflow(
                     x= wf,
                     parameters= select_best(t_wf, "rmse")) %>%
    fit(splits_train)
    }

glm_f<-fun_fwf(glm_wf, glm_t)
```

4.2 MARS

```
mars_m<- mars(num_terms   = tune(),prod_degree = tune()) %>%
  set_engine("earth") %>%   set_mode("regression")

mars_wf<- workflow() %>% add_recipe(rec_PC) %>% add_model(mars_m)

set.seed(69)
mars_t<- tune_grid(
    object= mars_wf,
    resamples= folds, para_info_NULL, grid=10, metrics=metrics_custom)

mars_f<- fun_fwf(mars_wf, mars_t)
```

4.3 RF

```
rf_m<- rand_forest(min_n = tune(), trees = tune()) %>%
    set_engine("ranger") %>% set_mode("regression")

rf_wf<- workflow() %>% add_recipe(rec_PC) %>% add_model(rf_m)

set.seed(69)
rf_t<- tune_grid(
    object=rf_wf,
    resamples = folds,
    grid=10,
    param_info = NULL, metrics = metrics_custom)
```

```
rf_f<-fun_fwf(rf_wf, rf_t)
```

4.4 XGB

```
xgb_m<- boost_tree(sample_size = tune(),
                   min_n = tune(),
                   tree_depth = tune(),
                   loss_reduction = tune(),
                   trees=tune()) %>%
    set_engine("xgboost") %>% set_mode("regression")

xgb_wf<- workflow() %>% add_recipe(rec_PC ) %>% add_model(xgb_m)

all_cores <- parallel::detectCores(logical = FALSE)
library(doParallel)
cl <- makePSOCKcluster(all_cores)
registerDoParallel(cl)

set.seed(69)
xgb_t<-tune_bayes(
    object= xgb_wf,
    resamples = folds,
    iter = 20,
    param_info = NULL, initial = 9,
    metrics = metrics_custom,
    control = control_bayes(no_improve = 20))

xgb_f<-fun_fwf(xgb_wf, xgb_t)
```

4.5 Prophet boost

```
pb_m<- prophet_boost(
    seasonality_daily = FALSE, seasonality_weekly = FALSE,
seasonality_yearly = FALSE,
                   min_n = tune(),
                   tree_depth = tune(),
                   loss_reduction = tune(),
                   trees=tune()) %>%
    set_engine("prophet_xgboost")

pb_wf<- workflow() %>% add_recipe(rec_Date) %>%  add_model(pb_m)

set.seed(69)
pb_t<- tune_bayes(
    object= pb_wf,
    resamples = folds,
    iter = 20,
    param_info = NULL, initial = 9, # Generate five at semi-random to
start
    metrics = metrics_custom,
    control = control_bayes(no_improve = 20))
```

```
pb_f<- fun_fwf(pb_wf, pb_t)
```

5. Evaluation

After machine learning was completed with `tidymodels`, the evaluation and forecasting was completed with `modeltime`. The finalized and fitted workflows were housed in a `modeltime_table`.

```
(e_table<-modeltime_table(glm_f, mars_f, rf_f, xgb_f, pb_f))
## # Modeltime Table
## # A tibble: 5 x 3
##   .model_id .model     .model_desc
##       <int> <list>     <chr>
## 1         1 <workflow> GLMNET
## 2         2 <workflow> EARTH
## 3         3 <workflow> RANGER
## 4         4 <workflow> XGBOOST
## 5         5 <workflow> PROPHET W/ XGBOOST ERRORS
```

The workflows in the `modeltime_table` were calibrated on the testing set.

```
(e_cal <- e_table %>% modeltime_calibrate(testing(splits)))
## Warning in bs(x = c(1588291200, 1590969600, 1593561600, 1596240000,
## 1598918400, : some 'x' values beyond boundary knots may cause ill-
conditioned
## bases

## Warning in bs(x = c(1588291200, 1590969600, 1593561600, 1596240000,
## 1598918400, : some 'x' values beyond boundary knots may cause ill-
conditioned
## bases

## Warning in bs(x = c(1588291200, 1590969600, 1593561600, 1596240000,
## 1598918400, : some 'x' values beyond boundary knots may cause ill-
conditioned
## bases
## # Modeltime Table
## # A tibble: 5 x 5
##   .model_id .model     .model_desc               .type
.calibration_data
##       <int> <list>     <chr>                     <chr> <list>
## 1         1 <workflow> GLMNET                    Test  <tibble [120
x 4]>
## 2         2 <workflow> EARTH                     Test  <tibble [120
x 4]>
## 3         3 <workflow> RANGER                    Test  <tibble [120
x 4]>
## 4         4 <workflow> XGBOOST                   Test  <tibble [120
x 4]>
## 5         5 <workflow> PROPHET W/ XGBOOST ERRORS Test  <tibble [120
x 4]>
```

The tree models performed well on the testing set. XGB performer poorer than prophet boost likely due to XGB's reduced ability to capture trends which is better captured by the Prophet part of Prophet Boost. Earlier EDA revealed that more hospitals have higher trend strength than seasonal strength (yearly).

```
e_cal %>% modeltime_accuracy(metric_set = metrics_custom) %>%
arrange(rmse, sort=T)
## # A tibble: 5 x 5
##   .model_id .model_desc               .type  rmse   mae
##       <int> <chr>                     <chr> <dbl> <dbl>
## 1         3 RANGER                    Test   549.  410.
## 2         5 PROPHET W/ XGBOOST ERRORS Test  1137.  799.
## 3         4 XGBOOST                   Test  1231.  888.
## 4         2 EARTH                     Test  3796. 3312.
## 5         1 GLMNET                    Test  9847. 8281.
```

Conclusion

The top 2 models, Random forest and Prophet Boost, were selected for retuning to improve performance. Retuning would be covered in the next post.

```
#The folllowing were exported to save code and computational time for
future posts
#1. Datasets and the cross validation splits AND folds
save(to_train, to_predictfuture, splits, splits_train, folds,
file="5Data_CV.RData")
#2. tuned grids and workflows (for retunining)
save(rf_t, rf_wf, pb_t, pb_wf, file="5Retunning_objects.RData")
#3. finalized and fitted wf (to add to modeltime table and compare
retunned and OG wf)
save(glm_f, mars_f, rf_f, xgb_f, pb_f, file="5FFWf.RData")
```