1. Intro
2. Data wrangling
   1. Long format with aggregated values
   2. Extend into the future
   3. External regressor
      1. Lags and rolling lags
      2. Covid
      3. Time series features
3. Splitting
4. Pre-processing recipes Pre-processing order
5. Modelling

Workflow

1. Evaluate
   1. Evaluate against the training set What’s inside the calibrated table

6.1 Evaluate with cross validation 8 Conclusion

# Intro

The aim of this series of blog is to predict monthly admissions to Singapore public acute adult hospitals.

library(tidyverse) library(tidymodels) library(timetk) library(modeltime)

# cleaned up dataset downloaded from my github. Clean up of OG dataset done in 1st post

raw<- read\_csv("https://raw.githubusercontent.com/notast/hierarchical-forecasting/main/ stat\_sg\_CLEAN.csv")

The admissions were treated as a hierarchical time series. Every country has a hierarchical order to its public hospitals. In Singapore, there are 3 levels:

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.)

In this post, different combinations of predictors were screened to determine the set of predictors for machine learning.

# Data wrangling

The dataset starts from Jan 2016 and ends in Feb 2021. The training set was from Jan 16 to Apr 20 (3 years, 4months) and the testing set was from May 20 to Feb 21 (10 months). The forecast horizon was 10 months as the data ended in Feb 21 and the goal was to forecast admissions till the end of 2021. The splitting of training, testing and future data are covered later in this section.

## Long format with aggregated values

When fpp3::reconcile was used for classical approach, each column represented the members of that particular level and aggregated values for each level were not required.

tribble(

~Level1\_Hospital, ~Level2\_Cluster, ~Admission, "CGH", "SHS", 100,

"SKH", "SHS", 200,

"TTSH", "NHG", 900)

## # A tibble: 3 x 3

## Level1\_Hospital Level2\_Cluster Admission ## <chr> <chr> <dbl>

## 1 CGH SHS 100

## 2 SKH SHS 200

## 3 TTSH NHG 900

In the machine learning approach, all the hierarchical levels form a variable and the corresponding subordinate members form another variable. Hence, the aggregated values had to be calculated. The variable of all levels and the variable of the corresponding members were treated as categorical variables in this machine learning problem.

tribble(

~Level, ~Name, ~Admission, "Level 1", "CGH", 100,

"Level 1", "SKH", 200,

"Level 1", "TTSH", 900,

"Level 2" ,"SHS", 300,

"Level 2", "NHG", 900,

"Level 3", "National", 1200) ## # A tibble: 6 x 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ## |  | Level |  | Name | Admission |
| ## |  | <chr> |  | <chr> | <dbl> |
| ## | 1 | Level | 1 | CGH | 100 |
| ## | 2 | Level | 1 | SKH | 200 |
| ## | 3 | Level | 1 | TTSH | 900 |
| ## | 4 | Level | 2 | SHS | 300 |
| ## | 5 | Level | 2 | NHG | 900 |
| ## | 6 | Level | 3 | National | 1200 |

df<-raw %>%

# add national

mutate(National\_id= "National") %>% # add id as suffix to all levels

rename\_with(.fn= ~paste0(.x, "\_id"), .cols = c(Hospital, Cluster ))

%>%

pivot\_longer(cols=ends\_with("id"), names\_to = "Level", values\_to="Name")%>%

group\_by(Level, Name, Date) %>% summarise(Admission=

sum(Admission,na.rm=T), .groups="drop")

## Extend into the future

The machine learning approach was supported with the modeltime meta-package which is like a time series equivalent to tidymodels. In modeltime when a machine learning approach is adopted with external regressors, the future period to be forecasted is appended to the dataset.

full<- df %>% group\_by(Level, Name) %>%

# .bind\_data to append

future\_frame(Date, .length\_out = 10, .bind\_data = TRUE) %>% ungroup()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| full | | %>% tail(10) |  |  | |
| ## # | | A tibble: 10 | x 4 |
| ## | | Level | Name | Date | Admission |
| ## | | <chr> | <chr> | <date> | <dbl> |
| ## | 1 | National\_id | National | 2021-03-01 | NA |
| ## | 2 | National\_id | National | 2021-04-01 | NA |
| ## | 3 | National\_id | National | 2021-05-01 | NA |
| ## | 4 | National\_id | National | 2021-06-01 | NA |
| ## | 5 | National\_id | National | 2021-07-01 | NA |
| ## | 6 | National\_id | National | 2021-08-01 | NA |
| ## | 7 | National\_id | National | 2021-09-01 | NA |
| ## | 8 | National\_id | National | 2021-10-01 | NA |
| ## | 9 | National\_id | National | 2021-11-01 | NA |
| ## | 10 | National\_id | National | 2021-12-01 | NA |

## External regressor

### Lags and rolling lags

Lags and rolling lags were external regressors included in the machine learning approach as these features have seen success in the M5 competition. While lags and rolling windows could have been added with step\_lags and step\_slidify during the recipe phase, these values couldn’t be calculated for future dates. Thus, it was calculated in this dataset which included future dates.

Lag periods were set to be the same as the forecast horizon and the rolling lags were based temporal periods heuristically associated with a year e.g. 3,6,12 months.

full<- full %>% group\_by(Level, Name) %>% tk\_augment\_lags(Admission, .lags = 10) %>%

tk\_augment\_slidify( Admission\_lag10,

.f = ~ mean(., na.rm = TRUE),

.period = c(3,6,12),

.align = "center",

.partial = TRUE) %>% ungroup()

**Impute**

Lags and rolling lags resulted in missing values for earlier observations.

full %>% head(10)

## # A tibble: 10 x 8

## Level Name Date Admission Admission\_lag10 Admission\_lag10\_roll\_3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ## | <chr> | <chr> | <date> | <dbl> | <dbl> |
| <dbl> |  |  |  |  |  |
| ## 1 | Cluster\_id | NHG | 2016-01-01 | 8035 | NA |
| NaN |  |  |  |  |  |
| ## 2 | Cluster\_id | NHG | 2016-02-01 | 7526 | NA |
| NaN |  |  |  |  |  |
| ## 3 | Cluster\_id | NHG | 2016-03-01 | 8419 | NA |
| NaN |  |  |  |  |  |
| ## 4 | Cluster\_id | NHG | 2016-04-01 | 7934 | NA |
| NaN |  |  |  |  |  |
| ## 5 | Cluster\_id | NHG | 2016-05-01 | 8048 | NA |
| NaN |  |  |  |  |  |
| ## 6 | Cluster\_id | NHG | 2016-06-01 | 8199 | NA |
| NaN |  |  |  |  |  |
| ## 7 | Cluster\_id | NHG | 2016-07-01 | 8230 | NA |
| NaN |  |  |  |  |  |
| ## 8 | Cluster\_id | NHG | 2016-08-01 | 8496 | NA |
| NaN |  |  |  |  |  |
| ## 9 | Cluster\_id | NHG | 2016-09-01 | 7991 | NA |
| NaN |  |  |  |  |  |
| ## 10 | Cluster\_id | NHG | 2016-10-01 | 8284 | NA |
| 8035 |  |  |  |  |  |

## # ... with 2 more variables: Admission\_lag10\_roll\_6 <dbl>, ## # Admission\_lag10\_roll\_12 <dbl>

These observations were either to be discarded or imputed. As discarding might result in less than ideal number of observations for training , these values were imputed. Missing lags and rolling lags could have been imputed with step\_ts\_clean or step\_ts\_impute. However, these step\_s do not operate on group data i.e. imputation at a global level and not for specific hospitals and clusters. Therefore, the imputation occurred earlier up here.

(full<-full %>% group\_by(Level, Name) %>% mutate(across(.cols= starts\_with("Admission\_"),

.fns = ~ ts\_impute\_vec(.x, period = 12))) %>% ungroup() %>% rowid\_to\_column(var = "row\_id"))

## # A tibble: 864 x 9

## 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 854 more rows, and 2 more variables: Admission\_lag10\_roll\_6 <dbl>

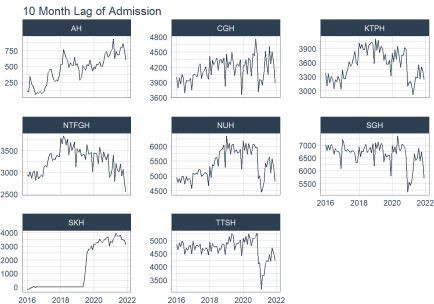
## # Admission\_lag10\_roll\_12 <dbl>

The imputed values were visually inspected.

full %>% filter(Level=="Hospital\_id") %>%

group\_by(Name) %>% plot\_time\_series(Date, Admission\_lag10,

.interactive=F, .smooth = F, .facet\_ncol = 3, .title = "10 Month Lag of Admission"



SKH had lags which were below 0 which is impossible as negative admission is absurd. There were also lags above 0 in 2016 which was not impossible as the hospital opened only in Jul 18.

full %>% filter(Name=="SKH") %>%

plot\_time\_series(Date, Admission\_lag10, .interactive=F, .smooth = F,

.t= "10 Month Lag of SKH Admission")



Thus, the imputations were revised:

* + - 1. The minimum values for lags and rolling lags were set to 0
      2. All lags and rolling lags for SKH before Jul 18 were set to 0

full<-full %>% mutate(across(.cols=(starts\_with("Admission\_lag")),

.fns = ~ ifelse(.x<0, 0, .x))) %>% mutate(across(.cols = (starts\_with("Admission\_lag")),

.fns = ~ifelse(Date<lubridate::ymd("2018-07-01")&

Name=="SKH",

0, .x)))

### Covid

Covid peak periods occurred between Jan 21 to Jul 21 and this period form another categorical variable.

full<- full %>% mutate(Covid= ifelse(

between(Date, lubridate::ymd("2020-01-01"), lubridate::ymd("2020-07-01")),

"yes", "no"))

### Time series features

Time series features and statistics have been used in machine learning approaches for forecasting. Likewise, the time series features of this hierarchical time series were used as predictors for our machine learning approach.

feat\_all\_url<-url("https://github.com/notast/hierarchical-forecasting/blob/ main/3feat\_all.RData?raw=true")

load(feat\_all\_url) close(feat\_all\_url)

full<- left\_join(full, feat\_all, by=c("Level", "Name"))

Appending the future forecast periods and adding the external regressors, increased the dataset from 744 rows to 864 rows and from 4 columns to 47 columns.

glimpse(full) ## Rows: 864

## Columns: 47

## $ row\_id <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,

12, 13, 14~

## $ Level <chr> "Cluster\_id", "Cluster\_id", "Cluster\_id", "Cl~

## $ Name <chr> "NHG", "NHG", "NHG", "NHG", "NHG", "NHG", "NH~

## $ Date <date> 2016-01-01, 2016-02-01,

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2016-03-01, | | 2016-04-~ | | | | | | | |
| ## $ Admission | | | <dbl> | | 8035, 7526, 8419, 7934, 8048, 8199, | | | | |
| 8230, 849~ | | |  | |  | | | | |
| ## $ Admission\_lag10 | | | | <dbl> | 8184.009, | 7801.307, | 8282.589, | | |
| 7925.547, 8167.~ | | | |  |  |  |  | | |
| ## $ Admission\_lag10\_roll\_3 | | | | <dbl> | 7839.890, | 8184.588, | 8071.753, | | |
| 8187.940, 8168.~ | | | |  |  |  |  | | |
| ## $ Admission\_lag10\_roll\_6 | | | | <dbl> | 8049.493, | 8091.948, | 8236.410, | | |
| 8140.612, 8162.~ | | | |  |  |  |  | | |
| ## $ Admission\_lag10\_roll\_12 | | | | <dbl> | 8057.390, | 8068.275, | 8064.694, | | |
| 8049.987, 8035.~ | | | |  |  |  |  | | |
| ## $ Covid | | | | <chr> | "no", "no", "no", "no", "no", "no", | | | | |
| "no", "no~ | | | |  |  | | | | |
| ## $ F\_trend\_strength | | | | <dbl> | 0.6829768, 0.6829768, 0.6829768, | | | | |
| 0.6829768, 0~ | | | |  |  | | | | |
| ## $ F\_seasonal\_strength\_year | | | | <dbl> | 0.4330629, 0.4330629, 0.4330629, | | | | |
| 0.4330629, 0~ | | | |  |  | | | | |
| ## | $ F\_seasonal\_peak\_year | | | <dbl> | 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, | | | | 0, |
| 0, | 0, 0, ~ | | |  |  | | | |  |
| ## | $ F\_seasonal\_trough\_year | | | <dbl> | 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, | | | | 2, |
| 2, | 2, 2, ~ | | |  |  | | | |  |
| ## | $ F\_spikiness | | | <dbl> | 10383595, 10383595, 10383595, | | | |  |
| 10383595, 10383~ | | |  | |  |  | |  | |
| ## $ F\_linearity | | | <dbl> | | -1526.063, | -1526.063, | | -1526.063, | |
| -1526.063, -~ | | |  | |  |  | |  | |
| ## $ F\_curvature | | | <dbl> | | -2821.471, | -2821.471, | | -2821.471, | |
| -2821.471, -~ | | |  | |  |  | |  | |
| ## $ F\_stl\_e\_acf1 | | | <dbl> | | 0.6609598, | 0.6609598, | | 0.6609598, | |
| 0.6609598, 0~ | | |  | |  |  | |  | |
| ## $ F\_stl\_e\_acf10 | | | <dbl> | | 1.239149, 1.239149, 1.239149, | | | | |
| 1.239149, 1.239~ | | |  | |  | | | | |
| ## $ F\_acf1 | | | <dbl> | | 0.6453483, 0.6453483, 0.6453483, | | | | |
| 0.6453483, 0~ | | |  | |  | | | | |
| ## $ F\_acf10 | | | <dbl> | | 1.279804, 1.279804, 1.279804, | | | | |
| 1.279804, 1.279~ | | |  | |  | | | | |
| ## $ F\_diff1\_acf1 | | | <dbl> | | -0.4949473, -0.4949473, -0.4949473, | | | | |
| -0.494947~ | | |  | |  | | | | |
| ## $ F\_diff1\_acf10 | | | <dbl> | | 0.6419415, 0.6419415, 0.6419415, | | | | |
| 0.6419415, 0~ | | |  | |  | | | | |

|  |  |  |
| --- | --- | --- |
| ## $ F\_diff2\_acf1 | <dbl> | -0.7519704, -0.7519704, -0.7519704, |
| -0.751970~ |  |  |
| ## $ F\_diff2\_acf10 | <dbl> | 1.209413, 1.209413, 1.209413, |
| 1.209413, 1.209~ |  |  |
| ## $ F\_season\_acf1 | <dbl> | 0.1026575, 0.1026575, 0.1026575, |
| 0.1026575, 0~ |  |  |
| ## $ F\_kpss\_stat | <dbl> | 0.410627, 0.410627, 0.410627, |
| 0.410627, 0.410~ |  |  |
| ## $ F\_kpss\_pvalue | <dbl> | 0.07257455, 0.07257455, 0.07257455, |
| 0.0725745~ |  |  |
| ## $ F\_pp\_stat | <dbl> | -3.333663, -3.333663, -3.333663, |
| -3.333663, -~ |  |  |
| ## $ F\_pp\_pvalue | <dbl> | 0.02307222, 0.02307222, 0.02307222, |
| 0.0230722~ |  |  |
| ## $ F\_ndiffs | <int> | 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, |
| 0, 0, 0, ~ |  |  |
| ## $ F\_bp\_stat | <dbl> | 25.82141, 25.82141, 25.82141, |
| 25.82141, 25.82~ |  |  |
| ## $ F\_bp\_pvalue | <dbl> | 3.745109e-07, 3.745109e-07, |
| 3.745109e-07, 3.7~ |  |  |
| ## $ F\_lb\_stat | <dbl> | 27.09132, 27.09132, 27.09132, |
| 27.09132, 27.09~ |  |  |
| ## $ F\_lb\_pvalue | <dbl> | 1.940678e-07, 1.940678e-07, |
| 1.940678e-07, 1.9~ |  |  |
| ## $ F\_var\_tiled\_var | <dbl> | 0.2356997, 0.2356997, 0.2356997, |
| 0.2356997, 0~ |  |  |
| ## $ F\_var\_tiled\_mean | <dbl> | 0.696821, 0.696821, 0.696821, |
| 0.696821, 0.696~ |  |  |
| ## $ F\_shift\_level\_max | <dbl> | 1404, 1404, 1404, 1404, 1404, 1404, |
| 1404, 140~ |  |  |
| ## $ F\_shift\_level\_index | <dbl> | 50, 50, 50, 50, 50, 50, 50, 50, 50, |
| 50, 50, 5~ |  |  |
| ## $ F\_shift\_var\_max | <dbl> | 1057581, 1057581, 1057581, 1057581, |
| 1057581, ~ |  |  |
| ## $ F\_shift\_var\_index | <dbl> | 43, 43, 43, 43, 43, 43, 43, 43, 43, |
| 43, 43, 4~ |  |  |
| ## $ F\_shift\_kl\_max | <dbl> | 2.08871, 2.08871, 2.08871, 2.08871, |
| 2.08871, ~ |  |  |
| ## $ F\_shift\_kl\_index | <dbl> | 49, 49, 49, 49, 49, 49, 49, 49, 49, |
| 49, 49, 4~ |  |  |
| ## $ F\_spectral\_entropy | <dbl> | 0.7173186, 0.7173186, 0.7173186, |
| 0.7173186, 0~ |  |  |
| ## $ F\_n\_crossing\_points | <int> | 18, 18, 18, 18, 18, 18, 18, 18, 18, |
| 18, 18, 1~ |  |  |
| ## $ F\_longest\_flat\_spot | <int> | 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, |
| 4, 4, 4, ~ |  |  |
| ## $ F\_stat\_arch\_lm | <dbl> | 0.5968071, 0.5968071, 0.5968071, |

0.5968071, 0~

# Splitting

The full dataset was split into

1. Training (to\_train) and future prediction (to\_predictfuture) dataset
2. The training dataset (to\_train) was further split into the analysis/training and assessment/testing sets.

# Datasets to train and predict

to\_train<- full %>% filter(!is.na(Admission)) to\_predictfuture<- full %>% filter(is.na(Admission))

# Spiltting `to\_train` 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.

# Pre-processing recipes

When using modeltime for forecasting, the date column is treated differently for each approach.

Classical: The date column is left untoched.

Machine learning: Features are engineered from the date column e.g. using

as.numeric(date) or in this case with step\_timeseries\_signature.

The date column is then

1. either dropped. In the case of using glmnet, the date format cannot be implicitly converted into numeric format for the needed matrix structure used in glmnet
2. or its role as a predictor is updated to something else.

4 combinations of predictors and pre-processing step\_s were screened to determine the best combination of features for machine learning.

1. Basic (rec\_basic)
   1. Lags (Admission\_lag10)
   2. Rolling lags (Admission\_lag10\_roll\_???)
   3. Covid peak period (Covid)
   4. Relevant features engineered from step\_timeseries\_signature
   5. Hierarchical levels (Level)
   6. Members in the corresponding level (Name)
2. Basic + Time series features and statistics (rec\_ft)
   1. Above rec\_basic
   2. Time series features. (F\_???)
3. Basic + PCA of the time series features and statistics (rec\_PC)
   1. Above rec\_basic
   2. The first 5 principal components of the time series features. 5 components were shown to summarised 88% of the variance.
4. Basic + kernel PCA of the time series and features and statistics (rec\_kPC)
   1. Above rec\_basic
   2. The first 5 principal components of the time series features. Kernel PCA would be more suitable if there were a non-linear relationship between the time series features and the number of admissions.

## Pre-processing order

Pre-processing tend to follow this order:

* + 1. Impute
    2. Handle Factor levels
    3. Individual transformations
    4. Discretize
    5. Lump minority observations
    6. Create dummy variables
    7. Create interactions
    8. Remove variables with near zero variance
    9. Normalize
    10. Create splines / Multivariate transformation (e.g. PCA, spatial)

fun\_rec<- function(R){ LHS= "Admission ~"

RHS = paste(R, collapse = "+") formula<- as.formula(paste0(LHS, RHS))

recipe(formula, data = training(splits)) %>% update\_role(c(row\_id, Date), new\_role = "id") %>% step\_timeseries\_signature(Date) %>%

step\_rm(matches("(.xts$)|(.iso$)|(hour)|(minute)|(second)|(am.pm)| (mweek)")) %>%

step\_rm(starts\_with("Date\_wday")) %>% step\_rm(starts\_with("Date\_ mday")) %>% step\_rm(Date\_day) %>%

step\_dummy(all\_nominal(), one\_hot = TRUE)}

col\_lag<- to\_train %>% select(starts\_with("Admission\_lag")) %>% colnames()

col\_bare<- c("Level", "Name", "row\_id", "Date", "Covid") col\_ft<- to\_train %>% select(starts\_with("F\_")) %>% colnames()

rec\_basic<-fun\_rec(c(col\_bare, col\_lag))

rec\_ft<-fun\_rec(c(col\_bare, col\_lag, col\_ft)) %>% step\_corr(!!col\_ft) %>% step\_nzv(all\_numeric\_predictors())

rec\_PC<- fun\_rec(c(col\_bare, col\_lag, col\_ft)) %>% step\_normalize(!!col\_ft) %>%

step\_pca(!!col\_ft)

rec\_kpca<- fun\_rec(c(col\_bare, col\_lag, col\_ft)) %>% step\_normalize(!!col\_ft) %>%

step\_kpca\_rbf(!!col\_ft)

# Modelling

Random Forest was used as the model to screen the combinations of features as Random Forest tends to do relatively well without hyperparameter tuning. It has also been used in the M4 competition with success where time series features and statistics were predictors.

mod<-rand\_forest(trees= 1000) %>% # increase from 500->100 set\_engine("ranger", verbose = TRUE) %>% # use ranger as faster,

increased trees set\_mode("regression")

## Workflow

The multiple combination of recipes was easily handed with workflowsets. (wfsets <- workflowsets::workflow\_set(

preproc = list(basic = rec\_basic, ft= rec\_ft, PC= rec\_PC, kPC=rec\_kpca),

models = list(rf=mod), cross = F))

## # A workflow set/tibble: 4 x 4

## wflow\_id info option result

## <chr> <list> <list> <list>

## 1 basic\_rf <tibble [1 x 4]> <wrkflw > <list [0]>

## 2 ft\_rf <tibble [1 x 4]> <wrkflw > <list [0]>

## 3 PC\_rf <tibble [1 x 4]> <wrkflw > <list [0]>

## 4 kPC\_rf <tibble [1 x 4]> <wrkflw > <list [0]>

# Evaluate

## 6.1 Evaluate against the training set

modeltime\_fit\_workflowset only described the models used. It did not not carry forward the wflow\_id.

(wfset\_table<-modeltime\_fit\_workflowset(wfsets, training(splits))) ## 2021-06-12 14:05:33: Calculating kernel PCA

## 2021-06-12 14:05:33: Trying to calculate reverse ## 2021-06-12 14:05:33: DONE

## # Modeltime Table ## # A tibble: 4 x 3

## .model\_id .model .model\_desc ## <int> <list> <chr>

## 1 1 <workflow> RANGER

## 2 2 <workflow> RANGER

## 3 3 <workflow> RANGER

## 4 4 <workflow> RANGER

Before the models with various features were evaluated, the .model\_descriptions were updated to reflect the recipes of the various features and models used. The models from

modeltime\_fit\_workflowset were arranged in the alphabetical order of the wflow\_id

which made updating of .model\_descriptions easier.

(wfset\_table<-wfset\_table %>% update\_model\_description(1, "rec\_basic") %>%

update\_model\_description(2, "rec\_ft") %>% update\_model\_description(3, "rec\_kPC") %>% update\_model\_description(4, "rec\_PC"))

## # Modeltime Table ## # A tibble: 4 x 3

## .model\_id .model .model\_desc ## <int> <list> <chr>

## 1 1 <workflow> rec\_basic

## 2 2 <workflow> rec\_ft

## 3 3 <workflow> rec\_kPC

## 4 4 <workflow> rec\_PC

Alternatively, one could refer to the recipe inside .model to infer the relevant feature combination

wfset\_table$.model[[1]]

## == Workflow [trained] ==============================

============================

## Preprocessor: Recipe ## Model: rand\_forest() ##

## -- Preprocessor - -

## 6 Recipe Steps ##

## \* step\_timeseries\_signature() ## \* step\_rm()

## \* step\_rm() ## \* step\_rm() ## \* step\_rm()

## \* step\_dummy() ##

## -- Model

-

## Ranger result ##

## Call:

## ranger::ranger(x = maybe\_data\_frame(x), y = y, num.trees = ~1000, verbose = ~TRUE, num.threads = 1, seed = sample.int(10^5, 1)) ##

## Type: Regression

## Number of trees: 1000

## Sample size: 624

## Number of independent variables: 44

## Mtry: 6

## Target node size: 5

## Variable importance mode: none

## Splitrule: variance

## OOB prediction error (MSE): 286072.1

## R squared (OOB): 0.9949897

Renaming could have been skipped with this very new package as it specifically deals with

workflowset for modeltime.

The models were calibrated on the training set.

(wfset\_calibrate<-wfset\_table %>% modeltime\_calibrate(training( splits)))

## # Modeltime Table ## # A tibble: 4 x 5

## .model\_id .model .model\_desc .type .calibration\_data ## <int> <list> <chr> <chr> <list>

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## | 1 | 1 | <workflow> | rec\_basic | Test | <tibble | [624 | x | 4]> |
| ## | 2 | 2 | <workflow> | rec\_ft | Test | <tibble | [624 | x | 4]> |
| ## | 3 | 3 | <workflow> | rec\_kPC | Test | <tibble | [624 | x | 4]> |
| ## | 4 | 4 | <workflow> | rec\_PC | Test | <tibble | [624 | x | 4]> |

### What’s inside the calibrated table

The .calibration\_data from modeltime\_calibrate contains the .acutal number of admissions, the .prediction and the .residuals.

# just one example wfset\_calibrate$.calibration\_data[[1]] ## # A tibble: 624 x 4

## Date .actual .prediction .residuals ## <date> <dbl> <dbl> <dbl>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ## | 1 | 2016-01-01 | 8035 | 8145. | -110. |
| ## | 2 | 2016-01-01 | 7861 | 8152. | -291. |
| ## | 3 | 2016-01-01 | 10659 | 10554. | 105. |
| ## | 4 | 2016-01-01 | 75 | 245. | -170. |
| ## | 5 | 2016-01-01 | 3977 | 3967. | 10.5 |
| ## | 6 | 2016-01-01 | 3214 | 3366. | -152. |
| ## | 7 | 2016-01-01 | 2944 | 3142. | -198. |
| ## | 8 | 2016-01-01 | 4842 | 4863. | -21.4 |
| ## | 9 | 2016-01-01 | 6682 | 6606. | 75.9 |
| ## | 10 | 2016-01-01 | 0 | 202. | -202. |

## # ... with 614 more rows modeltime\_accuracy(object=wfset\_calibrate,

metric\_set = metric\_set(rmse,mae)) %>% arrange(rmse, sort=T)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ##  ## | # | A tibble:  .model\_id | 4 x 5  .model\_desc | .type | rmse | mae |
| ## |  | <int> | <chr> | <chr> | <dbl> | <dbl> |
| ## | 1 | 4 | rec\_PC | Test | 264. | 151. |
| ## | 2 | 3 | rec\_kPC | Test | 266. | 151. |
| ## | 3 | 1 | rec\_basic | Test | 277. | 161. |
| ## | 4 | 2 | rec\_ft | Test | 314. | 182. |

## 6.1 Evaluate with cross validation

To increase the robustness of the accuracy, cross validation was executed. The cross validation scores were different than without cross validation.

set.seed(69)

folds <- vfold\_cv(training(splits), strata = Admission)

set.seed(69) (wfsets\_done<- wfsets %>%

workflow\_map("fit\_resamples", resamples = folds)) ## 2021-06-12 14:06:06: Calculating kernel PCA

## 2021-06-12 14:06:07: Trying to calculate reverse ## 2021-06-12 14:06:07: DONE

## 2021-06-12 14:06:08: Calculating kernel PCA

## 2021-06-12 14:06:08: Trying to calculate reverse ## 2021-06-12 14:06:08: DONE

## 2021-06-12 14:06:10: Calculating kernel PCA

## 2021-06-12 14:06:10: Trying to calculate reverse ## 2021-06-12 14:06:10: DONE

## 2021-06-12 14:06:11: Calculating kernel PCA

## 2021-06-12 14:06:11: Trying to calculate reverse ## 2021-06-12 14:06:11: DONE

## 2021-06-12 14:06:12: Calculating kernel PCA

## 2021-06-12 14:06:13: Trying to calculate reverse ## 2021-06-12 14:06:13: DONE

## 2021-06-12 14:06:14: Calculating kernel PCA

## 2021-06-12 14:06:14: Trying to calculate reverse ## 2021-06-12 14:06:14: DONE

## 2021-06-12 14:06:15: Calculating kernel PCA

## 2021-06-12 14:06:16: Trying to calculate reverse ## 2021-06-12 14:06:16: DONE

## 2021-06-12 14:06:17: Calculating kernel PCA

## 2021-06-12 14:06:17: Trying to calculate reverse ## 2021-06-12 14:06:17: DONE

## 2021-06-12 14:06:18: Calculating kernel PCA

## 2021-06-12 14:06:18: Trying to calculate reverse ## 2021-06-12 14:06:19: DONE

## 2021-06-12 14:06:20: Calculating kernel PCA

## 2021-06-12 14:06:20: Trying to calculate reverse ## 2021-06-12 14:06:20: DONE

## # A workflow set/tibble: 4 x 4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ## |  | wflow\_id | info |  |  | option | result |
| ## |  | <chr> | <list> |  |  | <list> | <list> |
| ## | 1 | basic\_rf | <tibble | [1 | x | 4]> <wrkflw > | <rsmp[+]> |
| ## | 2 | ft\_rf | <tibble | [1 | x | 4]> <wrkflw > | <rsmp[+]> |
| ## | 3 | PC\_rf | <tibble | [1 | x | 4]> <wrkflw > | <rsmp[+]> |
| ## | 4 | kPC\_rf | <tibble | [1 | x | 4]> <wrkflw > | <rsmp[+]> |

collect\_metrics(wfsets\_done, summarize = T ) %>% # summarised all folds filter(.metric=="rmse") %>%

select(wflow\_id, avg\_rmse=mean) %>% arrange(avg\_rmse,sort=T) ## # A tibble: 4 x 2

## wflow\_id avg\_rmse ## <chr> <dbl>

|  |  |  |  |
| --- | --- | --- | --- |
| ## | 1 | PC\_rf | 514. |
| ## | 2 | kPC\_rf | 516. |
| ## | 3 | basic\_rf | 526. |
| ## | 4 | ft\_rf | 543. |

# 8 Conclusion

The cross validation scores were different than without cross validation.The bottom 2 performing feature combinations were consistent (rec\_basic & rec\_ft). However, when the time series features were condensed into principal components the accuracy improved. The relationship between the time series features and the number of admissions was linear-ish as the performance between kernel PCA and PCA were close for evaluations with and without cross validation. For the marginal difference in rmse and shorter computational time, PCA was selected. The feature combination for machine learning forecasting in the next few posts would be

1. Lags
2. Rolling lags
3. Covid peak period
4. Relevant features engineered from step\_timeseries\_signature
5. The first 5 principal components (PCA) of the time series features
6. Hierarchical levels and their members

# save the dataset for machine learning and the dataset for future prediction; save the recipe

save(to\_train, to\_predictfuture, rec\_PC, file= "4Dataset4ML.rds")