Assignment 4 DATA423-20S1

Zhen Huang

5/22/2020

Question 1-Caret

Overview

Caret The `caret` package which described, by its first author/maintainer Max Kuhn, as a unified interface of predictive models. To be specific, it provides a set of tools (functions and classes) that attempt to streamline the process for creating classification and regression models.[1] The package contains tools such as data splitting, pre-processing and resampling, feature selection, model tuning, performance evaluation, models comparison, visualization for the models and the other functionality.

Python's scikit-learn Scikit-learn is an open-source machine learning library in Python that supports supervised and unsupervised learning. Similar to `caret`, it also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.[2]

R's mlr3 package One of the design principles of `mlr3` is focusing on computation. Together, the `mlr3` package and its ecosystem provide a generic framework for machine learning tasks for the R language. Similar to `caret` wrapping function `caret::train`, `mlr3` also provide a unified interface to many popular machine learing algorithms in R, which are objects of class `mlr3::learner`. [3]

Compare and Contrast the Frameworks

Tasks coverage The general data science tasks were taken into account in a comparison of functions coverage, which is shown in the below table[4].

Tasks Coverage	`caret`	`scikit-learn`	`mlr3`
Visualisation	*	`matplot`	`viz`
Pre-processing	* or work with `recipe`	* work with `numpy`	`pipelines`
Data spliting	*	*	`pipelines`
Standardised training/predicting	wrapping	*	wrapping
Feature engineering/selection	*	*	`filter`,`fswrap`
Method categorisation	*		
Resamplling	*	*	*
Hyperparameter tuning	*	*	`tuning`,`hyperband`,`mbo`,`paradox`
Model evaluation	*	*	`measures`
Model selection	*	*	
Parallel Processing	*	*	*

[1] The caret Package http://topepo.github.io/caret/index.html
[2] scikit-learn.org https://scikit-learn.org/stable/getting_started.html
[3] mlr3 Manual https://mlr3book.mlr-org.com/introduction.html
[4] *(asterisk) means functions or classes integrated in the package

Except for method categorisation and model selection, all three packages can handle most of the tasks in predicting models. There is much difference in details worthing to discuss, such as the way the packages achieving the tasks and specific functions or classes within these packages.

For the **visualisation** task, `caret` offers its own functions which suitable to the specific model, `scikit-learn` leaves this part of the job to `matplot` and `mlr3` offers through its extension package `viz`. This kind of extension function makes up the `mlr3` ecosystem, which is designed by following the principles in the `mlr3`.

It is critical for predicting a model by implementing an efficient and reasonable **pre-processing** step dealing with missing data, variable encoding and various feature transformations. Although three packages have their own tools, it is usually a good practice working together with other packages, such as `recipe` and `caret`, `numpy` and `scikit-learn`. There is a similar idea of `pipelines` in `mlr3` and `scikit-learn`, furthermore, `pipelines` in `mlr3` offers a more complicated structure by networking the pre-processing steps together in `Graph`, which is not only a visualisation for the inputs and outputs but making it possible to connect the steps of training/predicting/hyperparameter tuning with pre-processing together.

Both `caret` and `mlr3` wrap the methods of other packages in R, while `scikit-learn` implements a method in itself. So, depending packages need to be loaded when training/predicting models in `caret` and `mlr3` explicitly or implicitly.

Both `caret` and `scikit-learn` offer a range of options in feature selection/extraction/transformation/composition whatever they called in the packages. At the same time `mlr3` has less choice at the present and new extension `mlr3fselect` in an experiment.

Similar situations show in the tasks of resampling, hyperparameter tuning, model evaluation and model selection when comparing three packages in these tasks.

There is a very well **method categorisation** in `caret` in terms of both the quality of documents and helping tools.

Parallel Processing is a critical ability when facing a computational problem or a big dataset when the strategies to scale computationally, such as decreasing the data size, is not enough. There is much difference in the three packages. Basically, `caret` and `scikit-learn` utilize the other backend packages in R and Python, respectively, to leverage the parallel processing framework [5] [6]. These type of parallelism is based on using multiple CPU cores in a local machine. `mlr3` uses the `future` backends for parallelisation [7] which is different with `caret`. `future` enables `mlr3` has the ability to run on one or more local or remote machines. `spark-sklearn` is another attemp to extending `scikit-learn` in Python world focuses on problems that have a small amount of data, and that can be run in parallel [8] in Spark's distributed computational framework.

Models A simple comparison with the number of methods in three packages is shown below. [9] [10] [11]

	`caret`	`scikit-learn`	`mlr3`
Number of methods	238	140	51

The very newcomer is `mlr3`, launched from 2019, which supports fewer models than the other two packages. Under these simple numbers, `caret` focuses on the classification and regression methods, just as its acronym. Note that only `scikit-learn` supports cluster methods at present, while `mlr3` is going to achieve this through its extension package `mlr3cluster` in progress. So if we are facing an unsupervised clustering problem or anomoly detection, `scikit-learn` is the most appropriate choice with its nearest neighbour and novelty and

[5] https://topepo.github.io/caret/parallel-processing.html
[6] https://scikit-learn.org/stable/modules/computing.html#computational-performance
[7] https://mlr3book.mlr-org.com/parallelization.html#parallelization
[8] https://pypi.org/project/spark-sklearn/
[9] http://topepo.github.io/caret/available-models.html
[10] https://scikit-learn.org/stable/user_guide.html
[11] https://mlr3book.mlr-org.com/list-learners.html

outlier detection methods.

Strengths and weaknesses From the comparison above we have a basic knowledge of three packages. Here below are the summary and some additional discussion in their strengths/weaknesses and limitations.

For `caret`, the wide range of choices in classification and regression methods is the most outstanding strength compared with the other two packages. The ability of processing data and predicting model is limited by physical components in the local machine, which may be a bottleneck of the predicting in facing big data. In addition, there is less flexible in building a customed deep learning neuron network, which limits the ability of `care` in handling with image/audio/video segmentation/classification problem.

For `scikit-learn`, the diversity of its methods option is one of the advantages, such as clustering and outliers detection method. Scaling and parallelism limitation from the local machine also exist in `scikit-learn`. However, with the background of Python, `scikit-learn` can extend its application to Spark and empower the ability of distributed processing. As the changing interest in the python open source community, more contributes now toward PyTorch, Dash, which face the modern data science problems. This trend leads `scikit-learn` will focus on improving as an external components, which can be integrated into these frameworks. [12]

The first release of `mlr` is in 2013, while the rewritten `mlr3` has the ambition to build an ecosystem with its extendible structure including modern features in R including R6 objects, `future` backend and learners suitable to build neuron networks, such as `mlr3keras` in experimental.

Question 2 - Report assessment (Marks: 1/3)

Procedure followed

Missing and imputation

1. Providing the proportion of missing data, however, further investigation should be taken to determine whether there is a potential pattern of missing data in specific variables, observations , or just missing at random.
2. The imputation was ahead of the training/test split and on all the data. This imputation would lead to "data leakage", mainly, when missing data is not at random.
3. The method of imputation was not presented in the report. The possibility of omitting the missing data was not discussed in the report, whether omitting would be better than imputation?

Outliers

4. The criteria detecting outliers were not discussed in the report. Was it 1.5 IQR?
5. What about the other predictors? Whether there was a pattern of observations in multivariable?
6. Outliers deletion can only occur in the case that outliers will not occur in the future unseen data. Whether the observations were removed automatically or manually?

[12] https://scikit-learn.org/dev/roadmap.html

Feature engineering

7. An interaction terms "BodyMassIndex" was added without showing the explanation and examination of the relationship between "Weight" and "Height". Although this is intuitive, some exception might exist in the dataset.
8. "Net_productivity" is a combination of three variables whose relationship to each other should also be examed. Although related variables removed from predictors later, the "Sales", "Returned_sales" might relate to "Sales_value" and "Returned_sales_value". High "Sales" or low "Returned_sales" would be the effect of "Net_productivity". Thus, the potential data leakage might exist in predicting the model.
9. Feature engineering includes feature extraction and feature selection. What about the correlation of predictors? If there is a relationship between numeric predictors, PCA can be used to extract new feature variables. If there is a relationship between outcome and predictor, PLS can be used.

Role assignment

10. More investigation could be done when assigning the roles of variables. As the description of variables, "Branch" is a globally unique branch name, thus, "Branch"+"Employee ID" could be a unique observation identifier. Then "Country" might be kept as a predictor in this study because there was no other variable related to certain Nationalities that was part of the confirmation in this study.

(There may have a typo in the role assignment at "Company" which does not exist in the dataset.)

Feature Selection

11. The "Weight" and "Height" were kept in the predictors, while they consisted the interaction variable "BodyMassIndex". There was no discussion for the selection in the study. Whether "BodyMassIndex" is essential, and whether the interpretability changed, if "Weight" and "Height" were removed.
12. "Shift_length" was kept in the predictor, however, it was one of the variable consisted of the outcome variable. The outcome consisted of one predictor would lead to data leakage as mentioned above.

Standardising

13. Standardising before the training/test split potentially leaks information from the test to train dataset.

Converting/Encoding

14. `model.matrix` converted nominal variables into binary dummy-variables for the dataset. One more intercept column added also was added into the dataset which was suitable for a linear regression model, however, neither intercept column nor dummy-variables are necessary for a random forest model. There was no explanation whether this transformation was used in training the random forest model.
15. "Education level" is an ordinal predictor, which should not be encoding by binary dummy-variables.

Resampling

16. A train-validation-test split could be a better alternative than simple random split into training/test.
17. The potential risk of near-zero variance was not examed in the study, so a one-time train/test split was running the risk of NZV.
18. A supervised stratified split would be a better alternative, if a categorical variable proved to be an appropriate group data role.

Modelling candidates

19. There is no explanation about why choosing these as candidate methods. For example, a better practice can be reporting a list of available methods for the problem and shows how to do the filtering.

20. Hyperparameters chosen for the methods; however, there was one set of hyperparameter in GLMnet and Neural network methods, neither evidence nor the process were shown in the report of how these hyperparameters were chosen. For KNN and Random Forest, two different sets were shown without evidence in proving which one was better.

21. Neither the results nor the process of training was reported, if tuning grid was used, was there a problem of the hyperparameter at the edge of the grid?

22. Neural network is a generic name of a batch of methods/algorithms; the specific method should be reported.

Conclusion

21. As the resampling method was a one-time train/test split in the study, using test RMSE to choose the best model exclusively was not the best practice. Using one of cross-validation resampling method would be an improvement.

22. The test results of the other candidate methods were not reported. If the test RMSE of one method was close to the best model, and it has an advantage of interpretability over neuron network, the model was also worth to report.

23. The residuals had not to be examed whether there was a pattern which suggests the model had not explained some other information in the dataset.

24. The statistics of the outcome can be compared with the RMSE to show whether the model predicted sensible values.

25. No guarantee of the model always performs the best application. For example, there was no evidence that the model would be tolerant of the novelty of categorical data or missing data in the future unseen data. Broadly, there are two ways a model can not always performs the best (i.e. model decay), previously unseen variety of predictor data and new categories; the interpretation of the outcome changes with time while the general distribution of the predictor data doesnot.[13]

26. There was a suggestion to deploy the predictive model, however no evidence was shown that the model would tolerant outliers, novelities and missing value in future unseen data.

27. Without a monitoring plan, it is hard to guarentee the model always performs the best application.

Other problems in the report

28. Monitoring employees without their awareness and extracting personel files to the dataset were ethic problems in this study.

29. The economic statuses vary from different country; thus, it was not a fair comparison in the whole dataset without stratified or grouped employees.

30. Gender and religion also related to a certain group of employees which should be considered in the study, such as stratified resampling or fitting models for each group.

[13] Lecture note of 'Model Lifecycle'

Question 3 - Quality Control (Marks: 1/3)

```
library(qicharts2)
library(tidyverse)
data <- read_csv("/Users/neil/Workspace/Repositories/Rcode/DSI/Ass4/monitor.csv")
```

Load data Investing data structure and summary. The columns are loaded:

Column	Type	Description
Timestamp	datetime	the timestamp of a model prediction being run
ProcessMemory	double	the allocated memory (MB) of the relevant server process
Prediction	double	the value predicted by the model
PredictionTimeMS	double	the duration of the prediction task in milliseconds

```
str(data) # data structure
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 10000 obs. of  4 variables:
##  $ Timestamp       : POSIXct, format: "2019-01-01 13:14:17" "2019-01-01 13:30:41" ...
##  $ ProcessMemory   : num  12.4 12.1 12.5 12.6 11.3 ...
##  $ Prediction      : num  42.2 57 66.1 58.4 43.2 ...
##  $ PredictionTimeMS: num  232 433 357 303 252 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Timestamp = col_datetime(format = ""),
##   ..   ProcessMemory = col_double(),
##   ..   Prediction = col_double(),
##   ..   PredictionTimeMS = col_double()
##   .. )
```

```
summary(data) # distribution
```

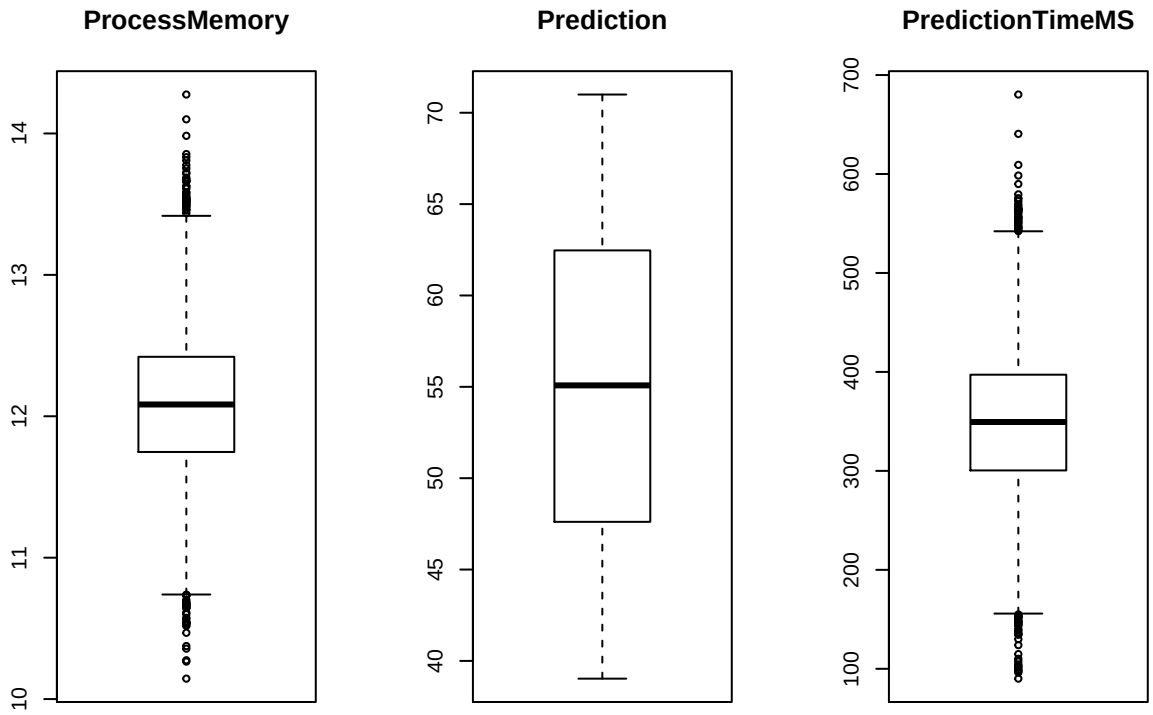
```
##    Timestamp                     ProcessMemory     Prediction    PredictionTimeMS
##  Min.   :2019-01-01 13:14:17   Min.   :10.14   Min.   :39.03   Min.   : 90.06
##  1st Qu.:2019-01-30 05:07:54   1st Qu.:11.75   1st Qu.:47.61   1st Qu.:300.44
##  Median :2019-02-19 16:55:26   Median :12.08   Median :55.08   Median :349.40
##  Mean   :2019-02-18 21:20:39   Mean   :12.08   Mean   :55.05   Mean   :349.27
##  3rd Qu.:2019-03-12 18:10:37   3rd Qu.:12.42   3rd Qu.:62.47   3rd Qu.:397.19
##  Max.   :2019-04-01 12:41:24   Max.   :14.28   Max.   :71.00   Max.   :680.30
```

```
data %>% is.na %>% apply(2, sum) # simple missing check
```

```
##         Timestamp     ProcessMemory        Prediction PredictionTimeMS
##                 0                 0                 0                 0
```

Visualisation of variable distributions and potential outliers.

```
par(mfrow=c(1,3))
data$ProcessMemory %>% boxplot(main = "ProcessMemory")
data$Prediction %>% boxplot(main = "Prediction")
data$PredictionTimeMS %>% boxplot(main = "PredictionTimeMS")
```



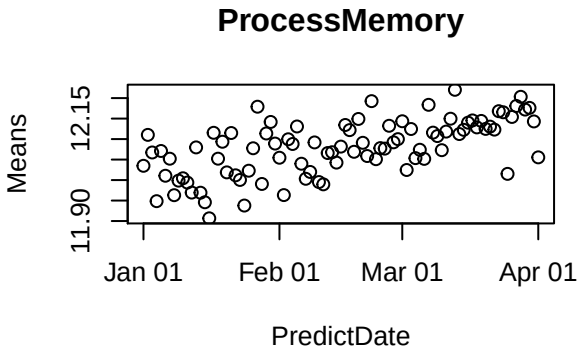
Creating a new column for grouping per day.

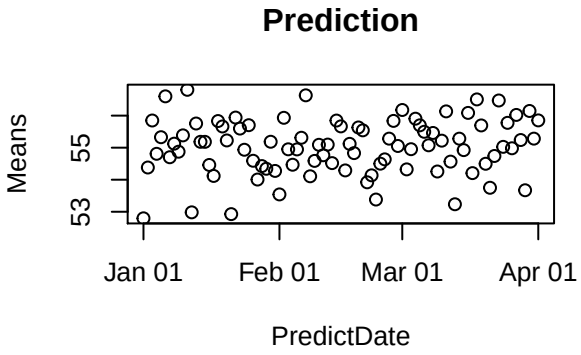
```
attach(data)
agg_data <- data %>%
  mutate(PredictDate=(as.Date(Timestamp)))

agg_data %>% distinct(PredictDate) %>% arrange(PredictDate) %>% head(20) %>% top_n(1)
```

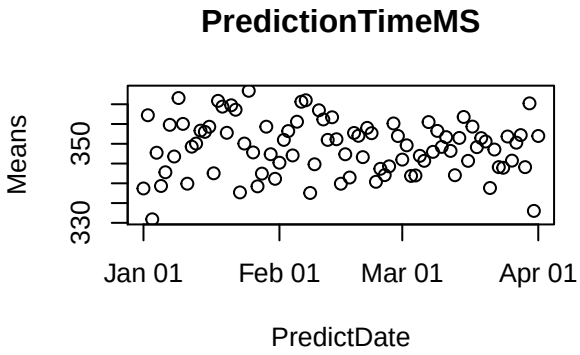
```
## # A tibble: 1 x 1
##   PredictDate
##   <date>
## 1 2019-01-20
```

Getting a intuitive about the variables.

```
par(mfrow=c(2,2))
agg_data %>% group_by(PredictDate) %>% summarise(Means = mean(ProcessMemory)) %>% plot(main = "ProcessM
agg_data %>% group_by(PredictDate) %>% summarise(Means = mean(Prediction)) %>% plot(main = "Prediction"
agg_data %>% group_by(PredictDate) %>% summarise(Means = mean(PredictionTimeMS)) %>% plot(main = "Predi
```





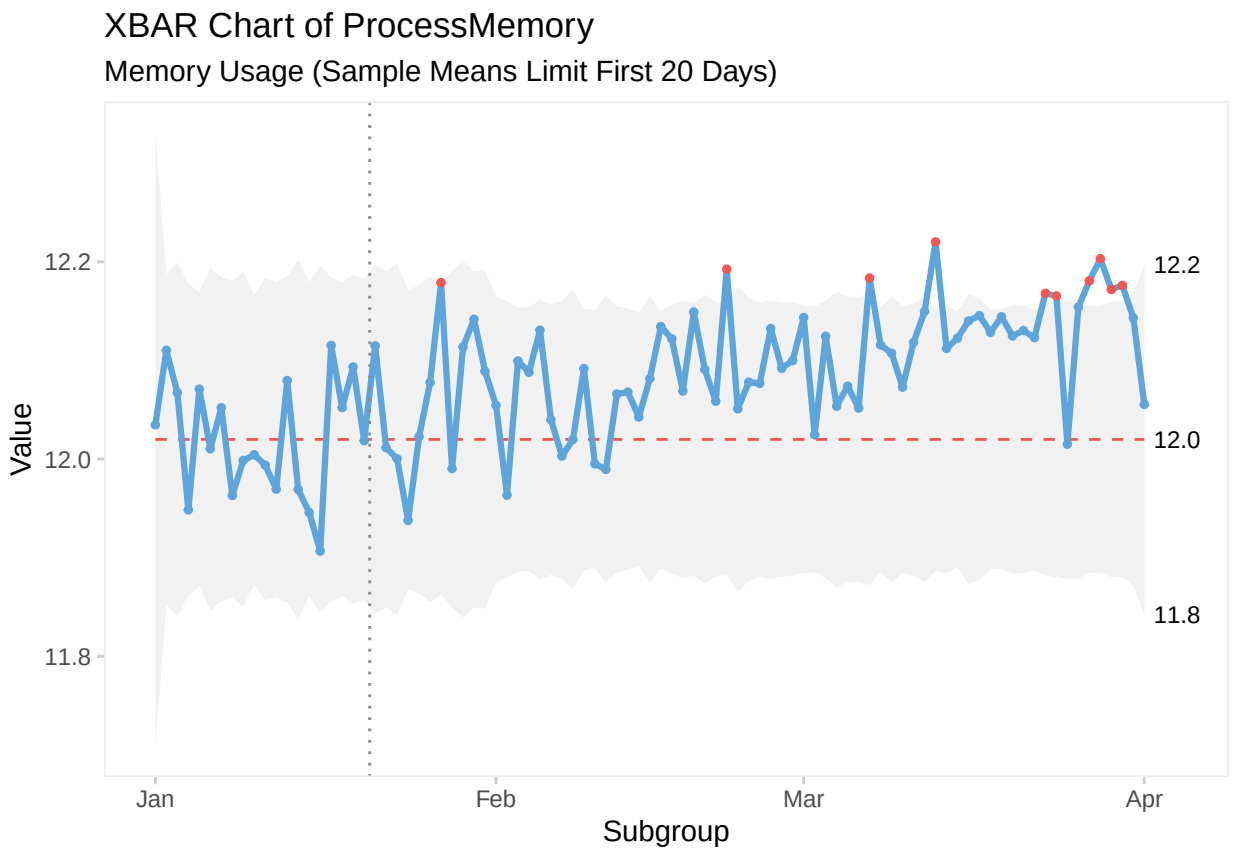


a) Is the memory usage of the server in control? Answer:

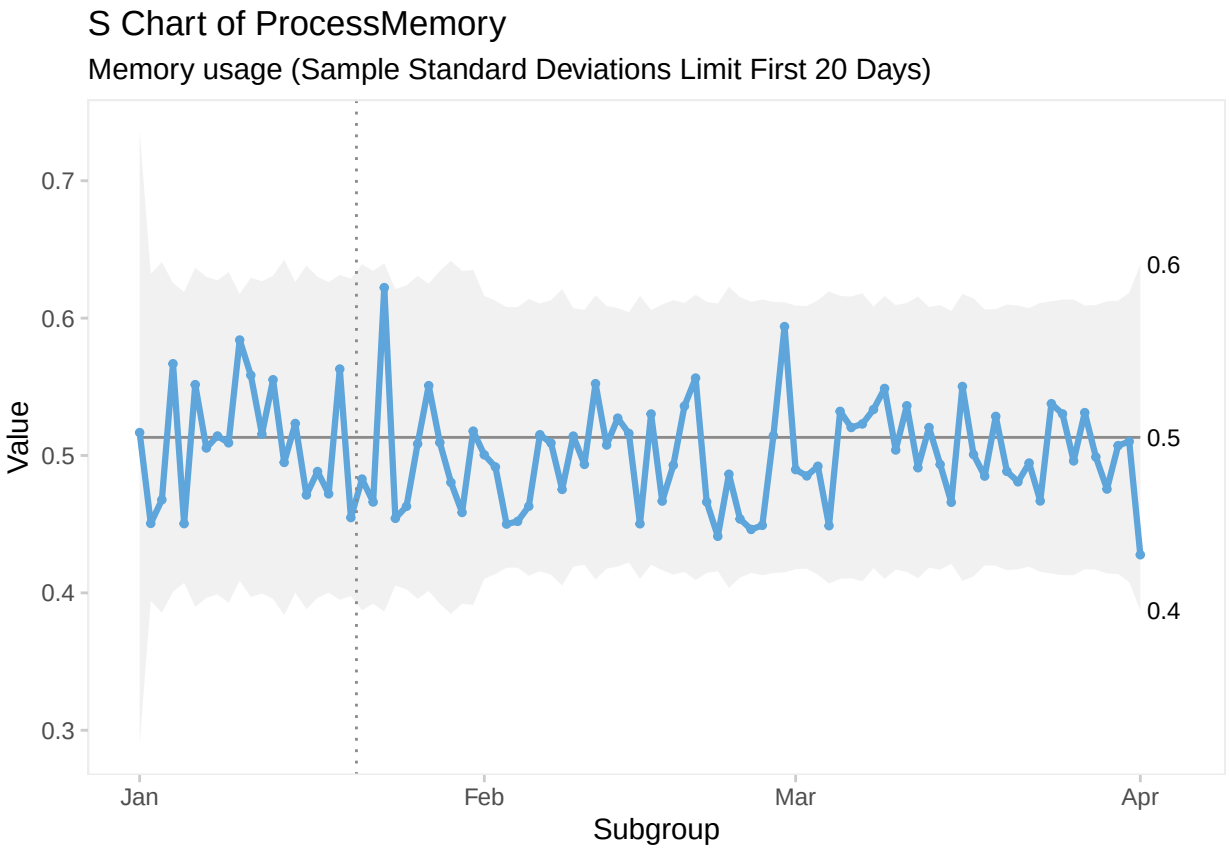
No, the memory usage is not in control with respect to the xbar chart.

- The longest.run breaches the longest.run.max. The "run" failure is signaled by runs.signal at 1. There are consecutive process points without crossing the mean line in a period from Feburary to April shown on the xbar chart.
- There are nine red points shown on the xbar chart and sigmar.signal is 9 indicates breaches identified by Shewhart's 3-sigma rule.

```
par(mfrow=c(1,1))
p_xbar <- qicharts2::qic(x = PredictDate, y = ProcessMemory, data = agg_data, chart = "xbar", freeze = 
p_std <- qicharts2::qic(x = PredictDate, y = ProcessMemory, data = agg_data, chart = "s", freeze = 20, 
plot(p_xbar)
```


XBAR Chart of ProcessMemory
Memory Usage (Sample Means Limit First 20 Days)
Value
12.2
12.0
11.8
12.2
12.0
11.8
Jan
Feb
Mar
Apr
Subgroup

```
plot(p_std)
```




```
df_summ <- bind_rows(summary(p_xbar), summary(p_std))
rownames(df_summ) <- c("xbar","s")

df_summ %>% select(longest.run, longest.run.max,n.crossings,n.crossings.min,runs.signal,sigma.signal)
```

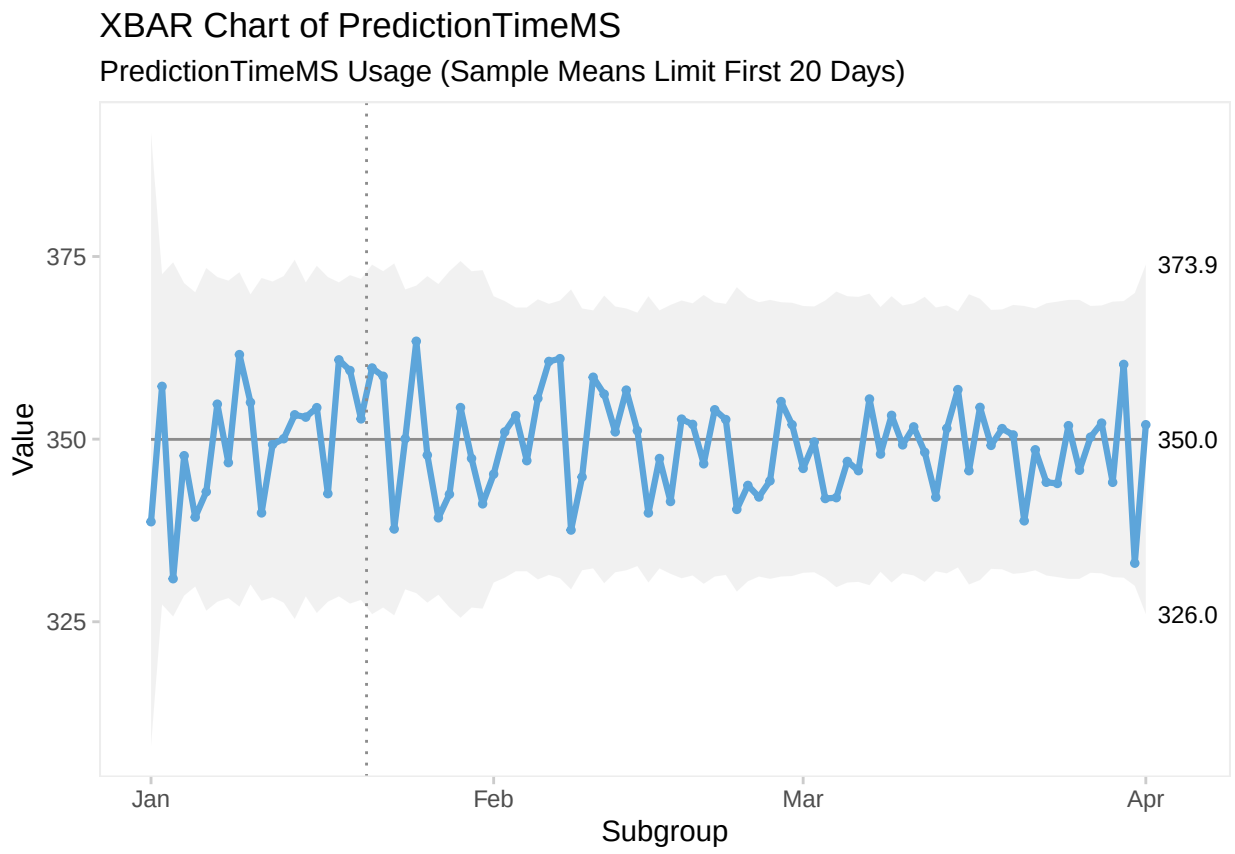
```
##      longest.run longest.run.max n.crossings n.crossings.min runs.signal
## xbar          41              10          22              37           1
## s              6              10          47              37           0
##      sigma.signal
## xbar           10
## s               0
```

b) Is the prediction time of the model in control? Answer:

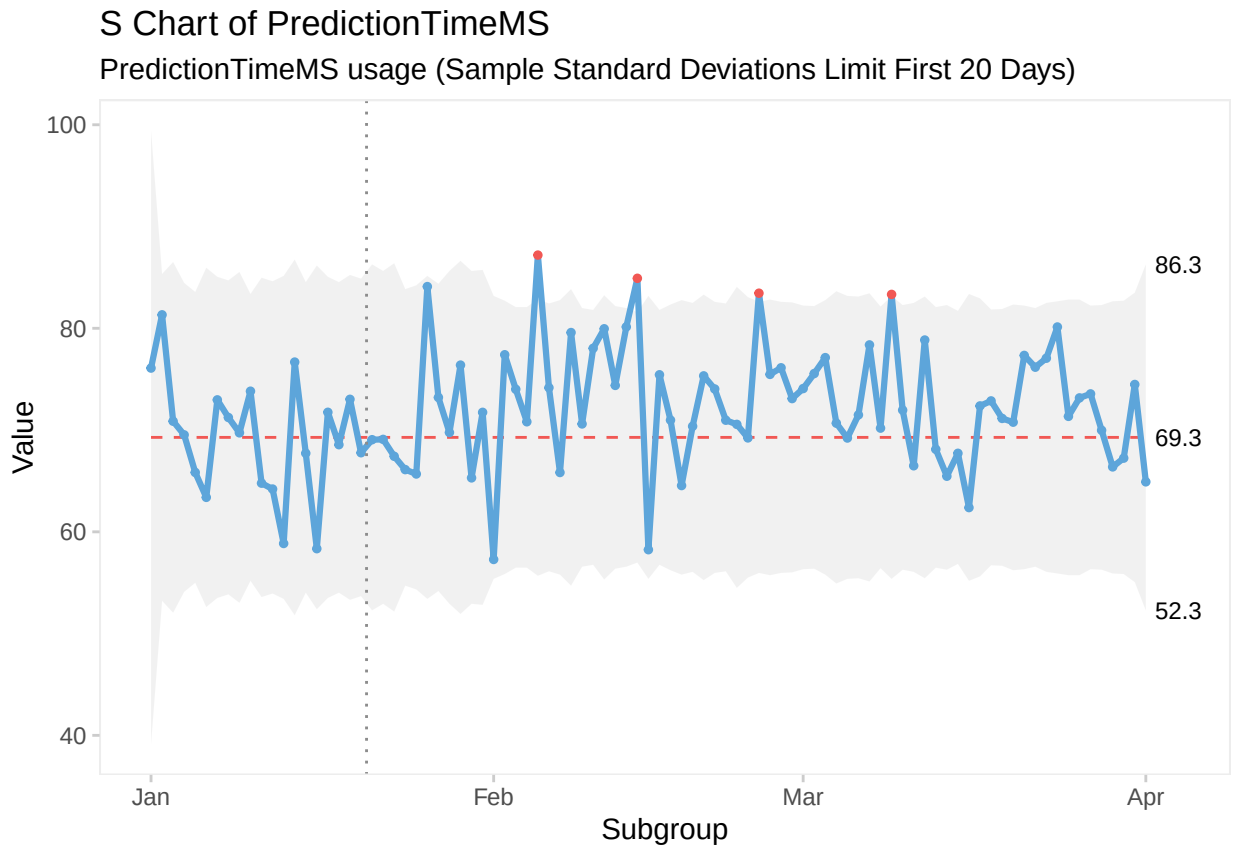
No, the prediction time is not in control with respect to the s chart.

- The longest.run at 12 breaches the longest.run.max at 10. The "run" failure is signaled by runs.signal at 1.
- The n.crossings at 31 breaches the n.crossing.min at 37. There is unusually few crossing the of the process points. The "run" failure is signaled by runs.signal at 1.
- There are four red points shown on the s chart and sigmar.signal is 4 indicates breaches.

```
p_xbar <- qicharts2::qic(x = PredictDate, y = PredictionTimeMS, data = agg_data, chart = "xbar", freeze
p_std <- qicharts2::qic(x = PredictDate, y = PredictionTimeMS, data = agg_data, chart = "s", freeze = 2
plot(p_xbar)
```


XBAR Chart of PredictionTimeMS
PredictionTimeMS Usage (Sample Means Limit First 20 Days)
375
350
325
373.9
350.0
326.0
Value
Jan
Feb
Mar
Apr
Subgroup

```
plot(p_std)
```




```
df_summ <- bind_rows(summary(p_xbar), summary(p_std))
rownames(df_summ) <- c("xbar","s")

df_summ %>% select(longest.run, longest.run.max,n.crossings,n.crossings.min,runs.signal,sigma.signal)
```

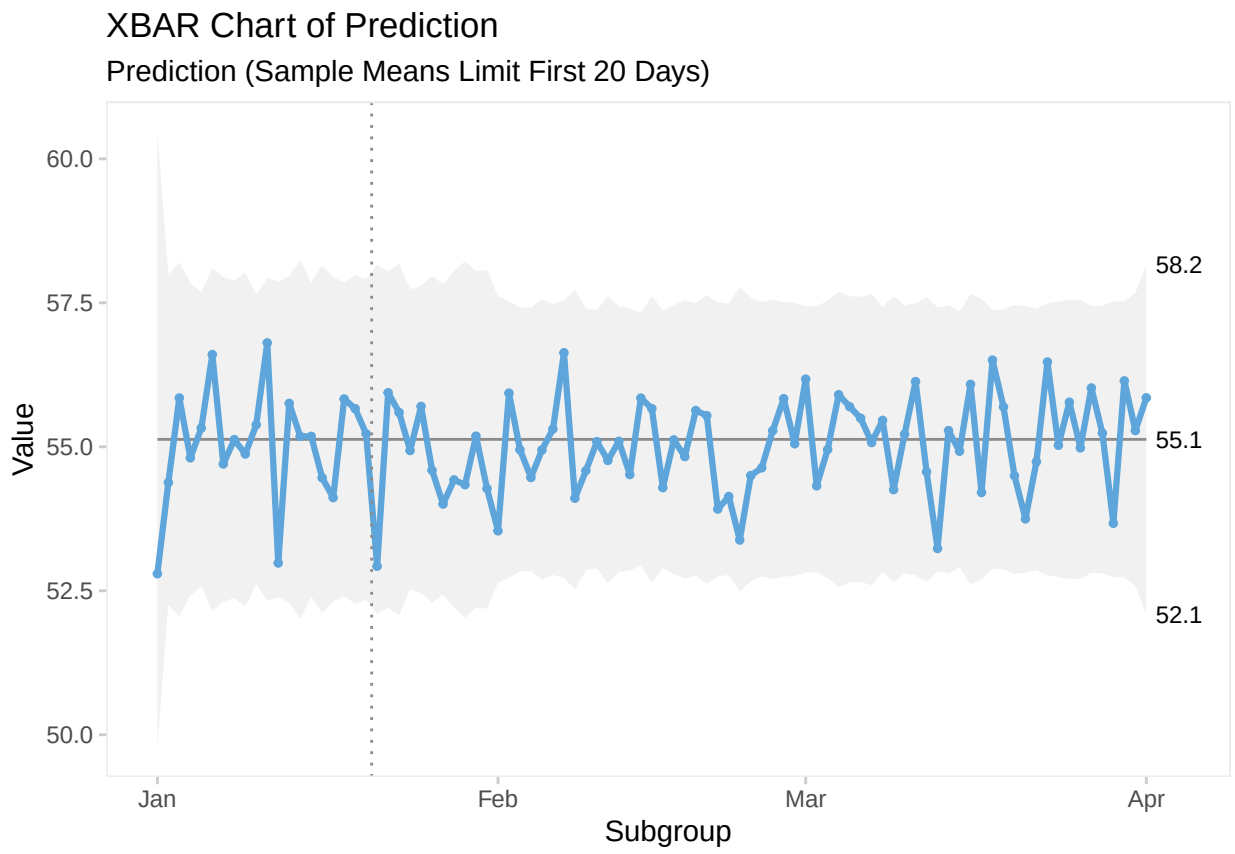
```
##      longest.run longest.run.max n.crossings n.crossings.min runs.signal
## xbar           6              10          45              37           0
## s             12              10          31              37           1
##      sigma.signal
## xbar            0
## s               4
```

c) Is the stream of predictions in control? Answer:

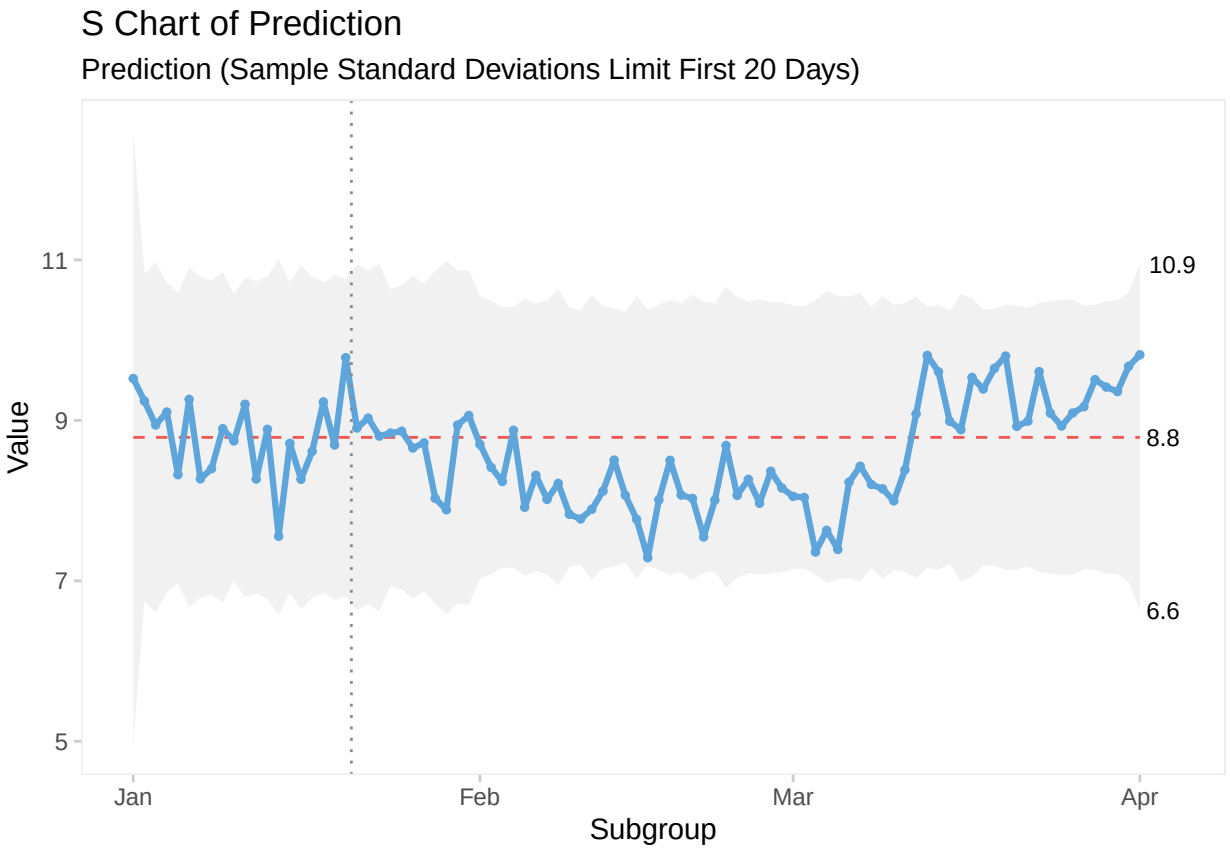
No, the stream of prediction is not in control with respect to the s chart.

- The longest.run at 35 breaches the longest.run.max at 10. The "run" failure is signaled by runs.signal at 1.
- The n.crossings at 18 breaches the n.crossing.min at 37. There is unusually few crossing the of the process points. The "run" failure is signaled by runs.signal at 1.

```
p_xbar <- qicharts2::qic(x = PredictDate, y = Prediction, data = agg_data, chart = "xbar", freeze = 20,
p_std <- qicharts2::qic(x = PredictDate, y = Prediction, data = agg_data, chart = "s", freeze = 20, sub
plot(p_xbar)
```


XBAR Chart of Prediction
Prediction (Sample Means Limit First 20 Days)
60.0
57.5
55.0
52.5
50.0
Value
58.2
55.1
52.1
Jan
Feb
Mar
Apr
Subgroup

```
plot(p_std)
```




```
df_summ <- bind_rows(summary(p_xbar), summary(p_std))
rownames(df_summ) <- c("xbar","s")

df_summ %>% select(longest.run, longest.run.max,n.crossings,n.crossings.min,runs.signal,sigma.signal)
```

```
##      longest.run longest.run.max n.crossings n.crossings.min runs.signal
## xbar           6              10          47              37           0
## s             35              10          18              37           1
##      sigma.signal
## xbar            0
## s               0
```