Overcoming the computational demands of time series:

Scaling R-based demand forecasting with RapidMiner

2/12/2020

Strategy and Insights
Global Center of Excellence

Goal

Provide Supply Chain with highly accurate, highly scalable food demand forecasts

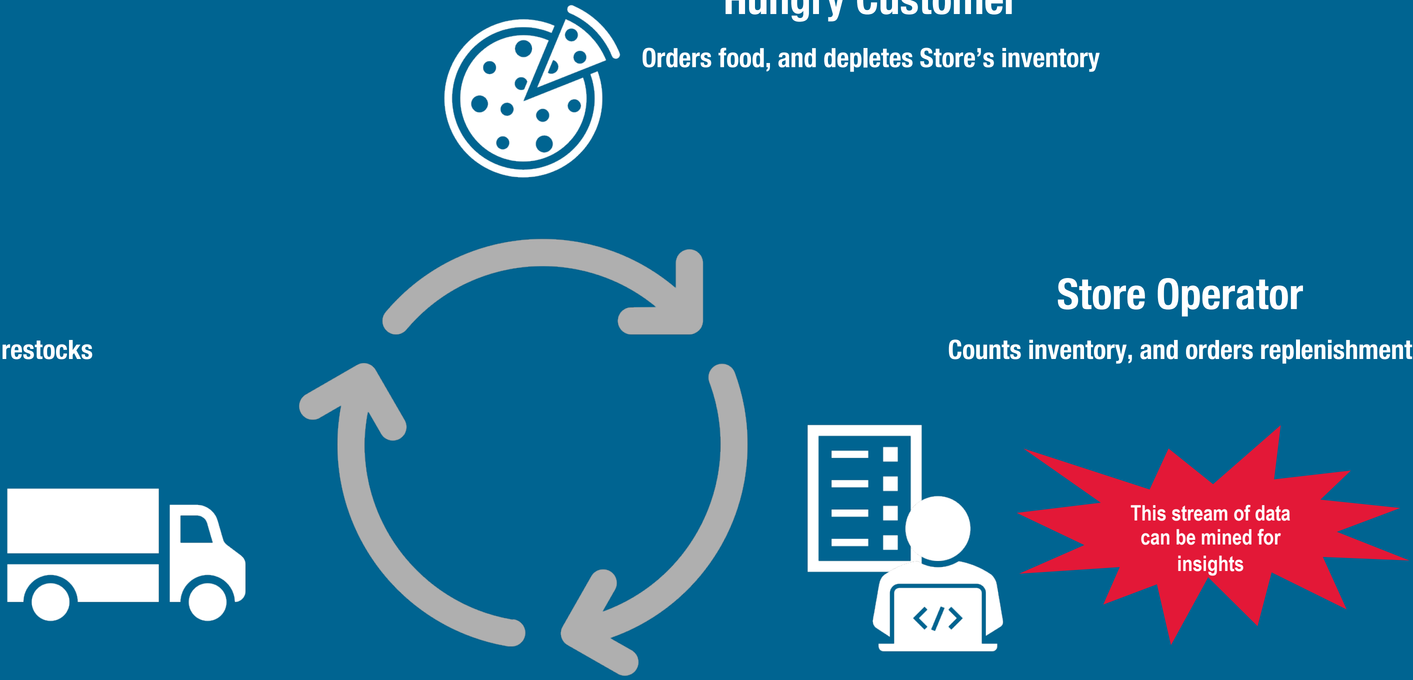
Problem

Shared resources limited; ecosystem of projects expanding rapidly

Solution

Make extensible open source time series forecasting tool; think creatively to keep footprint small

Store Inventory Lifecycle



Hungry Customer

Orders food, and depletes Store's inventory

Store Operator

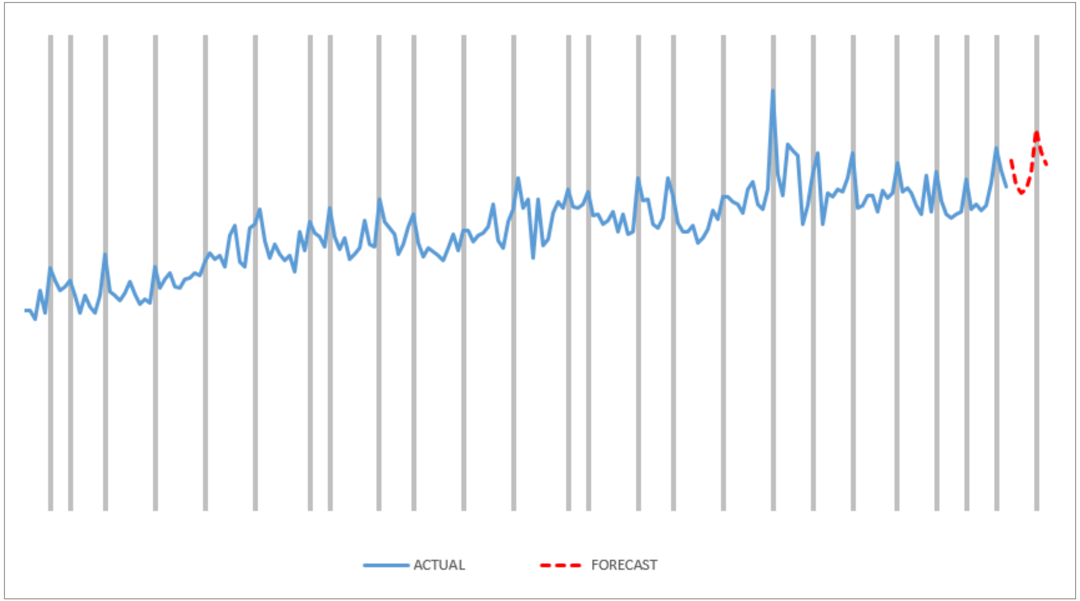
Counts inventory, and orders replenishment

This stream of data can be mined for insights

Supply Chain

Fulfills Store's food order, and restocks Store's inventory

Highly Accurate, Highly Scalable Demand Forecasts



Business Value

Improve Supplier Relations

Enable timely, and accurate purchase plan to suppliers

Reduce Food Waste

Avoid food spoilage

Scale Labor to Demand

Avoid idle labor and overtime

Available Resources

50+ Team Members

Many with advanced degrees:

PhD Chemistry, PhD Computer Science, PhD Physics, Masters in Applied Statistics, Masters in Electrical Engineering, Masters in Epidemiology, Masters in Industrial and Operations Engineering

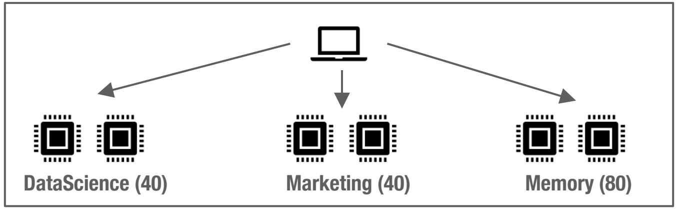
Comprehensive Tech Stack

<u>User Desktop</u>: RapidMiner Studio, R, Python, Jupyter, SSMS

<u>AI/ML</u>: RapidMiner, Jupyterhub, R Studio, Nvidia GPU Server, ArcGIS, Hive, Spark

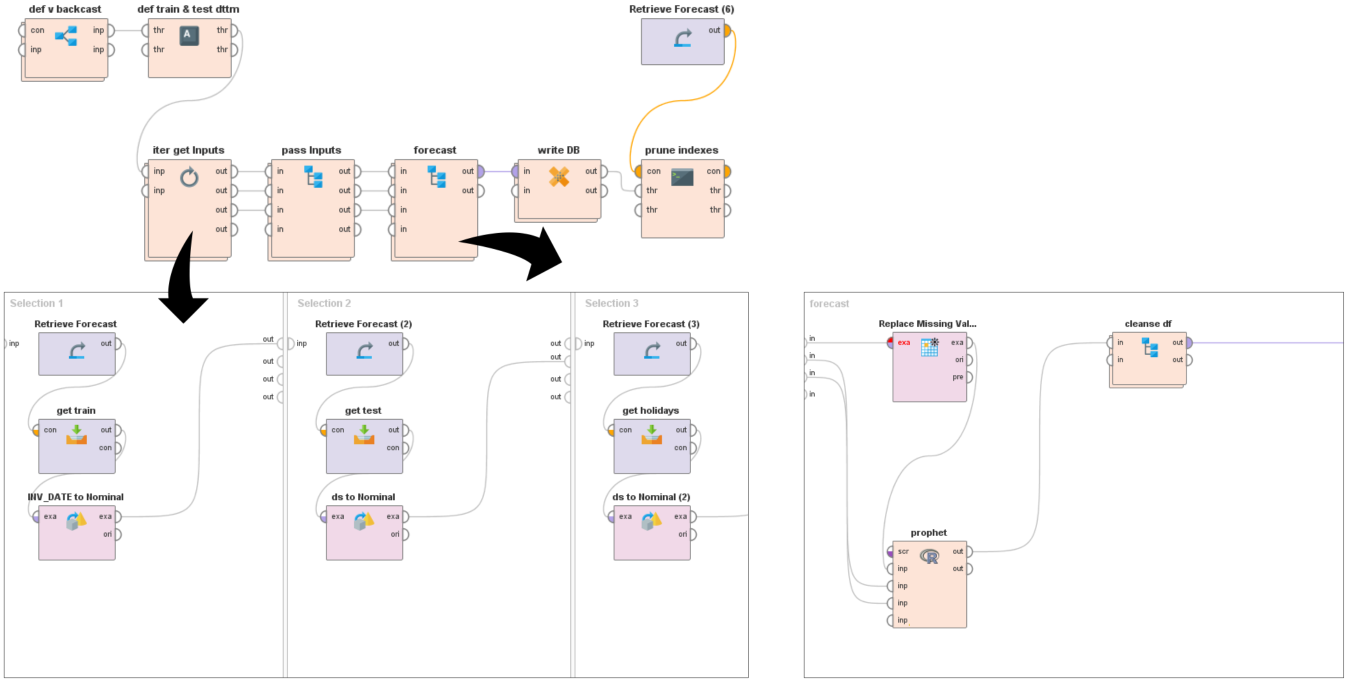
<u>Data Stores</u>: Sql Server, Hadoop

RapidMiner



Prototype

RapidMiner

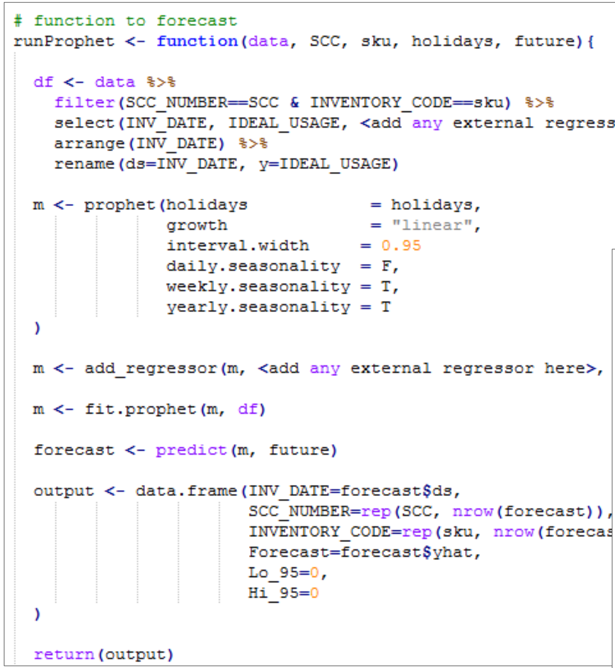


Read demand history, promotion history, planned future promotions, & important holiday dates from database

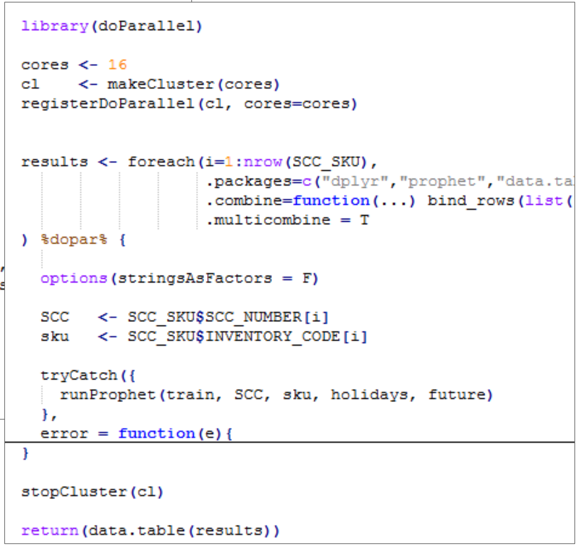
Pass as inputs into the R implementation of Facebook's opensource timeseries **forecast**ing package prophet

Write results into downstream applications

Prototype – (important bits of) the R Script



```
# function to forecast
runProphet <- function(data, SCC, sku, holidays, future){

  df <- data %>%
    filter(SCC_NUMBER==SCC & INVENTORY_CODE==sku) %>%
    select(INV_DATE, IDEAL_USAGE, <add any external regressor here>) %>%
    arrange(INV_DATE) %>%
    rename(ds=INV_DATE, y=IDEAL_USAGE)

  m <- prophet(holidays              = holidays,
               growth                = "linear",
               interval.width       = 0.95
               daily.seasonality  = F,
               weekly.seasonality = T,
               yearly.seasonality = T
  )

  m <- add_regressor(m, <add any external regressor here>,

  m <- fit.prophet(m, df)

  forecast <- predict(m, future)

  output <- data.frame(INV_DATE=forecast$ds,
                       SCC_NUMBER=rep(SCC, nrow(forecast)),
                       INVENTORY_CODE=rep(sku, nrow(forecas
                       Forecast=forecast$yhat,
                       Lo_95=0,
                       Hi_95=0
  )

  return(output)
```



```
library(doParallel)

cores <- 16
cl    <- makeCluster(cores)
registerDoParallel(cl, cores=cores)


results <- foreach(i=1:nrow(SCC_SKU),
                   .packages=c("dplyr","prophet","data.ta
                   .combine=function(...) bind_rows(list(
                   .multicombine = T
) %dopar% {

  options(stringsAsFactors = F)

  SCC   <- SCC_SKU$SCC_NUMBER[i]
  sku   <- SCC_SKU$INVENTORY_CODE[i]

  tryCatch({
    runProphet(train, SCC, sku, holidays, future)
  },
  error = function(e){
}

stopCluster(cl)

return(data.table(results))
```

R-script **receives from RM three inputs** retrieved from sql:

(1) 3-yrs history of demand

(2) Forecasting period (i.e., 8-weeks of future)

(3) List of important holidays

Forecast function **filters to a single scenario** (Supply Chain Center-SKU)

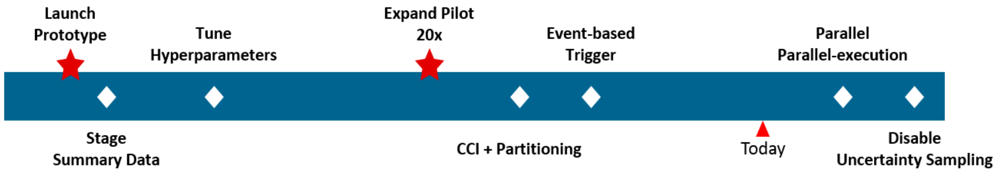
Forecast function defines prophet model, **fits it, & forecasts demand**, by week, for the next 8-week period

Forecast function is wrapped with in a doParallel process to use 16 cores, concurrently

https://facebook.github.io/prophet/

https://github.com/facebook/prophet

Timeline of Enhancements



Launch Prototype

Stage Summary Data

Tune Hyperparameters

Expand Pilot 20x

CCI + Partitioning

Event-based Trigger

Today

Parallel Parallel-execution

Disable Uncertainty Sampling

Launch Prototype

- □ 1 VM (single queue)
- □ 200 forecasts
- □ 8-hour run-time

Stage Inputs

- □ 7+ hours run-time for training history SELECT
- □ Remove cursor-based outlier replacement
- □ Remove self-joins, replace with windowing function
- □ Perform history aggregation once, save for later use

- □ 1 VM (single queue)
- □ 200 forecasts
- □ 15-minute run-time

Hyperparameters

- □ Thesis: single set of hyperparameters exists with performance better than default
- □ Use Grid Search + Bayesian Optimization
- □ Parallelize Grid Search over 6 VMs
- □ 10-hours elapsed vs. 60-hours

- □ MAPE improved from 6.5% to 6.23%

Expand Workload

- □ Stable results encouraged business
- □ Added SKUs & Supply Chain Centers
- □ **Data Volume & Compute needed expanded 20x**

- □ 1 VM (single queue)
- □ 4,000 forecasts
- □ 8+ hour run-time
- □ Staged Inputs db footprint > 150 GB

CCI

- □ Replaced Heap + Nonclustered Index with Clustered Columnstore Index (CCI)

- □ Staged Inputs db footprint ~ 5 GB

Event-based

- □ Time-based launch leaves slack in system

- □ Event-based trigger runs instantly after all dependents complete

Parallel, Parallel

- □ Use all available VMs to run mutually exclusive segments of workload

---

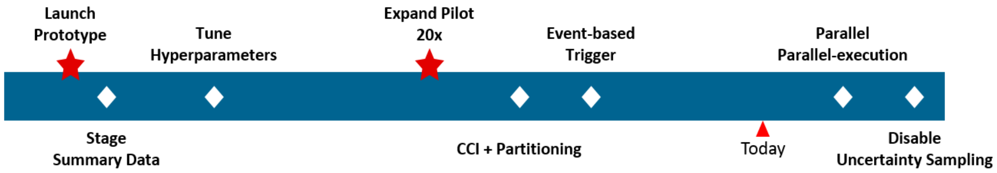
- □ 6 VM (3 queues)
- □ 4,000 forecasts
- □ 1.3 hours run-time

Uncertainty Off

- □ Disable Prophet's Uncertainty Intervals

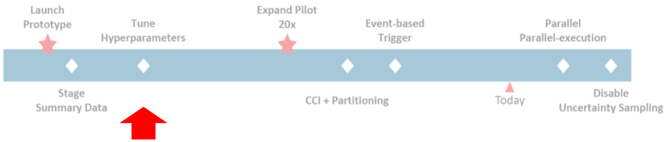
- □ 6 VM (3 queues)
- □ 4,000 forecasts
- □ 27-minute run-time

Tune Hyperparameters

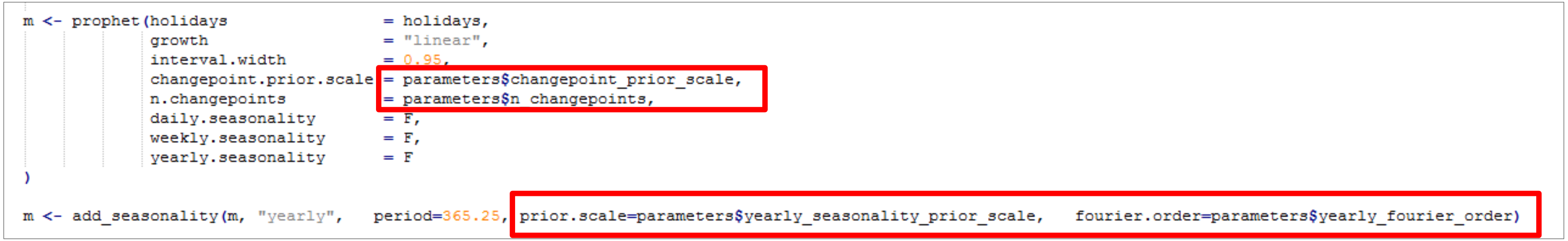


Launch Prototype	Stage Inputs	Hyperparameters	Expand Workload	CCI	Event-based	Parallel, Parallel	Uncertainty Off
□ 1 VM (single queue) □ 200 forecasts □ 8-hour run-time	□ 7+ hours run-time for training history SELECT □ Remove cursor-based outlier replacement □ Remove self-joins, replace with windowing function □ Perform history aggregation once, save for later use	□ Thesis: single set of hyperparameters exists with performance better than default □ Use Grid Search + Bayesian Optimization □ Parallelize Grid Search over 6 VMs □ 10-hours elapsed vs. 60-hours	□ Stable results encouraged business □ Added SKUs & Supply Chain Centers □ **Data Volume & Compute needed expanded 20x**	□ Replaced Heap + Nonclustered Index with Clustered Columnstore Index (CCI)	□ Time-based launch leaves slack in system	□ Use all available VMs to run mutually exclusive segments of workload	□ Disable Prophet's Uncertainty Intervals
	□ 1 VM (single queue) □ 200 forecasts □ 15-minute run-time	□ MAPE improved from 6.5% to 6.23%	□ 1 VM (single queue) □ 4,000 forecasts □ 8+ hour run-time □ Staged Inputs db footprint > 150 GB	□ Staged Inputs db footprint ~ 5 GB	□ Event-based trigger runs instantly after all dependents complete	□ 6 VM (3 queues) □ 4,000 forecasts □ 1.3 hours run-time	□ 6 VM (3 queues) □ 4,000 forecasts □ 27-minute run-time

Tune Hyperparameters: Grid Search

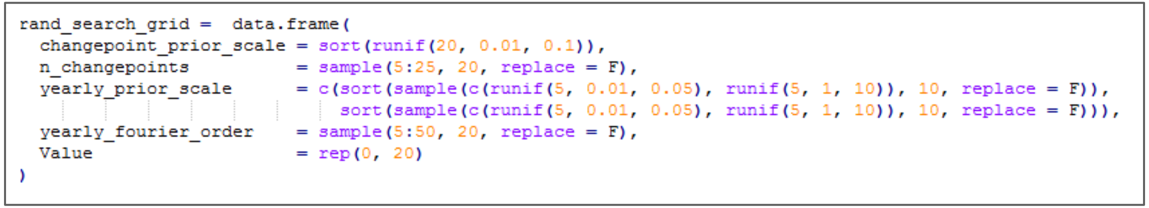


Parameterize the forecast function



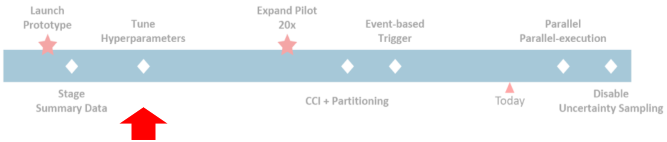
```
m <- prophet(holidays                = holidays,
             growth                  = "linear",
             interval.width          = 0.95,
             changepoint.prior.scale = parameters$changepoint_prior_scale,
             n.changepoints          = parameters$n_changepoints,
             daily.seasonality       = F,
             weekly.seasonality      = F,
             yearly.seasonality      = F
)

m <- add_seasonality(m, "yearly",   period=365.25, prior.scale=parameters$yearly_seasonality_prior_scale,   fourier.order=parameters$yearly_fourier_order)
```

Pass the new function some random values

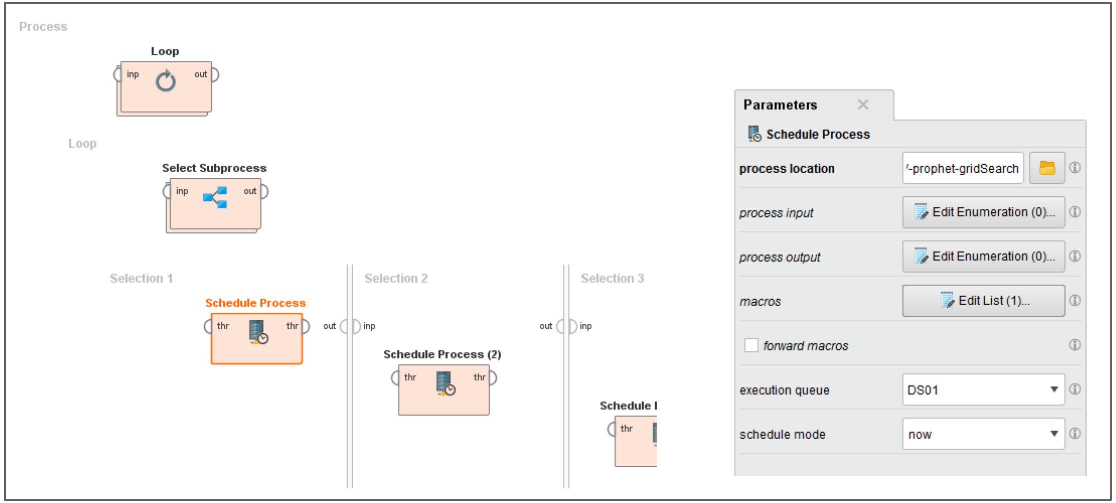


```
rand_search_grid =  data.frame(
  changepoint_prior_scale = sort(runif(20, 0.01, 0.1)),
  n_changepoints          = sample(5:25, 20, replace = F),
  yearly_prior_scale      = c(sort(sample(c(runif(5, 0.01, 0.05), runif(5, 1, 10)), 10, replace = F)),
                              sort(sample(c(runif(5, 0.01, 0.05), runif(5, 1, 10)), 10, replace = F))),
  yearly_fourier_order    = sample(5:50, 20, replace = F),
  Value                   = rep(0, 20)
)
```

Tune Hyperparameters: Parallelize Grid Search

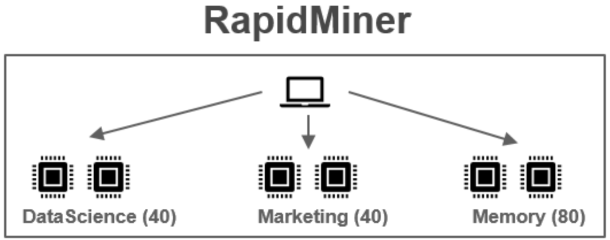


Run 6x Instances of Grid Search concurrently

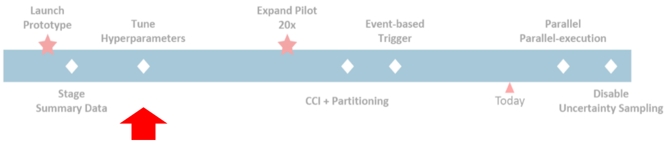


Schedule Process

process location	'-prophet-gridSearch
process input	Edit Enumeration (0)...
process output	Edit Enumeration (0)...
macros	Edit List (1)...
□ *forward macros*	
execution queue	DS01
schedule mode	now



Tune Hyperparameters: Bayesian Optimization



Launch Prototype

Tune Hyperparameters

Expand Pilot 20x

Event-based Trigger

Parallel Parallel-execution

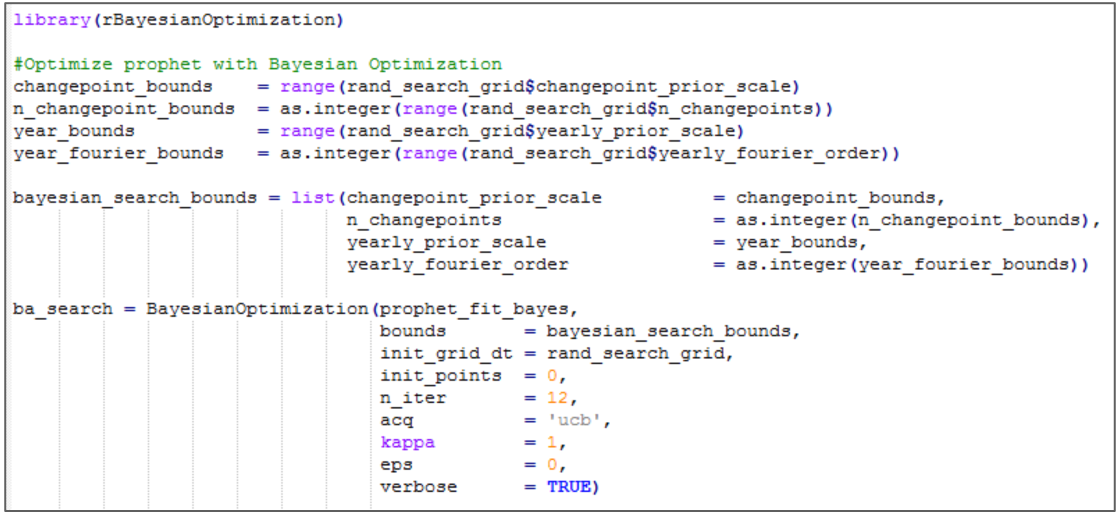
Stage Summary Data

CCI + Partitioning

Today

Disable Uncertainty Sampling

Wrap new forecast function with Bayesian Optimization



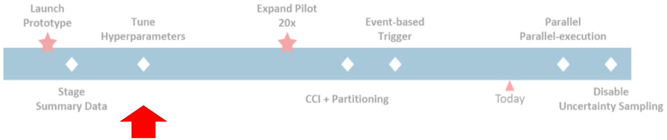
```
library(rBayesianOptimization)

#Optimize prophet with Bayesian Optimization
changepoint_bounds     = range(rand_search_grid$changepoint_prior_scale)
n_changepoint_bounds   = as.integer(range(rand_search_grid$n_changepoints))
year_bounds            = range(rand_search_grid$yearly_prior_scale)
year_fourier_bounds    = as.integer(range(rand_search_grid$yearly_fourier_order))

bayesian_search_bounds = list(changepoint_prior_scale          = changepoint_bounds,
                              n_changepoints                   = as.integer(n_changepoint_bounds),
                              yearly_prior_scale               = year_bounds,
                              yearly_fourier_order             = as.integer(year_fourier_bounds))

ba_search = BayesianOptimization(prophet_fit_bayes,
                                 bounds      = bayesian_search_bounds,
                                 init_grid_dt = rand_search_grid,
                                 init_points  = 0,
                                 n_iter       = 12,
                                 acq          = 'ucb',
                                 kappa        = 1,
                                 eps          = 0,
                                 verbose      = TRUE)
```

Seed Bayesian Optimization with Grid Search results

Search feature-space for global optimal value of the model evaluation metric (MAPE)

Since rBayesianOptimization seeks to maximize the target (MAPE) pass it MAPE x (-1)

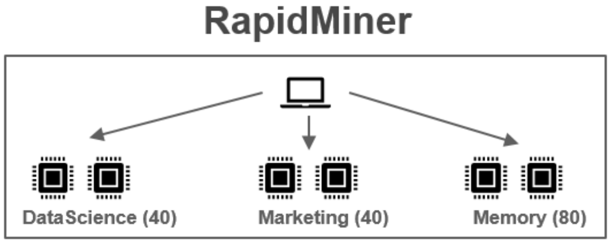
Tune Hyperparameters: Results

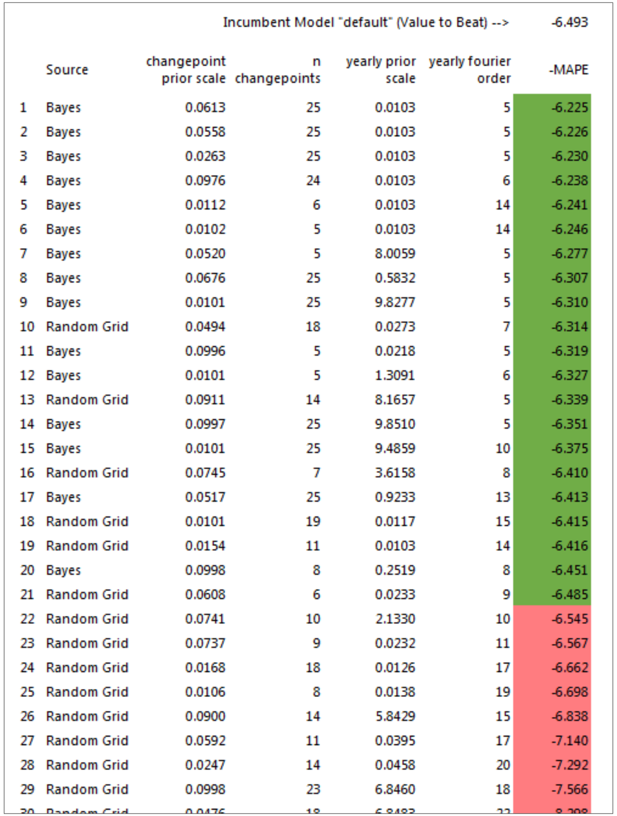


Parameter tuning generated MAPE improvement from 6.5% to 6.23% with negligible change in standard deviation of errors

Grid search with serial execution would have elapsed 60+ hours.

Parallel execution, across RapidMiner queues, on all nodes, took little more than 10-hours

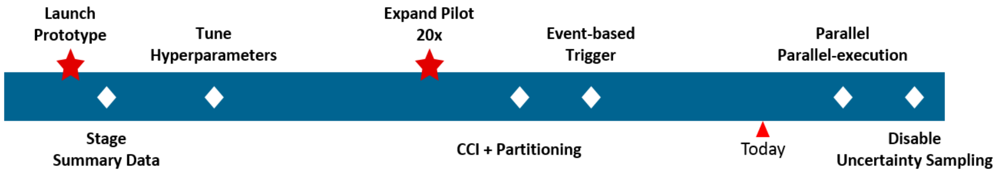




Incumbent Model "default" (Value to Beat) --> -6.493

	Source	changepoint prior scale	n changepoints	yearly prior scale	yearly fourier order	-MAPE
1	Bayes	0.0613	25	0.0103	5	-6.225
2	Bayes	0.0558	25	0.0103	5	-6.226
3	Bayes	0.0263	25	0.0103	5	-6.230
4	Bayes	0.0976	24	0.0103	6	-6.238
5	Bayes	0.0112	6	0.0103	14	-6.241
6	Bayes	0.0102	5	0.0103	14	-6.246
7	Bayes	0.0520	5	8.0059	5	-6.277
8	Bayes	0.0676	25	0.5832	5	-6.307
9	Bayes	0.0101	25	9.8277	5	-6.310
10	Random Grid	0.0494	18	0.0273	7	-6.314
11	Bayes	0.0996	5	0.0218	5	-6.319
12	Bayes	0.0101	5	1.3091	6	-6.327
13	Random Grid	0.0911	14	8.1657	5	-6.339
14	Bayes	0.0997	25	9.8510	5	-6.351
15	Bayes	0.0101	25	9.4859	10	-6.375
16	Random Grid	0.0745	7	3.6158	8	-6.410
17	Bayes	0.0517	25	0.9233	13	-6.413
18	Random Grid	0.0101	19	0.0117	15	-6.415
19	Random Grid	0.0154	11	0.0103	14	-6.416
20	Bayes	0.0998	8	0.2519	8	-6.451
21	Random Grid	0.0608	6	0.0233	9	-6.485
22	Random Grid	0.0741	10	2.1330	10	-6.545
23	Random Grid	0.0737	9	0.0232	11	-6.567
24	Random Grid	0.0168	18	0.0126	17	-6.662
25	Random Grid	0.0106	8	0.0138	19	-6.698
26	Random Grid	0.0900	14	5.8429	15	-6.838
27	Random Grid	0.0592	11	0.0395	17	-7.140
28	Random Grid	0.0247	14	0.0458	20	-7.292
29	Random Grid	0.0998	23	6.8460	18	-7.566
30	Random Grid	0.0476	18	6.8483	22	-8.208

Event-based Trigger



Launch Prototype

- ☐ 1 VM (single queue)
- ☐ 200 forecasts
- ☐ 8-hour run-time

Stage Inputs

- ☐ 7+ hours run-time for training history SELECT
- ☐ Remove cursor-based outlier replacement
- ☐ Remove self-joins, replace with windowing function
- ☐ Perform history aggregation once, save for later use

- ☐ 1 VM (single queue)
- ☐ 200 forecasts
- ☐ 15-minute run-time

Hyperparameters

- ☐ Thesis: single set of hyperparameters exists with performance better than default
- ☐ Use Grid Search + Bayesian Optimization
- ☐ Parallelize Grid Search over 6 VMs
- ☐ 10-hours elapsed vs. 60-hours

- ☐ MAPE improved from 6.5% to 6.23%

Expand Workload

- ☐ Stable results encouraged business
- ☐ Added SKUs & Supply Chain Centers
- ☐ **Data Volume & Compute needed expanded 20x**

- ☐ 1 VM (single queue)
- ☐ 4,000 forecasts
- ☐ 8+ hour run-time
- ☐ Staged Inputs db footprint > 150 GB

CCI

- ☐ Replaced Heap + Nonclustered Index with Clustered Columnstore Index (CCI)

- ☐ Staged Inputs db footprint ~ 5 GB

Event-based

- ☐ Time-based launch leaves slack in system

- ☐ Event-based trigger runs instantly after all dependents complete

Parallel, Parallel

- ☐ Use all available VMs to run mutually exclusive segments of workload

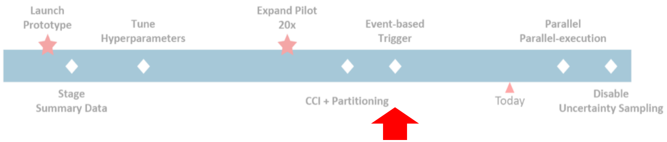
- ☐ 6 VM (3 queues)
- ☐ 4,000 forecasts
- ☐ 1.3 hours run-time

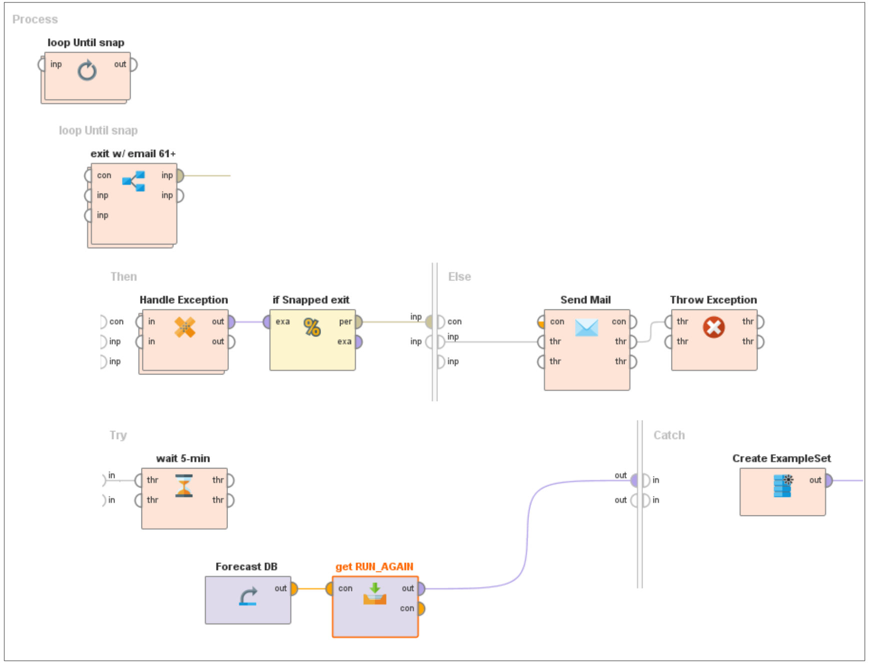
Uncertainty Off

- ☐ Disable Prophet's Uncertainty Intervals

- ☐ 6 VM (3 queues)
- ☐ 4,000 forecasts
- ☐ 27-minute run-time

Event-based Trigger





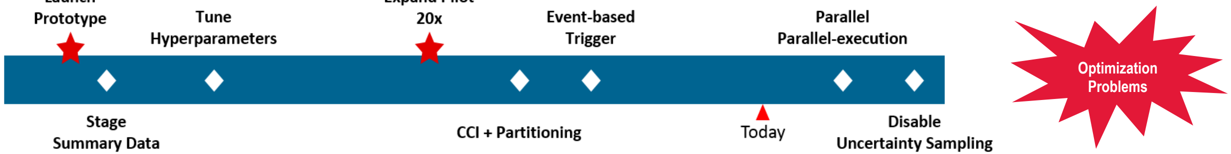
Time-boxed process start can lead to process launch before all dependents are ready, or to lost opportunity to begin ahead of schedule

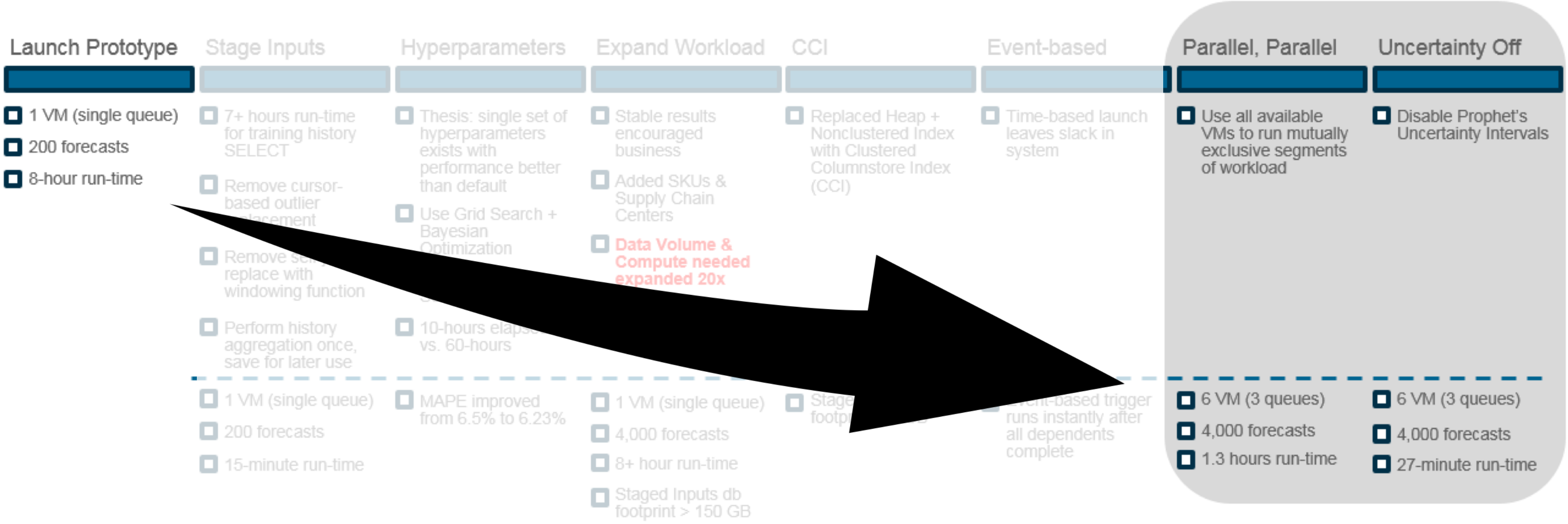
Read-only replicas of EDW databases are "snapped" to the Data Science environment daily, at 4 AM, but exact timing varies

This process checks for "snap" completion, and only then allows the down-stream forecasting process to begin

The Event-based process allows our forecasting model to start "as soon as it can"

What’s Next?





Launch Prototype

- □ 1 VM (single queue)
- □ 200 forecasts
- □ 8-hour run-time

Stage Inputs

- □ 7+ hours run-time for training history SELECT
- □ Remove cursor-based outlier replacement
- □ Remove se replace with windowing function
- □ Perform history aggregation once, save for later use

- □ 1 VM (single queue)
- □ 200 forecasts
- □ 15-minute run-time

Hyperparameters

- □ Thesis: single set of hyperparameters exists with performance better than default
- □ Use Grid Search + Bayesian Optimization
- □ 10-hours elaps vs. 60-hours

- □ MAPE improved from 6.5% to 6.23%

Expand Workload

- □ Stable results encouraged business
- □ Added SKUs & Supply Chain Centers
- □ **Data Volume & Compute needed expanded 20x**

- □ 1 VM (single queue)
- □ 4,000 forecasts
- □ 8+ hour run-time
- □ Staged Inputs db footprint > 150 GB

CCI

- □ Replaced Heap + Nonclustered Index with Clustered Columnstore Index (CCI)

- □ Stage footpr

Event-based

- □ Time-based launch leaves slack in system

- □ ent-based trigger runs instantly after all dependents complete

Parallel, Parallel

- □ Use all available VMs to run mutually exclusive segments of workload

- □ 6 VM (3 queues)
- □ 4,000 forecasts
- □ 1.3 hours run-time

Uncertainty Off

- □ Disable Prophet’s Uncertainty Intervals

- □ 6 VM (3 queues)
- □ 4,000 forecasts
- □ 27-minute run-time

RapidMiner Enabled Success

- Low-code interface
  - ➢ Speedy development
  - ➢ Speedy testing
- Integration of scripting languages
- Orchestration across systems
- Server-side hosting
- Parallel execution
- Event-based process

Goal

Highly accurate, highly scalable demand forecasts

Problem

Shared resources limited; ecosystem of competing projects expanding rapidly

Solution

Creating thinking to keep footprint small

QUESTIONS