HW6-ISYE6501x-Summer2018-OA

OA

6/23/2018

Objectives

In this week, the focus was on prescriptive analytics. We reviewed some probability-based models and techniques, as well as optimization. We also covered some advanced data preparation – what to do when data is missing. In this homework assignment we also learnt how to use simulation software.

Question 13.2 simulation

(done in Arena, appended to this pdf at the end)

Question 14.1

The breast cancer data set breast-cancer-wisconsin.data.txt from http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/ (description at http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29) has missing values. 1. Use the mean/mode imputation method to impute values for the missing data. 2. Use regression to impute values for the missing data. 3. Use regression with perturbation to impute values for the missing data. 4. (Optional) Compare the results and quality of classification models (e.g., SVM, KNN) build using (1) the data sets from questions 1,2,3; (2) the data that remains after data points with missing values are removed; and (3) the data set when a binary variable is introduced to indicate missing values.

```
dataFile <- "breast.cancer.data.txt"
if (!file.exists(dataFile)) {
  #crimeDataURL <- paste0(c("http://www.statsci.org/data/general/uscrime.txt"))
  bcURL <- paste0(c("http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/b

  download.file(bcURL, dataFile) }

breastCancerTable <- read.csv(dataFile, header = FALSE, na.strings = "\\?" )
```

From the website, the details for each column are : Attribute Information:

1. Sample code number: id number
2. Clump Thickness: 1 - 10
3. Uniformity of Cell Size: 1 - 10
4. Uniformity of Cell Shape: 1 - 10
5. Marginal Adhesion: 1 - 10
6. Single Epithelial Cell Size: 1 - 10
7. Bare Nuclei: 1 - 10
8. Bland Chromatin: 1 - 10
9. Normal Nucleoli: 1 - 10
10. Mitoses: 1 - 10
11. Class: (2 for benign, 4 for malignant)

- the above legend helps out craft our game plan:
- *we will do mean() for all values and round them up*

- *we will do mode() for the categorical column 11*
- *we will ofcourse skip column 1, its just a code number*
- Let's name each column appropriately,
- as well asl sample a little data to see where we do mean and where we apply mode:

```
colnames(breastCancerTable) <- c("sampleCodeNumber",
                                 "clumpThickness",
                                 "cellSizeUniformity",
                                 "cellShapeUniformity",
                                 "marginalAdhesion",
                                 "singleEpithelialCellSize",
                                 "bareNuclei",
                                 "blandChromatin",
                                 "normalNucleoli",
                                 "Mitoses",
                                 "Class")

  head(breastCancerTable, 3)
```

```
##   sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 1          1000025              5                  1                   1
## 2          1002945              5                  4                   4
## 3          1015425              3                  1                   1
##   marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 1                1                        2          1              3
## 2                5                        7         10              3
## 3                1                        2          2              3
##   normalNucleoli Mitoses Class
## 1              1       1     2
## 2              2       1     2
## 3              1       1     2
```

1. Use the mean/mode imputation method to impute values for the missing data.*

```
getMode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

breastCancerTable[which(breastCancerTable$bareNuclei == "?"),]
```

```
##     sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 24           1057013              8                  4                   5
## 41           1096800              6                  6                   6
## 140          1183246              1                  1                   1
## 146          1184840              1                  1                   3
## 159          1193683              1                  1                   2
## 165          1197510              5                  1                   1
## 236          1241232              3                  1                   4
## 250           169356              3                  1                   1
## 276           432809              3                  1                   3
## 293           563649              8                  8                   8
## 295           606140              1                  1                   1
## 298          61634            5                 4                 3
## 316         704168            4                 6                 5
## 322         733639            3                 1                 1
## 412        1238464            1                 1                 1
## 618        1057067            1                 1                 1
##     marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 24                 1                        2          ?              7
## 41                 9                        6          ?              7
## 140                1                        1          ?              2
## 146                1                        2          ?              2
## 159                1                        3          ?              1
## 165                1                        2          ?              3
## 236                1                        2          ?              3
## 250                1                        2          ?              3
## 276                1                        2          ?              2
## 293                1                        2          ?              6
## 295                1                        2          ?              2
## 298                1                        2          ?              2
## 316                6                        7          ?              4
## 322                1                        2          ?              3
## 412                1                        1          ?              2
## 618                1                        1          ?              1
##     normalNucleoli Mitoses Class
## 24               3       1     4
## 41               8       1     2
## 140              1       1     2
## 146              1       1     2
## 159              1       1     2
## 165              1       1     2
## 236              1       1     2
## 250              1       1     2
## 276              1       1     2
## 293             10       1     4
## 295              1       1     2
## 298              3       1     2
## 316              9       1     2
## 322              1       1     2
## 412              1       1     2
## 618              1       1     2
```

```r
errors <- breastCancerTable[which(breastCancerTable$bareNuclei == "?"), ]
percentNoData <- length(errors)/length(breastCancerTable$bareNuclei)

#storing the indices of missing data:
missingDataIndices <- which(breastCancerTable$bareNuclei == "?", arr.ind = TRUE)

#find mode of clean data:
mode_bareNuclei <- as.numeric(getMode(breastCancerTable[-missingDataIndices, "bareNuclei"]))
#find mean:
mean_bareNuclei <- round(mean(as.numeric(breastCancerTable[-missingDataIndices, "bareNuclei"])))

mean_bareNuclei
```

```
## [1] 3
```

```
#gsub("\\?","22",breastCancerTable[322,])
#breastCancerTable[322,]
```

- by using vector manipulation , we identified the missing data, and specifically the column with the missing data (bareNuclei)
- only 1.5% is supected data, which is less than the 5% watermark, and thus should allowed to do imputations.
- we simple now create a copy of the breastCancerTable and impute all the values with the mean. **i chose not to use the mode, since this cellNuclei is not a categorical variable, its just a value between a range.

```
breastCancerTableImputedByMean <- breastCancerTable
breastCancerTableImputedByMean[missingDataIndices, ]$bareNuclei <- as.integer(mean_bareNuclei)
```

- let's see how the original and the current missing data compares:

- first we print the first 3 lines of the observations which have missing data. Note the "?" in bareNuclei column

```
head(breastCancerTable[which(breastCancerTable$bareNuclei == "?"), ], 3)
```

```
##     sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 24           1057013              8                  4                   5
## 41           1096800              6                  6                   6
## 140          1183246              1                  1                   1
##     marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 24                 1                        2          ?              7
## 41                 9                        6          ?              7
## 140                1                        1          ?              2
##     normalNucleoli Mitoses Class
## 24               3       1     4
## 41               8       1     2
## 140              1       1     2
```

- now let's print the same lines from the new data table with imputed values:

```
#head(breastCancerTable[which(breastCancerTable$bareNuclei == "?"), ], 3)
breastCancerTableImputedByMean[24,]
```

```
##    sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 24          1057013              8                  4                   5
##    marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 24                1                        2          3              7
##    normalNucleoli Mitoses Class
## 24              3       1     4
```

```
breastCancerTableImputedByMean[41,]
```

```
##    sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 41          1096800              6                  6                   6
##    marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 41                9                        6          3              7
##    normalNucleoli Mitoses Class
## 41              8       1     2
```

```
breastCancerTableImputedByMean[140,]
```

```
##     sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 140          1183246              1                  1                   1
##     marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 140                1                        1          3              2
##     normalNucleoli Mitoses Class
## 140              1       1     2
```

Summary: the verification above confirms that the imputation has worked effectively and the missing data is "no more!"

2. Us24e regression to impute values for the missing data.

- do the same stuff as in step 1..
- first pulling out everything but the missing data and just the predictors (i.e ignoring the id (column1), and the response (class, column 11))
- and using those predictors to identify the values for the missing data using regression (this is a pretty cool way of doing it and more realistic instead of applying the flat mean or mode, the only issue is its more complicated as now we have to manage a regression model for the missing values and a regression value of the actual prediction!)

```
bcPredictors <- breastCancerTable[-missingDataIndices, 2:10]
# converting to integer so the lm can write back into this column without error
bcPredictors$bareNuclei <- as.integer(bcPredictors$bareNuclei)

# running the linear regression, to predict the bareNuclei  values (note its predicting ALL the values
bareNucleiLRM <- lm(bareNuclei~., data = bcPredictors)
summary(bareNucleiLRM)
```

```
##
## Call:
## lm(formula = bareNuclei ~ ., data = bcPredictors)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1137 -0.7185 -0.4731 -0.2994  7.3848
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1.862817   0.162497  11.464  < 2e-16 ***
## clumpThickness           0.068118   0.034746   1.960  0.05035 .
## cellSizeUniformity       0.087939   0.063482   1.385  0.16643
## cellShapeUniformity      0.110046   0.061190   1.798  0.07255 .
## marginalAdhesion        -0.076950   0.038270  -2.011  0.04475 *
## singleEpithelialCellSize  0.043216   0.052123   0.829  0.40733
## blandChromatin           0.044536   0.049211   0.905  0.36579
## normalNucleoli           0.119422   0.037076   3.221  0.00134 **
## Mitoses                  0.001405   0.049448   0.028  0.97733
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.896 on 674 degrees of freedom
## Multiple R-squared:  0.2326, Adjusted R-squared:  0.2235
## F-statistic: 25.54 on 8 and 674 DF,  p-value: < 2.2e-16
```

- clearly the more is pretty bad, looking at the r-squared value, so let's just pick up the specific predictors which have low p-value, which are:

- clumpThickness 0.068118 0.034746 1.960 0.05035 .

- cellShapeUniformity 0.110046 0.061190 1.798 0.07255 .

- marginalAdhesion -0.076950 0.038270 -2.011 0.04475 *

- normalNucleoli 0.119422 0.037076 3.221 0.00134 **

- and re-run the regression with just these:

```
bareNucleiLRM <- lm(bareNuclei ~ clumpThickness + cellShapeUniformity + marginalAdhesion + normalNucleol
summary(bareNucleiLRM)
```

```
##
## Call:
## lm(formula = bareNuclei ~ clumpThickness + cellShapeUniformity +
##     marginalAdhesion + normalNucleoli, data = bcPredictors)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9752 -0.7586 -0.4826 -0.3230  7.6159
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          1.95020    0.13693  14.242  < 2e-16 ***
## clumpThickness       0.07980    0.03432   2.325   0.0204 *
## cellShapeUniformity  0.20051    0.04284   4.680 3.46e-06 ***
## marginalAdhesion    -0.04953    0.03575  -1.385   0.1664
## normalNucleoli       0.14200    0.03525   4.028 6.25e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.897 on 678 degrees of freedom
## Multiple R-squared:  0.2268, Adjusted R-squared:  0.2222
## F-statistic: 49.71 on 4 and 678 DF,  p-value: < 2.2e-16
```

- hmm, the model has gone from bad to worse, perhaps this is the way it is. the data in this model is just insufficient in accurately predicting the missing Values.
- **MOVING ON!!**
- getting the predicted values for the missing bareNuclei from the last model..
- though the TA recommended this in her notes, i did not run k-fold test as I couldn't figure out how cv.kknn, or train.kknn which are classification models apply to linear regression.

```
bareNucleiPredicted <- predict(bareNucleiLRM, newdata = breastCancerTable[missingDataIndices, ])
breastCancerTableImputedByRegression <- breastCancerTable
breastCancerTableImputedByRegression[missingDataIndices, ]$bareNuclei <- round(bareNucleiPredicted)
breastCancerTableImputedByRegression$bareNuclei <- as.integer(breastCancerTableImputedByRegression$bareN
breastCancerTableImputedByRegression[missingDataIndices, ]$bareNuclei
```

```
##  [1] 6 6 4 5 5 5 5 4 5 8 4 5 6 4 4 4
```

- looks like a clean set of new bareNuclei data that would be inserted into the missing data slots.
- lets ensure we limit the range of the data from 1-10 which is the prescribed limit for bareNuclei

```
#make sure no bareNuclei values are outside of original range:
breastCancerTableImputedByRegression$bareNuclei[breastCancerTableImputedByRegression$bareNuclei > 10 ] <
breastCancerTableImputedByRegression$bareNuclei[breastCancerTableImputedByRegression$bareNuclei < 1 ] <-
```

that's it , we're done!

3. Use regression with perturbation to impute values for the missing data.

- this model is nothing more than adding a little error into the regression values created for the missing data..
- the raw values perturbed (unrounded values are printed here:)

```
# first, creating a normal distribution
bareNucleiPerturbed <- rnorm(nrow(breastCancerTable[missingDataIndices,]), bareNucleiPredicted, sd(bareN

bareNucleiPerturbed
```

```
##  [1] 4.537641 3.712837 2.824813 4.042514 1.819416 1.705341 3.504362
##  [8] 1.749486 2.243761 4.443305 1.621406 4.021406 3.714123 2.494358
## [15] 1.381322 1.366726
```

```
breastCancerTablePerturbed <- breastCancerTable
breastCancerTablePerturbed[missingDataIndices, ]$bareNuclei <- round(bareNucleiPerturbed)
breastCancerTablePerturbed$bareNuclei <- as.integer(breastCancerTablePerturbed$bareNuclei)

# round to integers


breastCancerTablePerturbed$bareNuclei[breastCancerTablePerturbed$bareNuclei > 10] <- 10
breastCancerTablePerturbed$bareNuclei[breastCancerTablePerturbed$bareNuclei < 1] <- 1
```

-

```
head(breastCancerTable[which(breastCancerTable$bareNuclei == "?"), ], 3)
```

```
##     sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 24           1057013              8                  4                   5
## 41           1096800              6                  6                   6
## 140          1183246              1                  1                   1
##     marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 24                 1                        2          ?              7
## 41                 9                        6          ?              7
## 140                1                        1          ?              2
##     normalNucleoli Mitoses Class
## 24               3       1     4
## 41               8       1     2
## 140              1       1     2
```

- now let's print the same lines from the new data table with imputed mean values:

```
#head(breastCancerTable[which(breastCancerTable$bareNuclei == "?"), ], 3)
breastCancerTableImputedByMean[24,]
```

```
##    sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 24          1057013              8                  4                   5
##    marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 24                1                        2          3              7
##    normalNucleoli Mitoses Class
## 24              3       1     4
```

```
breastCancerTableImputedByMean[41,]
```

```
##    sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 41          1096800              6                  6                   6
##    marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 41                9                        6          3              7
##    normalNucleoli Mitoses Class
## 41              8       1     2
```

```
breastCancerTableImputedByMean[140,]
```

```
##     sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 140          1183246              1                  1                   1
##     marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 140                1                        1          3              2
##     normalNucleoli Mitoses Class
## 140              1       1     2
```

- finally let's printed the same observations with perturbed bareNuclei values:

```
#head(breastCancerTable[which(breastCancerTable$bareNuclei == "?"), ], 3)
breastCancerTablePerturbed[24,]
```

```
##    sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 24          1057013              8                  4                   5
##    marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 24                1                        2          7              7
##    normalNucleoli Mitoses Class
## 24              3       1     4
```

```
breastCancerTablePerturbed[41,]
```

```
##    sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 41          1096800              6                  6                   6
##    marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 41                9                        6          6              7
##    normalNucleoli Mitoses Class
## 41              8       1     2
```

```
breastCancerTablePerturbed[140,]
```

```
##     sampleCodeNumber clumpThickness cellSizeUniformity cellShapeUniformity
## 140          1183246              1                  1                   1
##     marginalAdhesion singleEpithelialCellSize bareNuclei blandChromatin
## 140                1                        1          5              2
##     normalNucleoli Mitoses Class
## 140              1       1     2
```

** Summary we can clearly see a lot more variety in even the 3 imputed (perturbed) values above than the earlier imputation methods.

Question 15.1

Describe a situation or problem from your job, everyday life, current events, etc., for which optimization would be appropriate. What data would you need?

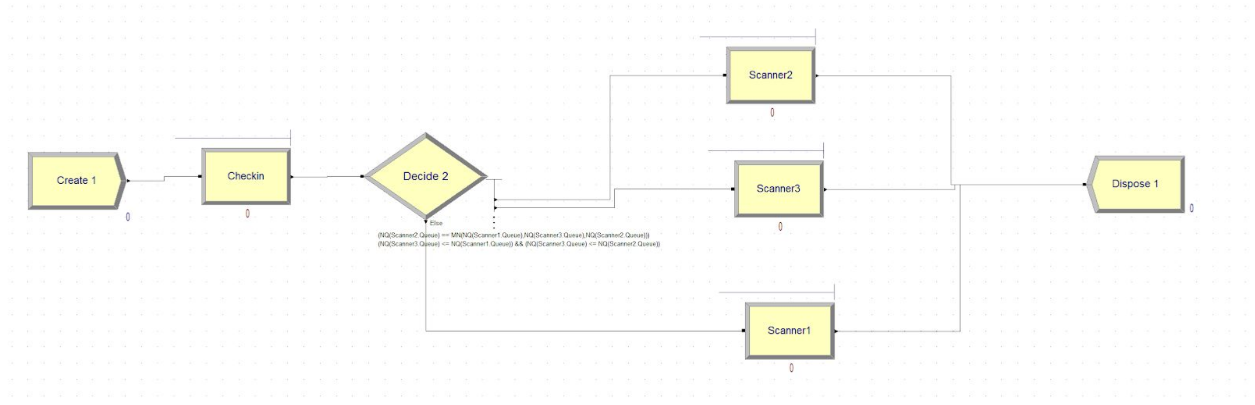
Optimization can be used for prescriptive analytics. I would use it to select which college my son can choose to go to based on financial, qualititative, and location preferences. information I'd need would be information on each school's costs for the program my son selects, information on the rankings and ratings from industry, internally and cross-peer (typically published by US news) and distance door to doo from our house to that school.

Question 13.2

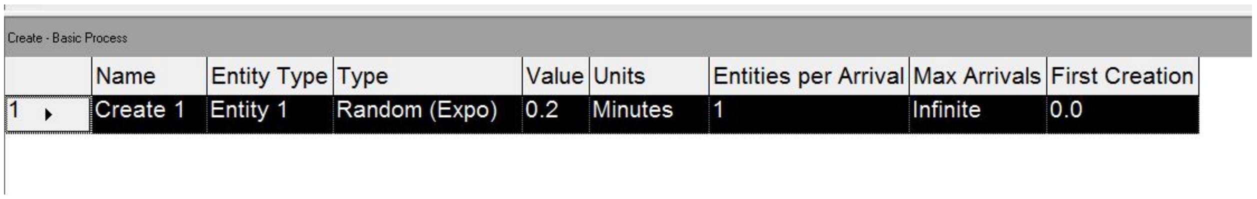
In this problem you, can simulate a simplified airport security system at a busy airport. Passengers arrive according to a Poisson distribution with $\lambda_1 = 5$ per minute (i.e., mean interarrival rate $\mu_1 = 0.2$ minutes) to the ID/boarding-pass check queue, where there are several servers who each have exponential service time with mean rate $\mu_2 = 0.75$ minutes. [Hint: model them as one block that has more than one resource.] After that, the passengers are assigned to the shortest of the several personal-check queues, where they go through the personal scanner (time is uniformly distributed between 0.5 minutes and 1 minute).

Use the Arena software (PC users) or Python with SimPy (PC or Mac users) to build a simulation of the system, and then vary the number of ID/boarding-pass checkers and personal-check queues to determine how many are needed to keep average wait times below 15 minutes. [If you're using SimPy, or if you have access to a non-student version of Arena, you can use $\lambda_1 = 50$ to simulate a busier airport.]

I used the Arena simulation to do this exercise.

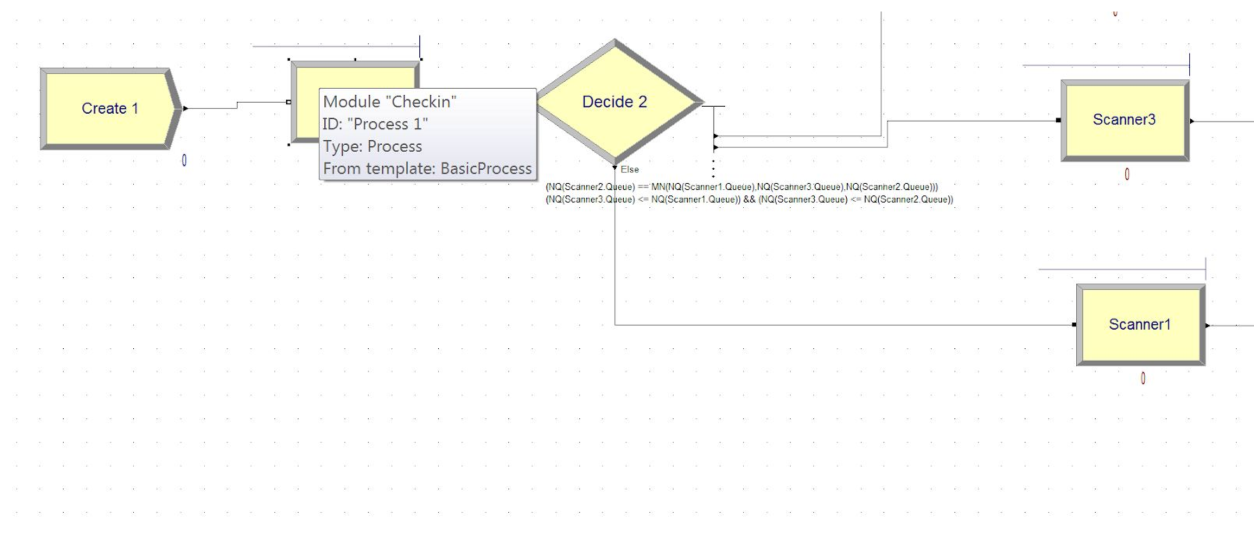

Create 1
0
Checkin
0
Decide 2
Else
(NQ(Scanner2.Queue) == MN(NQ(Scanner1.Queue),NQ(Scanner3.Queue),NQ(Scanner2.Queue)))
(NQ(Scanner3.Queue) <= NQ(Scanner1.Queue)) && (NQ(Scanner3.Queue) <= NQ(Scanner2.Queue))
Scanner2
0
Scanner3
0
Scanner1
0
Dispose 1
0

For the passenger input, here are my settings in the Create icon:


Create - Basic Process

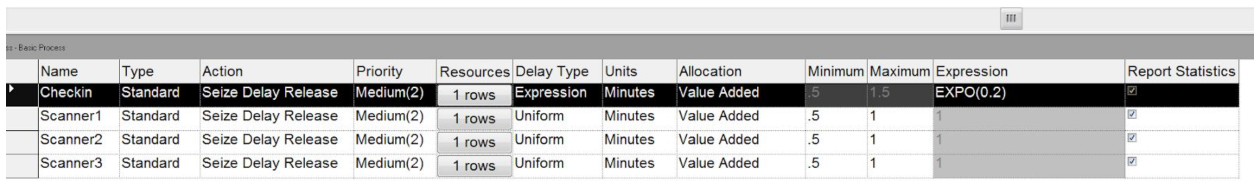
	Name	Entity Type	Type	Value	Units	Entities per Arrival	Max Arrivals	First Creation
1	Create 1	Entity 1	Random (Expo)	0.2	Minutes	1	Infinite	0.0

At the check in I have defined the appropriate distribution in the input queue delay type



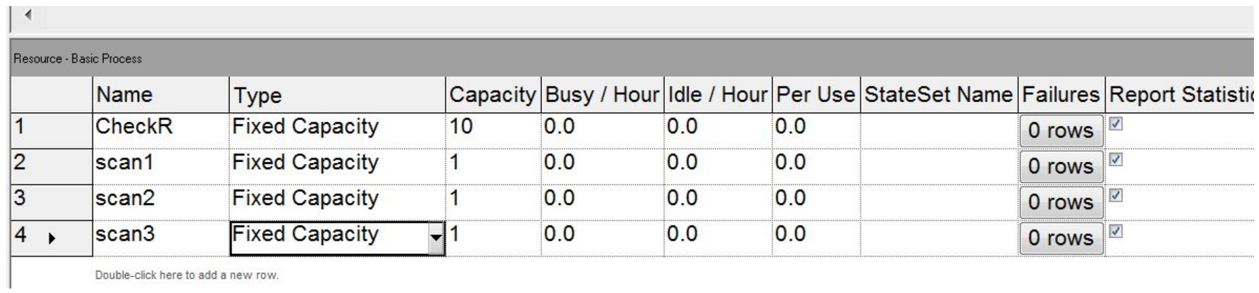
```
Module "Checkin"
ID: "Process 1"
Type: Process
From template: BasicProcess
```

```
(NQ(Scanner2.Queue) == MN(NQ(Scanner1.Queue),NQ(Scanner3.Queue),NQ(Scanner2.Queue)))
(NQ(Scanner3.Queue) <= NQ(Scanner1.Queue)) && (NQ(Scanner3.Queue) <= NQ(Scanner2.Queue))
```



Name	Type	Action	Priority	Resources	Delay Type	Units	Allocation	Minimum	Maximum	Expression	Report Statistics
Checkin	Standard	Seize Delay Release	Medium(2)	1 rows	Expression	Minutes	Value Added	.5	1.5	EXPO(0.2)	✓
Scanner1	Standard	Seize Delay Release	Medium(2)	1 rows	Uniform	Minutes	Value Added	.5	1	1	✓
Scanner2	Standard	Seize Delay Release	Medium(2)	1 rows	Uniform	Minutes	Value Added	.5	1	1	✓
Scanner3	Standard	Seize Delay Release	Medium(2)	1 rows	Uniform	Minutes	Value Added	.5	1	1	✓

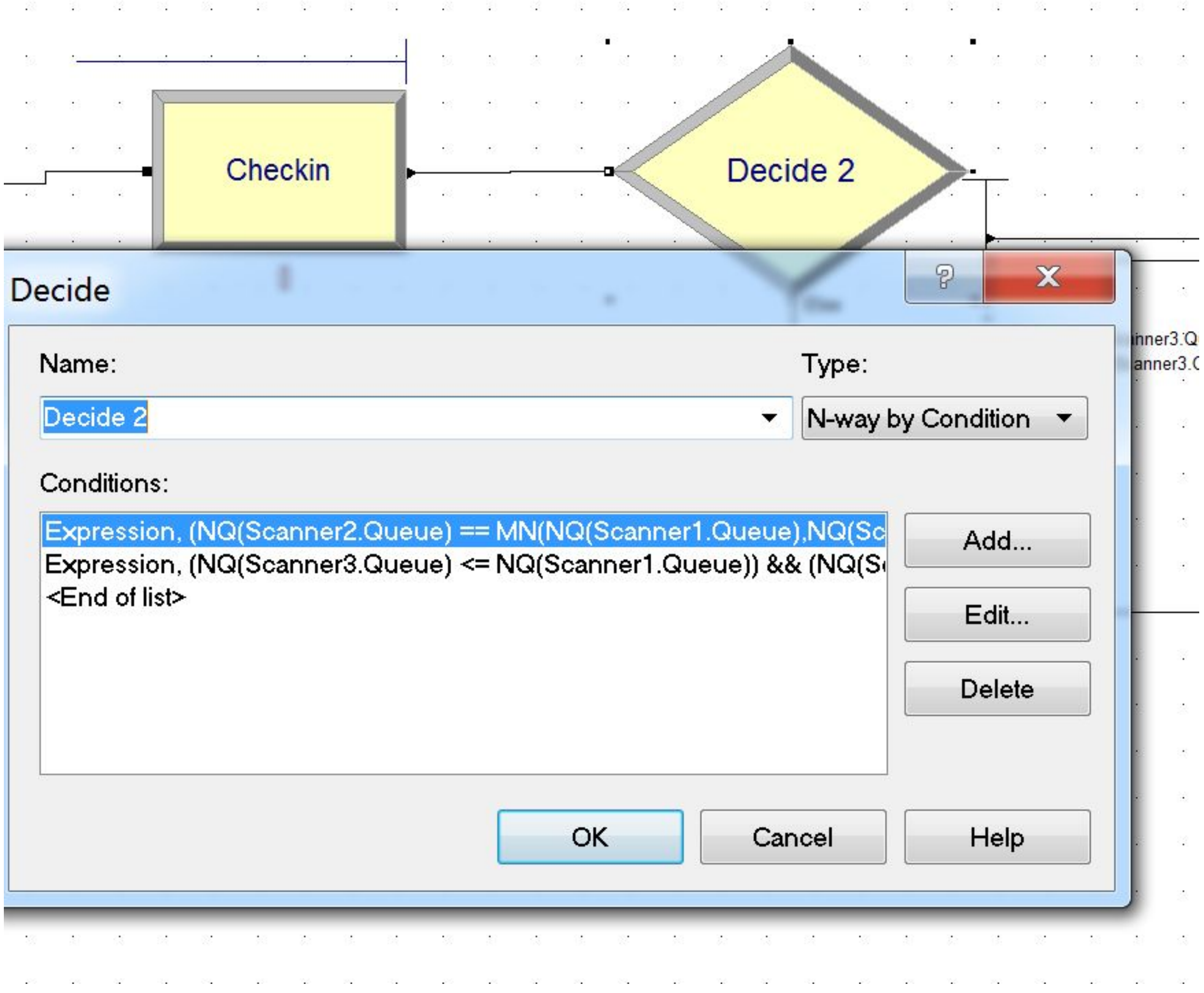
I separately created the resources via drag n drop:



	Name	Type	Capacity	Busy / Hour	Idle / Hour	Per Use	StateSet Name	Failures	Report Statisti
1	CheckR	Fixed Capacity	10	0.0	0.0	0.0		0 rows	✓
2	scan1	Fixed Capacity	1	0.0	0.0	0.0		0 rows	✓
3	scan2	Fixed Capacity	1	0.0	0.0	0.0		0 rows	✓
4	scan3	Fixed Capacity	1	0.0	0.0	0.0		0 rows	✓

And attached the Checkin , Scanner1, Scanner2 and Scanner3 to the resources , e,g,

The decide item in the flowchart was key



Name: Decide 2

Type: N-way by Condition

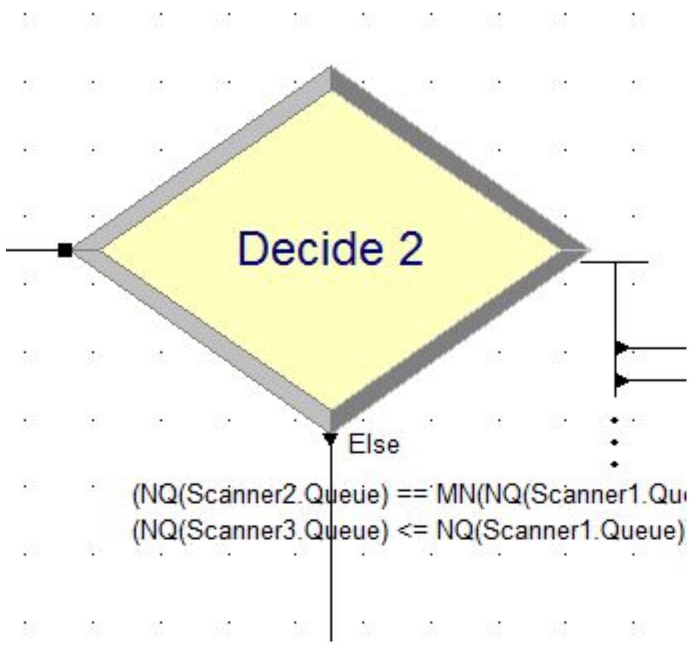
Conditions:

```
Expression, (NQ(Scanner2.Queue) == MN(NQ(Scanner1.Queue),NQ(Sc
Expression, (NQ(Scanner3.Queue) <= NQ(Scanner1.Queue)) && (NQ(S
<End of list>
```

It was type N-way by confition, which allowed me two if commands, and the else In programming this likely translates to

```
If (condition1) { dothis}
Elseif (condition2) {dothat}
Else (dothisInstead)
```

Every condition I defined created a new arror on the right of the decision diamond (notice the two lines comes out of the two if commands..


Decide 2
Else
(NQ(Scanner2.Queue) == MN(NQ(Scanner1.Qu
(NQ(Scanner3.Queue) <= NQ(Scanner1.Queue)

The HW requirement was that the traveler would go to the scanner with the LOWEST queue..

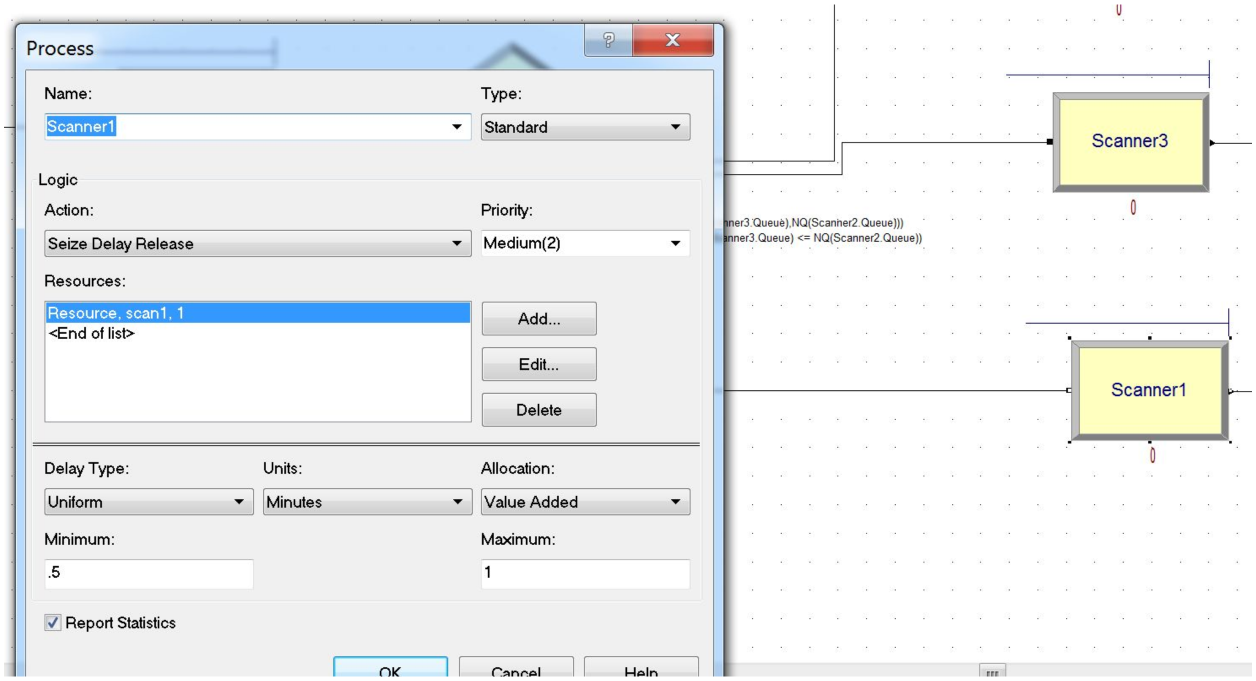
Condition 1 was to select the scanner queue if out of all the queues scanner2 was the lowest

```
(NQ(Scanner2.Queue) ==
MN(NQ(Scanner1.Queue),NQ(Scanner3.Queue),NQ(Scanner2.Queue)))
```

Condition 2 was to select SCanner 3 if it is smaller than 1 and 2:

```
(NQ(Scanner3.Queue) <= NQ(Scanner1.Queue)) && (NQ(Scanner3.Queue) <=
NQ(Scanner2.Queue))
```

The final “else” is the default option which went to scanner1 as implicitly being the lowest queue



Process

Name: Scanner1

Type: Standard

Logic

Action: Seize Delay Release

Priority: Medium(2)

Resources:

Resource, scan1, 1

<End of list>

Add... Edit... Delete

Delay Type: Uniform

Units: Minutes

Allocation: Value Added

Minimum: .5

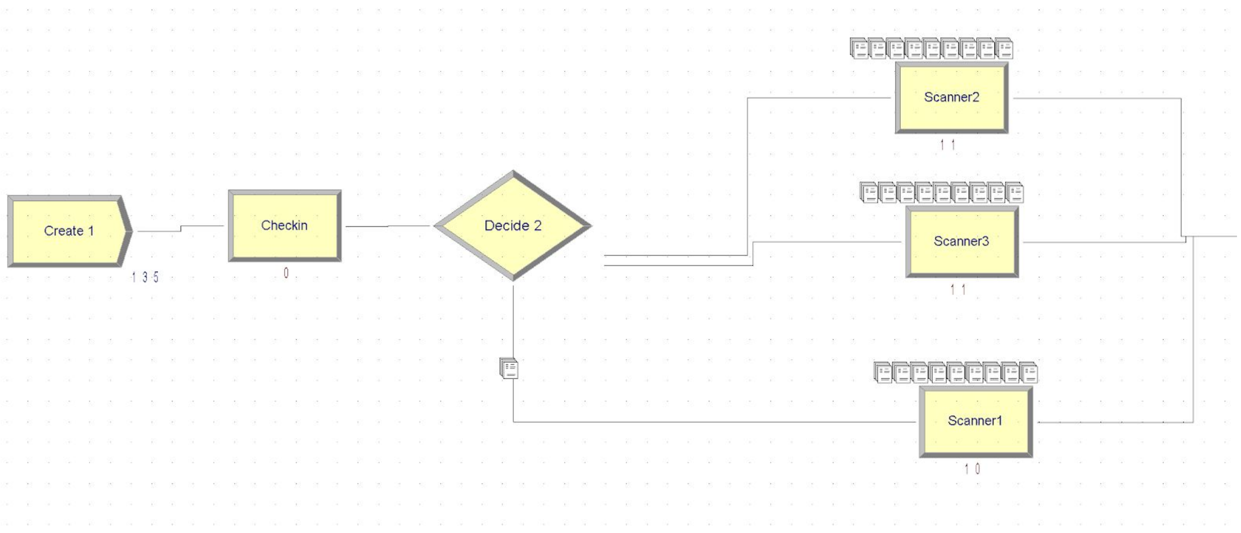
Maximum: 1

☑ Report Statistics

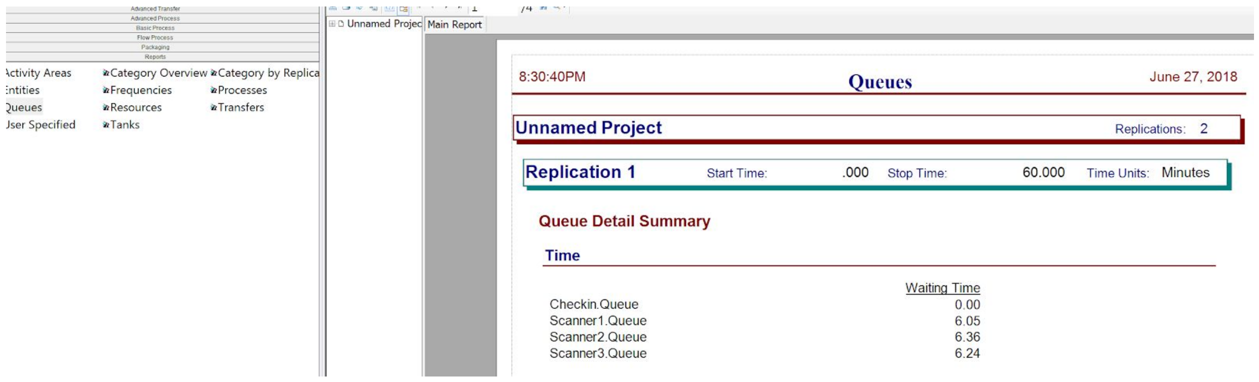
OK Cancel Help

nner3.Queue),NQ(Scanner2.Queue)))
anner3.Queue) <= NQ(Scanner2.Queue))
Scanner3
0
Scanner1
0

During the run I saw the queue increase on all the scanner people



In report the results showed that the queue times were less than 15 minutes



8:30:40PM Queues June 27, 2018

Unnamed Project Replications: 2

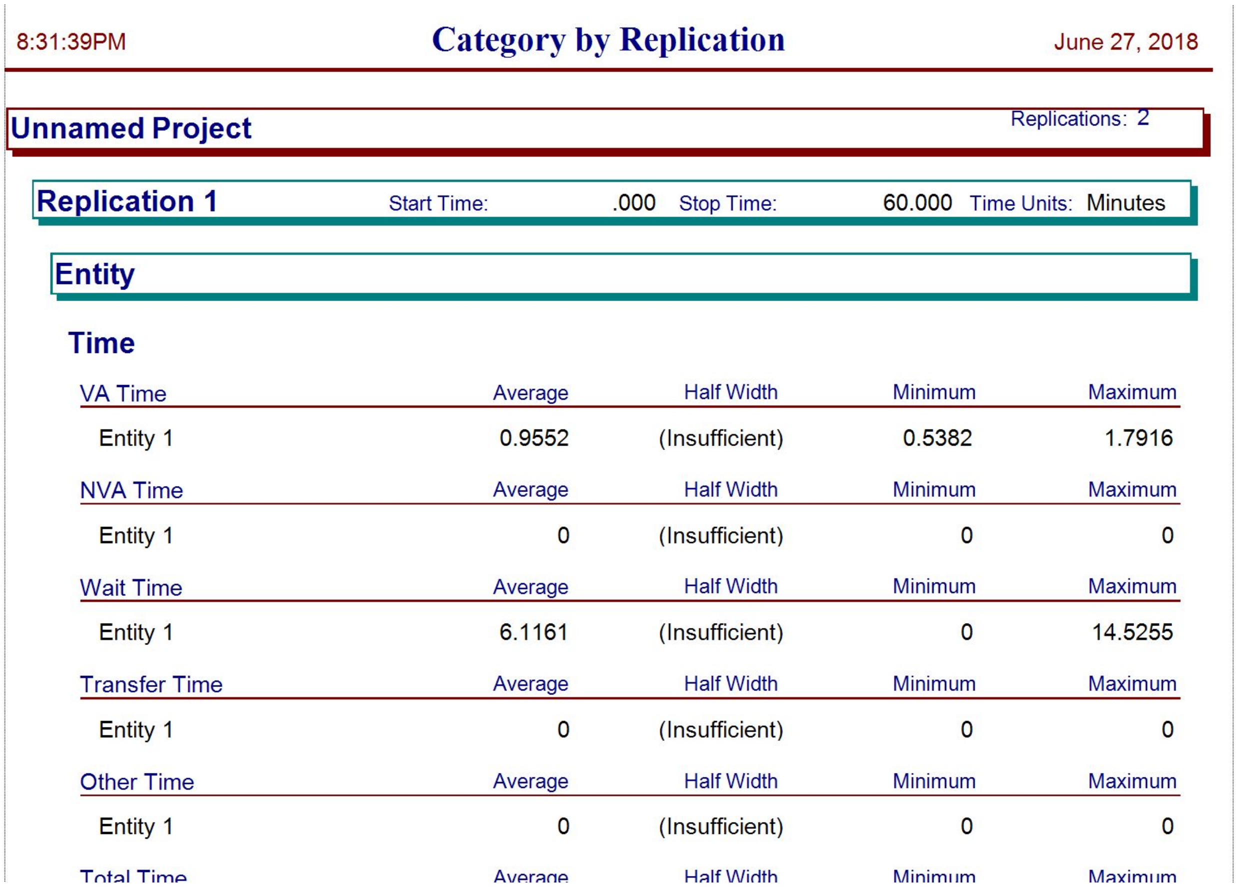
Replication 1 Start Time: .000 Stop Time: 60.000 Time Units: Minutes

Queue Detail Summary

Time

```
                                   Waiting Time
Checkin.Queue                          0.00
Scanner1.Queue                         6.05
Scanner2.Queue                         6.36
Scanner3.Queue                         6.24
```

With the maximum just under 15 minutes..



8:31:39PM **Category by Replication** June 27, 2018

Unnamed Project Replications: 2

Replication 1 Start Time: .000 Stop Time: 60.000 Time Units: Minutes

Entity

Time

VA Time	Average	Half Width	Minimum	Maximum
Entity 1	0.9552	(Insufficient)	0.5382	1.7916

NVA Time	Average	Half Width	Minimum	Maximum
Entity 1	0	(Insufficient)	0	0

Wait Time	Average	Half Width	Minimum	Maximum
Entity 1	6.1161	(Insufficient)	0	14.5255

Transfer Time	Average	Half Width	Minimum	Maximum
Entity 1	0	(Insufficient)	0	0

Other Time	Average	Half Width	Minimum	Maximum
Entity 1	0	(Insufficient)	0	0

Total Time Average Half Width Minimum Maximum

So apparently this model with 10 checkers and 3 scanners worked fine out of the box!