ML1000CourseProjectRtlGrp

Ignacio Palma, Jairo Melo and Vikram Khade

2019-03-13

## Information Technology Service Management Analysis

ITSM is an area of continues improvement and for major organizations every opportunity could represent major cost savings which translate into more affortable products for patiences and parents.

The file extracted from the ITSM system contains 1.2 year worth data for two major product lines.

## Loading data

You can include R code in the document as follows:

## [1] "/Users/jairomelo/Desktop/ML/YORK/CourseProject"

## Identify Anomalies/Cleaning the data

We will take care of duplicated, records with NA values, removing tickets that are not Resolved, as well as undertermine records, for example: Tickets with Support Level outside of the standards.

## [1] 21750

## [1] 41975

## [1] 21294

## [1] 21291

Now we will ensure no data issues to reduce the risk of miss interpretation We need to remove any of observation if any of the below is greater than 0 Zero

## [1] 0

## [1] 0

## [1] 0

## [1] 0

## [1] 0

## [1] 0

## [1] 0

## [1] 0

## [1] 0

## [1] 0

## [1] 0

## [1] 0

## Data Understanding

* incident: Number of the ticket incident. Not a significant variable as is sequencial counter.
* application: Number of the application of the reported issue. This is a relevant variable which a certantly number of tickets are assigned to one application.
* region: Region where the user is located. Significant as a region is associated to a particular population of users reporting issues of an application.
* prod\_line: Product Line is a group of related products under the same brand. For example, Web and Ecommerce, and also Internal Business process applications.
* opened: Date when the issues was opened. The ticket has 5 stages: Not Assigned, In Progress, Customer Action, Pending, Resolved, Closed. Not Assigned: The ticket was created/open, but still not been worked by the support team. In Progress: The ticket is assigned to a support group who is actively working on it. Customer Action: The ticket goes into a stand-by because additional information is requested from the user before the current support group can continue working. Pending: The ticket goes into a stand-by because there is an activity to be performed by a third party group before the current support group can continue working. Resolved: Once the issue is fixed, the user is notified by the Support team. Closed: Each resolved ticket moves into Closed after the user confirms, or automatically, the ticket is closed after n number of days. For our analysis, we will using only tickets that are Resolved. Closed might not be relevant as there is a strong correlation between Closed and Resolved.
* app\_category: Category of the Application. Relevant as this is the classification of the application.
* priority: Priority of the Issue. This is the result of Urgency and Impact.  
  Low Urgency - Limited Impact = Lower Priority. -> 4 High Urgency - Limite Impact = High Priority. -> 1 The “Priority” word can be removed from the field and use the numbers 1,2,3,4. Priority 4 is low, and 1 is the highest.
* urgency: How soon the issue should be resolved. There is a strong correlation between Urgency and Priority; which might cause to ommit the field when using Priority.
* impact: What’s the extension of the issue in terms of number of users. eg: Limited means small group usually 1 or 2 users, Spread-out means usually an area, department or even all organization. There is a strong correlation between Urgency and Priority; which might cause to ommit the field when using Priority.
* Closed: Date when the ticket was finally closed. Refer to the Opened field for explanation of the stages of the tickets.
* sup\_grp: Support Group providing resolution to the issue. This is relevant as the support group is responsible to effectively close a ticket as soon as it’s assigned.
* grp\_level: Support Group Level. There are 3 different groups of support level.

Level 1: Service Desk, primary group who handles all tickets and try to troubleshot the issue. Most of the tickets should be filtered by this team. This is less specialized team, and help to keep Level 2 and 3 focus on major activities. Level 2: This is the specialist team who has greater knowledge on how the application operates. This team takes care of tickets Level 1 is not able to resolved. Level 3: This is the Developers of the applications; has complete knowledge of the application and finally able to resolve the issues scalated by L2 team.

For JnJ, the L2 and L3 are more expensive, and the interest of the company is to identify ways to reduce cost translating activities from L3 to L2 and from L2 to L1.

The “Level” word can be removed from the field and use the numbers 1,2,3. Level 1 is the less specialized, and 3 is the most specialized, usually a lot more expensive than 1.

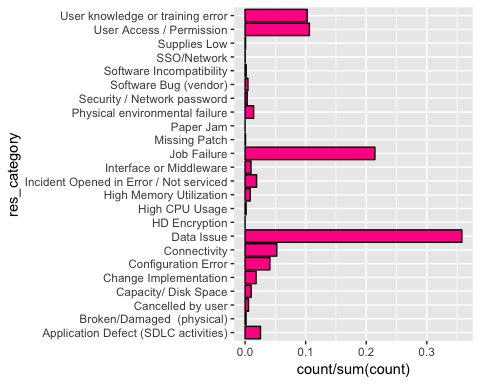
* resolved: Date when the issue was resolved. Refer to the Opened field for explanation of the stages of the tickets.
* res\_category: Category of the type of resolution support team completed.
* cust\_time: Time in seconds the ticket was waiting for Customer response. Refer to the Opened field for explanation of the stages of the tickets.
* pend\_time: Time ticket is on hold. Refer to the Opened field for explanation of the stages of the tickets.
* call\_log: Id of the phone call When a call is involved. Not a relevant attribute as not all tickets triggers a phone call.
* chat\_log: If of the chat session when user uses instance message with the support team. This new technology is not heavily used, so there are very few observations with this information.

Let’s chart the data to understand more about the variables associated to the support activities

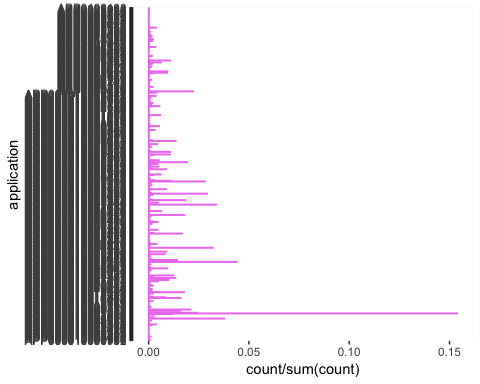
## Data Visualization

Let’s review what the data can tell us about supporting applications for JJTS:

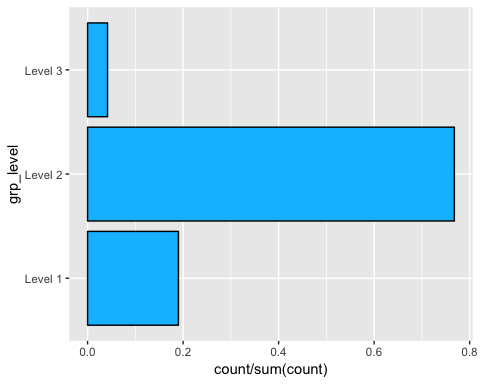
### Plot by Application Category

This feature contains great information on how a particular ticket was resolved. 

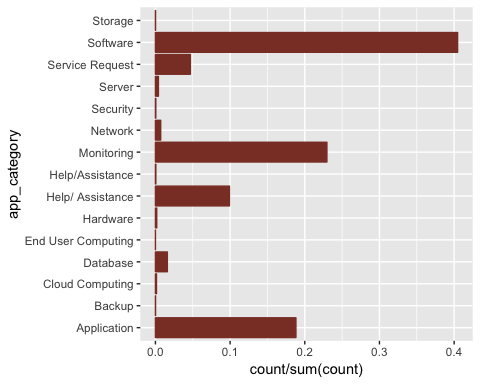
## Exploring Applications

There is more than 500 applications. This feature might not be the best for Supervised Algorithms. 

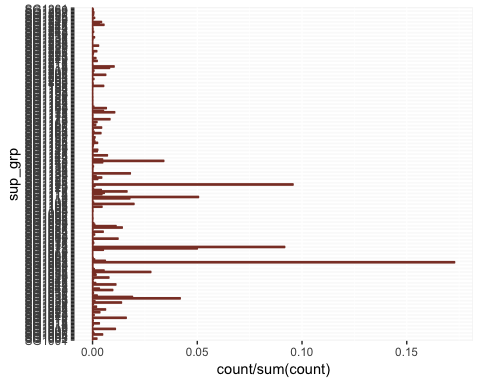
## Exploring Support Group Level

Support Group Level indicates the expertise of the support team. At the same time, the cost of the time goes up while more expert. 

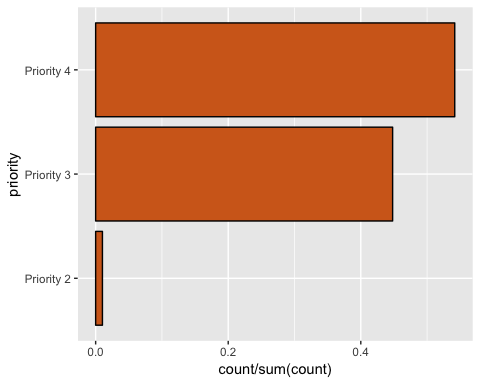
## Exploring Application Category



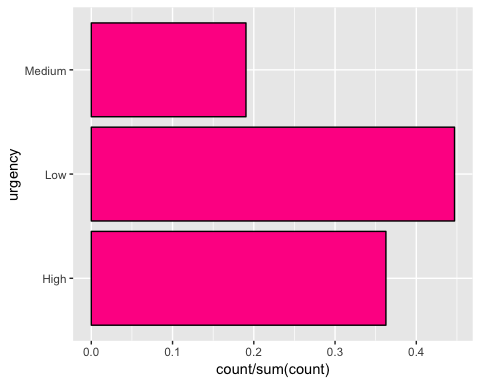
## Exploring Support Group



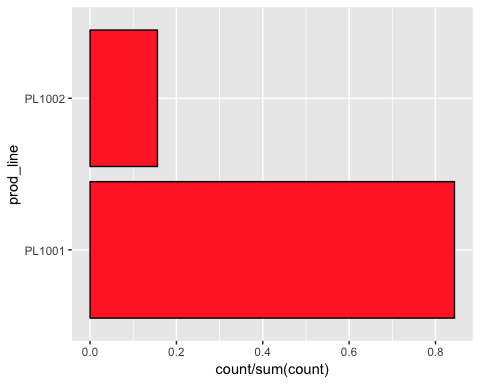
## Exploring Priority

Priority is one of the most important features, and it comes from the combination or Urgency and Impact. 

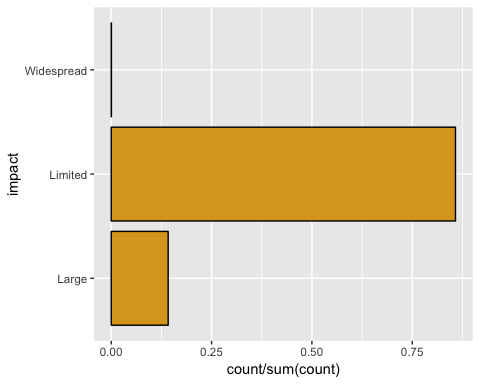
## Exploring Urgency

How soon the issue needs to be resolved. There is a strong correlation with Priority. 

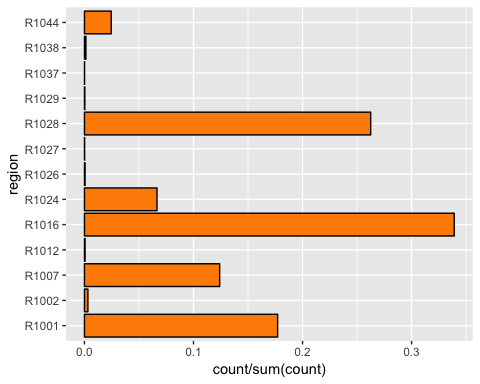
## Exploring Product Line

There are only two product lines in this data set. Its relevance might not be the highest but we’ll keep it while our analysis. 

# Exploring the Impact

This is associated to the number of users affected by the issue. 

# Exploring Region of where the Users are located

When a ticket is created, the location of the user is recorded as well. 

## Data Preparation

Below attributes will be removed from the Dataset due to the low analytical value: Incident: This is the ID of the ticket. We only use it to ensure there are not duplicates. cust\_time: We will focus on the time that the ticket is resolved, customer time with other teams is not relevant for our analysis Pend\_time: We will focus on the time that the ticket is resolved, pending time with other teams is not relevant for our analysis call\_log: This feature is not used mainly; while Support team uses Skype IM chat\_log: Less than 1% of the tickets are manages through chat from ServiceNow; support team usually use Skype IM, which is not recorded in the dataset. Closed: We will focus our analysis on Resolved tickets, close is an automatic process happening 12 days after the ticket was resolved.

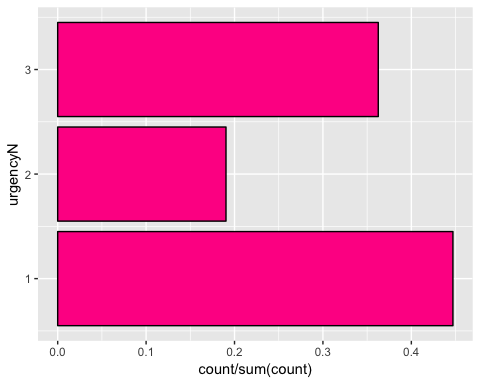
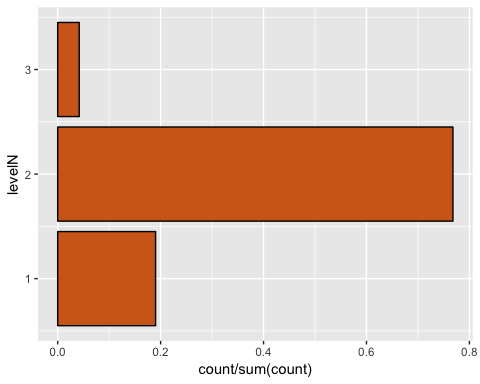
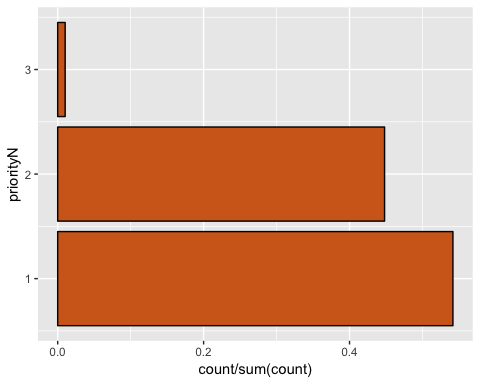
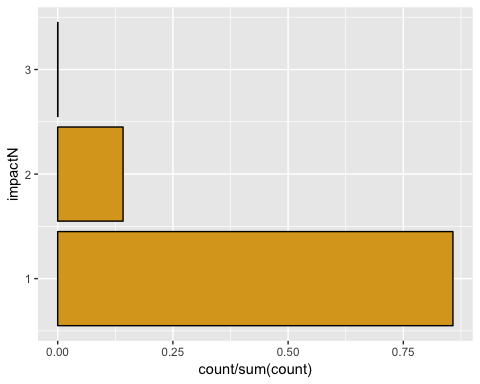
Here is the final data set

## 'data.frame': 21291 obs. of 12 variables:  
## $ application : Factor w/ 535 levels "APP000010000791",..: 375 523 286 375 188 69 148 49 126 226 ...  
## $ region : Factor w/ 13 levels "R1001","R1002",..: 9 9 5 9 3 9 3 9 1 3 ...  
## $ prod\_line : Factor w/ 2 levels "PL1001","PL1002": 2 2 1 2 2 1 1 1 1 2 ...  
## $ opened : Factor w/ 20047 levels "2018-01-01 20:03",..: 9697 6935 12718 9721 13647 2021 3796 6688 3270 16404 ...  
## $ app\_category: Factor w/ 15 levels "Application",..: 15 15 15 15 14 14 14 14 14 14 ...  
## $ priority : Factor w/ 3 levels "Priority 2","Priority 3",..: 3 3 2 2 3 2 3 3 3 3 ...  
## $ urgency : Factor w/ 3 levels "High","Low","Medium": 2 2 1 1 2 1 2 3 2 3 ...  
## $ impact : Factor w/ 3 levels "Large","Limited",..: 2 2 2 2 2 2 2 2 2 2 ...  
## $ sup\_grp : Factor w/ 261 levels "SG1001","SG1002",..: 230 39 62 123 136 248 18 114 33 87 ...  
## $ grp\_level : Factor w/ 5 levels "","3rd Party",..: 3 3 4 3 3 4 4 5 4 4 ...  
## $ resolved : Factor w/ 20314 levels "","2018-01-02 0:45",..: 9414 6806 13406 9321 13459 2002 3986 6544 3133 16987 ...  
## $ res\_category: Factor w/ 25 levels "","Application Defect (SDLC activities)",..: 25 15 9 13 8 9 9 9 9 8 ...

## New Numeric Variables

We are now creating Numeric representation of Impact -> impactN Urgency -> urgencyN Priority -> priorityN Group Level -> LevelN

## 'data.frame': 21291 obs. of 12 variables:  
## $ application : Factor w/ 535 levels "APP000010000791",..: 375 523 286 375 188 69 148 49 126 226 ...  
## $ region : Factor w/ 13 levels "R1001","R1002",..: 9 9 5 9 3 9 3 9 1 3 ...  
## $ prod\_line : Factor w/ 2 levels "PL1001","PL1002": 2 2 1 2 2 1 1 1 1 2 ...  
## $ opened : Factor w/ 20047 levels "2018-01-01 20:03",..: 9697 6935 12718 9721 13647 2021 3796 6688 3270 16404 ...  
## $ app\_category: Factor w/ 15 levels "Application",..: 15 15 15 15 14 14 14 14 14 14 ...  
## $ sup\_grp : Factor w/ 261 levels "SG1001","SG1002",..: 230 39 62 123 136 248 18 114 33 87 ...  
## $ resolved : Factor w/ 20314 levels "","2018-01-02 0:45",..: 9414 6806 13406 9321 13459 2002 3986 6544 3133 16987 ...  
## $ res\_category: Factor w/ 25 levels "","Application Defect (SDLC activities)",..: 25 15 9 13 8 9 9 9 9 8 ...  
## $ impactN : chr "1" "1" "1" "1" ...  
## $ urgencyN : chr "1" "1" "3" "3" ...  
## $ priorityN : chr "1" "1" "2" "2" ...  
## $ levelN : num 1 1 2 1 1 2 2 3 2 2 ...

Let’s see visually the new Numeric Features 

## Dates to chart and Duration of a Ticket

Now we will create the numeric representation of the Date variables and calculate the number of days that support team took to resolve an issue.

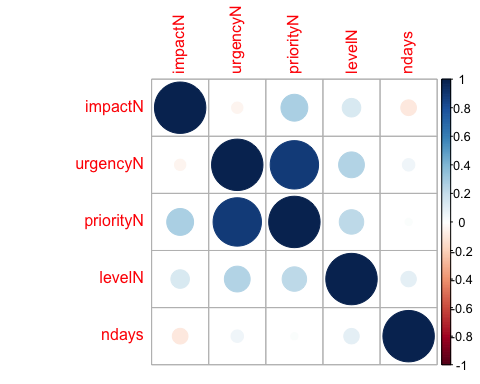
ndays is the time support team took to resolved the issue, in this case is calculated as Resolved - Opened

## 'data.frame': 21291 obs. of 13 variables:  
## $ application : Factor w/ 535 levels "APP000010000791",..: 375 523 286 375 188 69 148 49 126 226 ...  
## $ region : Factor w/ 13 levels "R1001","R1002",..: 9 9 5 9 3 9 3 9 1 3 ...  
## $ prod\_line : Factor w/ 2 levels "PL1001","PL1002": 2 2 1 2 2 1 1 1 1 2 ...  
## $ app\_category: Factor w/ 15 levels "Application",..: 15 15 15 15 14 14 14 14 14 14 ...  
## $ sup\_grp : Factor w/ 261 levels "SG1001","SG1002",..: 230 39 62 123 136 248 18 114 33 87 ...  
## $ res\_category: Factor w/ 25 levels "","Application Defect (SDLC activities)",..: 25 15 9 13 8 9 9 9 9 8 ...  
## $ impactN : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ urgencyN : num 1 1 3 3 1 3 1 2 1 2 ...  
## $ priorityN : num 1 1 2 2 1 2 1 1 1 1 ...  
## $ levelN : num 1 1 2 1 1 2 2 3 2 2 ...  
## $ open\_date : Date, format: "2018-10-03" "2018-07-30" ...  
## $ resolve\_date: Date, format: "2018-10-05" "2018-08-02" ...  
## $ ndays : num 2 3 11 0 0 3 14 3 1 3 ...

## Correlation Matrix

Based on our previous charts, we are now curious to see if there is any feature which its correlation might cause to have it dropped.

## impactN urgencyN priorityN levelN ndays  
## impactN 1.00000000 -0.04820566 0.27259729 0.13022254 -0.09302363  
## urgencyN -0.04820566 1.00000000 0.88568657 0.25126735 0.05838574  
## priorityN 0.27259729 0.88568657 1.00000000 0.22596517 0.01865288  
## levelN 0.13022254 0.25126735 0.22596517 1.00000000 0.09020365  
## ndays -0.09302363 0.05838574 0.01865288 0.09020365 1.00000000



From the figure we identify that urgency and priority are strongly correlated Priority and impact are weakly correlated (0.27) this could be because it is defined by the user during the ticket triaging. Similarly priority and level are weakly correlated. ndays has a very low correlation.

Since priority and urgency are highly correlated (0.89) urgency is dropped from further analysis.

## 'data.frame': 21291 obs. of 12 variables:  
## $ application : Factor w/ 535 levels "APP000010000791",..: 375 523 286 375 188 69 148 49 126 226 ...  
## $ region : Factor w/ 13 levels "R1001","R1002",..: 9 9 5 9 3 9 3 9 1 3 ...  
## $ prod\_line : Factor w/ 2 levels "PL1001","PL1002": 2 2 1 2 2 1 1 1 1 2 ...  
## $ app\_category: Factor w/ 15 levels "Application",..: 15 15 15 15 14 14 14 14 14 14 ...  
## $ sup\_grp : Factor w/ 261 levels "SG1001","SG1002",..: 230 39 62 123 136 248 18 114 33 87 ...  
## $ res\_category: Factor w/ 25 levels "","Application Defect (SDLC activities)",..: 25 15 9 13 8 9 9 9 9 8 ...  
## $ impactN : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ priorityN : num 1 1 2 2 1 2 1 1 1 1 ...  
## $ levelN : num 1 1 2 1 1 2 2 3 2 2 ...  
## $ open\_date : Date, format: "2018-10-03" "2018-07-30" ...  
## $ resolve\_date: Date, format: "2018-10-05" "2018-08-02" ...  
## $ ndays : num 2 3 11 0 0 3 14 3 1 3 ...

## Low Frequency Cleaning

In particular, App Category, Resolution category and Region contains very low frequency levels which could reduce accuracy for our predictions or computing time during our cluster analys. We choose the Threshold = 2.5% to remove observations.

## application region prod\_line   
## APP000010007299: 3199 R1001:3237 PL1001:14920   
## APP000010022869: 892 R1007:2013 PL1002: 2503   
## APP000010006077: 651 R1016:6372   
## APP000010027175: 608 R1024:1028   
## APP000010027900: 558 R1028:4773   
## APP000010027488: 498   
## (Other) :11017   
## app\_category sup\_grp   
## Application :3144 SG1062 :3266   
## Help/ Assistance:1735 SG1123 :1837   
## Monitoring :4374 SG1073 :1482   
## Service Request : 937 SG1112 : 990   
## Software :7233 SG1033 : 847   
## SG1072 : 775   
## (Other):8226   
## res\_category impactN priorityN   
## Configuration Error : 840 Min. :1.000 Min. :1.00   
## Connectivity :1049 1st Qu.:1.000 1st Qu.:1.00   
## Data Issue :7069 Median :1.000 Median :1.00   
## Job Failure :4484 Mean :1.148 Mean :1.45   
## User Access / Permission :2074 3rd Qu.:1.000 3rd Qu.:2.00   
## User knowledge or training error:1907 Max. :3.000 Max. :3.00   
##   
## levelN open\_date resolve\_date   
## Min. :1.000 Min. :2018-01-01 Min. :2018-01-02   
## 1st Qu.:2.000 1st Qu.:2018-06-04 1st Qu.:2018-06-08   
## Median :2.000 Median :2018-10-21 Median :2018-10-24   
## Mean :1.846 Mean :2018-09-13 Mean :2018-09-17   
## 3rd Qu.:2.000 3rd Qu.:2018-12-28 3rd Qu.:2018-12-31   
## Max. :3.000 Max. :2019-02-26 Max. :2019-02-26   
##   
## ndays   
## Min. : 0.000   
## 1st Qu.: 0.000   
## Median : 1.000   
## Mean : 3.969   
## 3rd Qu.: 4.000   
## Max. :265.000   
##

## [1] 17423

## Calculating Performance

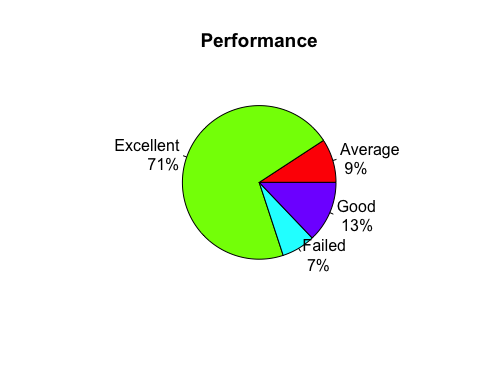
We are rating the servcies provided by the vendor the following table where the numbers are the days expected to have the issue resolved.

## Priority |P1 |P2 | P3

Excellent | 5 | 3 | 1 Good |10 | 5 | 3 Average |15 |10 | 5

Failed: For any other ticket is considered Bad performance is rated as failed service.

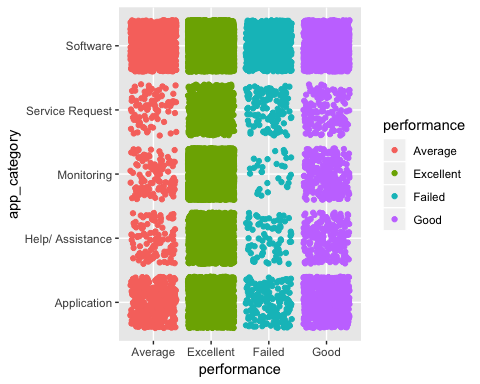
## Average Excellent Failed Good   
## 1605 12343 1232 2243



## Performance by most significant features

Since now we have performance, let’s inspect how the current support teams performs. ##Performance by Application Category Interesting is that across the applications support team is performing quite well. Howeverm software shows the highest issues.

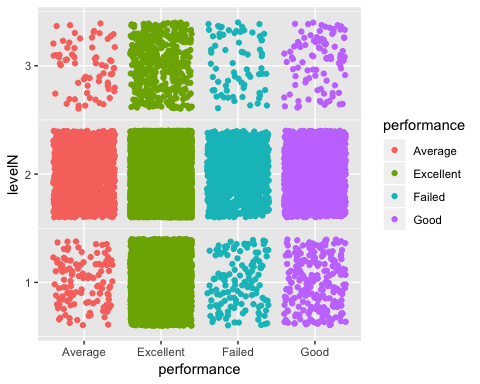
ggplot(data = sdata, mapping = aes(x = performance, y = app\_category)) + geom\_jitter(aes(colour = performance))



## Performance by Support Level

This is the Level of the Support Group level, 1 Service Desk, 2 Specialist, 3 Developer/Architect. From the below chat we can conclude that most of the tickets are resolved by Support Level 2. It would be interesting to look for opportunities to move L2 to L1. Perhaps our unsupervise analysis would provide us ideas.

ggplot(data = sdata, mapping = aes(x = performance, y = levelN)) + geom\_jitter(aes(colour = performance))

 ##Performance by Resolution Category Altough, resolution category can’t be use as predictor, it’s interesting to visualize Data Issue is the most common across all the performance.

ggplot(data = sdata, mapping = aes(x = performance, y = res\_category)) + geom\_jitter(aes(colour = performance))



# Supervised Learning

## Objective

For our analysis, we will predict what performance a ticket will have given an basic data related to the issue.

## What’s the Problem

Business users needs to know how long a ticket would take to be resolved so they can focus on additional activities, or look for workarounds that will reduce the impact of the issue.

## Feature selection

Let’s run a random forest to quantify the relative importance of these features. We will use features with less than 50 Levels, so Application and Support Group will go away. Also Dates should not be considered. Ndays is not a predictable variable because we can’t actually trying to Predict the Performance of resolving a ticket when Duration is provided since we won’t know how long the ticket will last unresolved, but we will know what Priority the ticket is raised.

Finally, Resolution category is unknown as we don’t know what the issue is. We cannot predict based on the resolution.

Summary: priority + grp\_level + app\_category + region + prod\_line + impactN

## MeanDecreaseGini  
## app\_category 354.58480  
## levelN 195.85496  
## region 154.66078  
## priorityN 143.65979  
## impactN 64.58817  
## prod\_line 49.78494

As shown in the table, Product Line has the lowest predictable power. Understandable because there is only two Product Lines; and the decision is either one of the other. Very limited predictibility power.

# Suppervised Modeling

## Classification and Regression Trees

For our Analysis, we will select 5 of the most predictable features:

Priority, Support Level, App Category, Resolution Category, Region

Let’s see the dimmensions for:

#Training  
dim(trainDF);

## [1] 13069 7

#Test  
dim(testDF);

## [1] 4354 7

All Algorithms

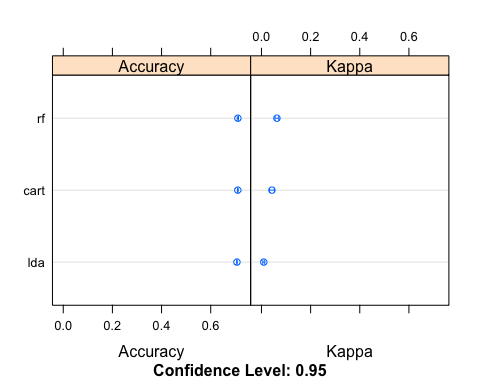
As per the table; the mean accuracy of random forest is the highest of all the three algorithms.

results <- resamples(list(lda=fit.lda, cart=fit.cart, rf=fit.rf))  
summary(results)

##   
## Call:  
## summary.resamples(object = results)  
##   
## Models: lda, cart, rf   
## Number of resamples: 10   
##   
## Accuracy   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lda 0.7031370 0.7063326 0.7068503 0.7066337 0.7075363 0.7087156 0  
## cart 0.7064220 0.7089067 0.7110602 0.7102308 0.7113323 0.7125382 0  
## rf 0.7051184 0.7086284 0.7106773 0.7105373 0.7116634 0.7176741 0  
##   
## Kappa   
## Min. 1st Qu. Median Mean 3rd Qu. Max.  
## lda -0.001277124 0.00737260 0.009210369 0.009439904 0.01387106 0.01632250  
## cart 0.018911855 0.03740403 0.042764849 0.042666165 0.04761031 0.06463698  
## rf 0.046462566 0.05588746 0.063110349 0.062839833 0.07084777 0.07791801  
## NA's  
## lda 0  
## cart 0  
## rf 0

## Visualize the accuracy of the models of the Training

dotplot(results)



The most important metric is prediction on the Testing data

## Let’s make a prediction (accuracy of testing dataset)

1. Linear Discriminant Analysis and Confusion Matrix

predictions <- predict(fit.lda, testDF)  
head(predictions)

## [1] Excellent Excellent Excellent Excellent Excellent Excellent  
## Levels: Average Excellent Failed Good

confusionMatrix(factor(predictions, levels = c("Excellent","Good","Average","Failed")),factor(testDF$performance, levels = c("Excellent","Good","Average","Failed")))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Excellent Good Average Failed  
## Excellent 3074 553 398 298  
## Good 0 0 0 2  
## Average 8 2 3 8  
## Failed 3 5 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.7067   
## 95% CI : (0.6929, 0.7202)  
## No Information Rate : 0.7085   
## P-Value [Acc > NIR] : 0.6124   
##   
## Kappa : 0.0087   
## Mcnemar's Test P-Value : <2e-16   
##   
## Statistics by Class:  
##   
## Class: Excellent Class: Good Class: Average  
## Sensitivity 0.99643 0.0000000 0.007481  
## Specificity 0.01576 0.9994729 0.995446  
## Pos Pred Value 0.71108 0.0000000 0.142857  
## Neg Pred Value 0.64516 0.8713235 0.908147  
## Prevalence 0.70854 0.1286174 0.092099  
## Detection Rate 0.70602 0.0000000 0.000689  
## Detection Prevalence 0.99288 0.0004593 0.004823  
## Balanced Accuracy 0.50610 0.4997364 0.501464  
## Class: Failed  
## Sensitivity 0.000000  
## Specificity 0.998023  
## Pos Pred Value 0.000000  
## Neg Pred Value 0.929130  
## Prevalence 0.070740  
## Detection Rate 0.000000  
## Detection Prevalence 0.001837  
## Balanced Accuracy 0.499011

The Balance accuracy is less than 70.5% for all performance and with the Balanced accuracy: Excellent Class Average Failed 0.50401 0.499341 0.5032167 0.5017638

1. Classification Tree / Recursive Partitioning and Confusion Matrix

predictions <- predict(fit.cart, testDF)  
head(predictions)

## [1] Excellent Excellent Excellent Excellent Excellent Excellent  
## Levels: Average Excellent Failed Good

confusionMatrix(factor(predictions, levels = c("Excellent","Good","Average","Failed")),factor(testDF$performance, levels = c("Excellent","Good","Average","Failed")))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Excellent Good Average Failed  
## Excellent 3070 545 392 293  
## Good 3 5 0 0  
## Average 2 3 6 3  
## Failed 10 7 3 12  
##   
## Overall Statistics  
##   
## Accuracy : 0.7104   
## 95% CI : (0.6967, 0.7238)  
## No Information Rate : 0.7085   
## P-Value [Acc > NIR] : 0.4021   
##   
## Kappa : 0.032   
## Mcnemar's Test P-Value : <2e-16   
##   
## Statistics by Class:  
##   
## Class: Excellent Class: Good Class: Average  
## Sensitivity 0.99514 0.008929 0.014963  
## Specificity 0.03073 0.999209 0.997976  
## Pos Pred Value 0.71395 0.625000 0.428571  
## Neg Pred Value 0.72222 0.872296 0.908986  
## Prevalence 0.70854 0.128617 0.092099  
## Detection Rate 0.70510 0.001148 0.001378  
## Detection Prevalence 0.98760 0.001837 0.003215  
## Balanced Accuracy 0.51294 0.504069 0.506469  
## Class: Failed  
## Sensitivity 0.038961  
## Specificity 0.995057  
## Pos Pred Value 0.375000  
## Neg Pred Value 0.931513  
## Prevalence 0.070740  
## Detection Rate 0.002756  
## Detection Prevalence 0.007350  
## Balanced Accuracy 0.517009

The Balance accuracy is 71% for all performance Excellent Class Average Failed 0.51989 0.5026489 0.509343 0.533771 is better than LDA; but not by far.

predictions <- predict(fit.rf, testDF)  
head(predictions)

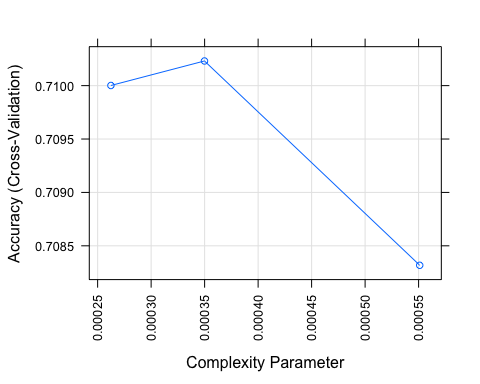
## [1] Excellent Excellent Excellent Excellent Excellent Excellent  
## Levels: Average Excellent Failed Good

confusionMatrix(factor(predictions, levels = c("Excellent","Good","Average","Failed")),factor(testDF$performance, levels = c("Excellent","Good","Average","Failed")))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Excellent Good Average Failed  
## Excellent 3038 539 377 271  
## Good 11 10 4 0  
## Average 9 5 10 8  
## Failed 27 6 10 29  
##   
## Overall Statistics  
##   
## Accuracy : 0.709   
## 95% CI : (0.6953, 0.7225)  
## No Information Rate : 0.7085   
## P-Value [Acc > NIR] : 0.481   
##   
## Kappa : 0.0609   
## Mcnemar's Test P-Value : <2e-16   
##   
## Statistics by Class:  
##   
## Class: Excellent Class: Good Class: Average  
## Sensitivity 0.98476 0.017857 0.024938  
## Specificity 0.06462 0.996046 0.994435  
## Pos Pred Value 0.71905 0.400000 0.312500  
## Neg Pred Value 0.63566 0.872950 0.909533  
## Prevalence 0.70854 0.128617 0.092099  
## Detection Rate 0.69775 0.002297 0.002297  
## Detection Prevalence 0.97037 0.005742 0.007350  
## Balanced Accuracy 0.52469 0.506952 0.509686  
## Class: Failed  
## Sensitivity 0.094156  
## Specificity 0.989372  
## Pos Pred Value 0.402778  
## Neg Pred Value 0.934844  
## Prevalence 0.070740  
## Detection Rate 0.006661  
## Detection Prevalence 0.016537  
## Balanced Accuracy 0.541764

The Balance accuracy is better with Random forest with 71.1% or accuray Excellent Class Average Failed Balanced Accuracy 0.52578 0.502751 0.511204 0.541517 Random forest is giving us the best accuracy of all three, altough, they are pretty close.

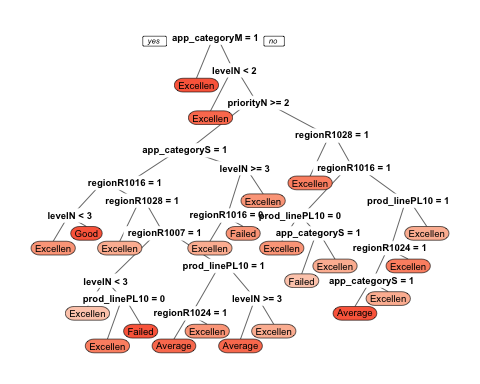
## Plotting Classification and Regression Trees (CART)

All three methods are giving us a close prediction accuracy of 71%. To better understand the process, we will inspect Classification Trees deeper. 

Accuracy decreases after CP = 0.00035. Complexity Parameter is the minimum improvement in the model needed at each node. The cp value is a stopping parameter. It helps speed up the search for splits because it can identify splits that don’t meet this criteria and prune them before going too far.

Let’s look at the Tree how the decision is being calculated:

prp(fit.cart$finalModel, box.palette = "Reds", tweak = 1.2)



## Deployment

Predicting outcome of an unseen data point

## [1] 1

## [1] Excellent  
## Levels: Average Excellent Failed Good

## Conclusion:

Saving the model for the Shyniapp

saveRDS(fit.cart,'jjts\_cart\_model')