# BAN 502 Course Project Phase 1

## Kelly Quesnel

### Exploratory Analysis

#### Libraries

library(tidyverse)  
library(tidymodels)  
library(mice)  
library(VIM)  
library(naniar)  
library(skimr)  
library(UpSetR)  
library(rpart)  
library(rpart.plot)   
library(RColorBrewer)   
library(rattle)  
library(caret)  
library(usemodels)  
library(glmnet)  
library(ROCR)  
library(ranger)   
library(randomForest)  
library(GGally)  
library(gridExtra)  
library(vip)  
library(ggcorrplot)  
library(MASS)  
library(leaps)  
library(lmtest)  
library(splines)  
library(car)  
library(e1071)  
library(arules)  
library(corrplot)  
library(vcd)

#### Data

Read in

train <- read\_csv("train.csv")  
test <- read\_csv("test.csv")

Summarize

str(train)

## spc\_tbl\_ [26,570 × 26] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ id : num [1:26570] 0 1 2 3 4 5 6 7 8 9 ...  
## $ product\_code : chr [1:26570] "A" "A" "A" "A" ...  
## $ loading : num [1:26570] 80.1 84.9 82.4 101.1 188.1 ...  
## $ attribute\_0 : chr [1:26570] "material\_7" "material\_7" "material\_7" "material\_7" ...  
## $ attribute\_1 : chr [1:26570] "material\_8" "material\_8" "material\_8" "material\_8" ...  
## $ attribute\_2 : num [1:26570] 9 9 9 9 9 9 9 9 9 9 ...  
## $ attribute\_3 : num [1:26570] 5 5 5 5 5 5 5 5 5 5 ...  
## $ measurement\_0 : num [1:26570] 7 14 12 13 9 11 12 4 9 10 ...  
## $ measurement\_1 : num [1:26570] 8 3 1 2 2 4 2 8 6 4 ...  
## $ measurement\_2 : num [1:26570] 4 3 5 6 8 0 4 8 5 7 ...  
## $ measurement\_3 : num [1:26570] 18 18.2 18.1 17.3 19.3 ...  
## $ measurement\_4 : num [1:26570] 12.5 11.5 11.7 11.2 12.9 ...  
## $ measurement\_5 : num [1:26570] 15.7 17.7 16.7 18.6 17 ...  
## $ measurement\_6 : num [1:26570] 19.3 17.9 18.2 18.3 15.7 ...  
## $ measurement\_7 : num [1:26570] 11.7 12.7 12.7 12.6 11.3 ...  
## $ measurement\_8 : num [1:26570] 20.2 17.9 18.3 19.1 18.1 ...  
## $ measurement\_9 : num [1:26570] 10.7 12.4 12.7 12.5 10.3 ...  
## $ measurement\_10: num [1:26570] 15.9 17.9 15.6 16.3 17.1 ...  
## $ measurement\_11: num [1:26570] 17.6 17.9 NA 18.4 19.9 ...  
## $ measurement\_12: num [1:26570] 15.2 11.8 13.8 10 12.4 ...  
## $ measurement\_13: num [1:26570] 15 14.7 16.7 15.2 16.2 ...  
## $ measurement\_14: num [1:26570] NA 15.4 18.6 15.6 12.8 ...  
## $ measurement\_15: num [1:26570] 13 14.4 14.1 16.2 13.2 ...  
## $ measurement\_16: num [1:26570] 14.7 15.6 17.9 17.2 16.4 ...  
## $ measurement\_17: num [1:26570] 764 682 663 826 580 ...  
## $ failure : chr [1:26570] "No" "No" "No" "No" ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. id = col\_double(),  
## .. product\_code = col\_character(),  
## .. loading = col\_double(),  
## .. attribute\_0 = col\_character(),  
## .. attribute\_1 = col\_character(),  
## .. attribute\_2 = col\_double(),  
## .. attribute\_3 = col\_double(),  
## .. measurement\_0 = col\_double(),  
## .. measurement\_1 = col\_double(),  
## .. measurement\_2 = col\_double(),  
## .. measurement\_3 = col\_double(),  
## .. measurement\_4 = col\_double(),  
## .. measurement\_5 = col\_double(),  
## .. measurement\_6 = col\_double(),  
## .. measurement\_7 = col\_double(),  
## .. measurement\_8 = col\_double(),  
## .. measurement\_9 = col\_double(),  
## .. measurement\_10 = col\_double(),  
## .. measurement\_11 = col\_double(),  
## .. measurement\_12 = col\_double(),  
## .. measurement\_13 = col\_double(),  
## .. measurement\_14 = col\_double(),  
## .. measurement\_15 = col\_double(),  
## .. measurement\_16 = col\_double(),  
## .. measurement\_17 = col\_double(),  
## .. failure = col\_character()  
## .. )  
## - attr(\*, "problems")=<externalptr>

summary(train)

## id product\_code loading attribute\_0   
## Min. : 0 Length:26570 Min. : 33.16 Length:26570   
## 1st Qu.: 6642 Class :character 1st Qu.: 99.99 Class :character   
## Median :13284 Mode :character Median :122.39 Mode :character   
## Mean :13284 Mean :127.83   
## 3rd Qu.:19927 3rd Qu.:149.15   
## Max. :26569 Max. :385.86   
## NA's :250   
## attribute\_1 attribute\_2 attribute\_3 measurement\_0   
## Length:26570 Min. :5.000 Min. :5.00 Min. : 0.000   
## Class :character 1st Qu.:6.000 1st Qu.:6.00 1st Qu.: 4.000   
## Mode :character Median :6.000 Median :8.00 Median : 7.000   
## Mean :6.754 Mean :7.24 Mean : 7.416   
## 3rd Qu.:8.000 3rd Qu.:8.00 3rd Qu.:10.000   
## Max. :9.000 Max. :9.00 Max. :29.000   
##   
## measurement\_1 measurement\_2 measurement\_3 measurement\_4   
## Min. : 0.000 Min. : 0.000 Min. :13.97 Min. : 8.008   
## 1st Qu.: 5.000 1st Qu.: 4.000 1st Qu.:17.12 1st Qu.:11.051   
## Median : 8.000 Median : 6.000 Median :17.79 Median :11.733   
## Mean : 8.233 Mean : 6.257 Mean :17.79 Mean :11.732   
## 3rd Qu.:11.000 3rd Qu.: 8.000 3rd Qu.:18.47 3rd Qu.:12.410   
## Max. :29.000 Max. :24.000 Max. :21.50 Max. :16.484   
## NA's :381 NA's :538   
## measurement\_5 measurement\_6 measurement\_7 measurement\_8   
## Min. :12.07 Min. :12.71 Min. : 7.968 Min. :15.22   
## 1st Qu.:16.44 1st Qu.:16.84 1st Qu.:11.045 1st Qu.:18.34   
## Median :17.13 Median :17.52 Median :11.712 Median :19.02   
## Mean :17.13 Mean :17.51 Mean :11.717 Mean :19.02   
## 3rd Qu.:17.80 3rd Qu.:18.18 3rd Qu.:12.391 3rd Qu.:19.71   
## Max. :21.43 Max. :21.54 Max. :15.419 Max. :23.81   
## NA's :676 NA's :796 NA's :937 NA's :1048   
## measurement\_9 measurement\_10 measurement\_11 measurement\_12   
## Min. : 7.537 Min. : 9.323 Min. :12.46 Min. : 5.167   
## 1st Qu.:10.757 1st Qu.:15.209 1st Qu.:18.17 1st Qu.:10.703   
## Median :11.430 Median :16.127 Median :19.21 Median :11.717   
## Mean :11.431 Mean :16.118 Mean :19.17 Mean :11.703   
## 3rd Qu.:12.102 3rd Qu.:17.025 3rd Qu.:20.21 3rd Qu.:12.709   
## Max. :15.412 Max. :22.479 Max. :25.64 Max. :17.663   
## NA's :1227 NA's :1300 NA's :1468 NA's :1601   
## measurement\_13 measurement\_14 measurement\_15 measurement\_16   
## Min. :10.89 Min. : 9.14 Min. : 9.104 Min. : 9.701   
## 1st Qu.:14.89 1st Qu.:15.06 1st Qu.:13.957 1st Qu.:15.268   
## Median :15.63 Median :16.04 Median :14.969 Median :16.436   
## Mean :15.65 Mean :16.05 Mean :14.996 Mean :16.461   
## 3rd Qu.:16.37 3rd Qu.:17.08 3rd Qu.:16.018 3rd Qu.:17.628   
## Max. :22.71 Max. :22.30 Max. :21.626 Max. :24.094   
## NA's :1774 NA's :1874 NA's :2009 NA's :2110   
## measurement\_17 failure   
## Min. : 196.8 Length:26570   
## 1st Qu.: 619.0 Class :character   
## Median : 701.0 Mode :character   
## Mean : 701.3   
## 3rd Qu.: 784.1   
## Max. :1312.8   
## NA's :2284

Factor conversion

train <- train %>%  
 mutate\_if(is.character, as\_factor)  
  
str(train)

## tibble [26,570 × 26] (S3: tbl\_df/tbl/data.frame)  
## $ id : num [1:26570] 0 1 2 3 4 5 6 7 8 9 ...  
## $ product\_code : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ loading : num [1:26570] 80.1 84.9 82.4 101.1 188.1 ...  
## $ attribute\_0 : Factor w/ 2 levels "material\_7","material\_5": 1 1 1 1 1 1 1 1 1 1 ...  
## $ attribute\_1 : Factor w/ 3 levels "material\_8","material\_5",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ attribute\_2 : num [1:26570] 9 9 9 9 9 9 9 9 9 9 ...  
## $ attribute\_3 : num [1:26570] 5 5 5 5 5 5 5 5 5 5 ...  
## $ measurement\_0 : num [1:26570] 7 14 12 13 9 11 12 4 9 10 ...  
## $ measurement\_1 : num [1:26570] 8 3 1 2 2 4 2 8 6 4 ...  
## $ measurement\_2 : num [1:26570] 4 3 5 6 8 0 4 8 5 7 ...  
## $ measurement\_3 : num [1:26570] 18 18.2 18.1 17.3 19.3 ...  
## $ measurement\_4 : num [1:26570] 12.5 11.5 11.7 11.2 12.9 ...  
## $ measurement\_5 : num [1:26570] 15.7 17.7 16.7 18.6 17 ...  
## $ measurement\_6 : num [1:26570] 19.3 17.9 18.2 18.3 15.7 ...  
## $ measurement\_7 : num [1:26570] 11.7 12.7 12.7 12.6 11.3 ...  
## $ measurement\_8 : num [1:26570] 20.2 17.9 18.3 19.1 18.1 ...  
## $ measurement\_9 : num [1:26570] 10.7 12.4 12.7 12.5 10.3 ...  
## $ measurement\_10: num [1:26570] 15.9 17.9 15.6 16.3 17.1 ...  
## $ measurement\_11: num [1:26570] 17.6 17.9 NA 18.4 19.9 ...  
## $ measurement\_12: num [1:26570] 15.2 11.8 13.8 10 12.4 ...  
## $ measurement\_13: num [1:26570] 15 14.7 16.7 15.2 16.2 ...  
## $ measurement\_14: num [1:26570] NA 15.4 18.6 15.6 12.8 ...  
## $ measurement\_15: num [1:26570] 13 14.4 14.1 16.2 13.2 ...  
## $ measurement\_16: num [1:26570] 14.7 15.6 17.9 17.2 16.4 ...  
## $ measurement\_17: num [1:26570] 764 682 663 826 580 ...  
## $ failure : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 2 1 1 ...

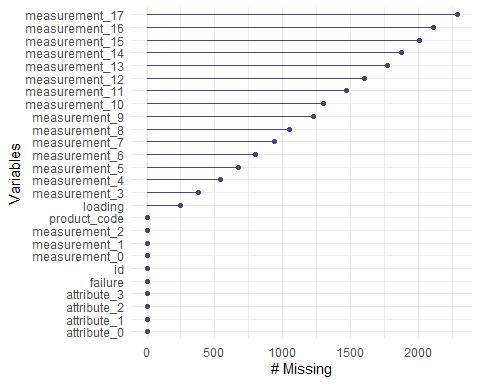
summary(train)

## id product\_code loading attribute\_0   
## Min. : 0 A:5100 Min. : 33.16 material\_7:21320   
## 1st Qu.: 6642 B:5250 1st Qu.: 99.99 material\_5: 5250   
## Median :13284 C:5765 Median :122.39   
## Mean :13284 D:5112 Mean :127.83   
## 3rd Qu.:19927 E:5343 3rd Qu.:149.15   
## Max. :26569 Max. :385.86   
## NA's :250   
## attribute\_1 attribute\_2 attribute\_3 measurement\_0   
## material\_8:10865 Min. :5.000 Min. :5.00 Min. : 0.000   
## material\_5:10362 1st Qu.:6.000 1st Qu.:6.00 1st Qu.: 4.000   
## material\_6: 5343 Median :6.000 Median :8.00 Median : 7.000   
## Mean :6.754 Mean :7.24 Mean : 7.416   
## 3rd Qu.:8.000 3rd Qu.:8.00 3rd Qu.:10.000   
## Max. :9.000 Max. :9.00 Max. :29.000   
##   
## measurement\_1 measurement\_2 measurement\_3 measurement\_4   
## Min. : 0.000 Min. : 0.000 Min. :13.97 Min. : 8.008   
## 1st Qu.: 5.000 1st Qu.: 4.000 1st Qu.:17.12 1st Qu.:11.051   
## Median : 8.000 Median : 6.000 Median :17.79 Median :11.733   
## Mean : 8.233 Mean : 6.257 Mean :17.79 Mean :11.732   
## 3rd Qu.:11.000 3rd Qu.: 8.000 3rd Qu.:18.47 3rd Qu.:12.410   
## Max. :29.000 Max. :24.000 Max. :21.50 Max. :16.484   
## NA's :381 NA's :538   
## measurement\_5 measurement\_6 measurement\_7 measurement\_8   
## Min. :12.07 Min. :12.71 Min. : 7.968 Min. :15.22   
## 1st Qu.:16.44 1st Qu.:16.84 1st Qu.:11.045 1st Qu.:18.34   
## Median :17.13 Median :17.52 Median :11.712 Median :19.02   
## Mean :17.13 Mean :17.51 Mean :11.717 Mean :19.02   
## 3rd Qu.:17.80 3rd Qu.:18.18 3rd Qu.:12.391 3rd Qu.:19.71   
## Max. :21.43 Max. :21.54 Max. :15.419 Max. :23.81   
## NA's :676 NA's :796 NA's :937 NA's :1048   
## measurement\_9 measurement\_10 measurement\_11 measurement\_12   
## Min. : 7.537 Min. : 9.323 Min. :12.46 Min. : 5.167   
## 1st Qu.:10.757 1st Qu.:15.209 1st Qu.:18.17 1st Qu.:10.703   
## Median :11.430 Median :16.127 Median :19.21 Median :11.717   
## Mean :11.431 Mean :16.118 Mean :19.17 Mean :11.703   
## 3rd Qu.:12.102 3rd Qu.:17.025 3rd Qu.:20.21 3rd Qu.:12.709   
## Max. :15.412 Max. :22.479 Max. :25.64 Max. :17.663   
## NA's :1227 NA's :1300 NA's :1468 NA's :1601   
## measurement\_13 measurement\_14 measurement\_15 measurement\_16   
## Min. :10.89 Min. : 9.14 Min. : 9.104 Min. : 9.701   
## 1st Qu.:14.89 1st Qu.:15.06 1st Qu.:13.957 1st Qu.:15.268   
## Median :15.63 Median :16.04 Median :14.969 Median :16.436   
## Mean :15.65 Mean :16.05 Mean :14.996 Mean :16.461   
## 3rd Qu.:16.37 3rd Qu.:17.08 3rd Qu.:16.018 3rd Qu.:17.628   
## Max. :22.71 Max. :22.30 Max. :21.626 Max. :24.094   
## NA's :1774 NA's :1874 NA's :2009 NA's :2110   
## measurement\_17 failure   
## Min. : 196.8 No :20921   
## 1st Qu.: 619.0 Yes: 5649   
## Median : 701.0   
## Mean : 701.3   
## 3rd Qu.: 784.1   
## Max. :1312.8   
## NA's :2284

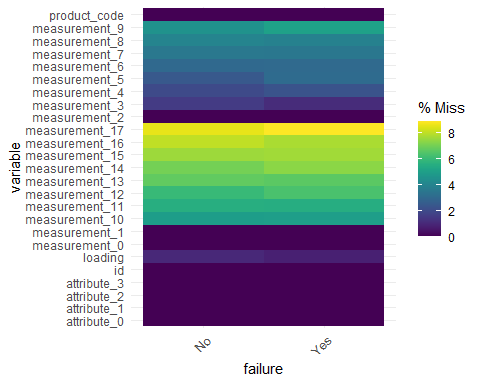
#### Deal with missing values

Visualize missing data

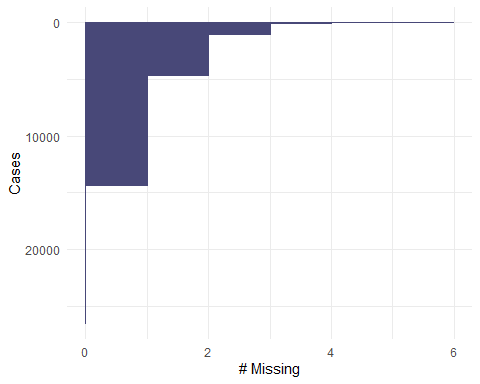
gg\_miss\_var(train)



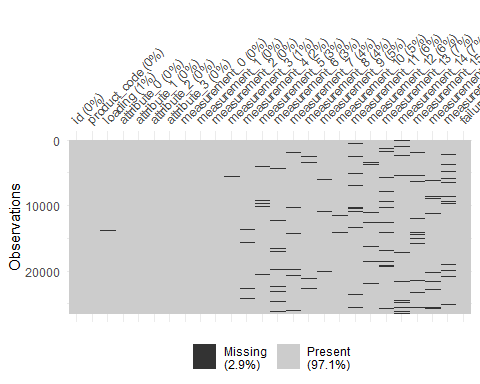
gg\_miss\_fct(x = train, fct = failure)



gg\_miss\_case(train)



vis\_miss(train)



Imputation

set.seed(1234)  
imp\_measures <-  
 mice(train, m=5, method='pmm', printFlag=FALSE)

## Warning: Number of logged events: 756

summary(imp\_measures)

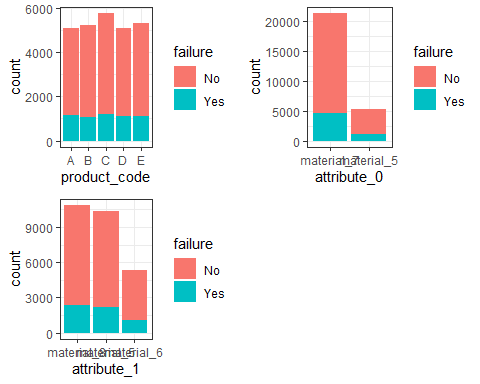
## Class: mids  
## Number of multiple imputations: 5   
## Imputation methods:  
## id product\_code loading attribute\_0 attribute\_1   
## "" "" "pmm" "" ""   
## attribute\_2 attribute\_3 measurement\_0 measurement\_1 measurement\_2   
## "" "" "" "" ""   
## measurement\_3 measurement\_4 measurement\_5 measurement\_6 measurement\_7   
## "pmm" "pmm" "pmm" "pmm" "pmm"   
## measurement\_8 measurement\_9 measurement\_10 measurement\_11 measurement\_12   
## "pmm" "pmm" "pmm" "pmm" "pmm"   
## measurement\_13 measurement\_14 measurement\_15 measurement\_16 measurement\_17   
## "pmm" "pmm" "pmm" "pmm" "pmm"   
## failure   
## ""   
## PredictorMatrix:  
## id product\_code loading attribute\_0 attribute\_1 attribute\_2  
## id 0 1 1 1 1 1  
## product\_code 1 0 1 1 1 1  
## loading 1 1 0 1 1 1  
## attribute\_0 1 1 1 0 1 1  
## attribute\_1 1 1 1 1 0 1  
## attribute\_2 1 1 1 1 1 0  
## attribute\_3 measurement\_0 measurement\_1 measurement\_2  
## id 1 1 1 1  
## product\_code 1 1 1 1  
## loading 1 1 1 1  
## attribute\_0 1 1 1 1  
## attribute\_1 1 1 1 1  
## attribute\_2 1 1 1 1  
## measurement\_3 measurement\_4 measurement\_5 measurement\_6  
## id 1 1 1 1  
## product\_code 1 1 1 1  
## loading 1 1 1 1  
## attribute\_0 1 1 1 1  
## attribute\_1 1 1 1 1  
## attribute\_2 1 1 1 1  
## measurement\_7 measurement\_8 measurement\_9 measurement\_10  
## id 1 1 1 1  
## product\_code 1 1 1 1  
## loading 1 1 1 1  
## attribute\_0 1 1 1 1  
## attribute\_1 1 1 1 1  
## attribute\_2 1 1 1 1  
## measurement\_11 measurement\_12 measurement\_13 measurement\_14  
## id 1 1 1 1  
## product\_code 1 1 1 1  
## loading 1 1 1 1  
## attribute\_0 1 1 1 1  
## attribute\_1 1 1 1 1  
## attribute\_2 1 1 1 1  
## measurement\_15 measurement\_16 measurement\_17 failure  
## id 1 1 1 1  
## product\_code 1 1 1 1  
## loading 1 1 1 1  
## attribute\_0 1 1 1 1  
## attribute\_1 1 1 1 1  
## attribute\_2 1 1 1 1  
## Number of logged events: 756   
## it im dep meth  
## 1 1 1 loading pmm  
## 2 1 1 loading pmm  
## 3 1 1 measurement\_3 pmm  
## 4 1 1 measurement\_3 pmm  
## 5 1 1 measurement\_4 pmm  
## 6 1 1 measurement\_4 pmm  
## out  
## 1 product\_codeB, product\_codeD, attribute\_0material\_5, attribute\_1material\_5, attribute\_3  
## 2 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, but the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique respondent names?  
## 3 product\_codeB, loading, attribute\_1material\_6, measurement\_0, measurement\_4  
## 4 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, but the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique respondent names?  
## 5 product\_codeB, attribute\_1material\_5, attribute\_1material\_6, measurement\_0, measurement\_2  
## 6 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, but the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique respondent names?

train\_complete <-  
 complete(imp\_measures)  
  
summary(train\_complete)

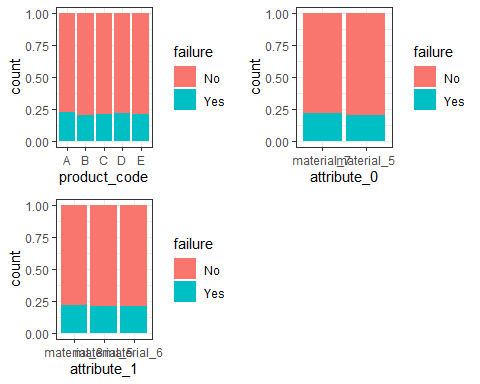
## id product\_code loading attribute\_0   
## Min. : 0 A:5100 Min. : 33.16 material\_7:21320   
## 1st Qu.: 6642 B:5250 1st Qu.: 99.96 material\_5: 5250   
## Median :13284 C:5765 Median :122.36   
## Mean :13284 D:5112 Mean :127.82   
## 3rd Qu.:19927 E:5343 3rd Qu.:149.16   
## Max. :26569 Max. :385.86   
## attribute\_1 attribute\_2 attribute\_3 measurement\_0   
## material\_8:10865 Min. :5.000 Min. :5.00 Min. : 0.000   
## material\_5:10362 1st Qu.:6.000 1st Qu.:6.00 1st Qu.: 4.000   
## material\_6: 5343 Median :6.000 Median :8.00 Median : 7.000   
## Mean :6.754 Mean :7.24 Mean : 7.416   
## 3rd Qu.:8.000 3rd Qu.:8.00 3rd Qu.:10.000   
## Max. :9.000 Max. :9.00 Max. :29.000   
## measurement\_1 measurement\_2 measurement\_3 measurement\_4   
## Min. : 0.000 Min. : 0.000 Min. :13.97 Min. : 8.008   
## 1st Qu.: 5.000 1st Qu.: 4.000 1st Qu.:17.12 1st Qu.:11.013   
## Median : 8.000 Median : 6.000 Median :17.79 Median :11.716   
## Mean : 8.233 Mean : 6.257 Mean :17.80 Mean :11.710   
## 3rd Qu.:11.000 3rd Qu.: 8.000 3rd Qu.:18.48 3rd Qu.:12.405   
## Max. :29.000 Max. :24.000 Max. :21.50 Max. :16.484   
## measurement\_5 measurement\_6 measurement\_7 measurement\_8   
## Min. :12.07 Min. :12.71 Min. : 7.968 Min. :15.22   
## 1st Qu.:16.39 1st Qu.:16.84 1st Qu.:11.049 1st Qu.:18.38   
## Median :17.10 Median :17.52 Median :11.713 Median :19.07   
## Mean :17.08 Mean :17.51 Mean :11.717 Mean :19.12   
## 3rd Qu.:17.79 3rd Qu.:18.18 3rd Qu.:12.391 3rd Qu.:19.81   
## Max. :21.43 Max. :21.54 Max. :15.419 Max. :23.81   
## measurement\_9 measurement\_10 measurement\_11 measurement\_12   
## Min. : 7.537 Min. : 9.323 Min. :12.46 Min. : 5.167   
## 1st Qu.:10.734 1st Qu.:15.173 1st Qu.:18.24 1st Qu.:10.715   
## Median :11.420 Median :16.110 Median :19.32 Median :11.690   
## Mean :11.414 Mean :16.083 Mean :19.28 Mean :11.695   
## 3rd Qu.:12.065 3rd Qu.:16.975 3rd Qu.:20.35 3rd Qu.:12.699   
## Max. :15.412 Max. :22.479 Max. :25.64 Max. :17.663   
## measurement\_13 measurement\_14 measurement\_15 measurement\_16   
## Min. :10.89 Min. : 9.14 Min. : 9.104 Min. : 9.701   
## 1st Qu.:14.85 1st Qu.:15.08 1st Qu.:13.820 1st Qu.:15.263   
## Median :15.65 Median :16.07 Median :14.806 Median :16.534   
## Mean :15.64 Mean :16.09 Mean :14.812 Mean :16.718   
## 3rd Qu.:16.31 3rd Qu.:17.13 3rd Qu.:15.918 3rd Qu.:17.846   
## Max. :22.71 Max. :22.30 Max. :21.626 Max. :24.094   
## measurement\_17 failure   
## Min. : 196.8 No :20921   
## 1st Qu.: 614.1 Yes: 5649   
## Median : 698.6   
## Mean : 697.3   
## 3rd Qu.: 783.6   
## Max. :1312.8

#### Visualize and explore relationships

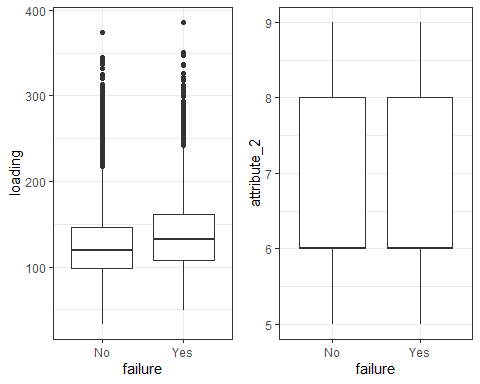
p1 <-  
 ggplot(train\_complete, aes(x=product\_code, fill = failure)) +  
 geom\_bar() +  
 theme\_bw()  
  
p2 <-  
 ggplot(train\_complete, aes(x=attribute\_0, fill = failure)) +  
 geom\_bar() +  
 theme\_bw()  
  
p3 <-   
 ggplot(train\_complete, aes(x=attribute\_1, fill = failure)) +  
 geom\_bar() +  
 theme\_bw()  
  
grid.arrange(p1,p2,p3,ncol=2)



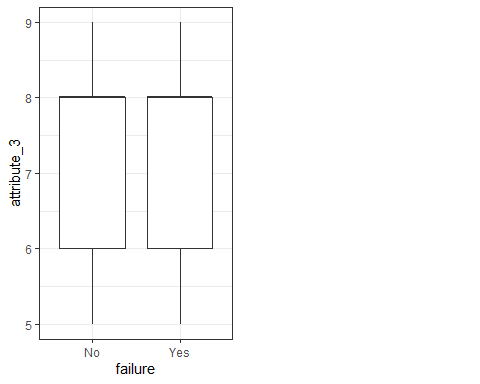
p1 <-  
 ggplot(train\_complete, aes(x=product\_code, fill = failure)) +  
 geom\_bar(position = "fill") +  
 theme\_bw()  
  
p2 <-   
 ggplot(train\_complete, aes(x=attribute\_0, fill = failure)) +  
 geom\_bar(position = "fill") +  
 theme\_bw()  
  
p3 <-   
 ggplot(train\_complete, aes(x=attribute\_1, fill = failure)) +  
 geom\_bar(position = "fill") +  
 theme\_bw()  
  
grid.arrange(p1,p2,p3,ncol=2)



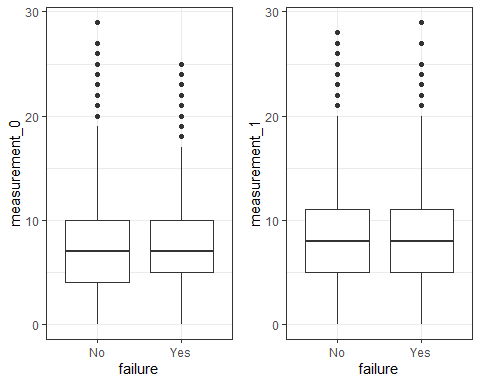
p1 <-  
 ggplot(train\_complete, aes(x = failure, y = loading)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
p2 <-  
 ggplot(train\_complete, aes(x = failure, y = attribute\_2)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
grid.arrange(p1,p2,ncol=2)



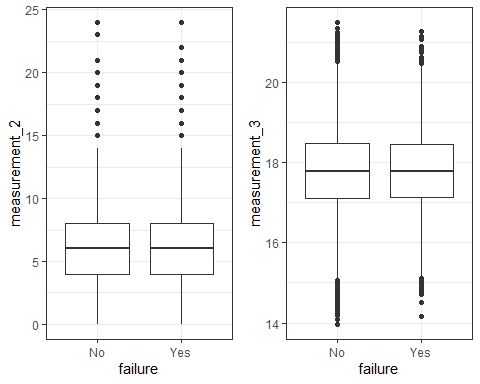
p3 <-  
 ggplot(train\_complete, aes(x = failure, y = attribute\_3)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
grid.arrange(p3,ncol=2)



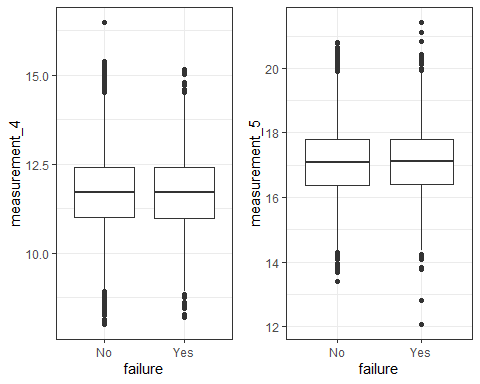
p1 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_0)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
p2 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_1)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
grid.arrange(p1,p2,ncol=2)



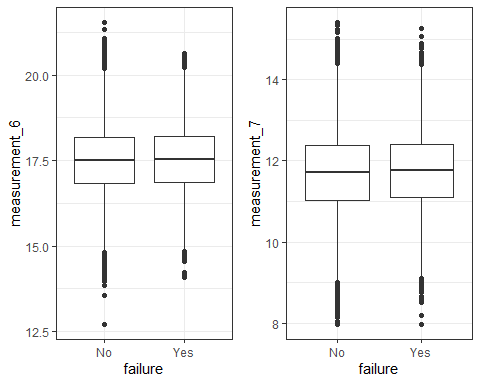
p3 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_2)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
p4 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_3)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
grid.arrange(p3,p4,ncol=2)



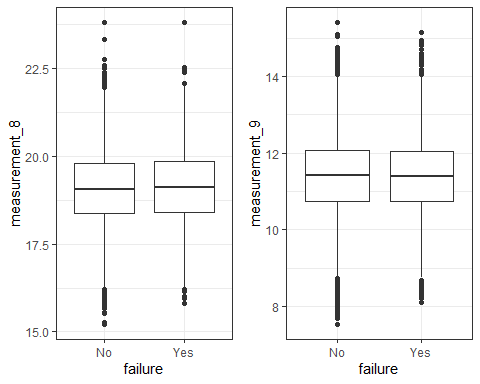
p1 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_4)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
p2 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_5)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
grid.arrange(p1,p2,ncol=2)



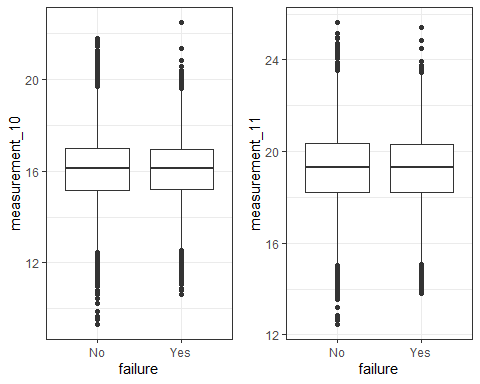
p3 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_6)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
p4 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_7)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
grid.arrange(p3,p4,ncol=2)



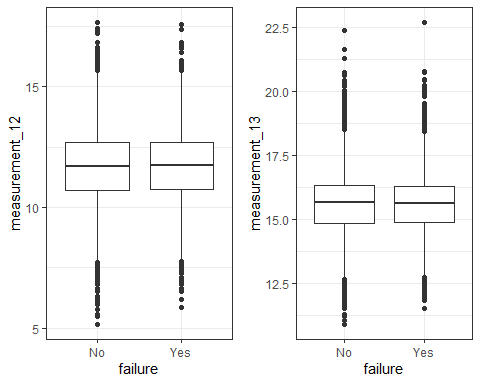
p1 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_8)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
p2 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_9)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
grid.arrange(p1,p2,ncol=2)



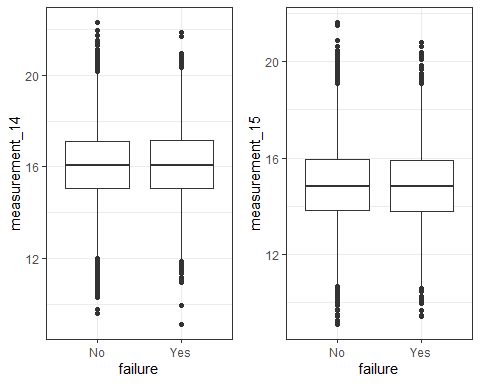
p3 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_10)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
p4 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_11)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
grid.arrange(p3,p4,ncol=2)



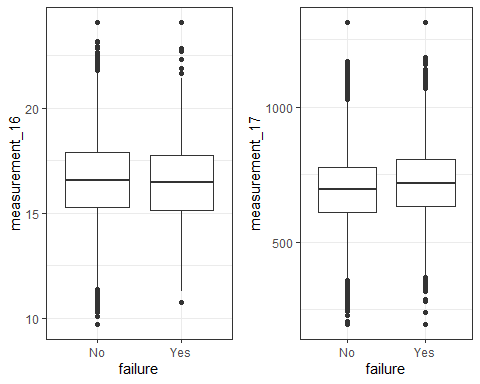
p1 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_12)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
p2 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_13)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
grid.arrange(p1,p2,ncol=2)



p3 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_14)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
p4 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_15)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
grid.arrange(p3,p4,ncol=2)

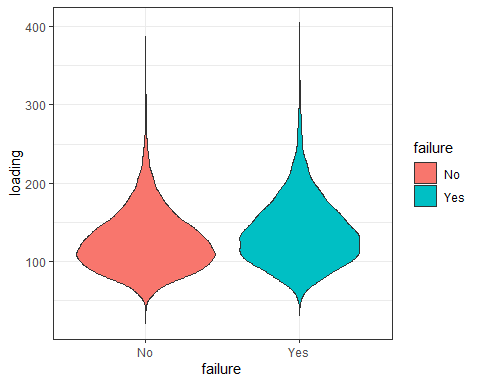


p1 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_16)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
p2 <-  
 ggplot(train\_complete, aes(x = failure, y = measurement\_17)) +  
 geom\_boxplot() +  
 theme\_bw()  
  
grid.arrange(p1,p2,ncol=2)

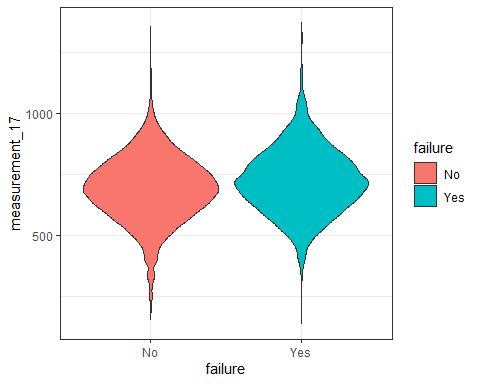


Look at a few variables of possible interest

ggplot(train\_complete, aes(x = failure, y = loading, fill = failure)) +   
 geom\_violin(trim = FALSE) +  
 theme\_bw()



ggplot(train\_complete, aes(x = failure, y = measurement\_17, fill = failure)) +   
 geom\_violin(trim = FALSE) +  
 theme\_bw()



Look at summary stats with groups split by failure

train\_failyes <- train\_complete[(which(train\_complete$failure %in% "Yes")),]  
train\_failno <- train\_complete[(which(train\_complete$failure %in% "No")),]

summary(train\_failyes)

## id product\_code loading attribute\_0   
## Min. : 6 A:1159 Min. : 49.64 material\_7:4597   
## 1st Qu.: 6282 B:1052 1st Qu.:107.29 material\_5:1052   
## Median :13182 C:1220 Median :131.83   
## Mean :13173 D:1112 Mean :137.44   
## 3rd Qu.:19840 E:1106 3rd Qu.:161.05   
## Max. :26559 Max. :385.86   
## attribute\_1 attribute\_2 attribute\_3 measurement\_0   
## material\_8:2379 Min. :5.000 Min. :5.000 Min. : 0.000   
## material\_5:2164 1st Qu.:6.000 1st Qu.:6.000 1st Qu.: 5.000   
## material\_6:1106 Median :6.000 Median :8.000 Median : 7.000   
## Mean :6.772 Mean :7.187 Mean : 7.492   
## 3rd Qu.:8.000 3rd Qu.:8.000 3rd Qu.:10.000   
## Max. :9.000 Max. :9.000 Max. :25.000   
## measurement\_1 measurement\_2 measurement\_3 measurement\_4   
## Min. : 0.000 Min. : 0.000 Min. :14.17 Min. : 8.196   
## 1st Qu.: 5.000 1st Qu.: 4.000 1st Qu.:17.13 1st Qu.:10.985   
## Median : 8.000 Median : 6.000 Median :17.79 Median :11.710   
## Mean : 8.145 Mean : 6.357 Mean :17.81 Mean :11.689   
## 3rd Qu.:11.000 3rd Qu.: 8.000 3rd Qu.:18.46 3rd Qu.:12.398   
## Max. :29.000 Max. :24.000 Max. :21.27 Max. :15.164   
## measurement\_5 measurement\_6 measurement\_7 measurement\_8   
## Min. :12.07 Min. :14.09 Min. : 7.968 Min. :15.82   
## 1st Qu.:16.40 1st Qu.:16.87 1st Qu.:11.099 1st Qu.:18.40   
## Median :17.13 Median :17.55 Median :11.758 Median :19.11   
## Mean :17.10 Mean :17.54 Mean :11.754 Mean :19.15   
## 3rd Qu.:17.80 3rd Qu.:18.20 3rd Qu.:12.416 3rd Qu.:19.85   
## Max. :21.43 Max. :20.62 Max. :15.269 Max. :23.81   
## measurement\_9 measurement\_10 measurement\_11 measurement\_12   
## Min. : 8.103 Min. :10.63 Min. :13.80 Min. : 5.867   
## 1st Qu.:10.729 1st Qu.:15.19 1st Qu.:18.24 1st Qu.:10.728   
## Median :11.406 Median :16.10 Median :19.31 Median :11.729   
## Mean :11.407 Mean :16.08 Mean :19.27 Mean :11.710   
## 3rd Qu.:12.054 3rd Qu.:16.95 3rd Qu.:20.32 3rd Qu.:12.703   
## Max. :15.154 Max. :22.48 Max. :25.43 Max. :17.594   
## measurement\_13 measurement\_14 measurement\_15 measurement\_16   
## Min. :11.50 Min. : 9.14 Min. : 9.425 Min. :10.73   
## 1st Qu.:14.87 1st Qu.:15.08 1st Qu.:13.779 1st Qu.:15.15   
## Median :15.63 Median :16.07 Median :14.804 Median :16.45   
## Mean :15.63 Mean :16.11 Mean :14.795 Mean :16.46   
## 3rd Qu.:16.30 3rd Qu.:17.18 3rd Qu.:15.888 3rd Qu.:17.74   
## Max. :22.71 Max. :21.85 Max. :20.784 Max. :24.09   
## measurement\_17 failure   
## Min. : 196.8 No : 0   
## 1st Qu.: 634.1 Yes:5649   
## Median : 717.8   
## Mean : 722.5   
## 3rd Qu.: 807.5   
## Max. :1312.8

summary(train\_failno)

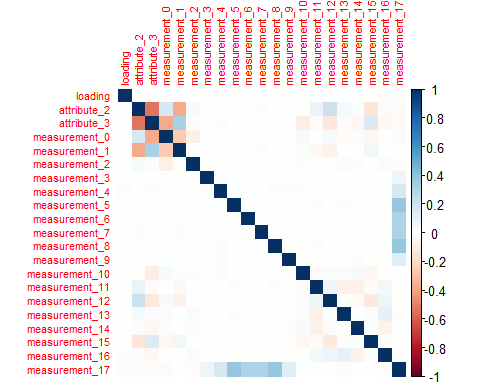
## id product\_code loading attribute\_0   
## Min. : 0 A:3941 Min. : 33.16 material\_7:16723   
## 1st Qu.: 6733 B:4198 1st Qu.: 98.09 material\_5: 4198   
## Median :13309 C:4545 Median :119.90   
## Mean :13315 D:4000 Mean :125.22   
## 3rd Qu.:19955 E:4237 3rd Qu.:145.94   
## Max. :26569 Max. :374.33   
## attribute\_1 attribute\_2 attribute\_3 measurement\_0   
## material\_8:8486 Min. :5.000 Min. :5.000 Min. : 0.000   
## material\_5:8198 1st Qu.:6.000 1st Qu.:6.000 1st Qu.: 4.000   
## material\_6:4237 Median :6.000 Median :8.000 Median : 7.000   
## Mean :6.749 Mean :7.255 Mean : 7.395   
## 3rd Qu.:8.000 3rd Qu.:8.000 3rd Qu.:10.000   
## Max. :9.000 Max. :9.000 Max. :29.000   
## measurement\_1 measurement\_2 measurement\_3 measurement\_4   
## Min. : 0.000 Min. : 0.000 Min. :13.97 Min. : 8.008   
## 1st Qu.: 5.000 1st Qu.: 4.000 1st Qu.:17.12 1st Qu.:11.020   
## Median : 8.000 Median : 6.000 Median :17.79 Median :11.719   
## Mean : 8.256 Mean : 6.229 Mean :17.80 Mean :11.715   
## 3rd Qu.:11.000 3rd Qu.: 8.000 3rd Qu.:18.48 3rd Qu.:12.407   
## Max. :28.000 Max. :24.000 Max. :21.50 Max. :16.484   
## measurement\_5 measurement\_6 measurement\_7 measurement\_8   
## Min. :13.39 Min. :12.71 Min. : 7.973 Min. :15.22   
## 1st Qu.:16.39 1st Qu.:16.83 1st Qu.:11.036 1st Qu.:18.37   
## Median :17.10 Median :17.51 Median :11.705 Median :19.06   
## Mean :17.08 Mean :17.50 Mean :11.708 Mean :19.11   
## 3rd Qu.:17.79 3rd Qu.:18.17 3rd Qu.:12.383 3rd Qu.:19.80   
## Max. :20.79 Max. :21.54 Max. :15.419 Max. :23.81   
## measurement\_9 measurement\_10 measurement\_11 measurement\_12   
## Min. : 7.537 Min. : 9.323 Min. :12.46 Min. : 5.167   
## 1st Qu.:10.736 1st Qu.:15.168 1st Qu.:18.24 1st Qu.:10.714   
## Median :11.423 Median :16.113 Median :19.32 Median :11.681   
## Mean :11.415 Mean :16.083 Mean :19.28 Mean :11.691   
## 3rd Qu.:12.067 3rd Qu.:16.982 3rd Qu.:20.36 3rd Qu.:12.699   
## Max. :15.412 Max. :21.761 Max. :25.64 Max. :17.663   
## measurement\_13 measurement\_14 measurement\_15 measurement\_16   
## Min. :10.89 Min. : 9.593 Min. : 9.104 Min. : 9.701   
## 1st Qu.:14.85 1st Qu.:15.075 1st Qu.:13.826 1st Qu.:15.287   
## Median :15.65 Median :16.076 Median :14.808 Median :16.555   
## Mean :15.64 Mean :16.086 Mean :14.817 Mean :16.789   
## 3rd Qu.:16.31 3rd Qu.:17.115 3rd Qu.:15.926 3rd Qu.:17.882   
## Max. :22.39 Max. :22.303 Max. :21.626 Max. :24.094   
## measurement\_17 failure   
## Min. : 196.8 No :20921   
## 1st Qu.: 609.4 Yes: 0   
## Median : 693.5   
## Mean : 690.5   
## 3rd Qu.: 776.5   
## Max. :1312.8

Split numerical and categorical for correlation analysis

num\_vars <- train\_complete %>%  
 select\_if(is.numeric) %>%  
 dplyr::select(-id)  
  
cat\_vars <- train\_complete %>%  
 select\_if(is.factor)

Correlation matrix of numerical variables

cor\_matrix <- cor(num\_vars, use = "pairwise.complete.obs")  
corrplot(cor\_matrix, method = "color", tl.cex = 0.7)



Chi-squared test for categorical variables

for(var in names(cat\_vars)) {  
 print(var)  
 print(chisq.test(table(cat\_vars[[var]], train\_complete$failure)))  
}

## [1] "product\_code"  
##   
## Pearson's Chi-squared test  
##   
## data: table(cat\_vars[[var]], train\_complete$failure)  
## X-squared = 13, df = 4, p-value = 0.01127  
##   
## [1] "attribute\_0"  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(cat\_vars[[var]], train\_complete$failure)  
## X-squared = 5.7525, df = 1, p-value = 0.01646  
##   
## [1] "attribute\_1"  
##   
## Pearson's Chi-squared test  
##   
## data: table(cat\_vars[[var]], train\_complete$failure)  
## X-squared = 4.5013, df = 2, p-value = 0.1053  
##   
## [1] "failure"  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(cat\_vars[[var]], train\_complete$failure)  
## X-squared = 26564, df = 1, p-value < 2.2e-16

# BAN 502 Course Project Phase 2

## Kelly Quesnel

### Model Building and Predictions

#### Libraries

library(tidyverse)  
library(tidymodels)  
library(mice)  
library(VIM)  
library(naniar)  
library(skimr)  
library(UpSetR)  
library(rpart)  
library(rpart.plot)   
library(RColorBrewer)   
library(rattle)  
library(caret)  
library(usemodels)  
library(glmnet)  
library(ROCR)  
library(ranger)   
library(randomForest)  
library(GGally)  
library(gridExtra)  
library(vip)  
library(ggcorrplot)  
library(MASS)  
library(leaps)  
library(lmtest)  
library(splines)  
library(car)  
library(e1071)  
library(arules)  
library(corrplot)  
library(vcd)  
library(doParallel)  
library(themis)

#### Copy to new data frame and rename for clarity

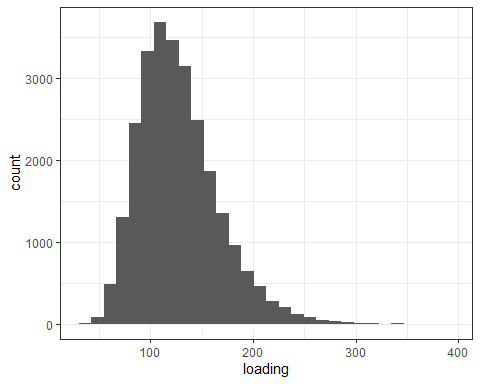
dataset <- train\_complete

#### Look for Outliers

*Going back to exploratory analysis for a moment to explore and deal with outliers as suggested.*

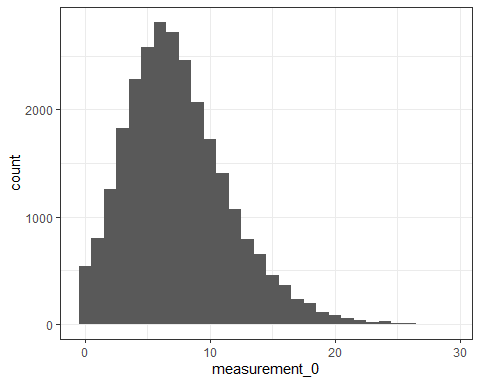
ggplot(dataset, aes(x=loading)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



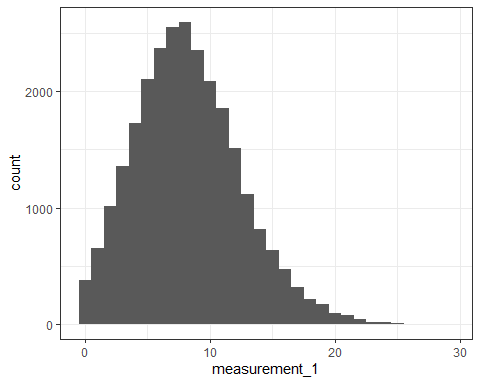
ggplot(dataset, aes(x=measurement\_0)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



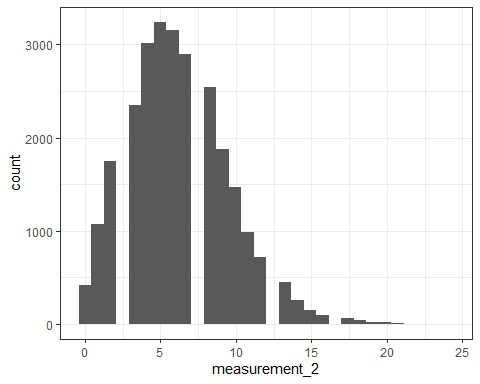
ggplot(dataset, aes(x=measurement\_1)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



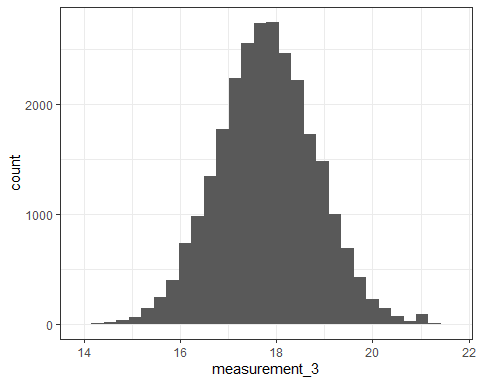
ggplot(dataset, aes(x=measurement\_2)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



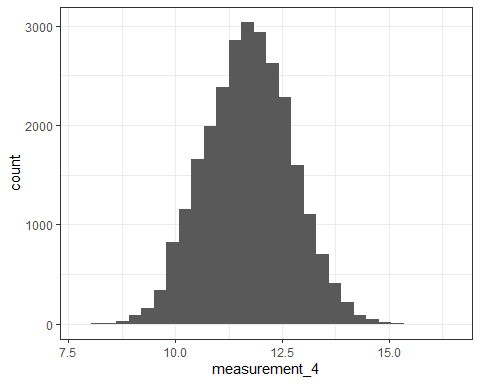
ggplot(dataset, aes(x=measurement\_3)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



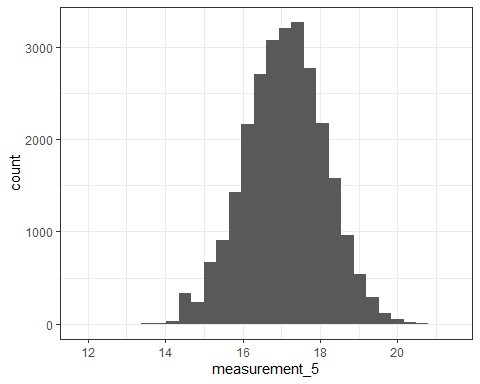
ggplot(dataset, aes(x=measurement\_4)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



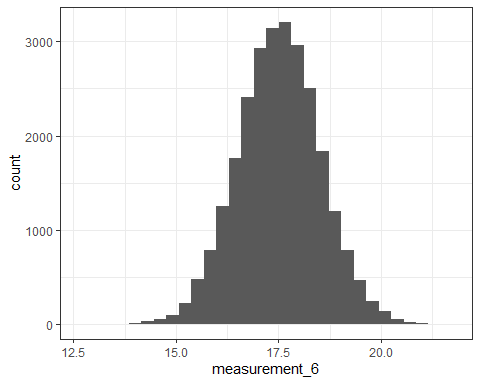
ggplot(dataset, aes(x=measurement\_5)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



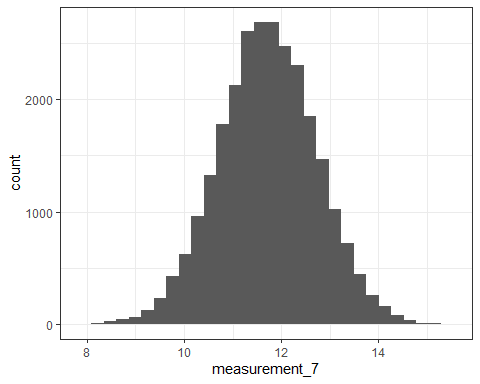
ggplot(dataset, aes(x=measurement\_6)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



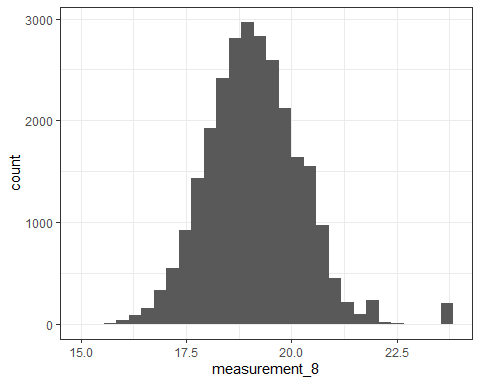
ggplot(dataset, aes(x=measurement\_7)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



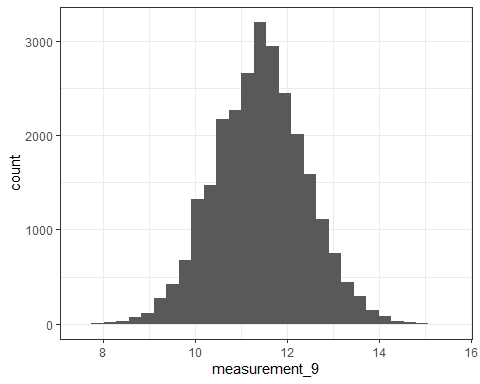
ggplot(dataset, aes(x=measurement\_8)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



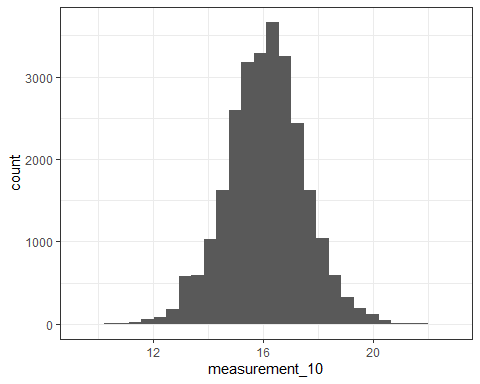
ggplot(dataset, aes(x=measurement\_9)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



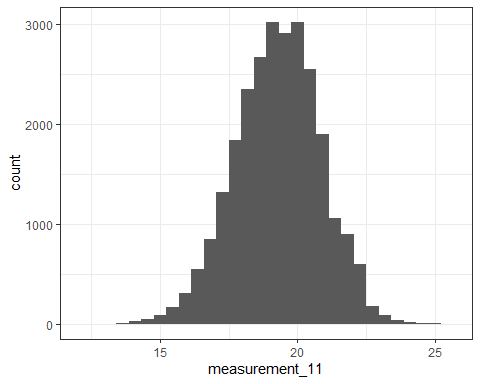
ggplot(dataset, aes(x=measurement\_10)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



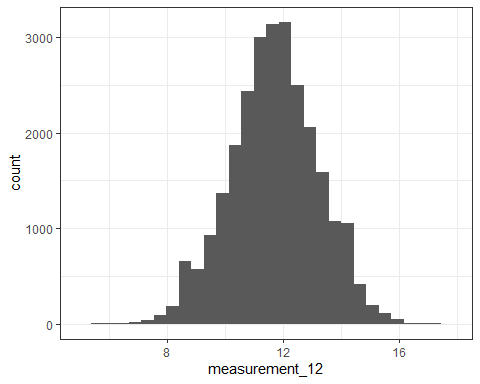
ggplot(dataset, aes(x=measurement\_11)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



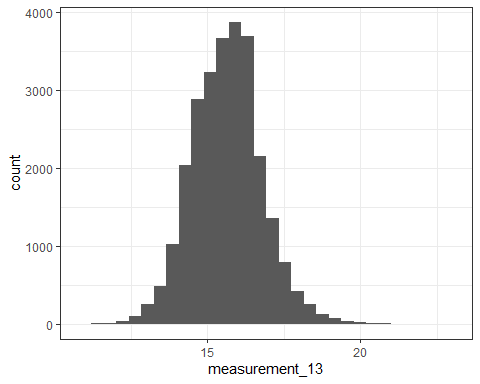
ggplot(dataset, aes(x=measurement\_12)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



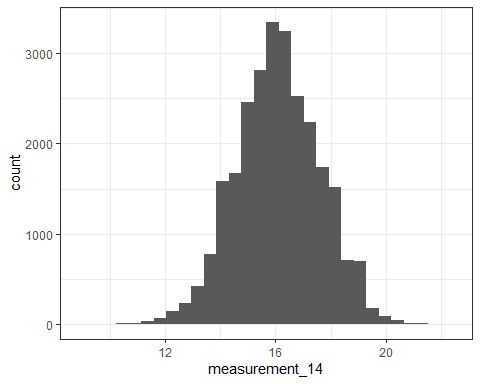
ggplot(dataset, aes(x=measurement\_13)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



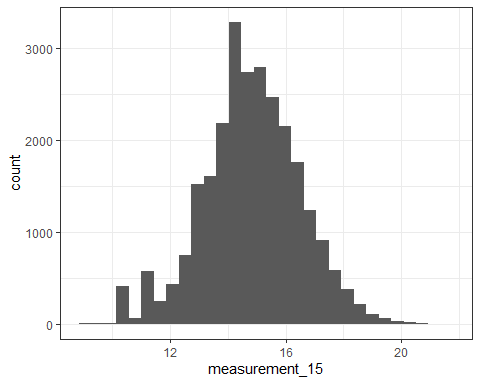
ggplot(dataset, aes(x=measurement\_14)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



ggplot(dataset, aes(x=measurement\_15)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



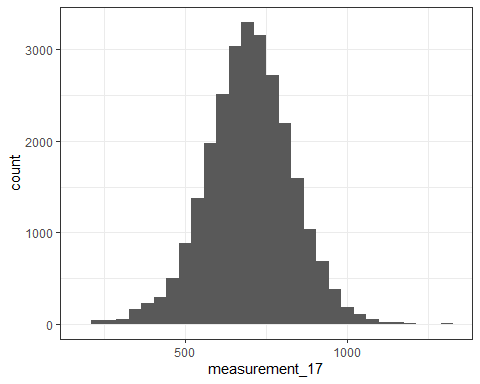
ggplot(dataset, aes(x=measurement\_16)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



ggplot(dataset, aes(x=measurement\_17)) +  
 geom\_histogram() +  
 theme\_bw()

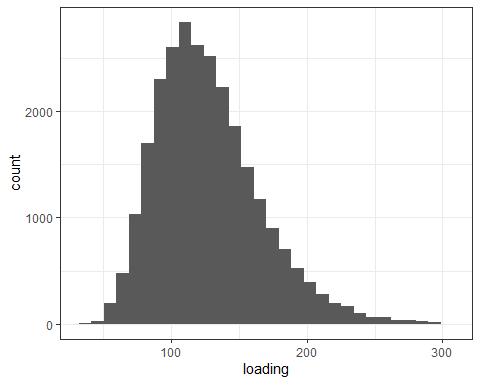
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



*Not immediately concerned with anything except perhaps loading.*

dataset <- dataset %>%  
 filter(loading < 300)  
  
ggplot(dataset, aes(x=loading)) +  
 geom\_histogram() +  
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



#### Remove ID column

dataset <- dataset %>%  
 dplyr::select(-id)

#### Split data for modeling

set.seed(123)  
data\_split <- initial\_split(dataset, prop = 0.80, strata = failure)  
data\_train <- training(data\_split)  
data\_test <- testing(data\_split)

#### Create k folds

set.seed(123)  
folds <- vfold\_cv(data\_train, v = 5)

#### Logistic Regression

Build model

model\_logreg <-  
 logistic\_reg(mode = "classification") %>%  
 set\_engine("glm")  
  
recipe\_logreg <-  
 recipe(failure ~., data\_train)  
  
wf\_logreg <-  
 workflow() %>%  
 add\_recipe(recipe\_logreg) %>%  
 add\_model(model\_logreg)  
  
set.seed(123)  
cv\_results <-   
 tune\_grid(wf\_logreg,  
 resamples = folds)

## Warning: No tuning parameters have been detected, performance will be evaluated  
## using the resamples with no tuning. Did you want to [tune()] parameters?

fit\_logreg <-  
 fit(wf\_logreg, data\_train)

Summarize model

summary(fit\_logreg$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Coefficients: (5 not defined because of singularities)  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 6.0824432 1.0143409 5.996 2.02e-09 \*\*\*  
## product\_codeB -0.0794745 0.0627645 -1.266 0.205429   
## product\_codeC -0.0766105 0.0598599 -1.280 0.200605   
## product\_codeD -0.0628186 0.0627333 -1.001 0.316653   
## product\_codeE -0.1346088 0.0639194 -2.106 0.035212 \*   
## loading 0.0074916 0.0004255 17.605 < 2e-16 \*\*\*  
## attribute\_0material\_5 NA NA NA NA   
## attribute\_1material\_5 NA NA NA NA   
## attribute\_1material\_6 NA NA NA NA   
## attribute\_2 NA NA NA NA   
## attribute\_3 NA NA NA NA   
## measurement\_0 0.0010453 0.0045701 0.229 0.819089   
## measurement\_1 -0.0010678 0.0047153 -0.226 0.820854   
## measurement\_2 0.0160031 0.0054431 2.940 0.003281 \*\*   
## measurement\_3 -0.0171207 0.0169991 -1.007 0.313862   
## measurement\_4 -0.0876981 0.0174640 -5.022 5.12e-07 \*\*\*  
## measurement\_5 -0.1463659 0.0191176 -7.656 1.92e-14 \*\*\*  
## measurement\_6 -0.0965416 0.0189829 -5.086 3.66e-07 \*\*\*  
## measurement\_7 -0.0957374 0.0189881 -5.042 4.61e-07 \*\*\*  
## measurement\_8 -0.1274277 0.0183484 -6.945 3.79e-12 \*\*\*  
## measurement\_9 -0.0622270 0.0177282 -3.510 0.000448 \*\*\*  
## measurement\_10 -0.0165795 0.0122765 -1.351 0.176853   
## measurement\_11 -0.0051708 0.0112956 -0.458 0.647114   
## measurement\_12 0.0182019 0.0117273 1.552 0.120640   
## measurement\_13 0.0291015 0.0153363 1.898 0.057754 .   
## measurement\_14 -0.0009377 0.0113743 -0.082 0.934294   
## measurement\_15 -0.0018697 0.0104080 -0.180 0.857433   
## measurement\_16 -0.0796316 0.0085957 -9.264 < 2e-16 \*\*\*  
## measurement\_17 0.0034741 0.0002014 17.252 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21943 on 21224 degrees of freedom  
## Residual deviance: 21179 on 21201 degrees of freedom  
## AIC: 21227  
##   
## Number of Fisher Scoring iterations: 4

Predict on training

logreg\_train\_pred <- predict(fit\_logreg, data\_train, type = "class")  
  
logreg\_train\_results <- data\_train %>%  
 mutate(logreg\_pred\_class = logreg\_train\_pred$.pred\_class)  
  
logreg\_train\_acc <- accuracy(logreg\_train\_results, truth = failure, estimate = logreg\_pred\_class)  
  
print(logreg\_train\_acc)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 accuracy binary 0.790

confusionMatrix(logreg\_train\_pred$.pred\_class,data\_train$failure,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 16677 4422  
## Yes 43 83  
##   
## Accuracy : 0.7896   
## 95% CI : (0.7841, 0.7951)  
## No Information Rate : 0.7878   
## P-Value [Acc > NIR] : 0.2539   
##   
## Kappa : 0.0246   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.018424   
## Specificity : 0.997428   
## Pos Pred Value : 0.658730   
## Neg Pred Value : 0.790417   
## Prevalence : 0.212250   
## Detection Rate : 0.003910   
## Detection Prevalence : 0.005936   
## Balanced Accuracy : 0.507926   
##   
## 'Positive' Class : Yes   
##

Predict on testing

logreg\_test\_pred <- predict(fit\_logreg, data\_test, type = "class")  
  
logreg\_test\_results <- data\_test %>%  
 mutate(logreg\_pred\_class = logreg\_test\_pred$.pred\_class)  
  
logreg\_test\_acc <- accuracy(logreg\_test\_results, truth = failure, estimate = logreg\_pred\_class)  
  
print(logreg\_test\_acc)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 accuracy binary 0.791

confusionMatrix(logreg\_test\_pred$.pred\_class,data\_test$failure,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 4179 1108  
## Yes 2 19  
##   
## Accuracy : 0.7909   
## 95% CI : (0.7797, 0.8018)  
## No Information Rate : 0.7877   
## P-Value [Acc > NIR] : 0.2906   
##   
## Kappa : 0.0255   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.016859   
## Specificity : 0.999522   
## Pos Pred Value : 0.904762   
## Neg Pred Value : 0.790429   
## Prevalence : 0.212321   
## Detection Rate : 0.003580   
## Detection Prevalence : 0.003956   
## Balanced Accuracy : 0.508190   
##   
## 'Positive' Class : Yes   
##

#### Logistic Regression #2

Split data

set.seed(123)  
data\_split2 <- initial\_split(dataset, prop = 0.70, strata = failure)  
data\_train2 <- training(data\_split2)  
data\_test2 <- testing(data\_split2)

Create folds

set.seed(123)  
folds10 <- vfold\_cv(data\_train2, v = 10)

Build model

model\_logreg2 <-  
 logistic\_reg(mode = "classification") %>%  
 set\_engine("glm")  
  
recipe\_logreg2 <-  
 recipe(failure ~ loading + measurement\_2 +  
 measurement\_4 + measurement\_5 +  
 measurement\_6 + measurement\_7 +   
 measurement\_8 + measurement\_9 +   
 measurement\_16 + measurement\_17 +  
 attribute\_0,  
 data\_train2)  
  
wf\_logreg2 <-  
 workflow() %>%  
 add\_recipe(recipe\_logreg2) %>%  
 add\_model(model\_logreg2)  
  
set.seed(123)  
cv\_results <-   
 tune\_grid(wf\_logreg2,  
 resamples = folds10)

## Warning: No tuning parameters have been detected, performance will be evaluated  
## using the resamples with no tuning. Did you want to [tune()] parameters?

fit\_logreg2 <-  
 fit(wf\_logreg2, data\_train2)  
  
summary(fit\_logreg$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Coefficients: (5 not defined because of singularities)  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 6.0824432 1.0143409 5.996 2.02e-09 \*\*\*  
## product\_codeB -0.0794745 0.0627645 -1.266 0.205429   
## product\_codeC -0.0766105 0.0598599 -1.280 0.200605   
## product\_codeD -0.0628186 0.0627333 -1.001 0.316653   
## product\_codeE -0.1346088 0.0639194 -2.106 0.035212 \*   
## loading 0.0074916 0.0004255 17.605 < 2e-16 \*\*\*  
## attribute\_0material\_5 NA NA NA NA   
## attribute\_1material\_5 NA NA NA NA   
## attribute\_1material\_6 NA NA NA NA   
## attribute\_2 NA NA NA NA   
## attribute\_3 NA NA NA NA   
## measurement\_0 0.0010453 0.0045701 0.229 0.819089   
## measurement\_1 -0.0010678 0.0047153 -0.226 0.820854   
## measurement\_2 0.0160031 0.0054431 2.940 0.003281 \*\*   
## measurement\_3 -0.0171207 0.0169991 -1.007 0.313862   
## measurement\_4 -0.0876981 0.0174640 -5.022 5.12e-07 \*\*\*  
## measurement\_5 -0.1463659 0.0191176 -7.656 1.92e-14 \*\*\*  
## measurement\_6 -0.0965416 0.0189829 -5.086 3.66e-07 \*\*\*  
## measurement\_7 -0.0957374 0.0189881 -5.042 4.61e-07 \*\*\*  
## measurement\_8 -0.1274277 0.0183484 -6.945 3.79e-12 \*\*\*  
## measurement\_9 -0.0622270 0.0177282 -3.510 0.000448 \*\*\*  
## measurement\_10 -0.0165795 0.0122765 -1.351 0.176853   
## measurement\_11 -0.0051708 0.0112956 -0.458 0.647114   
## measurement\_12 0.0182019 0.0117273 1.552 0.120640   
## measurement\_13 0.0291015 0.0153363 1.898 0.057754 .   
## measurement\_14 -0.0009377 0.0113743 -0.082 0.934294   
## measurement\_15 -0.0018697 0.0104080 -0.180 0.857433   
## measurement\_16 -0.0796316 0.0085957 -9.264 < 2e-16 \*\*\*  
## measurement\_17 0.0034741 0.0002014 17.252 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21943 on 21224 degrees of freedom  
## Residual deviance: 21179 on 21201 degrees of freedom  
## AIC: 21227  
##   
## Number of Fisher Scoring iterations: 4

Predict on train

logreg\_train\_pred2 <- predict(fit\_logreg2, data\_train2, type = "class")  
  
logreg\_train\_results2 <- data\_train2 %>%  
 mutate(logreg\_pred\_class2 = logreg\_train\_pred2$.pred\_class)  
  
logreg\_train\_acc2 <- accuracy(logreg\_train\_results2, truth = failure, estimate = logreg\_pred\_class2)  
  
print(logreg\_train\_acc2)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 accuracy binary 0.789

confusionMatrix(logreg\_train\_pred2$.pred\_class,data\_train2$failure,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 14598 3878  
## Yes 32 64  
##   
## Accuracy : 0.7895   
## 95% CI : (0.7835, 0.7953)  
## No Information Rate : 0.7877   
## P-Value [Acc > NIR] : 0.2863   
##   
## Kappa : 0.0218   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.016235   
## Specificity : 0.997813   
## Pos Pred Value : 0.666667   
## Neg Pred Value : 0.790106   
## Prevalence : 0.212255   
## Detection Rate : 0.003446   
## Detection Prevalence : 0.005169   
## Balanced Accuracy : 0.507024   
##   
## 'Positive' Class : Yes   
##

Predict on test

logreg\_test\_pred2 <- predict(fit\_logreg2, data\_test2, type = "class")  
  
logreg\_test\_results2 <- data\_test2 %>%  
 mutate(logreg\_pred\_class2 = logreg\_test\_pred2$.pred\_class)  
  
logreg\_test\_acc2 <- accuracy(logreg\_test\_results2, truth = failure, estimate = logreg\_pred\_class2)  
  
print(logreg\_test\_acc2)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 accuracy binary 0.790

confusionMatrix(logreg\_test\_pred2$.pred\_class,data\_test2$failure,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 6264 1664  
## Yes 7 26  
##   
## Accuracy : 0.7901   
## 95% CI : (0.781, 0.799)  
## No Information Rate : 0.7877   
## P-Value [Acc > NIR] : 0.3068   
##   
## Kappa : 0.0222   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.015385   
## Specificity : 0.998884   
## Pos Pred Value : 0.787879   
## Neg Pred Value : 0.790111   
## Prevalence : 0.212285   
## Detection Rate : 0.003266   
## Detection Prevalence : 0.004145   
## Balanced Accuracy : 0.507134   
##   
## 'Positive' Class : Yes   
##

**LOG REG EVALUATION: The model results in just under 80% accuracy on both training and testing for both log reg models. This is an okay level of accuracy, but could probably be better. The good news is that the accuracy consistency shows little evidence of overfitting..**

#### Classification Tree

Build model

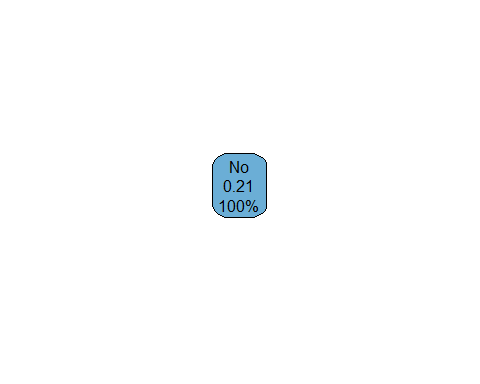
tree\_recipe <-   
 recipe(failure ~., data\_train)  
  
tree\_model <-  
 decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>%  
 set\_mode("classification")  
  
tree\_wf <-  
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(tree\_recipe)  
  
set.seed(123)  
cv\_tree <- tune\_grid(  
 tree\_wf,  
 resamples = folds)

## Warning: No tuning parameters have been detected, performance will be evaluated  
## using the resamples with no tuning. Did you want to [tune()] parameters?

tree\_fit <-   
 fit(tree\_wf, data\_train)  
  
tree\_best <- select\_best(cv\_tree, metric = "accuracy")  
  
tree\_wf\_final <- finalize\_workflow(tree\_wf, tree\_best)  
  
tree\_fit\_final <- fit(tree\_wf\_final, data\_train)  
  
tree <-   
 tree\_fit\_final %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

## Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
## ℹ Please use `extract\_fit\_parsnip()` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

rpart.plot(tree)



**I tried many, many edits to this and tuning options and no matter what cannot get more than one node. I even subsetted the data, and that resulted in a branched tree, but no matter what I just can’t get this one to branch. I probably should not have deleted out the other 10+ attempts at this that I made that resulted in the same single node, but I did. I’m going to move on to random forest modeling and see what happens there.**

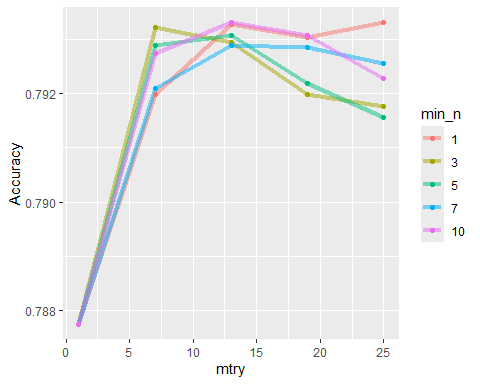
#### Random Forest

Build model

cl <- makePSOCKcluster(detectCores() - 1) # Borrowed code to try and speed processing  
registerDoParallel(cl)  
  
rf\_recipe <-  
 recipe(failure ~., data\_train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
rf\_model <-  
 rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>%  
 set\_engine("ranger") %>%   
 set\_mode("classification")  
  
rf\_wf <-   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(rf\_recipe)  
  
grid <- grid\_regular(  
 mtry(range = c(1, 25)),  
 min\_n(range = c(1, 10)),   
 levels = 5)   
  
set.seed(123)  
rf\_res <- tune\_grid(  
 rf\_wf,  
 resamples = folds,  
 grid = grid)  
  
stopCluster(cl)  
registerDoSEQ()

rf\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 mutate(min\_n = factor(min\_n)) %>%  
 ggplot(aes(mtry, mean, color = min\_n)) +  
 geom\_line(alpha = 0.5, size = 1.5) +  
 geom\_point() +  
 labs(y = "Accuracy")

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



Final fit

rf\_best <-  
 select\_best(rf\_res, metric = "accuracy")  
  
rf\_final <-  
 finalize\_workflow(rf\_wf, rf\_best)  
  
rf\_final

## ══ Workflow ════════════════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: rand\_forest()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 1 Recipe Step  
##   
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## Random Forest Model Specification (classification)  
##   
## Main Arguments:  
## mtry = 25  
## trees = 100  
## min\_n = 1  
##   
## Computational engine: ranger

rf\_final\_fit <-  
 fit(rf\_final, data\_train)

Predict on train

rf\_train\_pred <-  
 predict(rf\_final\_fit, data\_train)  
  
head(rf\_train\_pred)

## # A tibble: 6 × 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 No   
## 4 No   
## 5 No   
## 6 No

confusionMatrix(rf\_train\_pred$.pred\_class,  
 data\_train$failure,  
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 16720 0  
## Yes 0 4505  
##   
## Accuracy : 1   
## 95% CI : (0.9998, 1)  
## No Information Rate : 0.7878   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.2122   
## Detection Rate : 0.2122   
## Detection Prevalence : 0.2122   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##

Predict on test

rf\_test\_pred <-  
 predict(rf\_final\_fit, data\_test)  
  
head(rf\_test\_pred)

## # A tibble: 6 × 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 No   
## 4 No   
## 5 No   
## 6 No

confusionMatrix(rf\_test\_pred$.pred\_class,  
 data\_test$failure,  
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 4154 1089  
## Yes 27 38  
##   
## Accuracy : 0.7898   
## 95% CI : (0.7785, 0.8006)  
## No Information Rate : 0.7877   
## P-Value [Acc > NIR] : 0.3633   
##   
## Kappa : 0.0416   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.033718   
## Specificity : 0.993542   
## Pos Pred Value : 0.584615   
## Neg Pred Value : 0.792294   
## Prevalence : 0.212321   
## Detection Rate : 0.007159   
## Detection Prevalence : 0.012246   
## Balanced Accuracy : 0.513630   
##   
## 'Positive' Class : Yes   
##

**There appears to be significant overfitting happening here. On data\_test, we got Accuracy: 0.9999, Sensitivity: 0.9993, Specificity: 1, No Information Rate: 0.7878. On data\_test, that turned into Accuracy: 0.7924, Sensitivity: 0.040816, Specificity: 0.994977, No Information Rate: 0.7877.**

*Plan for RF #2: Use ideal values of mtry and min\_n found above. Let’s see what happens…*

#### Random Forest #2

Build model

cl <- makePSOCKcluster(detectCores() - 1) # Borrowed code to try and speed processing  
registerDoParallel(cl)  
  
set.seed(123) # Plugging this in again just to make sure, though pretty sure it's already fine.  
folds <- vfold\_cv(data\_train, v = 5)   
  
rf\_recipe2 <-  
 recipe(failure ~., data\_train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
rf\_model2 <-   
 rand\_forest(mtry = tune(),   
 min\_n = tune(),   
 trees = 100) %>%  
 set\_engine("ranger") %>%  
 set\_mode("classification")  
  
rf\_wf2 <- workflow() %>%  
 add\_model(rf\_model2) %>%  
 add\_recipe(rf\_recipe2)  
  
grid2 <-   
 grid\_regular(mtry(range = c(13, 13)),  
 min\_n(range = c(3, 3)),  
 levels = 5)  
  
set.seed(123)  
rf\_res2 <-   
 tune\_grid(rf\_wf2,  
 resamples = folds,  
 grid = grid2)  
  
stopCluster(cl)  
registerDoSEQ()

Final fit

rf\_best2 <-  
 select\_best(rf\_res2, metric = "accuracy")  
  
rf\_final2 <-  
 finalize\_workflow(rf\_wf2, rf\_best2)  
  
rf\_final2

## ══ Workflow ════════════════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: rand\_forest()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 1 Recipe Step  
##   
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## Random Forest Model Specification (classification)  
##   
## Main Arguments:  
## mtry = 13  
## trees = 100  
## min\_n = 3  
##   
## Computational engine: ranger

rf\_final\_fit2 <-  
 fit(rf\_final2, data\_train)

Predict on train

rf\_train\_pred2 <-  
 predict(rf\_final\_fit2, data\_train)  
  
head(rf\_train\_pred2)

## # A tibble: 6 × 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 No   
## 4 No   
## 5 No   
## 6 No

confusionMatrix(rf\_train\_pred2$.pred\_class,  
 data\_train$failure,  
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 16720 2  
## Yes 0 4503  
##   
## Accuracy : 0.9999   
## 95% CI : (0.9997, 1)  
## No Information Rate : 0.7878   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9997   
##   
## Mcnemar's Test P-Value : 0.4795   
##   
## Sensitivity : 0.9996   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.9999   
## Prevalence : 0.2122   
## Detection Rate : 0.2122   
## Detection Prevalence : 0.2122   
## Balanced Accuracy : 0.9998   
##   
## 'Positive' Class : Yes   
##

Predict on test

rf\_test\_pred2 <-  
 predict(rf\_final\_fit2, data\_test)  
  
head(rf\_test\_pred2)

## # A tibble: 6 × 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 No   
## 4 No   
## 5 No   
## 6 No

confusionMatrix(rf\_test\_pred2$.pred\_class,  
 data\_test$failure,  
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 4163 1089  
## Yes 18 38  
##   
## Accuracy : 0.7914   
## 95% CI : (0.7803, 0.8023)  
## No Information Rate : 0.7877   
## P-Value [Acc > NIR] : 0.257   
##   
## Kappa : 0.045   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.033718   
## Specificity : 0.995695   
## Pos Pred Value : 0.678571   
## Neg Pred Value : 0.792650   
## Prevalence : 0.212321   
## Detection Rate : 0.007159   
## Detection Prevalence : 0.010550   
## Balanced Accuracy : 0.514706   
##   
## 'Positive' Class : Yes   
##

**Still looks like overfitting, although upon further research I am seeing a lot of data science professionals saying that with random forests this is not only common but also not super concerning, like it would be for other models like log regs and classification trees. So I’m going to let this ride.**

*Random code I borrowed to attempt to estimate how much time running the RF model would take*

cl <- makePSOCKcluster(detectCores() - 1) # Borrowed code to try and speed processing registerDoParallel(cl)

set.seed(123) data\_sample <- data\_train %>% sample\_n(1000)

set.seed(123) folds\_sample <- vfold\_cv(data\_sample, v = 5)

rf\_recipe\_samp <- recipe(failure ~., data\_sample) %>% step\_dummy(all\_nominal(), -all\_outcomes())

rf\_model\_samp <- rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>% set\_engine(“ranger”) %>% set\_mode(“classification”)

rf\_wf\_samp <- workflow() %>% add\_model(rf\_model\_samp) %>% add\_recipe(rf\_recipe\_samp)

grid <- grid\_regular( mtry(range = c(1, 25)), min\_n(range = c(1, 10)),  
levels = 3)

start\_time <- Sys.time()

set.seed(123) rf\_res <- tune\_grid( rf\_wf\_samp, resamples = folds\_sample, grid = grid)

end\_time <- Sys.time() single\_iteration\_time <- end\_time - start\_time print(single\_iteration\_time)

single\_iteration\_seconds <- as.numeric(single\_iteration\_time, units = “secs”) num\_resamples <- 5 num\_combinations <- 25 total\_estimated\_time\_seconds <- single\_iteration\_seconds \* num\_resamples \* num\_combinations total\_estimated\_time\_minutes <- total\_estimated\_time\_seconds / 60 total\_estimated\_time\_hours <- total\_estimated\_time\_minutes / 60

print(total\_estimated\_time\_minutes) print(total\_estimated\_time\_hours)