Although we are fairly sure that our target variable (time\_delayed) is not accurate, it is all we have. We can thank the bus drivers of New York that we have anything at all, I suppose, but the inaccuracy of this variable is probably one of the reasons not very many people have touched this data set.

**Data preparation**

Some data preparation first. We have initially included the variable 'time\_diff\_report' in the ii main data set. This is the time between the incident and when the incident is called in to operations. The difficult with this number is that it is obviously included in the time estimate given for the breakdown. The time difference is only ever less than the breakdown time for cases which are reported after they are resolved. Obviously, this number can be used as a strong predictor, while also providing no useful predictive information for our real problem: working out how much longer the bus will be delayed. We correct all of this by subtracting this value from the delayed time, so that 'time\_delayed' becomes the amount of time from after operations is called that the bus is delayed. We also remove cases which are resolved before they are reported.

#remove zero variance columns

zero\_var <- function(dat) {

out <- lapply(dat, function(x) length(unique(x)))

keep <- rownames(data.frame(which(out > 1)))

dat %>%

select(keep)

}

in\_csv <- "../output/intermediate/ii\_spread.csv"

ii\_spread <- read\_csv(in\_csv) %>%

mutate(

time\_delayed = as.numeric(time\_delayed - time\_diff\_report),

time\_diff\_report = NULL

) %>%

filter(reported\_before\_resolved == 1) %>%

select(-School\_Year, -Route\_Number, -Schools\_Serviced, -Bus\_Company\_Name, -Occurred\_On) %>%

zero\_var

halfFold <- createFolds(ii\_spread$time\_delayed, k = 5, list=TRUE)

ii\_small <- ii\_spread[halfFold[[2]],]

ii\_test <-ii\_spread[halfFold[[3]],]

ii\_train <- ii\_spread %>%

anti\_join(ii\_test, by = "Busbreakdown\_ID")

rctrl\_manual <- trainControl(method = "none", returnResamp = "all")

rctrl\_repcv <- trainControl(method = "repeatedcv", number = 2, repeats = 5, returnResamp = "all")

**Cross fold validation testing**

Since I am running all of this locally on a laptop which 'benchmarkme' tells me is the slowest system to have been benchmarked in 2018, I build data set 'ii\_small' to do cross-fold validation testing, with root mean square error as my target. During my run, I test different numbers of committees and rules. The numbers presented here are what I ended up with. Neighbours always degrade the solution in this case are fixed to zero. We can see the full rule set of the model by calling 'summary', but this results in a lot of spam. A more succinct presentation is available by looking at the 'finalModel' components of the derived model. 'splits' gives a list of the derived rules, where multiple entries per rule indicates an 'AND' condition (both conditions satisfied). 'coefficients' gives the resulting slope and intercept of the application of that rule. Finally, we can get a full listing of how many times a predictor appears in the ruleset and the coefficient set by looking at 'usage': this is somewhat equivalent to a variable importance call.

#---cubist

cubist\_grd <- expand.grid(

committees = 1,

neighbors = 0

)

set.seed(849)

cubist\_cv <- train(time\_delayed ~ . -Busbreakdown\_ID,

data = ii\_small,

method = "cubist",

metric="RMSE",

na.action = na.pass,

trControl = rctrl\_repcv,

tuneGrid = cubist\_grd,

control = Cubist::cubistControl(unbiased = T, rules = 20, sample = 0)

)

cubist\_cv

*## Cubist*

*##*

*## 40908 samples*

*## 25 predictor*

*##*

*## No pre-processing*

*## Resampling: Cross-Validated (2 fold, repeated 5 times)*

*## Summary of sample sizes: 20454, 20454, 20453, 20455, 20455, 20453, ...*

*## Resampling results:*

*##*

*## RMSE Rsquared MAE*

*## 13.52728 0.1336734 9.883126*

*##*

*## Tuning parameter 'committees' was held constant at a value of 1*

*##*

*## Tuning parameter 'neighbors' was held constant at a value of 0*

#summary(cubist\_cv)

cubist\_cv$finalModel$splits %>% glimpse

*## Observations: 89*

*## Variables: 8*

*## $ committee 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...*

*## $ rule 1, 1, 1, 1, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, ...*

*## $ variable School\_Age, Boro\_Bronx, Reason\_HeavyTraffic, Number...*

*## $ dir <=, <=, >, <=, >, >, >, >, <=, >, <=, >, <=, <=, <=...*

*## $ value 0, 0, 0, 6, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...*

*## $ category , , , , , , , , , , , , , , , , , , , , , , , ,*

*## $ type "type2", "type2", "type2", "type2", "type2", "type2...*

*## $ percentile 0.13063459, 0.75171116, 0.29549232, 0.88540139, 0.9...*

cubist\_cv$finalModel$coefficients %>% glimpse

*## Observations: 20*

*## Variables: 27*

*## $ `(Intercept)` 16.8, 14.9, 13.8, 21.0, 14.7, ...*

*## $ Has\_Contractor\_Notified\_Schools NA, NA, 1.4, 8.7, 1.4, 2.2, NA...*

*## $ Has\_Contractor\_Notified\_Parents NA, 7.2, 4.0, -7.9, NA, 1.2, 3...*

*## $ Have\_You\_Alerted\_OPT NA, 6.1, NA, -2.6, NA, -1.9, 3...*

*## $ Number\_Of\_Students\_On\_The\_Bus -0.78, -0.56, -0.17, -0.70, -0...*

*## $ School\_Age NA, NA, NA, -1.2, NA, -2.3, 5....*

*## $ Reason\_Accident NA, NA, 10, 2, NA, 6, 10, NA, ...*

*## $ Reason\_DelayedbySchool NA, NA, NA, NA, NA, -5, -5, NA...*

*## $ Reason\_FlatTire NA, NA, 7, 1, NA, 4, NA, NA, 1...*

*## $ Reason\_HeavyTraffic NA, NA, -2.0, 1.9, NA, -1.2, N...*

*## $ Reason\_LatereturnfromFieldTrip NA, NA, NA, NA, NA, 2.1, 5.6, ...*

*## $ Reason\_MechanicalProblem NA, NA, 6.7, 14.6, NA, 5.0, 6....*

*## $ Reason\_ProblemRun NA, NA, NA, NA, NA, 2, 8, NA, ...*

*## $ Reason\_WeatherConditions NA, NA, 10.3, 3.6, NA, 3.4, NA...*

*## $ Reason\_WontStart NA, NA, NA, NA, NA, 2, NA, NA,...*

*## $ Boro\_Bronx NA, NA, NA, NA, NA, -2.5, NA, ...*

*## $ Boro\_Brooklyn NA, 0.4, NA, NA, 3.7, 1.1, 4.8...*

*## $ Boro\_Connecticut NA, NA, NA, NA, NA, NA, NA, NA...*

*## $ Boro\_Manhattan NA, 0.6, NA, NA, NA, 2.6, 4.3,...*

*## $ Boro\_NassauCounty NA, NA, NA, NA, 8, 3, NA, NA, ...*

*## $ Boro\_NewJersey NA, NA, NA, NA, NA, 4, 5, NA, ...*

*## $ Boro\_Queens NA, 0.5, NA, NA, 6.8, 2.2, 3.6...*

*## $ Boro\_RocklandCounty NA, NA, NA, NA, NA, 4, NA, NA,...*

*## $ Boro\_StatenIsland NA, NA, NA, NA, NA, -2.5, NA, ...*

*## $ Boro\_Westchester NA, 1.2, NA, NA, NA, 5.1, 13.6...*

*## $ committee "1", "1", "1", "1", "1", "1", ...*

*## $ rule "1", "2", "3", "4", "5", "6", ...*

cubist\_cv$finalModel$usage %>% head

*## Conditions Model Variable*

*## 1 68 98 Number\_Of\_Students\_On\_The\_Bus*

*## 2 66 6 Boro\_Bronx*

*## 3 65 29 Reason\_HeavyTraffic*

*## 4 52 79 Has\_Contractor\_Notified\_Parents*

*## 5 46 45 School\_Age*

*## 6 38 47 Boro\_Manhattan*

**Validation on test data**

We could look through this data set for human-readable rules, but we can see from the high RMSE and low \(R^2\) value that we do not have a very good fit. To look at exactly how rough a model this is, we use our manual fold splits to predict from our 'ii\_test' data set:

cubist\_man <- train(time\_delayed ~ . -Busbreakdown\_ID,

data = ii\_train,

method = "cubist",

metric="RMSE",

na.action = na.pass,

trControl = rctrl\_manual,

tuneGrid = cubist\_grd,

control = Cubist::cubistControl(unbiased = T, rules = 20, sample = 0)

)

ii\_withpred <- ii\_test %>%

mutate(

time\_predicted = predict(cubist\_man, newdata = ii\_test)

)

ii\_withpred %>%

summarise(

rms = (mean((time\_predicted-time\_delayed)^2))^0.5,

sum\_sq = sum((time\_predicted-time\_delayed)^2),

tot\_sum\_sq = sum((time\_delayed - mean(time\_delayed))^2),

r\_sq = 1- sum\_sq / tot\_sum\_sq

) #1,20: 13.5, 0.141

*## # A tibble: 1 x 4*

*## rms sum\_sq tot\_sum\_sq r\_sq*

*##*

*## 1 13.6 7542210. 8829766. 0.146*

protec <- ii\_withpred %>%

mutate(Reason\_Other = 1L) %>% #dummy flag to get the name in

select(-contains("Reason")) %>%

colnames()

pal <- brewer.pal(10, "Paired")

ii\_withpred %>%

mutate(Reason\_Other = ifelse(Reason\_Accident + Reason\_DelayedbySchool + Reason\_FlatTire + Reason\_HeavyTraffic + Reason\_LatereturnfromFieldTrip + Reason\_MechanicalProblem + Reason\_ProblemRun + Reason\_WeatherConditions + Reason\_WontStart == 1, 0L, 1L)) %>%

gather(key = Reason, value = dummy, -one\_of(protec)) %>%

filter(dummy == 1) %>%

mutate(dummy = NULL) %>%

filter(row\_number() %% 10 == 0) %>%

ggplot(aes(x = time\_predicted, y = time\_delayed, fill = Reason, color = Reason)) +

geom\_point(alpha = 0.75, shape = 21) +

scale\_fill\_manual(values = pal) +

geom\_abline(intercept = 0, slope = 1, color = "red") +

coord\_cartesian(expand = c(0,0), x = c(0,60)) +

theme\_classic() +

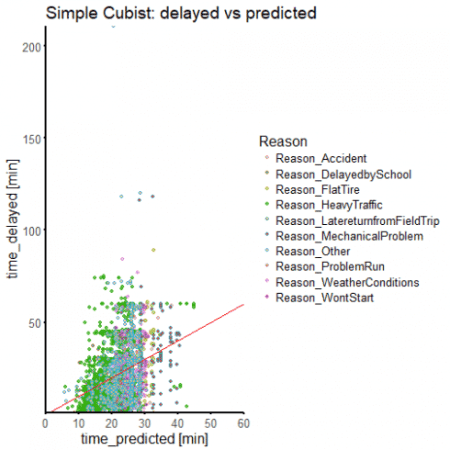
labs(x="time\_predicted [min]",y="time\_delayed [min]") +

theme(text=element\_text(family="Arial", size=16)) +

theme(axis.line = element\_line(colour = 'black', size = 1)) +

theme(axis.ticks = element\_line(colour = "black", size = 1)) +

ggtitle("Simple Cubist: delayed vs predicted")

[](https://i0.wp.com/datascienceplus.com/wp-content/uploads/2018/11/unnamed-chunk-3-1-13.png?ssl=1)

We also do not know whether a regression approach is really going to be useful with this data, and indeed, later on we will switch over to a two class classification problem, and simply look at whether a delay will be greater or less than a cut off time.