Kaggle - Restaurant Revenue Prediction (or: Regression with little training data)

Jacques Sauve

January 9th, 2016

Executive Summary

A regression analysis using many different models was performed in order to compare model performance. A regression problem from Kaggle was chosen: Restaurant Revenue Prediction. About 20 models were trained and evaluated. Unfortunately, the amount of training data was so small that there was not much performance difference between models using the RMSE metric. The best model found uses Gradient Boosting Machine and achieved position #57 on the private Kaggle leaderboard, out of 2257 teams. The most important predictor is the restaurant's opening date, which is not useful to predict the revenue of future restaurants.

Description from the Kaggle site

Objective

Predict annual restaurant sales based on objective measurements.

With over 1,200 quick service restaurants across the globe, TFI is the company behind some of the world's most well-known brands: Burger King, Sbarro, Popeyes, Usta Donerci, and Arby's. They employ over 20,000 people in Europe and Asia and make significant daily investments in developing new restaurant sites.

Right now, deciding when and where to open new restaurants is largely a subjective process based on the personal judgement and experience of development teams. This subjective data is difficult to accurately extrapolate across geographies and cultures.

New restaurant sites take large investments of time and capital to get up and running. When the wrong location for a restaurant brand is chosen, the site closes within 18 months and operating losses are incurred.

Finding a mathematical model to increase the effectiveness of investments in new restaurant sites would allow TFI to invest more in other important business areas, like sustainability, innovation, and training for new employees. Using demographic, real estate, and commercial data, this competition challenges you to predict the annual restaurant sales of 100,000 regional locations.

Data Fields

- Id : Restaurant id.
- Open Date : opening date for a restaurant
- City : City that the restaurant is in. Note that there are unicode in the names.
- City Group: Type of the city. Big cities, or Other.
- Type: Type of the restaurant. FC: Food Court, IL: Inline, DT: Drive Thru, MB: Mobile
- P1, P2 - P37: There are three categories of these obfuscated data. *Demographic data* are gathered from third party providers with GIS systems. These include population in any given area, age and gender distribution, development scales. *Real estate* data mainly relate to the m2 of the location, front facade of the location, car park availability. *Commercial data* mainly include the existence of points of interest including schools, banks, other QSR operators.

- Revenue: The revenue column indicates a (transformed) revenue of the restaurant in a given year and is the target of predictive analysis. Please note that the values are transformed so they don't mean real dollar values.

Evaluation by Root Mean Squared Error (RMSE)

Data Exploration

Load the data and see how much we have.

```r
# get train data from
# https://www.kaggle.com/c/restaurant-revenue-prediction/download/train.csv.zip
train <- read.csv("../input/train.csv", stringsAsFactors=F)
# get test data from
# https://www.kaggle.com/c/restaurant-revenue-prediction/download/test.csv.zip
test <- read.csv("../input/test.csv", stringsAsFactors=F)
dim(train)
```

```
## [1] 137  43
```

```r
dim(test)
```

```
## [1] 100000     42
```

Here is the main problem: only 137 observations for training. As a result:

- We will not split the training data in train/validation subsets since the resulting datasets would be too small.
- We can use all of the training data to train a model, using resampling to validate and adjust the tuning parameters.

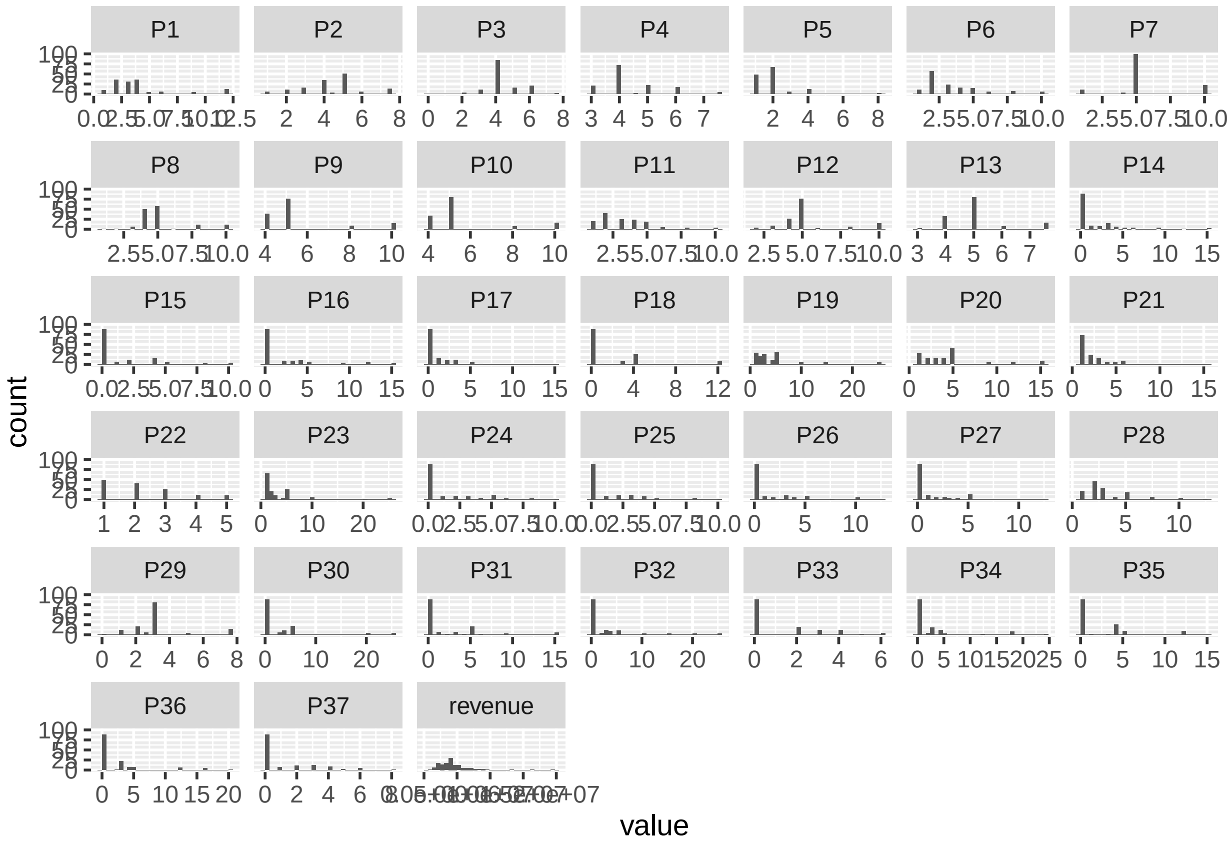
```r
sum(!complete.cases(train))
```

```
## [1] 0
```

There are no missing cases, aparently.

Let's have a look at the numerical data as histograms (ref).

```r
library(ggplot2)
library(reshape)
d <- melt(train[,-c(1:5)])
ggplot(d,aes(x = value)) +
  facet_wrap(~variable,scales = "free_x") +
  geom_histogram()
```



We can see several variables with a lot of zeros (P14, P15, ...). Other than these zeros, there doesn't seem to be a lot of skew in the predictors. There is some skew in revenue. Let's investigate the zeros more:

```r
# Display missing-data patterns.
# don't look at Id or revenue
library(mice)
temp = train[,c(-1,-ncol(train))]
temp[temp==0] = NA
md.pattern(temp)
```

```
##    P1 P2 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P19 P20 P21 P22 P23 P28 P3 P29
## 48  1  1  1  1  1  1  1  1   1   1   1   1   1   1   1   1   1   1  1   1
##  1  1  1  1  1  1  1  1  1   1   1   1   1   1   1   1   1   1   1  1   1
## 85  1  1  1  1  1  1  1  1   1   1   1   1   1   1   1   1   1   1  1   1
##  1  1  1  1  1  1  1  1  1   1   1   1   1   1   1   1   1   1   1  0   1
##  2  1  1  1  1  1  1  1  1   1   1   1   1   1   1   1   1   1   1  1   0
##     0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0  1   2
##    P14 P15 P16 P17 P18 P24 P25 P26 P30 P31 P32 P33 P34 P35 P36 P37 P27
## 48   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   0
## 85   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##  2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##      88  88  88  88  88  88  88  88  88  88  88  88  88  88  88  88  89
##    Open.Date City City.Group Type
## 48         0    0          0    0    4
##  1         0    0          0    0    5
## 85          0    0           0    0   21
##  1          0    0           0    0   22
##  2          0    0           0    0   22
##            137  137         137  137 2048
```

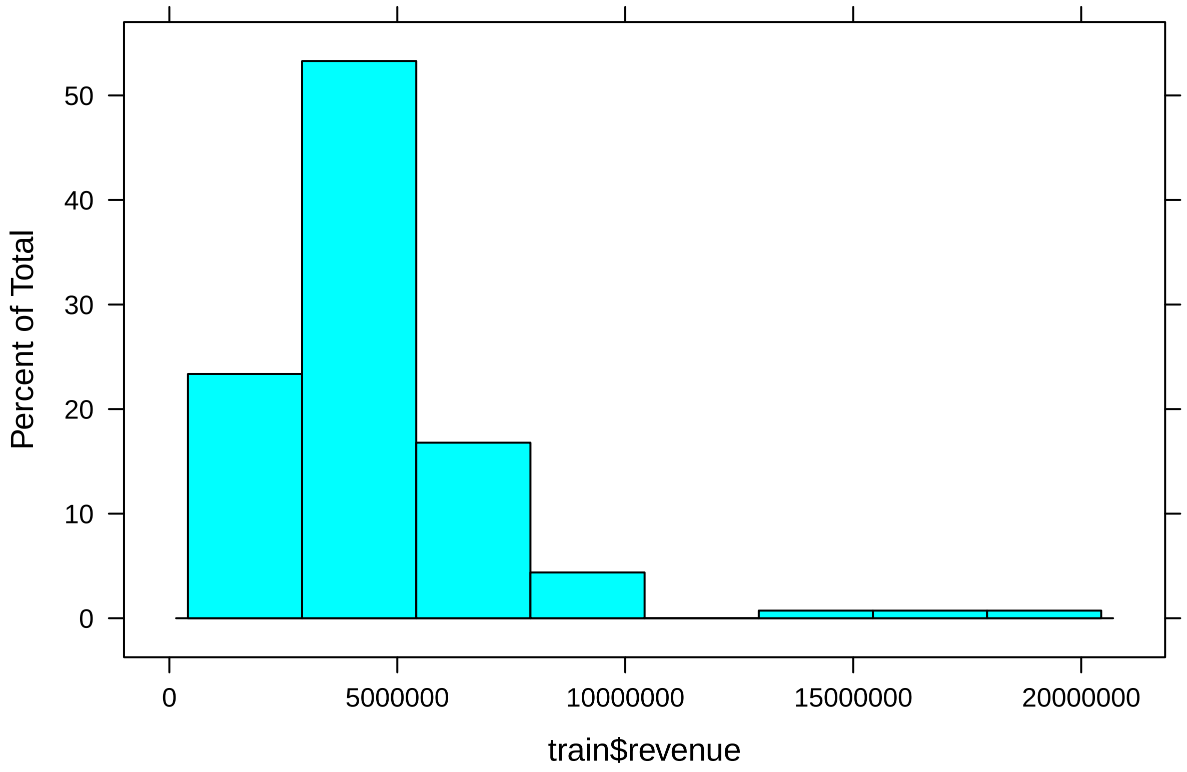
Looking at row counts:

- 48 complete observations, 1 missing P27
- 85 observations missing P14 P15 P16 P17 P18 P24 P25 P26 P30 P31 P32 P33 P34 P35 P36 P37 P27
- 1 observation missing P3
- 2 observations missing P29 P14 P15 P16 P17 P18 P24 P25 P26 P30 P31 P32 P33 P34 P35 P36 P37 P27

Column counts indicate that many predictors have 88 zeros. They appear to be a "cluster of NAs"

We can also look at the distribution of the outcome (revenue):

```
library(lattice)
histogram(train$revenue)
```



Conclusions from initial data exploration:

- No real need to transform predictors;
- There is a bit of skew in revenue;
- All these zeros look like missing data and will be converted to NAs prior to imputation;
- The Open.Date field may be converted into new features (days since opening date, year, month, day)

There is an interesting visual representation of the data here as a geomap of average revenue. It doesn't help for modeling but is a nice visual touch.

```r
library(methods)
library(googleVis)
library(plyr)
x <- train[, c(3, 43)]
y <- aggregate(x$revenue, list(City=x$City), mean)
colnames(y)[2] <- "Average Revenue"
z <- count(x, 'City')
colnames(z)[2] <- "Restaurants"
x <- merge(y, z, by = "City")
x$Restaurants <- paste0("Restaurants: ", x$Restaurants)
TFI <- x
GT <- gvisGeoMap(TFI, locationvar = 'City', numvar = 'Average Revenue',
                 hovervar = "Restaurants",
                 options = list(region = 'TR', height = 350,
                                dataMode = 'regions'))
print(GT, file = "../output/geomap.html")
```

Missing Data, Data Transformation and Feature Creation

Imputation for missing data

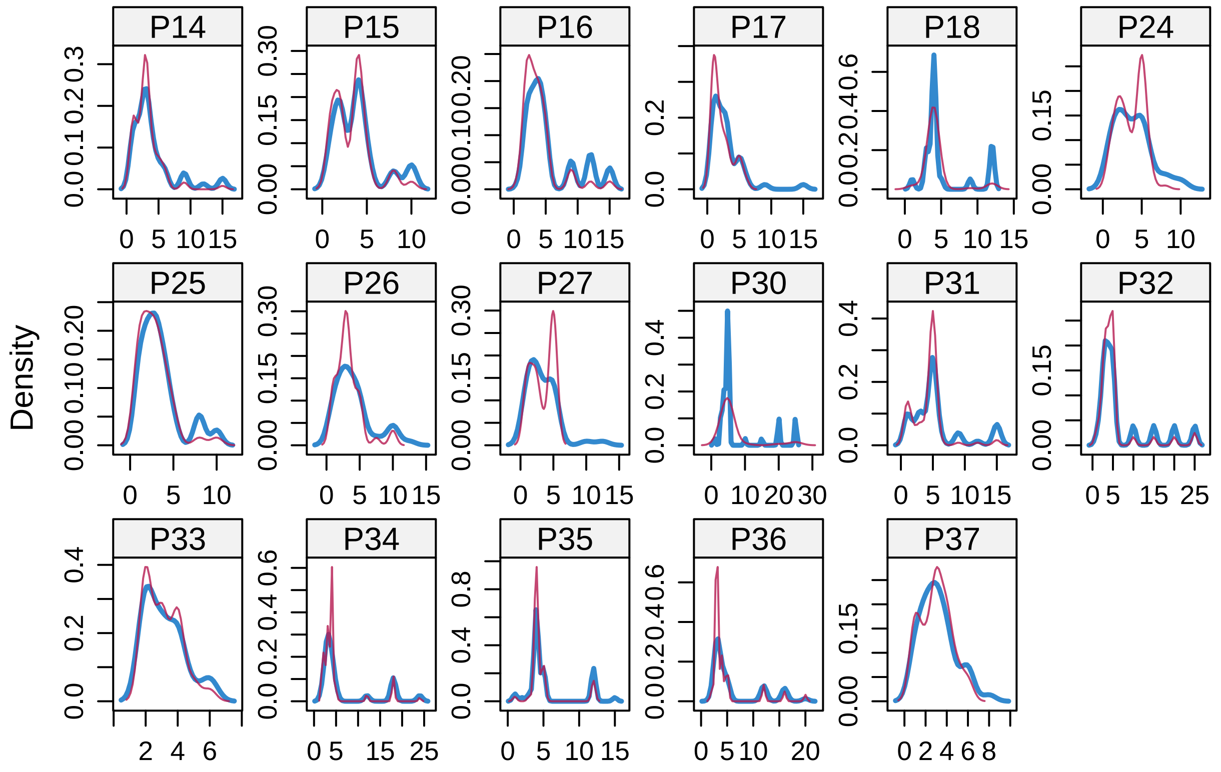
Impute using Multivariate Imputation by Chained Equations (mice)

- For imputation methods, see methods(mice)
- I used the `fastpmm` method and small values for m and maxit due for speed rf is used normally, but other methods (sample, for example), when there is a lot of data

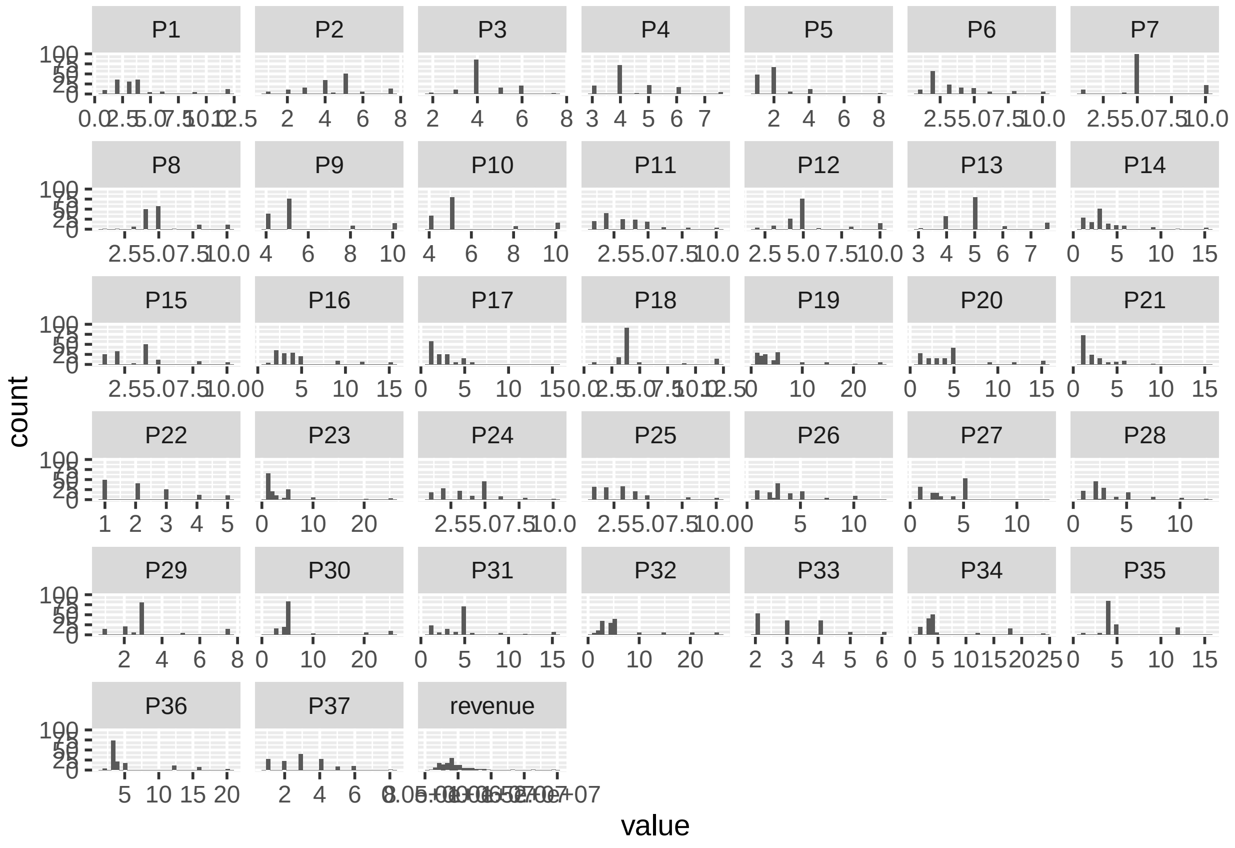
```r
library(mice)
library(caret)
train[train==0] = NA
tempData <- mice(train, m=1, maxit=2, meth='rf',
                 seed=501, printFlag=FALSE)
train <- complete(tempData, 1)
```

The data after imputation (imputed in magenta, observed in blue):

```r
densityplot(tempData)
```



```r
d <- melt(train[,-c(1:5)])
ggplot(d, aes(x = value)) +
  facet_wrap(~variable,scales = "free_x") +
  geom_histogram()
```



Yep! The values at zero are gone. Also there is little skew, so we'll just transform revenue.

Predictor removal: near-zero variance

- We don't have so many predictors that we need to remove some due to scale
- However, we need to check NZV and correlation between predictors

```
# near zero variance
library(caret)
print (nearZeroVar(train, saveMetrics = FALSE))
```

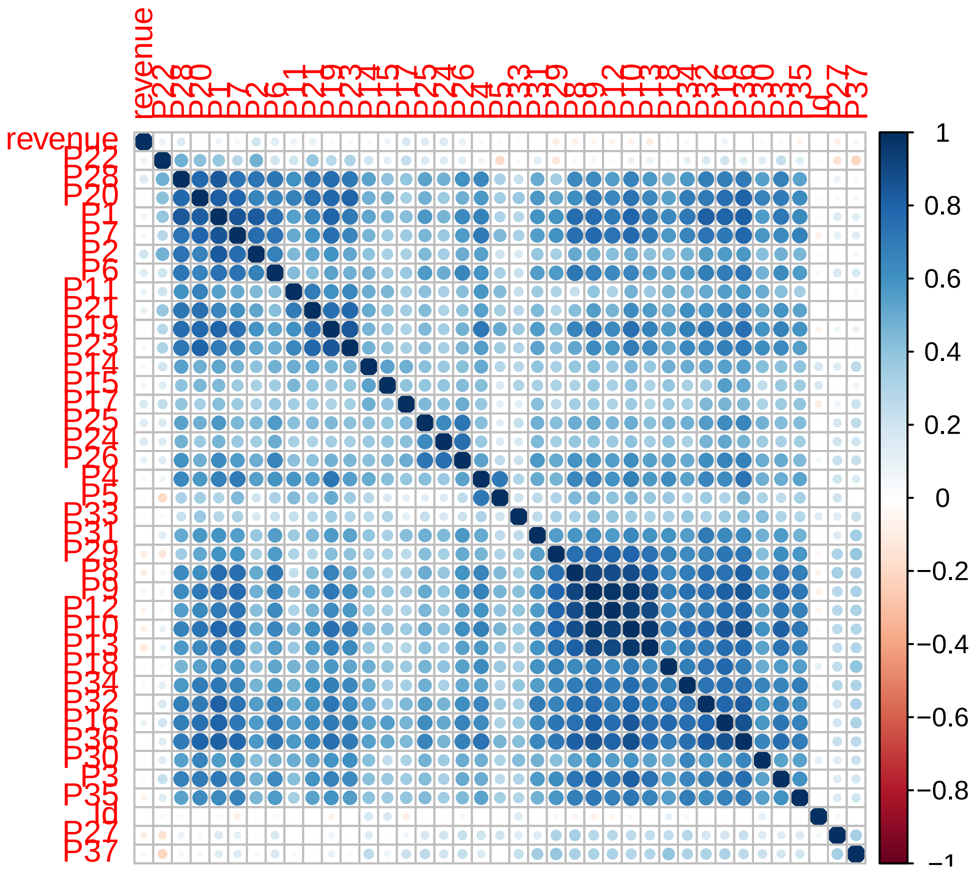
```
## integer(0)
```

Result: nothing to remove due to near-zero variance.

Predictor removal: correlation

Let's have a visual look at correlations between numerical variables.

```
library(corrplot)
nums <- sapply(train, is.numeric)
correlations = cor(train[,nums])
corrplot(correlations, order = "hclust")
```

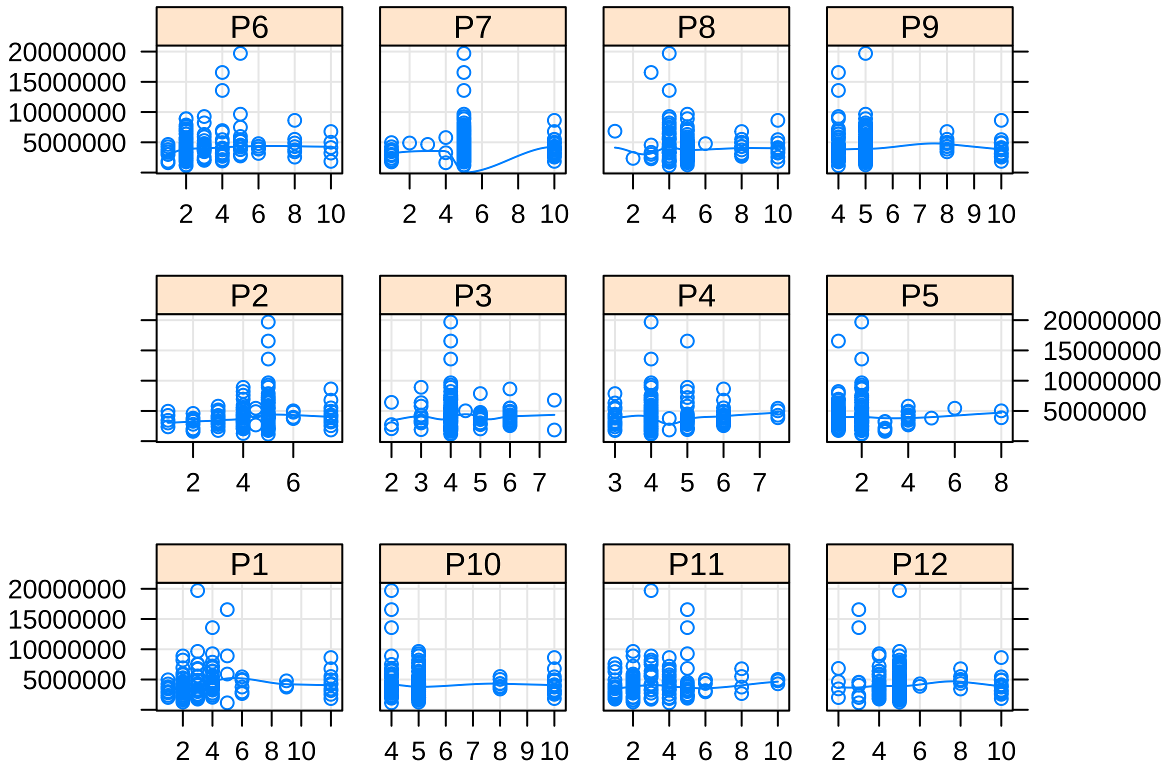


This plot shows that there is some correlation between numerical predictors

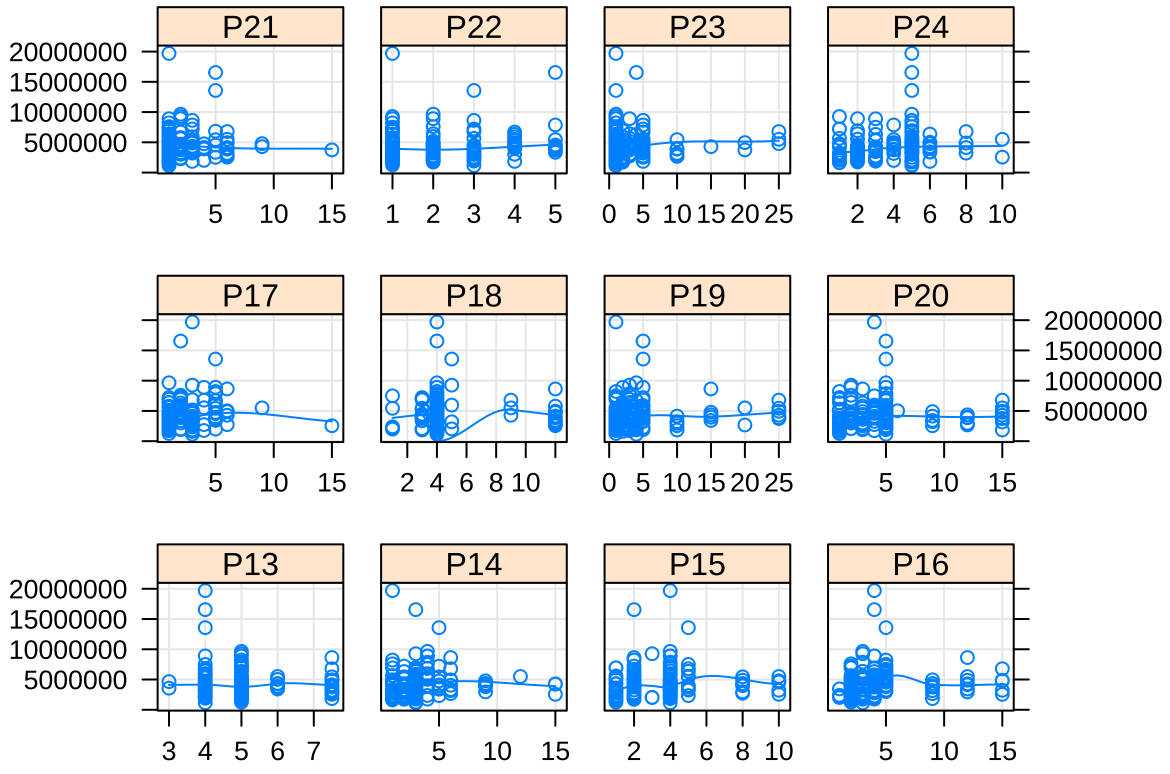
- We will investigate the removal of correlated predictors during modeling
- There is unfortunately very little correlation between predictors and revenue, so that we don't expect well-performing models.

Let's look at some scatterplots to see relationships.

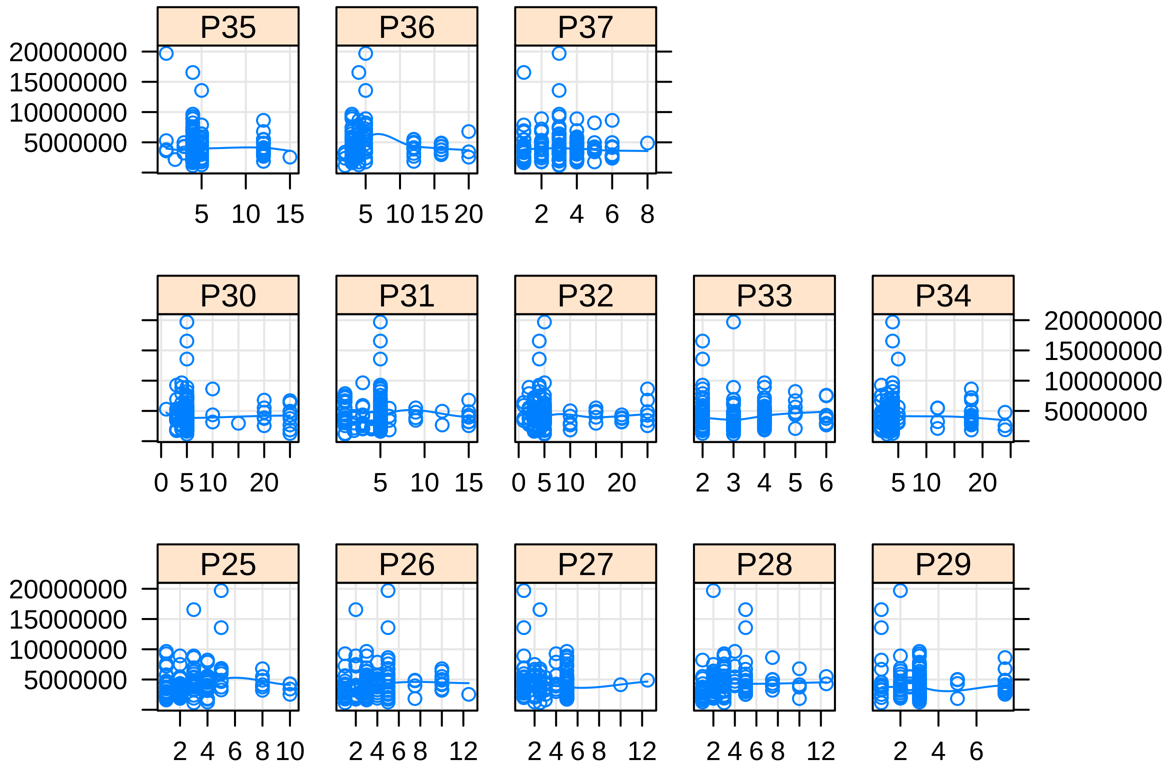
```r
featurePlot(train[,c('P1','P2','P3','P4','P5','P6','P7','P8','P9','P10','P11','P12')],
            train$revenue,
            between = list(x = 1, y = 1),
            type = c("g", "p", "smooth"),
            labels = rep("", 2))
```



```r
featurePlot(train[,c('P13','P14','P15','P16','P17','P18','P19', 'P20','P21','P22','P23','P24')],
            train$revenue,
            between = list(x = 1, y = 1),
            type = c("g", "p", "smooth"),
            labels = rep("", 2))
```



```r
featurePlot(train[,c('P25','P26','P27','P28','P29','P30', 'P31','P32','P33','P34','P35','P36', 'P37')],
            train$revenue,
            between = list(x = 1, y = 1),
            type = c("g", "p", "smooth"),
            labels = rep("", 2))
```



- There doesn't seem to be much change in revenue with respect to these predictors, so we can expect high RMSE and low R^2.
- There are non-linearities so that parametric regression (if used) will have to include polynomial terms.

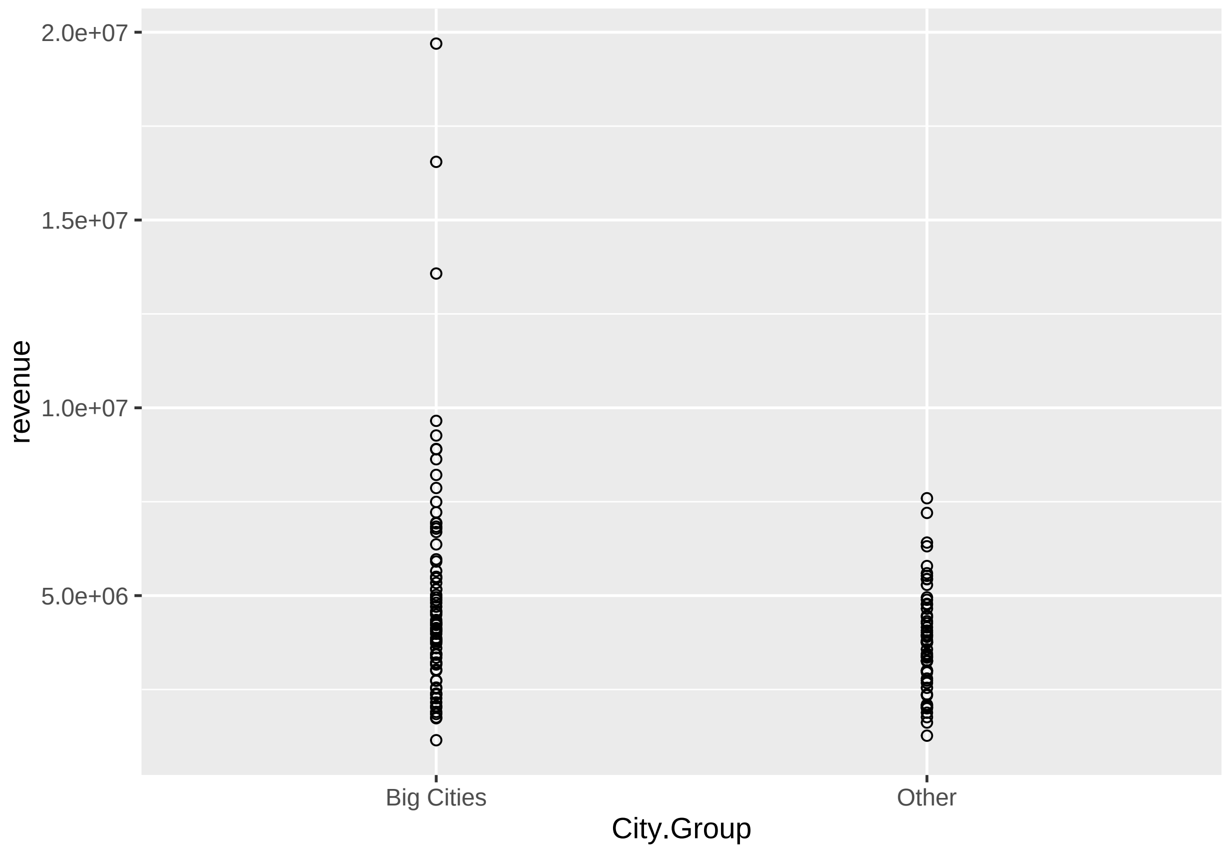
Outliers

- Won't consider this for now.

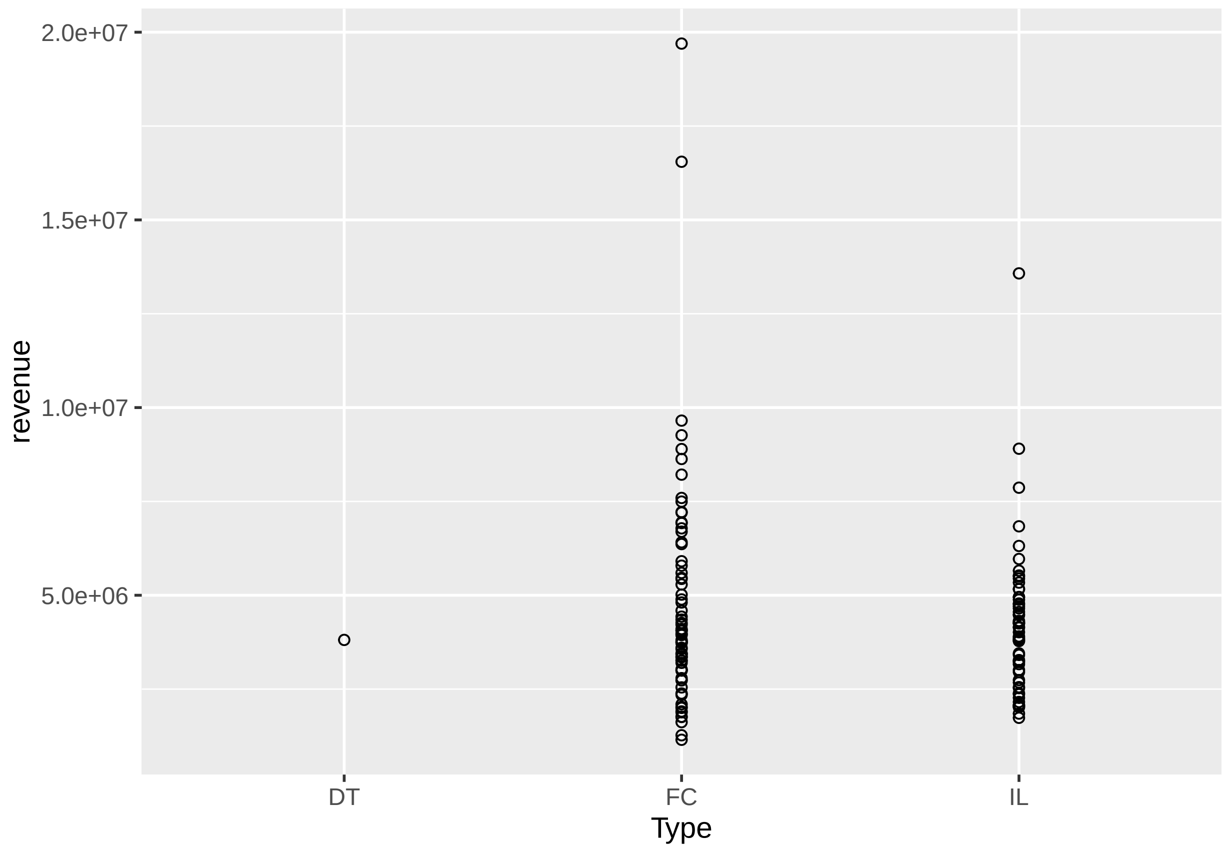
Examine data for relationships, trends

- Scatter plots above already show the weak, non-linear relationship with P* predictors.
- Now for the other 2 predictors:

```r
ggplot(train, aes(x=City.Group, y=revenue)) +
        geom_point(shape=1) +
        geom_smooth()
```



```
ggplot(train, aes(x=Type, y=revenue)) +
        geom_point(shape=1) +
        geom_smooth()
```



Modeling

Many different models were tested and tuned using the caret package. These are the models tested:

- Ordinary Least Squares
- Partial Least Squares
- Elastic Net (ridge/lasso)
- Neural Net
- MARS
- SVM radial
- Tree
- Bagged Tree
- Conditional Tree
- Random Forest
- Boosted Tree (Gradient Boosting Machine)
- eXtreme Gradient Boosting
- Cubist

Much code was borrowed from Kuhn and Johnson's book: Applied Predictive Modeling. See Kuhn's comparison table of models in Appendix A.

There was not much difference between the models since we have so little data. I will only show the best performing models here, all based on Gradient Boosting Machine (gbm).

Since we dont have much data, we will not split the training data to get a validation set; we will use cross-validation, even if the results will not be very indicative of the test RMSE. The final test will have to come from Kaggle.

```r
library(caret)
library(mice)
### WARNING 2: The RWeka package does not work well with some forms of
### parallel processing, such as mutlicore (i.e. doMC).
library(doMC)
registerDoMC(4)

### Create a control object for train()
ctrl = trainControl(method = "cv", number = 10)
read.data = function(filename) {
    data = read.csv(filename, stringsAsFactors=F)
    data$Type = factor(data$Type, levels=c('IL', 'FC', 'DT', 'MB'))
    data$City.Group = as.factor(data$City.Group)
    return (data)
}
# get example submission from
# https://www.kaggle.com/c/restaurant-revenue-prediction/download/sampleSubmission.csv
prepare.submission = function(fit, test, fit.name, fun = NULL) {
    result = data.frame(Id = test$Id)
    result$Prediction = predict(fit, test)
    if (!is.null(fun)) {
        result$Prediction = fun(result$Prediction)
    }
    write.csv(result, file = paste0('../output/',fit.name,'_results.csv'),
              quote = FALSE, row.names = FALSE)
}
# Impute using Multivariate Imputation by Chained Equations (mice)
# For imputation methods, see methods(mice)
# rf is used normally, but other methods (sample, for example),
# when there is a lot of data
impute = function(data, method='rf') {
    data[data==0] = NA
    tempData <- mice(data, m=1, maxit=2, meth=method,
                     seed=501, printFlag=FALSE)
    return (complete(tempData, 1))
}
remove.correlated.predictors = function(data) {
    nums <- sapply(data, is.numeric)
    correlations = cor(data[,nums])
    highCorr = findCorrelation(correlations, cutoff = .75)
    num.pred = data[,nums][, -highCorr]
    return (cbind(num.pred, data[,c('City.Group', 'Type', 'revenue')]))
}
numerical.predictors = function(data) {
    return (names(data)[grepl('^P', names(data))])
}
train = read.data('../input/train.csv')
train[,numerical.predictors(train)] = impute(train[,numerical.predictors(train)])
test = read.data('../input/test.csv')
```

Code for gbm models:

```r
do.gbm = function(formula, data) {
    gbmGrid <- expand.grid(interaction.depth = seq(1, 7, by = 2),
                           n.trees = seq(100, 1000, by = 50),
                           shrinkage = c(0.01, 0.1),
                           n.minobsinnode = c(5,10))
    set.seed(669)
    # for some reason, using a formula rather than (predictors,outcome)
    # gave slightly better results
    gbmFit <- train(form=formula, data=data,
                    method = "gbm",
                    tuneGrid = gbmGrid,
                    verbose = FALSE,
                    trControl = ctrl)
    print (getTrainPerf(gbmFit))
    return (gbmFit)
}
```

A basic model

- Characteristics of model
  - No imputation of missing training data
  - No transformation
  - No removal due to correlation
  - No additional features

```r
train = read.data('../input/train.csv')
train = train[, -(1:3)]
train$Type = factor(train$Type, levels=c("DT","FC","IL","MB"))
gbmFit = do.gbm(revenue ~ ., data=train)
```

```
##   TrainRMSE TrainRsquared method
## 1   2341155     0.1125955    gbm
```

```r
# ni_nt_nc_nf = 'no imputation, no transformation,
# no removal due to correlation, no additional features
test = read.data('../input/test.csv')
prepare.submission(gbmFit, test, 'gbm_ni_nt_nc_nf')
leaderboard = data.frame(fit='gbm_ni_nt_nc_nf',
                         rmse=1844142.14327, position=770)
```

Result: position 770 of 2257 teams on Kaggle.

One can see that the R^2 value is very low, as expected due to the small training dataset. The line `leaderboard = ...` indicates the RMSE on the private Kaggle leaderboard as well as the position achieved ou of 2257 competing teams.

With imputation

- Characteristics of model

  - Imputation of missing training data
  - No transformation
  - No removal due to correlation
  - No additional features

```r
train = read.data('../input/train.csv')
train = train[, -(1:3)]
gbmFit = do.gbm(revenue ~ ., data=train)
```

```
##   TrainRMSE TrainRsquared method
## 1   2340561     0.1130712    gbm
```

```r
test = read.data('../input/test.csv')
prepare.submission(gbmFit, test, 'gbm_it_nt_nc_nf')
leaderboard = rbind(leaderboard,
                    data.frame(fit='gbm_it_nt_nc_nf',
                               rmse=1845526.06265, position=785))
```

Result: about the same.

Removing correlated predictors

- Characteristics of model
  - Imputation of missing training data
  - No transformation
  - Removal of correlated predictors
  - No additional features

```r
train = read.data('../input/train.csv')
train = train[, -(1:3)]
train = remove.correlated.predictors(train)
gbmFit = do.gbm(revenue ~ ., data=train)
```

```
##   TrainRMSE TrainRsquared method
## 1   2399388     0.1035093    gbm
```

```r
test = read.data('../input/test.csv')
prepare.submission(gbmFit, test, 'gbm_it_nt_rc_nf')
leaderboard = rbind(leaderboard,
                    data.frame(fit='gbm_it_nt_rc_nf',
                               rmse=1845879.34674, position=787))
```

Result: about the same.

With additional features

- Characteristics of model
  - Imputation of missing training data

    - No transformation
    - No removal due to correlation
    - Additional features

```
add.features = function(data) {
    data$year <- as.numeric(substr(as.character(data$Open.Date),7,10))
    data$month <- as.numeric(substr(as.character(data$Open.Date),1,2))
    data$day <- as.numeric(substr(as.character(data$Open.Date),4,5))
    data$Date <- as.Date(strptime(data$Open.Date, "%m/%d/%Y"))
    min.year = head(sort(data$year))[1]
    data$days <- as.numeric(data$Date - as.Date(paste0(min.year,"-01-01")))
    return (data)
}
train = read.data('../input/train.csv')
train = add.features(train)
train = train[, -(1:3)]
gbmFit = do.gbm(revenue ~ ., data=train)
```

```
##   TrainRMSE TrainRsquared method
## 1   2372292    0.09188579    gbm
```

```
test = read.data('../input/test.csv')
test = add.features(test)
prepare.submission(gbmFit, test, 'gbm_it_nt_nc_wf')
leaderboard = rbind(leaderboard,
                    data.frame(fit='gbm_it_nt_nc_wf',
                               rmse=1832814.59420, position=618))
```

Result: Good improvement: position = 618.

With transformation

- Characteristics of model
    - Imputation of missing training data
    - log transformation of revenue
    - No removal due to correlation
    - Additional features

```
train = read.data('../input/train.csv')
train$revenue = log(train$revenue)
train = add.features(train)
train = train[, -(1:3)]
gbm_it_wlt_nc_wf = do.gbm(revenue ~ ., data=train)
```

```
##   TrainRMSE TrainRsquared method
## 1 0.4434942     0.1894891    gbm
```

```r
test = read.data('../input/test.csv')
test = add.features(test)
prepare.submission(gbm_it_wlt_nc_wf, test, 'gbm_it_wlt_nc_wf', fun=exp)
leaderboard = rbind(leaderboard,
                    data.frame(fit='gbm_it_wlt_nc_wf',
                               rmse=1780045.02033, position=57))
```

Result: Good improvement: position = 57.

With imputation of missing test data as well

Is there a lot of missing (zero) test data?

```r
test = read.data('../input/test.csv')
test[test==0] = NA
print (sum(!complete.cases(test)))
```

```
## [1] 92414
```

Yes, there is: let's do imputation. To impute, we need revenue data. So we first use a simpler model to predict test revenue, and then impute missing test data.

- Characteristics of model
  - Imputation of missing training data and testing data
  - log transformation of revenue
  - No removal due to correlation
  - Additional features

```r
train = read.data('../input/train.csv')
train$revenue = log(train$revenue)
train = add.features(train)
train = train[, -(1:3)]
gbmFit = do.gbm(revenue ~ ., data=train)
```

```
##   TrainRMSE TrainRsquared method
## 1 0.4434942     0.1894891    gbm
```

```r
test = read.data('../input/test.csv')
test = add.features(test)
test.revenue = predict(gbmFit, test)
test = read.data('../input/test.csv')
Id = test$Id
Open.Date = test$Open.Date
test = test[, -(1:3)]
test$revenue = test.revenue
test = impute(test, method='sample')
test$Open.Date = Open.Date
test$Id = Id
test = add.features(test)
```

```
prepare.submission(gbmFit, test, 'gbm_itt_wlt_nc_wf', fun=exp)
leaderboard = rbind(leaderboard,
                    data.frame(fit='gbm_itt_wlt_nc_wf',
                               rmse=1785557.44521, position=79))
```

Result: no improvement.

With multiple imputation

- Characteristics of model
  - Several imputations of missing training data
  - log transformation of revenue
  - No removal due to correlation
  - Additional features

```
NUM.IMPUTATIONS = 10
train = read.data('../input/train.csv')
train$revenue = log(train$revenue)
test = read.data('../input/test.csv')
test = add.features(test)
train[train==0] = NA
tempData <- mice(train, m=NUM.IMPUTATIONS, maxit=2, meth='rf',
                 seed=501, printFlag=FALSE)
predictions = data.frame(row.names=1:nrow(test))
for (i in (1:NUM.IMPUTATIONS)) {
    train = complete(tempData, i)
    train = add.features(train)
    train = train[, -(1:3)]
    gbmFit = do.gbm(revenue ~ ., data=train)
    predictions = cbind(predictions, predict(gbmFit, test))[,seq(1,i)]
}
```

```
##   TrainRMSE TrainRsquared method
## 1 0.4266123     0.2335522    gbm
##   TrainRMSE TrainRsquared method
## 1 0.4089761     0.2990659    gbm
##   TrainRMSE TrainRsquared method
## 1 0.4392566     0.1916227    gbm
##   TrainRMSE TrainRsquared method
## 1 0.3978416     0.3820618    gbm
##   TrainRMSE TrainRsquared method
## 1 0.4304478     0.2428719    gbm
##   TrainRMSE TrainRsquared method
## 1  0.412855     0.2824168    gbm
##   TrainRMSE TrainRsquared method
## 1 0.3890207     0.3589299    gbm
##   TrainRMSE TrainRsquared method
## 1 0.4242296     0.2564065    gbm
##   TrainRMSE TrainRsquared method
## 1 0.4350309     0.2551545    gbm
##   TrainRMSE TrainRsquared method
## 1 0.4243594     0.2400595    gbm
```

```
result = data.frame(Id = test$Id)
result$Prediction = exp(rowMeans(predictions))
write.csv(result, file = '../output/gbm_mi_wlt_nc_wf_results.csv',
          quote = FALSE, row.names = FALSE)
leaderboard = rbind(leaderboard,
                    data.frame(fit='gbm_mi_wlt_nc_wf',
                               rmse=1953404.65481, position=1725))
```

Result: no improvement. Wow! Something went wrong here. I only used multiple imputation compared to last model: it should have improved. Actually, the RMSE value is quite sensitive. Let's try without transformation.

With multiple imputation, no transformation

- Characteristics of model
  - Several imputations of missing training data
  - log transformation of revenue
  - No removal due to correlation
  - Additional features

```
NUM.IMPUTATIONS = 10
train = read.data('../input/train.csv')
test = read.data('../input/test.csv')
test = add.features(test)
train[train==0] = NA
tempData <- mice(train, m=NUM.IMPUTATIONS, maxit=2, meth='rf',
                 seed=501, printFlag=FALSE)
predictions = data.frame(row.names=1:nrow(test))
for (i in (1:NUM.IMPUTATIONS)) {
    train = complete(tempData, i)
    train = add.features(train)
    train = train[, -(1:3)]
    gbmFit = do.gbm(revenue ~ ., data=train)
    predictions = cbind(predictions, predict(gbmFit, test))[,seq(1,i)]
}
```

```
##   TrainRMSE TrainRsquared method
## 1   2242378     0.2768812    gbm
##   TrainRMSE TrainRsquared method
## 1   2294232     0.1913558    gbm
##   TrainRMSE TrainRsquared method
## 1   2219306     0.2326222    gbm
##   TrainRMSE TrainRsquared method
## 1   2394075     0.1028401    gbm
##   TrainRMSE TrainRsquared method
## 1   2263339     0.2301466    gbm
##   TrainRMSE TrainRsquared method
## 1   2377993     0.1153267    gbm
##   TrainRMSE TrainRsquared method
## 1   2376634     0.1186616    gbm
##   TrainRMSE TrainRsquared method
## 1   2384817    0.09363069    gbm
##   TrainRMSE TrainRsquared method
## 1   2248620     0.2194459    gbm
##   TrainRMSE TrainRsquared method
## 1   2343950     0.1386338    gbm
```

```r
result = data.frame(Id = test$Id)
result$Prediction = rowMeans(predictions)
write.csv(result, file = '../output/gbm_mi_nt_nc_wf_results.csv',
          quote = FALSE, row.names = FALSE)
leaderboard = rbind(leaderboard,
                    data.frame(fit='gbm_mi_nt_nc_wf',
                               rmse=1796661.54897, position=146))
```

Result: no improvement.

Model Performance

Here are the Kaggle results:

```r
print (leaderboard)
```

```
##                 fit    rmse position
## 1   gbm_ni_nt_nc_nf 1844142      770
## 2   gbm_it_nt_nc_nf 1845526      785
## 3   gbm_it_nt_rc_nf 1845879      787
## 4   gbm_it_nt_nc_wf 1832815      618
## 5  gbm_it_wlt_nc_wf 1780045       57
## 6 gbm_itt_wlt_nc_wf 1785557       79
## 7  gbm_mi_wlt_nc_wf 1953405     1725
## 8   gbm_mi_nt_nc_wf 1796662      146
```

Not bad! Position 57 out of 2257 teams on the private leaderboard. Of course, this was done with hints from other competitors after the competition was closed (see references below).

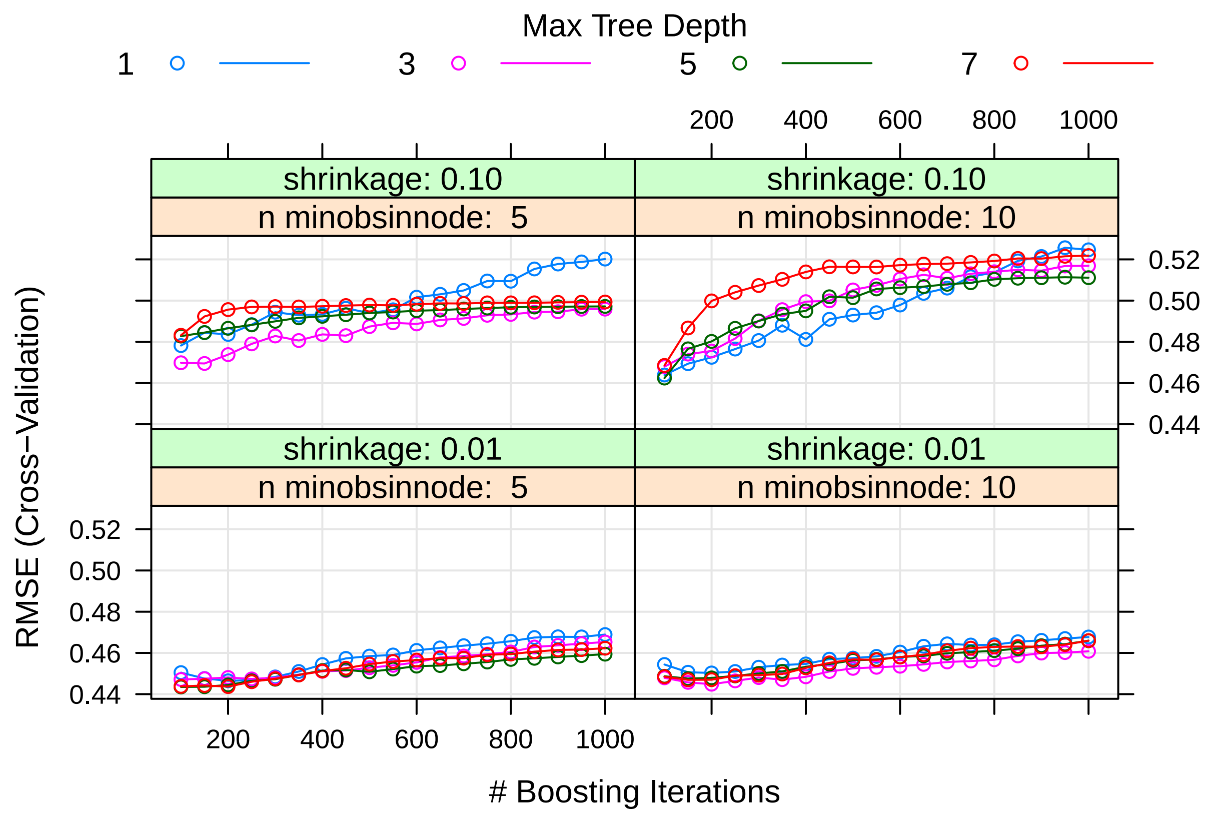
It is not really useful here to compare model performance since there is very little data and models yield almost the same results. But see Kuhn chapter 10 for good plots on model comparison.

Must repeat this sort of comparison with a Kaggle competition that has more training data.

Let us examine additional details about the winning model.

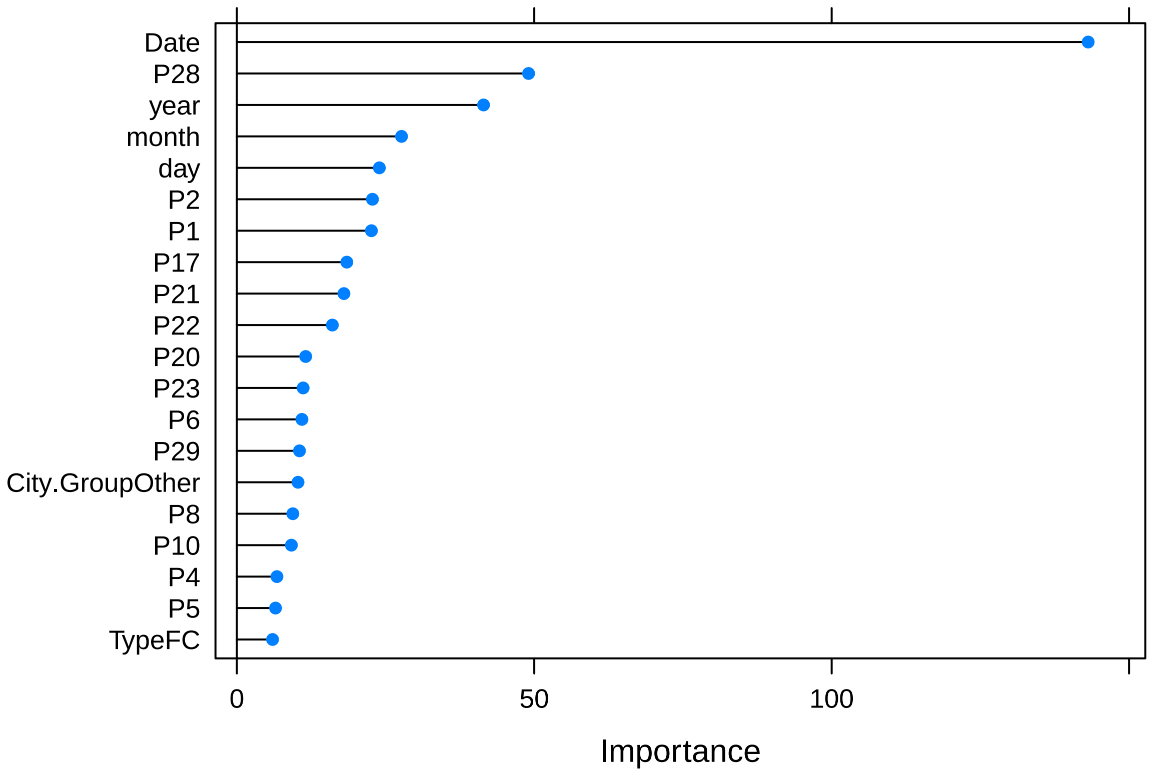
First, some details about the tuning process.

```r
plot(gbm_it_wlt_nc_wf)
```



Now, details about variable importance:

```
gbmImp <- varImp(gbm_it_wlt_nc_wf, scale = FALSE)
plot(gbmImp, top = 20)
```



As we saw before, the P* predictors did not correlate well with revenue. Here we see that Date, year, month, day are quite important. It doesn't appear that this would be a very useful model for predicting revenue for *future* restaurants.

References

- Kaggle site
- Kaggle forum
- Some ideas from first place winner
- Additional ideas from code
- Some code from Kuhn and Johnson book: Applied predictive Modeling
  - To see code from Kuhn and Johnson's book:

```
library(AppliedPredictiveModeling)
scriptLocation()
```