In the previous article in this series, we showed that common ensemble models like random forest and gradient boosting are *uncalibrated*: they are not guaranteed to estimate aggregates or rollups of the data in an unbiased way. However, they can be preferable to calibrated models such as linear or generalized linear regression, when they make more accurate predictions on individuals. In this article, we’ll demonstrate one ad-hoc method for calibrating an uncalibrated model with respect to *specific* grouping variables. This "polishing step" potentially returns a model that estimates certain rollups in an unbiased way, while retaining good performance on individual predictions.

**Example: Predicting income**

We’ll continue the example from the previous posts in the series: predicting income from demographic variables (sex, age, employment, education). The data is from the 2016 US Census American Community Survay (ACS) Public Use Microdata Sample (PUMS) for our example. More information about the data can be found [here](https://github.com/WinVector/PDSwR2/tree/master/PUMS). First, we’ll get the training and test data, and show how the expected income varies along different groupings (by sex, by employment, and by education):

library(zeallot)

library(wrapr)

incomedata <- readRDS("incomedata.rds")

c(test, train) %<-% split(incomedata, incomedata$gp)

# get the rollups (mean) by grouping variable

show\_conditional\_means <- function(d, outcome = "income") {

cols <- qc(sex, employment, education)

lapply(

cols := cols,

function(colname) {

aggregate(d[, outcome, drop = FALSE],

d[, colname, drop = FALSE],

FUN = mean)

})

}

display\_tables <- function(tlist) {

for(vi in tlist) {

print(knitr::kable(vi))

}

}

display\_tables(

show\_conditional\_means(train))

| **sex** | **income** |
| --- | --- |
| Male | 55755.51 |
| Female | 47718.52 |

| **employment** | **income** |
| --- | --- |
| Employee of a private for profit | 51620.39 |
| Federal government employee | 64250.09 |
| Local government employee | 54740.93 |
| Private not-for-profit employee | 53106.41 |
| Self employed incorporated | 66100.07 |
| Self employed not incorporated | 41346.47 |
| State government employee | 53977.20 |

| **education** | **income** |
| --- | --- |
| no high school diploma | 31883.18 |
| Regular high school diploma | 38052.13 |
| GED or alternative credential | 37273.30 |
| some college credit, no degree | 42991.09 |
| Associate’s degree | 47759.61 |
| Bachelor’s degree | 65668.51 |
| Master’s degree | 79225.87 |
| Professional degree | 97772.60 |
| Doctorate degree | 91214.55 |

**A random forest model**

For this post, we’ll train a random forest model to predict income.

library(randomForest)

model\_rf\_1stage <- randomForest(income ~ age+sex+employment+education,

data=train)

train$pred\_rf\_raw <- predict(model\_rf\_1stage, newdata=train, type="response")

# doesn't roll up

display\_tables(

show\_conditional\_means(train,

qc(income, pred\_rf\_raw)))

| **sex** | **income** | **pred\_rf\_raw** |
| --- | --- | --- |
| Male | 55755.51 | 55292.47 |
| Female | 47718.52 | 48373.40 |

| **employment** | **income** | **pred\_rf\_raw** |
| --- | --- | --- |
| Employee of a private for profit | 51620.39 | 51291.36 |
| Federal government employee | 64250.09 | 61167.31 |
| Local government employee | 54740.93 | 55425.30 |
| Private not-for-profit employee | 53106.41 | 54174.31 |
| Self employed incorporated | 66100.07 | 63714.20 |
| Self employed not incorporated | 41346.47 | 46415.34 |
| State government employee | 53977.20 | 55599.89 |

| **education** | **income** | **pred\_rf\_raw** |
| --- | --- | --- |
| no high school diploma | 31883.18 | 41673.91 |
| Regular high school diploma | 38052.13 | 42491.11 |
| GED or alternative credential | 37273.30 | 43037.49 |
| some college credit, no degree | 42991.09 | 44547.89 |
| Associate’s degree | 47759.61 | 46815.79 |
| Bachelor’s degree | 65668.51 | 63474.48 |
| Master’s degree | 79225.87 | 69953.53 |
| Professional degree | 97772.60 | 76861.44 |
| Doctorate degree | 91214.55 | 75940.24 |

As we observed before, the random forest model predictions do not match the true rollups, even on the training data.

**Polishing the model**

Suppose that we wish to make individual predictions of subjects’ incomes, *and* estimate mean income as a function of employment type. An ad-hoc way to do this is to adjust the predictions from the random forest, depending on subjects’ employment type, so that the resulting polished model is calibrated with respect to employment. Since linear models are calibrated, we might try fitting a linear model to the random forest model’s predictions, along with employment.

(Of course, we could use a Poisson model as well, but for this example we’ll just use a simple linear model for the polishing step).

One caution: *we shouldn’t use the same data to fit both the random forest model and the polishing model.* This leads to nested-model bias, a potential source of overfit. Either we must split the training data into two sets: one to train the random forest model and another to train the polishing model; or we have to use cross-validation to simulate having two sets. This second procedure is the same procedure used when stacking multiple models; you can think of this polishing procedure as being a form of stacking, where some of the sub-learners are simply single variables.

Let’s use 5-fold cross-validation to "stack" the random forest model and the employment variable. We’ll use vtreat to create the cross-validation plan.

set.seed(2426355)

# build a schedule for 5-way crossval

crossplan <- vtreat::kWayCrossValidation(nrow(train), 5)

The crossplan is a list of five splits of the data (described by row indices); each split is itself a list of two disjoint index vectors: split$train and split$app. For each fold, we want to train a model using train[split$train, , drop=FALSE] and then apply the model to train[split$app, , drop=FALSE].

train$pred\_uncal <- 0

# use cross validation to get uncalibrated predictions

for(split in crossplan) {

model\_rf\_2stage <- randomForest(income ~ age+sex+employment+education,

data=train[split$train, , drop=FALSE])

predi <- predict(model\_rf\_2stage,

newdata=train[split$app, , drop=FALSE],

type="response")

train$pred\_uncal[split$app] <- predi

}

The vector train$pred\_uncal is now a vector of random forest predictions on the training data; every prediction is made using a model that was not trained on the datum in question.

Now we can use these random forest predictions to train the linear polishing model.

# learn a polish/calibration for employment

rf\_polish <- lm(income - pred\_uncal ~ employment,

data=train)

# get rid of pred\_uncal, as it's no longer needed

train$pred\_uncal <- NULL

Now, take the predictions from the original random forest model (the one trained on all the data, earlier), and polish them with the polishing model.

# get the predictions from the original random forest model

train$pred\_rf\_raw <- predict(model\_rf\_1stage, newdata=train, type="response")

# polish the predictions so that employment rolls up correctly

train$pred\_cal <-

train$pred\_rf\_raw +

predict(rf\_polish, newdata=train, type="response")

# see how close the rollups get to ground truth

rollups <- show\_conditional\_means(train,

qc(income, pred\_cal, pred\_rf\_raw))

display\_tables(rollups)

| **sex** | **income** | **pred\_cal** | **pred\_rf\_raw** |
| --- | --- | --- | --- |
| Male | 55755.51 | 55343.35 | 55292.47 |
| Female | 47718.52 | 48296.93 | 48373.40 |

| **employment** | **income** | **pred\_cal** | **pred\_rf\_raw** |
| --- | --- | --- | --- |
| Employee of a private for profit | 51620.39 | 51640.44 | 51291.36 |
| Federal government employee | 64250.09 | 64036.19 | 61167.31 |
| Local government employee | 54740.93 | 54739.80 | 55425.30 |
| Private not-for-profit employee | 53106.41 | 53075.76 | 54174.31 |
| Self employed incorporated | 66100.07 | 66078.76 | 63714.20 |
| Self employed not incorporated | 41346.47 | 41341.37 | 46415.34 |
| State government employee | 53977.20 | 53946.07 | 55599.89 |

| **education** | **income** | **pred\_cal** | **pred\_rf\_raw** |
| --- | --- | --- | --- |
| no high school diploma | 31883.18 | 41526.88 | 41673.91 |
| Regular high school diploma | 38052.13 | 42572.57 | 42491.11 |
| GED or alternative credential | 37273.30 | 43104.09 | 43037.49 |
| some college credit, no degree | 42991.09 | 44624.38 | 44547.89 |
| Associate’s degree | 47759.61 | 46848.84 | 46815.79 |
| Bachelor’s degree | 65668.51 | 63468.93 | 63474.48 |
| Master’s degree | 79225.87 | 69757.13 | 69953.53 |
| Professional degree | 97772.60 | 76636.17 | 76861.44 |
| Doctorate degree | 91214.55 | 75697.59 | 75940.24 |

Note that the rolled up predictions from the polished model almost match the true rollups for employment, but *not* for the other grouping variables (sex and education). To see this better, let’s look at the total absolute error of the estimated rollups.

err\_mag <- function(x, y) {

sum(abs(y-x))

}

preds = qc(pred\_rf\_raw, pred\_cal)

errframes <- lapply(rollups,

function(df) {

lapply(df[, preds],

function(p)

err\_mag(p, df$income)) %.>%

as.data.frame(.)

})

errframes <- lapply(rollups,

function(df) {

gp = names(df)[[1]]

errs <- lapply(df[, preds],

function(p)

err\_mag(p, df$income))

as.data.frame(c(grouping=gp, errs))

})

display\_tables(errframes)

| **grouping** | **pred\_rf\_raw** | **pred\_cal** |
| --- | --- | --- |
| sex | 1117.927 | 990.5685 |

| **grouping** | **pred\_rf\_raw** | **pred\_cal** |
| --- | --- | --- |
| employment | 14241.51 | 323.2577 |

| **grouping** | **pred\_rf\_raw** | **pred\_cal** |
| --- | --- | --- |
| education | 70146.37 | 70860.7 |

We can reduce the rollup errors substantially for the variables that the polishing model was exposed to. For variables that the polishing model is not exposed to, there is no improvement; it’s likely that those estimated rollups will in many cases be worse.

**Model performance on holdout data**

Let’s see the performance of the polished model on test data.

# get the predictions from the original random forest model

test$pred\_rf\_raw <- predict(model\_rf\_1stage, newdata=test, type="response")

# polish the predictions so that employment rolls up correctly

test$pred\_cal <-

test$pred\_rf\_raw +

predict(rf\_polish, newdata=test, type="response")

# compare the rollups on employment

preds <- qc(pred\_rf\_raw, pred\_cal)

employment\_rollup <-

show\_conditional\_means(test,

c("income", preds))$employment

knitr::kable(employment\_rollup)

| **employment** | **income** | **pred\_rf\_raw** | **pred\_cal** |
| --- | --- | --- | --- |
| Employee of a private for profit | 50717.96 | 51064.25 | 51413.32 |
| Federal government employee | 66268.05 | 61401.94 | 64270.82 |
| Local government employee | 52565.89 | 54878.96 | 54193.47 |
| Private not-for-profit employee | 52887.52 | 54011.64 | 52913.09 |
| Self employed incorporated | 67744.61 | 63664.51 | 66029.07 |
| Self employed not incorporated | 41417.25 | 46215.42 | 41141.44 |
| State government employee | 51314.92 | 55395.96 | 53742.14 |

# see how close the rollups get to ground truth for employment

lapply(employment\_rollup[, preds],

function(p) err\_mag(p, employment\_rollup$income)) %.>%

as.data.frame(.) %.>%

knitr::kable(.)

| **pred\_rf\_raw** | **pred\_cal** |
| --- | --- |
| 21608.9 | 8764.302 |

The polished model estimates rollups with respect to employment better than the uncalibrated random forest model. Its performance on individual predictions (as measured by root mean squared error) is about the same.

# predictions on individuals

rmse <- function(x, y) {

sqrt(mean((y-x)^2))

}

lapply(test[, preds],

function(p) rmse(p, test$income)) %.>%

as.data.frame(.) %.>%

knitr::kable(.)

| **pred\_rf\_raw** | **pred\_cal** |
| --- | --- |
| 31780.39 | 31745.12 |

**Conclusion**

We’ve demonstrated a procedure that mitigates bias issues with ensemble models, or any other uncalibrated model. This potentially allows the data scientist to balance the requirement for highly accurate predictions on individuals with the need to correctly estimate specific aggregate quantities.

This method is ad-hoc, and may be somewhat brittle. In addition, it requires that the data scientist knows ahead of time which rollups will be desired in the future. However, if you find yourself in a situation where you must balance accurate individual prediction with accurate aggregate estimates, this may be a useful trick to have in your data science toolbox.