Supplmentary Materials

Tian

Contents

1 Modeling

NOTE: I re-used random forest and smoothing spline models from earlier

1.1 Preprocessing

The same pre-processing is applied in both models but some new variables are only used in random forest (details below).

1.1.1 Transformation

- price: performed box-cox transformation on the response variate price.
- saledate: transformed as the number of days since 1970/01/01, then take the power of 5
- landarea: take log
- stories: take log
- rmdl_diff: log(rmdl_diff + 1) because some of them are zero
- all categorical variables is treated as factors
- level of `cndtn` is specified

1.1.2 Level Matching

In order to solve the level mismatching problem during 5-fold cross-validation, we gather some categories with small number of observation (typically under 10 - 30) and merge them as a new category `Other`.

Variables with categories that has less than 10 observation (such categories are merged as `Other`):

- heat
- style
- roof
- ward

Variables with categories that has less than 20 observation (such categories are merged as `Other`):

- grade

Variables with categories that has less than 30 observation (such categories are merged as `Poor/Fair`):

- cndtn

1.1.3 New Variables

- if_rmdl : indicator whether the house is remodeled
- rmdl_diff: numerical rep the difference between remodel date and sale date; 0 if remodel is after sale
- saleyear: year of the sale
- buy_first: indicator whether the house is build first or buy first
- total_bath: combine number of bathroom and number of half-bathrooms (bathrm+0.5hf_bathrm)
- build_age: the years between sold year and year of build
- avg_room_size: divide `gba` by (the number of rooms + 1) to get the average size of rooms
- inter1: interaction term between latitude and saledate
- inter2: interaction term between longitude and saledate

- inter3: interaction term between gba and saledate
- inter4: interaction term between landarea and longitude
- inter5: interaction term between eyb and ayb
- inter6: interaction term between build_age and latitude

Note: interaction 1-6 is used only in random forest, they're not involved in the model building process of smoothing spline at all.

1.1.4 Missing data handling

- yr_rmdl: missing yr_rmdl is recoded as 0 (we use if_rmdl and rmdl_diff instead of yr_rmdl in the model)
- ayb: missing ayb is recoded as eyb - avg_gap between ayb and eyb
- quadrant: missing quadrant is recoded as "NW" bc "NW" is most popular
- kitchen and stories: omitted (5 observations)

1.2 Summary of Model Building/Tuning

1.2.1 Random Forest

Main package used: `randomforest`, `ranger`, `caret`

Parameters tuned and their optimal values:

- mtry: 37
- min.node.size: 5
- splitrule: extratrees

Cross-validation error (RMSE in caret) is used for tuning.

- Fitted a naive random forest model
  - obtained variable importance and plotted it
  - ran cross-validation (`rfcv`) on the naive model to exam the relationship between number of variables and cross-validation error. The error stops decreasing after the the number of predictors hits 17. Since more predictors don't increase error, we use all the predictors in the model.
- Tuning using caret

```
# tuning grid used
set.seed(444)
train_control <- trainControl(method="cv", number=5)
tuneGrid <- expand.grid(.mtry=c(8, 9, 10, 14, 20, 30, 37),
                        .min.node.size = c(5, 6, 7, 9, 10),
                        .splitrule = c("variance", "extratrees"))

tr1 = train(
  x = dtrain_full[ , names(dtrain_full) != 'price'],
  y = dtrain_full[ , names(dtrain_full) == 'price'],
  method = 'ranger', trControl = train_control, tuneGrid = tuneGrid
)

tr1
```

- Best parameters from this tuning result is used as our final model

Final model:

```
fit.rf <- ranger::ranger(price ~ . - fold,
                 data = dat_full,
                 mtry = 37, splitrule = "extratrees",
                 min.node.size = 5)
```

1.2.2 Smoothing Spline

Main package used: `mgcv::gam`

AIC is used to compare the performance between models

- Fitted all predictors naively
- Based on the summary of the naive model, I deleted insignificant ones and adjust number of knots for `ayb`
- Forward selection to examine significance of all interaction terms
- Added interaction (`by` in `s()`) between continuous and categorical
  - intuitively, it makes sense that the effect of earliest time the main portion of the building was built on price varies by different level of house conditions so I added it. And adding it indeed gives a lower AIC score.
- Based on p-values from the forward selection summary, I added important (p-value less than 0.05 at least, and I chose the ones with p-values as small as possible) interaction (`ti`) between continuous predictors.
  - even though I only chose only the significant interaction terms, there are still a lot options. But more interaction terms increase the running time significantly, I aimed to include less than 8 interaction terms.
  - fitted several models with different important interaction terms and compared AIC score between models.
  - AIC score hit a bottleneck after trying a few different models, and the model with the lowest AIC score gave me a good score on Kaggle, I believe it was reasonble to use that as my final model.

Final Model:

```
fit.sm <- mgcv::gam(price ~ s(rooms)
             +s(total_bath)+s(rmdl_diff)
             +s(bedrm)+s(ayb, k = 20, by = cndtn)+s(eyb)+s(saledate)+s(gba)
             +fireplaces
             +s(landarea)+s(latitude)+s(longitude)
             + heat+ac+style+grade+cndtn+roof+kitchens+ward
             +if_rmdl+buy_first
             + ti(eyb, ayb) + ti(gba,landarea)+ti(longitude,gba)
             + ti(longitude, ayb)
             +ti(longitude, eyb)+ti(saledate,latitude)
             ,data = dat_full)
```

2 Preprocessing Details

```
dat$fold <- fold

# ========== train ===============
avg_gap_train <- mean(dat$eyb-dat$ayb, na.rm = T)
# missing ayb is recoded as eyb - avg_gap between ayb and eyb
dat$ayb <- ifelse(is.na(dat$ayb), dat$eyb-avg_gap_train, dat$ayb)
# missing quadrant is recoded as "NW" bc "NW" is most popular
dat$quadrant <- ifelse(is.na(dat$quadrant), "NW", dat$quadrant)

# binary variable check whether the house is remodeled
dat$if_rmdl <- ifelse(is.na(dat$yr_rmdl), 0, 1)
dat$if_rmdl <- as.factor(dat$if_rmdl)

# year of the house sold
dat$saleyear<-as.numeric(substr(dat$saledate, 1, 4))

# the difference between sale year and the remodel year, if remodel is after sale
# then 0
for (i in seq(nrow(dat))) {
  if (is.na(dat$yr_rmdl[i])) {
    dat$rmdl_diff[i] <- 0
  } else if (dat$saleyear[i] <= dat$yr_rmdl[i]) {
    dat$rmdl_diff[i] <- 0
  } else if (dat$saleyear[i] > dat$yr_rmdl[i])
    dat$rmdl_diff[i] <- dat$saleyear[i] - dat$yr_rmdl[i]
}

# average room size
dat$avg_room_size <- dat$gba / (dat$rooms +1)

# year since build
dat$build_age <- as.numeric(substr(dat$saledate, 1,4)) - dat$ayb


# combine bathroom and half_bathroom
dat$total_bath <- dat$bathrm+0.5*dat$hf_bathrm

# whether it's sold first or build first
dat$buy_first <- as.factor(as.numeric(dat$saleyear < dat$ayb))



# manage level mismatching
tbl_heat <- table(dat$heat)
dat$heat[dat$heat %in% names(tbl_heat[tbl_heat<10])] <- "Other"
tbl_style <- table(dat$style)
dat$style[dat$style %in% names(tbl_style[tbl_style<10])] <- "Other"
tbl_grade <- table(dat$grade)
dat$grade[dat$grade %in% names(tbl_grade[tbl_grade<20])] <- "Other"
tbl_cndtn <- table(dat$cndtn)
dat$cndtn[dat$cndtn %in% names(tbl_cndtn[tbl_cndtn<30])] <- "Poor/Fair"
```

```r
dat$roof[dat$roof == "Typical"] <- "Comp Shingle"
tbl_roof <- table(dat$roof)
dat$roof[dat$roof %in% names(tbl_roof[tbl_roof<10])] <- "Other"
dat$intwall[dat$intwall == "Default"] <- "Hardwood"
tbl_intwall <- table(dat$intwall)
dat$intwall[dat$intwall %in% names(tbl_intwall[tbl_intwall<10])] <- "Other"
tbl_ward <- table(dat$ward)
dat$ward[dat$ward %in% names(tbl_ward[tbl_ward<10])] <- "Other"


# fill the na for yr_rmdl as 0
dat$yr_rmdl <- ifelse(is.na(dat$yr_rmdl), 0, dat$yr_rmdl)
# remove haft bathrooom
dat <- dat[, -which(names(dat) == "hf_bathrm")]
# remove yr_rmdl
dat <- dat[, -which(names(dat) == "yr_rmdl")]
```

```r
# omit other na
dat_full <- na.omit(dat)


# the number of days since 1970/01/01
dat_full$saledate <- as.numeric(as.Date(dat_full$saledate))

# to factor
dat_full[] <- lapply(dat_full, function(x) if(is.character(x)) factor(x) else x)

condition_levels <- c("Poor/Fair", "Average", "Good", "Very Good", "Excellent")
dat_full$cndtn <- factor(dat_full$cndtn, levels = condition_levels)


dat_full$gba <- log(dat_full$gba)
dat_full$landarea <- log(dat_full$landarea)
dat_full$saledate <- dat_full$saledate^5
dat_full$stories <- log(dat_full$stories)
dat_full$rmdl_diff <- log(dat_full$rmdl_diff + 1)
dat_full$price <- (dat_full$price^0.101-1)/0.101


dat_full$inter1 <- with(dat_full, latitude * saledate)
dat_full$inter2 <- with(dat_full, longitude * saledate)
dat_full$inter3 <- with(dat_full, gba * saledate)
dat_full$inter4 <- with(dat_full, landarea * longitude)
dat_full$inter5 <- with(dat_full, eyb * ayb)
dat_full$inter6 <- with(dat_full, build_age * latitude)

fold <- dat_full$fold
```

3 Cross-validation

We have separate cross validation function for smoothing spline model and random forest model, where the only difference is the fitted model.

The cross validation is done through the following key steps:

- separate data into training set (4 folds) and testing set (1 fold).
- fit the model using the training data set
- make prediction on the testing data set
- calculate the RMLSE

```
cv_sm <- function(data) {

  errors_sm <- numeric(5)

  for (i in 1:5) {
    train_data <- data[fold != i, ]
    test_data <- data[fold == i, ]

    sm_model <- gam(price ~ s(rooms)
              +s(total_bath)+s(rmdl_diff)
              +s(bedrm)+s(ayb, k = 20, by = cndtn)+s(eyb)+s(saledate)+s(gba)
              +fireplaces
              +s(landarea)+s(latitude)+s(longitude)
              + heat+ac+style+grade+cndtn+roof+kitchens+ward
              +if_rmdl+buy_first
              + ti(eyb, ayb) + ti(gba,landarea)+ti(longitude,gba)
              + ti(longitude, ayb)
              +ti(longitude, eyb)+ti(saledate,latitude)
              ,data = train_data)

    pred <- predict(sm_model, newdata = test_data)
    # reverse box-cox transformation
    pred_reverse <- (pred * 0.101 + 1)^(1/0.101)
    actual_responses <- test_data[['price']]
    # reverse box-cox transformation
    actual_responses <- (actual_responses * 0.101 + 1)^(1/0.101)

    errors_sm[i] <-  sqrt(mean((log(pred_reverse)-
                                log(actual_responses))^2))
  }
  return(errors_sm)
}
```

```
cv_rf <- function(data) {
  errors_rf <- numeric(5)

  for (i in 1:5) {
    train_data <- data[fold != i, ]
    test_data <- data[fold == i, ]

    rf_model <- ranger(price ~ . - fold,
```

```
                data = train_data,
                mtry = 37, splitrule = "extratrees",
                min.node.size = 5)

    pred <- predict(rf_model, data = test_data)$predictions
    # reverse box-cox transformation
    pred_reverse <- (pred * 0.101 + 1)^(1/0.101)
    actual_responses <- test_data[['price']]
    # reverse box-cox transformation
    actual_responses <- (actual_responses * 0.101 + 1)^(1/0.101)

    errors_rf[i] <- sqrt(mean((log(pred_reverse)-
                               log(actual_responses))^2))
  }
  return(errors_rf)
}
```

The 5 fold cross-validation score for `smoothing`:

```
set.seed(42)
cv_result_sm <- cv_sm(dat_full)
print(paste("5-fold cross-validatoin:" ,cv_result_sm))
```

```
## [1] "5-fold cross-validatoin: 0.16781614017129"
## [2] "5-fold cross-validatoin: 0.197484661024805"
## [3] "5-fold cross-validatoin: 0.192162056601297"
## [4] "5-fold cross-validatoin: 0.208260761794989"
## [5] "5-fold cross-validatoin: 0.188157206556455"
```

```
print(paste("Average: ", mean(cv_result_sm)))
```

```
## [1] "Average:  0.190776165229767"
```

The 5 fold cross-validation score for `random forest`:

```
set.seed(42)
cv_result_rf <- cv_rf(dat_full)
print(paste("5-fold cross-validatoin:" ,cv_result_rf))
```

```
## [1] "5-fold cross-validatoin: 0.181719803189755"
## [2] "5-fold cross-validatoin: 0.206810603266541"
## [3] "5-fold cross-validatoin: 0.200137545836101"
## [4] "5-fold cross-validatoin: 0.198220889590503"
## [5] "5-fold cross-validatoin: 0.192456821531735"
```

```
print(paste("Average: ", mean(cv_result_rf)))
```

```
## [1] "Average:  0.195869132682927"
```