STAT 444 Winter 2024 Final Project Supplemental Materials

Bruce Liu

UW ID: 20843361

Smoothing

Preprocessing

The following preprocessing has been done to the dataset.

Transformation (if any)

- price: Applied a power transform of 1/15 on the response variate price.
- saledate: Converted it to a numeric value of days since January 1st, 1970 (the beginning of Unix time)
- roof, extwall, intwall, nbhd: Applied factor() and converted into a numeric vector for better use in the model.
- cndtn, ac, style, grade, ward, heat, quadrant: Applied factor(), converted into a numeric vector, and ordered by increase in price

New Variables

- proximity: This variable is defined as the distance from the most expensive property in the dataset. The measurement is in degrees.

Missing data handling

- heat: replace the missing values in heat in dtrain with "No Data".
- stories, kitchens: replace the missing values in stories and kitchens in dtrain by imputing the k nearest neighbours to each missing value.
- quadrant: replace the missing values in quadrant by imputing the k nearest neighbours to each missing value.

```
datSmoothPreProc <- function(dtrain){
  dtrain$grade <- as.numeric(factor(dtrain$grade, levels = c("Low Quality",
                                                             "Fair Quality", "Average",
                                                             "Above Average", "Good Quality",
                                                             "Very Good", "Excellent",
                                                             "Superior","Exceptional-A",
                                                             "Exceptional-B", "Exceptional-C",
                                                             "Exceptional-D")))
  maxpricelat <- dtrain$latitude[which.max(dtrain$price)]
  maxpricelon <- dtrain$longitude[which.max(dtrain$price)]
  proximity <- sqrt((dtrain$latitude - maxpricelat)^2 + (dtrain$longitude - maxpricelon)^2)
  dtrain$proximity <- proximity
  dtrain$saledate <- as.numeric(as.Date(dtrain$saledate))
  dtrain$cndtn <- as.numeric(factor(dtrain$cndtn, levels = c("Poor","Fair","Average",
                                                             "Good", "Very Good", "Excellent")))
  dtrain$style <- as.numeric(factor(dtrain$style,levels = c("Split Foyer","1 Story",
```

```
                                                        "1.5 Story Fin", "1.5 Story Unfin",
                                                        "2.5 Story Unfin", "2 Story",
                                                        "Bi-Level","Split Level", "Default",
                                                        "2.5 Story Fin", "3 Story", "4 Story")))
dtrain$ac <- factor(dtrain$ac, levels = c("N", "Y"))
dtrain$ward <- factor(dtrain$ward, levels = c("Ward 7", "Ward 8", "Ward 5",
                                                       "Ward 6", "Ward 4", "Ward 1",
                                                       "Ward 3", "Ward 2"))
dtrain$nbhd <- as.numeric(factor(dtrain$nbhd))
dtrain$heat <- factor(dtrain$heat, levels = c("Wall Furnace", "Air-Oil", "Elec Base Brd",
                                              "Gravity Furnace", "No Data","Evp Cool",
                                              "Hot Water Rad", "Forced Air", "Water Base Brd",
                                              "Ht Pump","Warm Cool","Air Exchng"))
dtrain$roof <- as.numeric(factor(dtrain$roof))
dtrain$extwall <- as.numeric(factor(dtrain$extwall))
dtrain$intwall <- as.numeric(factor(dtrain$intwall))
library(impute)
# Handling of missing values in dtrain

for (i in 1:6000) {
  if (is.na(dtrain[i,"heat"])) {
    dtrain[i, "heat"] <- "No Data"
  }
}
dtrainYears <- dtrain[,c(7:9)]
knnYears <- impute::impute.knn(t(dtrainYears))
imputedYears <- t(knnYears$data)
dtrain$ayb[which(is.na(dtrain$ayb))] <- imputedYears[which(is.na(dtrain$ayb))]
dtrain$yr_rmdl[which(is.na(dtrain$yr_rmdl))] <- imputedYears[6000 + which(is.na(dtrain$yr_rmdl))]
dtrainRooms <- dtrain[, c("bathrm","hf_bathrm","rooms","bedrm",
                          "stories","kitchens","fireplaces")]
knnRooms <- impute::impute.knn(t(dtrainRooms))
imputedRooms <- t(knnRooms$data)
dtrain$stories[which(is.na(dtrain$stories))] <- imputedRooms[24000 + which(is.na(dtrain$stories))]
dtrain$kitchens[which(is.na(dtrain$kitchens))] <- imputedRooms[30000 + which(is.na(dtrain$kitchens))]
dtrain$RoomRatio <- as.numeric(unlist((dtrain["bathrm"]+dtrain["hf_bathrm"]+
                                       dtrain["bedrm"]+dtrain["kitchens"]+
                                       dtrain["fireplaces"])/(dtrain["rooms"] + 1)))
dtrain$stories[which(is.na(dtrain$stories))] <- imputedRooms[24000 + which(is.na(dtrain$stories))]
dtrain$kitchens[which(is.na(dtrain$kitchens))] <- imputedRooms[30000 + which(is.na(dtrain$kitchens))]
dtrainLocation <- dtrain[, c(23:27)]
dtrainLocation$nbhd <- as.numeric(dtrainLocation$nbhd)
dtrainLocation$ward <- as.numeric(factor(dtrainLocation$ward))
dtrainLocation$quadrant <- as.numeric(factor(dtrainLocation$quadrant))
knnLocation <- impute::impute.knn(t(dtrainLocation))
imputedLocation <- t(knnLocation$data)


for (i in c(1:6000)) {
  if (is.na(dtrain$quadrant[i])) {
    if (abs(imputedLocation[24000 + i]) <= 1) {
      dtrain$quadrant[i] <- "NE"
    } else if (abs(imputedLocation[24000 + i]) <= 2) {
```

```
        dtrain$quadrant[i] <- "NW"
      } else if (abs(imputedLocation[24000 + i]) <= 3) {
        dtrain$quadrant[i] <- "SE"
      } else {
        dtrain$quadrant[i] <- "SW"
      }
    }
  }
  dtrain$quadrant <- as.numeric(factor(dtrain$quadrant, levels = c("SW","SE","NE","NW")))
  return(dtrain)
}
datSmooth <- datSmoothPreProc(dat)
```

Model Building

A trial-and-error approach was used with `bam` for efficiency. Given that we have 5 folds to work with, we employ a 5-fold CV with the main metric being the APSE of the predictions. Adding, removing, and applying different transformations to the model and using the APSE where lower is better to determine the best model to run.

The following code is employed to calculate the APSE 5-fold cross validation error.

```
smoothCV <- function(dat){
  rmlseMetrics <- c()
  apseMetrics <- c()
  for (i in c(min(fold):max(fold))) {
    datTest <- datSmooth[which(fold == i),]
    datTrain <- datSmooth[which(fold != i),]
    for (i in 1:1200) {
      if (datTest$roof[i] == "Composition Ro") {
        datTest$roof[i] <- "Comp Shingle"
      } else if (datTest$roof[i] == "Metal- Pre") {
        datTest$roof[i] <- "Metal- Cpr"
      }
      if (datTest[i, "intwall"] == "Terrazo") {
        datTest[i, "intwall"] <- "Lt Concrete"
      } else if (datTest[i, "intwall"] == "Vinyl Sheet") {
        datTest[i, "intwall"] <- "Default"
      } else if (datTest[i, "intwall"] == "Parquet") {
        datTest[i, "intwall"] <- "Wood Floor"
      }
      if (datTest$extwall[i] == "Stucco Block") {
        datTest$extwall[i] <- "Stucco"
      }
    }
    fitSmooth <- mgcv::gam(price^(1/15)~s(proximity, by = ward, k = 26)+s(saledate,ayb,eyb)
                           +te(gba, landarea)+s(grade)+s(rooms,bedrm,bathrm)+s(fireplaces, k = 5)
                           +s(as.numeric(heat),by = ac)+s(hf_bathrm, k = 4)+s(cndtn, k = 6)
                           +roof +extwall+intwall+s(nbhd)
                           , data=datTrain)
    pred <- mgcv::predict.gam(fitSmooth, newdata=datTest, type = "response")
    rmlseMetrics <- c(rmlseMetrics, sqrt(mean((log(pred^15)-log(datTest$price))^2)))
    apseMetrics <- c(apseMetrics, sqrt(mean((pred^15-datTest$price)^2)))
  }
```

```
  return(mean(apseMetrics))
}
```

Much like in the Smoothing Model project, we have variables in the test fold that are not in the training fold. For this, we look at the closest analogue of the variable. For example, for the `intwall` variable, we have that "Terrazo" is in one fold but not the others. We learn that it is most similar to concrete. So, we impute the training fold value of "Lt Concrete" to the test fold. Similarly, we impute "Default" to "Vinyl Sheet". We continue with this imputation for the roof and exterior wall variables in the test fold until the training fold and the test fold have the same variables.

Random Forests

Preprocessing

All preprocessing for the random forest model is from the Random Forests project. In short,

Transformation (if any)

- price: Applied a power transform of 1/15 on the response variate price.
- saledate: Converted it to a numeric value of days since January 1st, 1970 (the beginning of Unix time)
- roof, extwall, intwall, nbhd: Applied factor() for better use in the model.
- cndtn, ac, style, grade, ward, heat, quadrant: Applied factor() and ordered by increase in price

New Variables

- proximity: This variable is defined as the distance from the most expensive property in the dataset. The measurement is in degrees.
- RpS: This variable is defined as the number of rooms per story of a building.
- yrDiff: This variable is defined as the difference in the years from the remodelling year and the year in which the oldest part of the building is built. (i.e. `ayb - yr_rmdl`)

Missing data handling

- heat: replace the missing values in heat in dtrain with "No Data".
- stories, kitchens: replace the missing values in stories and kitchens in dtrain by imputing the k nearest neighbours to each missing value.
- quadrant: replace the missing values in quadrant by imputing the k nearest neighbours to each missing value.

```
datRFPreProc <- function(dtrain){
  maxpricelat <- dtrain$latitude[which.max(dtrain$price)]
  maxpricelon <- dtrain$longitude[which.max(dtrain$price)]
  proximity <- sqrt((dtrain$latitude - maxpricelat)^2 + (dtrain$longitude - maxpricelon)^2)
  dtrain$proximity <- proximity
  dtrain$grade <- factor(dtrain$grade, levels = c("Low Quality",
                                                  "Fair Quality", "Average",
                                                  "Above Average", "Good Quality",
                                                  "Very Good", "Excellent",
                                                  "Superior","Exceptional-A",
                                                  "Exceptional-B", "Exceptional-C",
                                                  "Exceptional-D"))
  dtrain$saledate <- as.numeric(as.Date(dtrain$saledate))
  dtrain$cndtn <- factor(dtrain$cndtn, levels = c("Poor","Fair","Average",
                                                  "Good", "Very Good", "Excellent"))
  dtrain$ac <- factor(dtrain$ac, levels = c("N", "Y"))
```

```
dtrain$ward <- factor(dtrain$ward, levels = c("Ward 7", "Ward 8", "Ward 5",
                                              "Ward 6", "Ward 4", "Ward 1",
                                              "Ward 3", "Ward 2"))
dtrain$heat <- factor(dtrain$heat, levels = c("Wall Furnace", "Air-Oil", "Elec Base Brd",
                                              "Gravity Furnace", "No Data","Evp Cool",
                                              "Hot Water Rad", "Forced Air", "Water Base Brd",
                                              "Ht Pump","Warm Cool","Air Exchng"))
dtrain$nbhd <- factor(dtrain$nbhd)
dtrain$roof <- factor(dtrain$roof)
dtrain$extwall <- factor(dtrain$extwall)
dtrain$intwall <- factor(dtrain$intwall)
dtrain$style <- factor(dtrain$style, levels = c("Split Foyer","1 Story",
                                                "1.5 Story Fin", "1.5 Story Unfin",
                                                "2.5 Story Unfin", "2 Story",
                                                "Bi-Level","Split Level", "Default",
                                                "2.5 Story Fin", "3 Story", "4 Story"))
library(impute)
# Handling of missing values in dtrain

for (i in 1:6000) {
  if (is.na(dtrain[i,"heat"])) {
    dtrain[i, "heat"] <- "No Data"
  }
}
dtrainYears <- dtrain[,c(7:9)]
knnYears <- impute::impute.knn(t(dtrainYears))
imputedYears <- t(knnYears$data)
dtrain$ayb[which(is.na(dtrain$ayb))] <- imputedYears[which(is.na(dtrain$ayb))]
dtrain$yr_rmdl[which(is.na(dtrain$yr_rmdl))] <- imputedYears[6000 + which(is.na(dtrain$yr_rmdl))]
dtrainRooms <- dtrain[, c("bathrm","hf_bathrm","rooms","bedrm",
                          "stories","kitchens","fireplaces")]
knnRooms <- impute::impute.knn(t(dtrainRooms))
imputedRooms <- t(knnRooms$data)
dtrain$stories[which(is.na(dtrain$stories))] <- imputedRooms[24000 + which(is.na(dtrain$stories))]
dtrain$kitchens[which(is.na(dtrain$kitchens))] <- imputedRooms[30000 + which(is.na(dtrain$kitchens))]
dtrain$RoomRatio <- as.numeric(unlist((dtrain["bathrm"]+dtrain["hf_bathrm"]+
                                       dtrain["bedrm"]+dtrain["kitchens"]+
                                       dtrain["fireplaces"])/(dtrain["rooms"] + 1)))
dtrain$stories[which(is.na(dtrain$stories))] <- imputedRooms[24000 + which(is.na(dtrain$stories))]
dtrain$kitchens[which(is.na(dtrain$kitchens))] <- imputedRooms[30000 + which(is.na(dtrain$kitchens))]
dtrainLocation <- dtrain[, c(23:27)]
dtrainLocation$nbhd <- as.numeric(dtrainLocation$nbhd)
dtrainLocation$ward <- as.numeric(factor(dtrainLocation$ward))
dtrainLocation$quadrant <- as.numeric(factor(dtrainLocation$quadrant))
knnLocation <- impute::impute.knn(t(dtrainLocation))
imputedLocation <- t(knnLocation$data)


for (i in c(1:6000)) {
  if (is.na(dtrain$quadrant[i])) {
    if (abs(imputedLocation[24000 + i]) <= 1) {
      dtrain$quadrant[i] <- "NE"
    } else if (abs(imputedLocation[24000 + i]) <= 2) {
```

```
        dtrain$quadrant[i] <- "NW"
      } else if (abs(imputedLocation[24000 + i]) <= 3) {
        dtrain$quadrant[i] <- "SE"
      } else {
        dtrain$quadrant[i] <- "SW"
      }
    }
  }
  dtrain$quadrant <- factor(dtrain$quadrant, levels = c("SW","SE","NE","NW"))
  dtrain$RpS <- dtrain$rooms/dtrain$stories
  dtrain$yrDiff <- dtrain$yr_rmdl - dtrain$ayb
  dtrain$styleStories <- as.numeric(dtrain$style)*dtrain$stories
  return(dtrain)
}
datRF <- datRFPreProc(dat)
```

Model Building

A trial-and-error approach was used with `ranger`. Given that we have 5 folds to work with, we employ a 5-fold CV with the main metric being the RMSE of the predictions. Adding, removing, and applying different transformations to the model and using the APSE where lower is better to determine the best model to run. The variables selected are based on the variable importance plot in the Random Forests project. Starting from the variable of highest importance, add variables until the APSE is greater than the lowest APSE achieved until that point.

Parameters tuned and their optimal values:

- `mtry`: 7
- `max.depth`: 42
- `min.bucket`: 1
- `min.node.size`: 3
- `respect.unordered.factors`: TRUE

The following code is employed to calculate the APSE 5-fold cross validation error.

```
rfCV <- function(dat, j){
  # j is the paramater tuning value for the RF model, can be unused
  rmlseMetrics <- c() # vector for the RMSE of the data
  apseMetrics <- c()
  for (i in c(min(fold):max(fold))) { # runs for however many folds in the dataset
    datTest <- datRF[which(fold == i),] # test split
    datTrain <- datRF[which(fold != i),] # training split
    fitRF <- ranger::ranger(price^(1/15) ~ saledate + nbhd + longitude + proximity +
                              gba + grade + bathrm + eyb + ward + cndtn + latitude,
                            data = datRF, mtry = 7, respect.unordered.factors = TRUE,
                            min.bucket = 1, min.node.size = 3, max.depth = j)
    # fit the random forest based on a value of a parameter
    pred <- predict(fitRF, data=datTest)
    rmlseMetrics <- c(rmlseMetrics, sqrt(mean((log(pred$predictions^15)-log(datTest$price))^2)))
    apseMetrics <- c(apseMetrics, sqrt(mean((pred$predictions^15-datTest$price)^2)))
  }
  return(mean(apseMetrics)) # returns the average APSE
}
```

Producing the Cross Validation Error Scores

```
smoothCV(datSmooth)
```

```
## [1] 223731.4
```

```
system.time(smoothCV(datSmooth))
```

```
##    user  system elapsed
##  176.41    1.57  191.11
```

```
rfCV(datRF, 42)
```

```
## [1] 103260.7
```

```
system.time(rfCV(datRF,42))
```

```
##    user  system elapsed
##   59.11    0.71    5.92
```