Home Price Analysis

Cindy Cao

1/15/2019

Outline:

1. Problem Statement
2. Feature Engineering
3. Exploratory Data Analysis
4. Linear Regression
5. Gradient Boosting
6. Conclusion
7. Next Step

1. Problem Statement

Home price is a popular and well discussed topic, not only among home buyers, but also among investors, real-estate agencies, online listing platforms, etc. A traditional way to predict home prices is by analyzing house characteritics, such as location, square feet, building year, building materials, etc. However, with the increasing data availabilities, more complex and accurate models utilizing unconventional datasets or modeling techniques are emerging. This project aims to explore the relationship between home prices and various weather observations to better predict home prices.

2. Feature Engineering

####Load depository path and functions

```
path <- "/Users/xucao/Google Drive/MyWorkStation/Projects/ODG-Weather-HousingPrice/"
dataloc <- paste0(path, "DATA/")
codeloc <- paste0(path, "CODE/")
output <- paste0(path, "OUTPUT/")
source(paste0(codeloc, "functions.r"))
source(paste0(codeloc, "attr.bivar.R"))  #self defined bivariate plot function
```

####Read in text files

#####read in housing data

```
housing_data <- fread(paste0(dataloc, "housing_data_ACS_15_5YR_DP04_with_ann.csv"),
    header = TRUE)
# rename zip code column
colnames(housing_data)[2] <- "ZIP"
# convert zip code column to numeric
```

```r
housing_data$ZIP = as.numeric((housing_data$ZIP))
# drop the top row with variables descriptions
housing_data <- housing_data[2:nrow(housing_data),
    ]
```

#####read in geographic data

```r
Gaz_zcta_national <- fread(paste0(dataloc, "2015_Gaz_zcta_national.txt"),
    header = TRUE)
colnames(Gaz_zcta_national)[1] <- "ZIP"
# convert integer64 columns to numeric
Gaz_zcta_national$ALAND <- as.numeric(Gaz_zcta_national$ALAND)
Gaz_zcta_national$AWATER <- as.numeric(Gaz_zcta_national$AWATER)
```

#####read in location data

```r
weather_allstations <- read_fwf(file = paste0(dataloc,
    "weather_allstations.txt"), fwf_widths(c(12,
    9, 10, 7, 3, 31, 4, 4, 6)))
# renaming columns
names(weather_allstations) <- c("STATION_ID", "LAT",
    "LONG", "ELEVATION", "STATE", "STATION_NAME",
    "SOURCE1", "SOURCE2", "ZIP_STATION")
# subset useful variables
weather_allstations <- weather_allstations[, c("STATION_ID",
    "ELEVATION", "STATE")]
```

#####read in zipcode to weather station mapping table

```r
weather_zipcodes_stations <- read_fwf(file = paste0(dataloc,
    "weather_zipcodes-normals-stations.txt"), fwf_widths(c(12,
    6, 50)))
# Renaming columns
names(weather_zipcodes_stations) <- c("STATION_ID",
    "ZIP", "CITY")
# Formatting column 'CITY' and 'ZIP', all 'ZIP'
# columns are converted to numeric for matching
# purposes
weather_zipcodes_stations$CITY <- toupper(weather_zipcodes_stations$CITY)
weather_zipcodes_stations$ZIP <- as.numeric(weather_zipcodes_stations$ZIP)
# Create primary key
weather_zipcodes_stations$STATION <- paste0(formatC(weather_zipcodes_stations$ZIP,
    width = 5, flag = "0"), toupper(weather_zipcodes_stations$CITY))
```

#####read in weather data

```r
namelist <- c("STATION_ID", "Weat_Jan", "Weat_Jan_Type",
    "Weat_Feb", "Weat_Feb_Type", "Weat_Mar", "Weat_Mar_Type",
    "Weat_Apr", "Weat_Apr_Type", "Weat_May", "Weat_May_Type",
    "Weat_Jun", "Weat_Jun_Type", "Weat_Jul", "Weat_Jul_Type",
    "Weat_Aug", "Weat_Aug_Type", "Weat_Sep", "Weat_Sep_Type",
    "Weat_Oct", "Weat_Oct_Type", "Weat_Nov", "Weat_Nov_Type",
```

```
    "Weat_Dec", "Weat_Dec_Type")

weather_prcp <- read_fwf(file = paste0(dataloc, "weather_mly-prcp-normal.txt"),
    fwf_widths(c(17, 6, 1, 6, 1, 6, 1, 6, 1, 6, 1,
        6, 1, 6, 1, 6, 1, 6, 1, 6, 1, 6, 1, 6, 1)))
weather_tavg <- read_fwf(file = paste0(dataloc, "weather_mly-tavg-normal.txt"),
    fwf_widths(c(17, 6, 1, 6, 1, 6, 1, 6, 1, 6, 1,
        6, 1, 6, 1, 6, 1, 6, 1, 6, 1, 6, 1, 6, 1)))
weather_tmax <- read_fwf(file = paste0(dataloc, "weather_mly-tmax-normal.txt"),
    fwf_widths(c(17, 6, 1, 6, 1, 6, 1, 6, 1, 6, 1,
        6, 1, 6, 1, 6, 1, 6, 1, 6, 1, 6, 1, 6, 1)))
weather_tmin <- read_fwf(file = paste0(dataloc, "weather_mly-tmin-normal.txt"),
    fwf_widths(c(17, 6, 1, 6, 1, 6, 1, 6, 1, 6, 1,
        6, 1, 6, 1, 6, 1, 6, 1, 6, 1, 6, 1, 6, 1)))

names(weather_prcp) <- paste0(namelist, "_prcp")
names(weather_prcp)[1] <- "STATION_ID"
names(weather_tavg) <- paste0(namelist, "_tavg")
names(weather_tavg)[1] <- "STATION_ID"
names(weather_tmax) <- paste0(namelist, "_tmax")
names(weather_tmax)[1] <- "STATION_ID"
names(weather_tmin) <- paste0(namelist, "_tmin")
names(weather_tmin)[1] <- "STATION_ID"

# precipitation cannot be negative, so we convert
# negative values to NA
for (i in 1:ncol(weather_prcp)) {
    if (class(weather_prcp[, i]) == "numeric") {
        weather_prcp[, i] <- ifelse(weather_prcp[,
            i] < 0, "NA", weather_prcp[, i])
    }
}


clpcdy15 <- separate(read_fwf(file = paste0(dataloc,
    "clpcdy15.txt"), fwf_widths(c(38, 3, 3, 3, 3,
    3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
    3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
    3, 3, 3, 4, 4, 4)), skip = 2), col = "X1", into = c("STATION",
    "STATE"), sep = ",")
names(clpcdy15) = c("STATION", "STATE", "YRS", "Jan_CL",
    "Jan_PC", "Jan_CD", "Feb_CL", "Feb_PC", "Feb_CD",
    "Mar_CL", "Mar_PC", "Mar_CD", "Apr_CL", "Apr_PC",
    "Apr_CD", "May_CL", "May_PC", "May_CD", "Jun_CL",
    "Jun_PC", "Jun_CD", "Jul_CL", "Jul_PC", "Jul_CD",
    "Aug_CL", "Aug_PC", "Aug_CD", "Sep_CL", "Sep_PC",
    "Sep_CD", "Oct_CL", "Oct_PC", "Oct_CD", "Nov_CL",
    "Nov_PC", "Nov_CD", "Dec_CL", "Dec_PC", "Dec_CD",
    "Ann_CL", "Ann_PC", "Ann_CD")

# Base on Master Location Identifier Database
# downloaded from
# 'http://www.weathergraphics.com/identifiers/',
```

```r
# WBAN is the weather station identifiersm. The
# first column of dataset clpcdy15 and pctpos15
# consists of WBAN and CITY name.
clpcdy15$WBAN <- as.numeric(substr(clpcdy15$STATION,
    1, 5))
MLID <- fread(file = paste0(dataloc, "MLID.csv"),
    header = TRUE)
MLID <- MLID %>% select(WBAN, (CITY))
MLID$CITY <- toupper(MLID$CITY)
clpcdy15 <- clpcdy15 %>% left_join(MLID, by = "WBAN")
clpcdy15$CITY <- coalesce(clpcdy15$CITY, substr(clpcdy15$STATION,
    6, 50))
clpcdy15$STATION = paste0(clpcdy15$WBAN, clpcdy15$CITY)

pctpos15 <- separate(read_fwf(file = paste0(dataloc,
    "pctpos15.txt"), fwf_widths(c(37, 16, 6, 6, 6,
    6, 6, 6, 6, 6, 6, 6, 6, 6, 6)), skip = 2), col = "X1",
    into = c("STATION", "STATE"), sep = ",")
names(pctpos15) = c("STATION", "STATE", "POR", "JAN",
    "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG",
    "SEP", "OCT", "NOV", "DEC", "ANN")
pctpos15$WBAN <- as.numeric(substr(pctpos15$STATION,
    1, 5))
pctpos15$CITY <- substr(pctpos15$STATION, 6, 50)
```

####Check variable datatype

```r
# sapply(housing_data, class) We only keep the
# actual values (HC01), and drop other statistics
# columns, such as margin of error, percentage,
# etc.
housing_data <- housing_data %>% select(ZIP, starts_with("HC01_"))
# convert to numerica data
housing = as.data.frame(sapply(housing_data, as.numeric))
# sapply(Gaz_zcta_national, class)
# sapply(weather_allstations, class)
weather_allstations$ELEVATION <- as.numeric(weather_allstations$ELEVATION)
# sapply(weather_zipcodes_stations, class)
# sapply(weather_prcp, class)
# sapply(weather_tavg, class) convert character
# tempurature data into numeric data
weather_tavg$Weat_Jul_tavg <- as.numeric(weather_tavg$Weat_Jul_tavg)
weather_tavg$Weat_Aug_tavg <- as.numeric(weather_tavg$Weat_Aug_tavg)
# sapply(weather_tmax, class) convert character
# tempurature data into numeric data
weather_tmax$Weat_May_tmax <- as.numeric(weather_tmax$Weat_May_tmax)
weather_tmax$Weat_Jun_tmax <- as.numeric(weather_tmax$Weat_Jun_tmax)
weather_tmax$Weat_Jul_tmax <- as.numeric(weather_tmax$Weat_Jul_tmax)
weather_tmax$Weat_Aug_tmax <- as.numeric(weather_tmax$Weat_Aug_tmax)
weather_tmax$Weat_Sep_tmax <- as.numeric(weather_tmax$Weat_Sep_tmax)
# sapply(weather_tmin, class) sapply(clpcdy15,
# class)
clpcdy15 <- cbind(clpcdy15[1:3], lapply(clpcdy15[4:42],
```

```r
    as.numeric), clpcdy15[43:44])
# sapply(pctpos15, class)
```

####Check datasets primary key duplicates

```r
# clpcdy15$STATION[duplicated(clpcdy15$STATION)]
# dedup on primary key
clpcdy15 <- clpcdy15[!duplicated(clpcdy15$STATION),
    ]
# pctpos15$STATION[duplicated(pctpos15$STATION)]
pctpos15 <- pctpos15[!duplicated(pctpos15$STATION),
    ]
# housing$ZIP[duplicated(housing$ZIP)]
# Gaz_zcta_national$ZIP[duplicated(Gaz_zcta_national$ZIP)]
# weather_allstations$STATION_ID[duplicated(weather_allstations$STATION_ID)]
# weather_zipcodes_stations$STATION_ID[duplicated(weather_zipcodes_stations$STATION_ID)]
# weather_prcp$STATION_ID[duplicated(weather_prcp$STATION_ID)]
# weather_tavg$STATION_ID[duplicated(weather_tavg$STATION_ID)]
# weather_tmax$STATION_ID[duplicated(weather_tmax$STATION_ID)]
# weather_tmin$STATION_ID[duplicated(weather_tmin$STATION_ID)]
```

####Merge datasets

```r
data <- weather_zipcodes_stations %>% left_join(weather_allstations,
    by = "STATION_ID") %>% left_join(weather_prcp,
    by = "STATION_ID") %>% left_join(weather_tavg,
    by = "STATION_ID") %>% left_join(weather_tmax,
    by = "STATION_ID") %>% left_join(weather_tmin,
    by = "STATION_ID") %>% left_join(housing, by = "ZIP") %>%
    left_join(Gaz_zcta_national, by = "ZIP") %>%
    left_join(pctpos15, by = "STATION") %>% left_join(clpcdy15,
    by = "STATION")
```

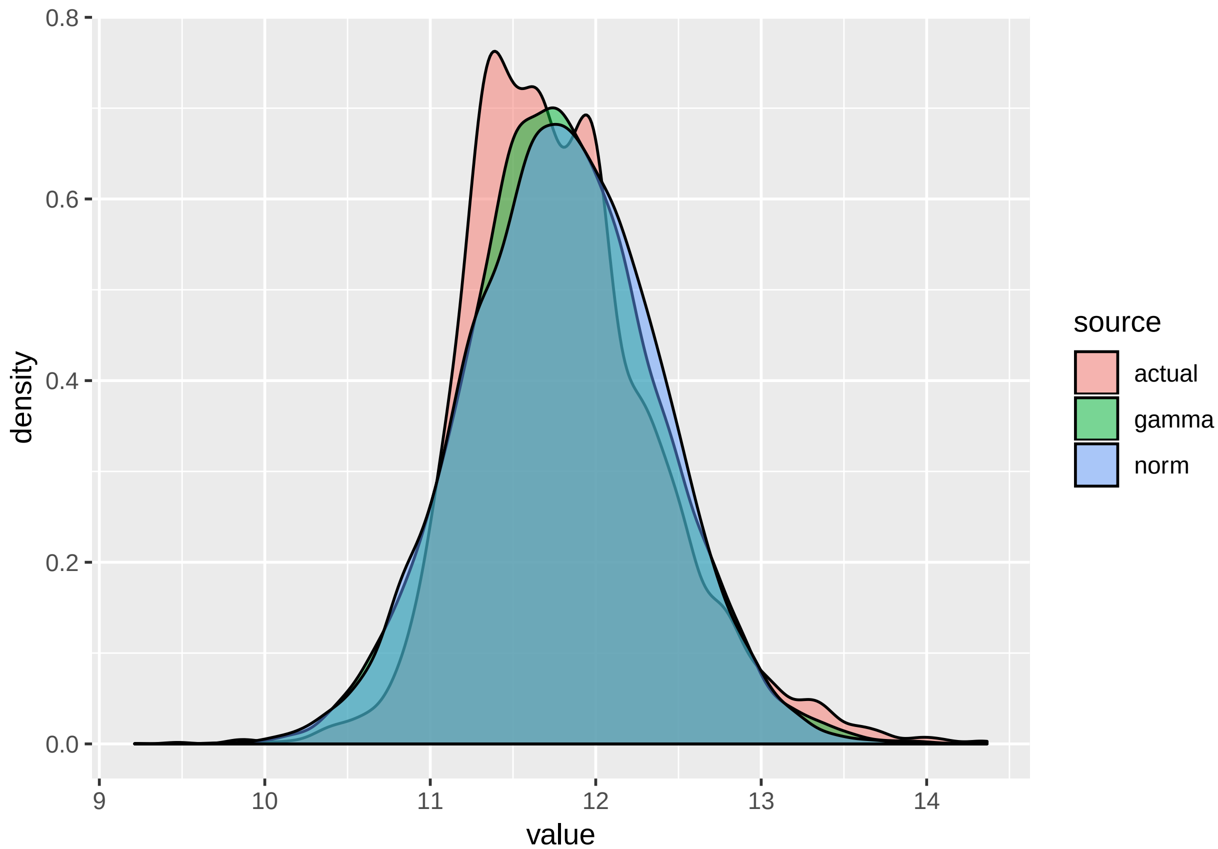
####Examine target

```r
summary(data$HC01_VC128)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   13000   88200  123700  159727  181300 1732700     371
```

```r
# drop records with NA target, these records
# without target informaiton are not useful to
# our supervised learning analysis.
data <- data[!is.na(data$HC01_VC128), ]

# simulate gamma distribution
theta = (sd(log(data$HC01_VC128)))^2/mean(log(data$HC01_VC128))
k = mean(log(data$HC01_VC128))/theta
gamma <- as.data.frame(rgamma(nrow(data), shape = k,
    scale = theta))
gamma$source <- "gamma"
```

```r
names(gamma) = c("value", "source")
# simulate normal distribution
mean <- mean(log(data$HC01_VC128))
sd <- sd(log(data$HC01_VC128))
norm <- as.data.frame(rnorm(nrow(data), mean = mean,
    sd = sd))
norm$source = "norm"
names(norm) = c("value", "source")
# plot actual dependent variable distribution
actual <- as.data.frame(log(data$HC01_VC128))
actual$source = "actual"
names(actual) = c("value", "source")
dist <- rbind(gamma, norm, actual)
dist$value <- as.numeric(dist$value)
ggplot(dist, aes(value, fill = source)) + geom_density(alpha = 0.5)
```




```r
# based on the above analysis, by comparing the
# distribution shape, a log normal distribution
# will be used for the target.
data$Target <- log(data$HC01_VC128)
```

####Split Train/Test/Holdout: 50%/25%/25%

This step is to prepare data for future modeling practice, and we only conduct featue engineering and exploratory analysis on the training set.

```r
set.seed(1)
inTrainingSet = createDataPartition(log(data$HC01_VC128),
    p = 0.5, list = FALSE)
dt = data[inTrainingSet, ]
trainset = data[-inTrainingSet, ]
set.seed(123)
inTrainingSet = createDataPartition(log(dt$HC01_VC128),
    p = 0.5, list = FALSE)
holdout = dt[inTrainingSet, ]
testset = dt[-inTrainingSet, ]

gbm.trainset <- trainset
gbm.testset <- testset
```

Check dependent variable distributions in different datasets. We are looking for similar dependent variable distributions among train/test/holdout sets to ensure that we do not have biased sample selection.

```r
## train number of records:
nrow(trainset)
```

```
## [1] 4710
```

```r
summary(log(trainset$HC01_VC128))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   9.473  11.387  11.726  11.793  12.108  14.362
```

```r
train_target_dist <- cbind(log(trainset$HC01_VC128),
    "trainset")
## test number of records:
nrow(testset)
```

```
## [1] 2355
```

```r
summary(log(testset$HC01_VC128))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.10   11.38   11.72   11.79   12.11   14.33
```

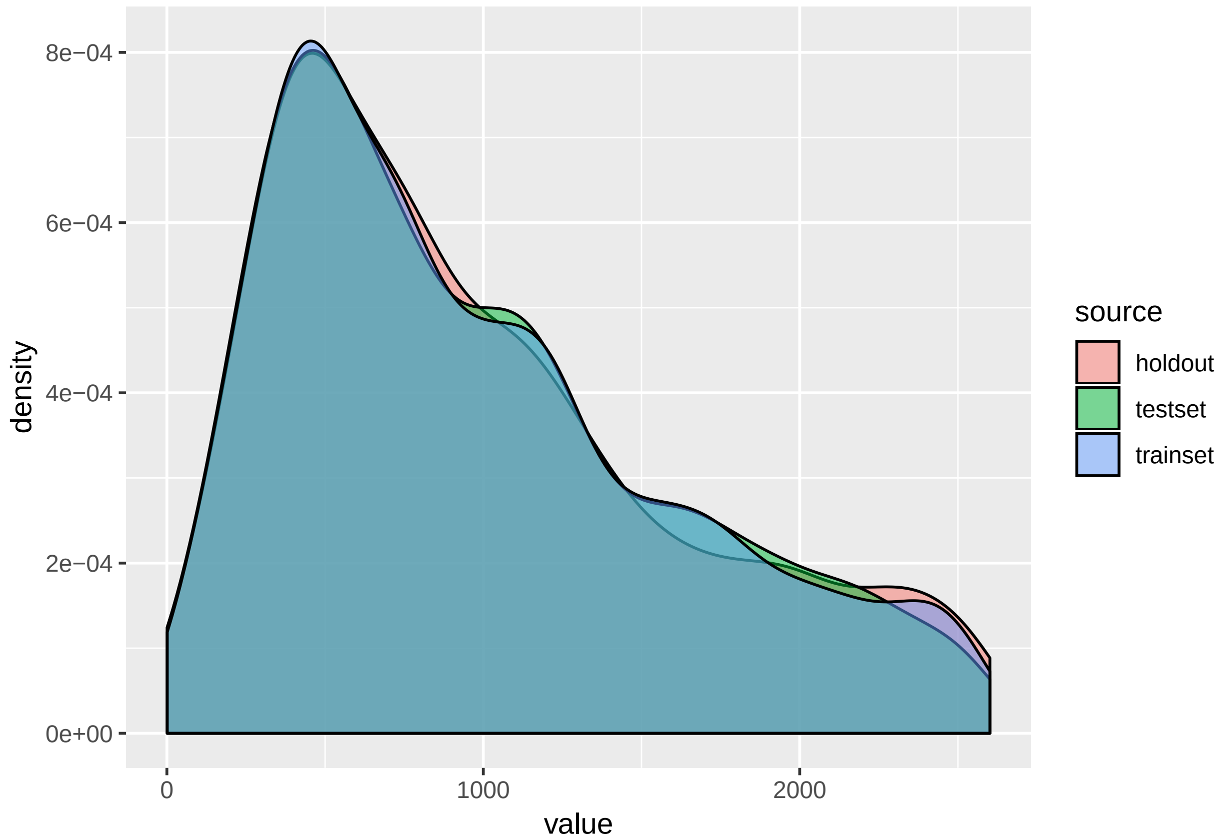
```r
test_target_dist <- cbind(log(testset$HC01_VC128),
    "testset")
## holdout number of records:
nrow(holdout)
```

```
## [1] 2358
```

```r
summary(log(holdout$HC01_VC128))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   9.473  11.387  11.726  11.799  12.107  14.365
```

```r
holdout_target_dist <- cbind(log(holdout$HC01_VC128),
    "holdout")

target_dist <- as.data.frame(rbind(train_target_dist,
    test_target_dist, holdout_target_dist))
names(target_dist) <- c("value", "source")
target_dist$value <- as.numeric(target_dist$value)
ggplot(target_dist, aes(value, fill = source)) +
    geom_density(alpha = 0.5)
```



Before imputing missing values, we create missing flags and cap outliers first, as missing flags carry meaningful information, and outliers can impact missing value imputation.

####Create missing flags

```r
for (i in 1:ncol(trainset)) {
    if (colnames(trainset)[i] != "HC01_VC128" & colnames(trainset)[i] !=
        "Target") {
        trainset[, paste0("flag_", colnames(trainset)[i])] <- as.factor((ifelse(is.na(trainset[,
            i]), 1, 0)))
    }
}
```

####Split Train/Test/Holdout: 50%/25%/25%

This step is to prepare data for future modeling practice, and we only conduct featue engineering and exploratory analysis on the training set.

```
set.seed(1)
inTrainingSet = createDataPartition(log(data$HC01_VC128),
    p = 0.5, list = FALSE)
dt = data[inTrainingSet, ]
trainset = data[-inTrainingSet, ]
set.seed(123)
inTrainingSet = createDataPartition(log(dt$HC01_VC128),
    p = 0.5, list = FALSE)
holdout = dt[inTrainingSet, ]
testset = dt[-inTrainingSet, ]

gbm.trainset <- trainset
gbm.testset <- testset
```

Check dependent variable distributions in different datasets. We are looking for similar dependent variable distributions among train/test/holdout sets to ensure that we do not have biased sample selection.

```
## train number of records:
nrow(trainset)
```

```
## [1] 4710
```

```
summary(log(trainset$HC01_VC128))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   9.473  11.387  11.726  11.793  12.108  14.362
```

```
train_target_dist <- cbind(log(trainset$HC01_VC128),
    "trainset")
## test number of records:
nrow(testset)
```

```
## [1] 2355
```

```
summary(log(testset$HC01_VC128))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.10   11.38   11.72   11.79   12.11   14.33
```

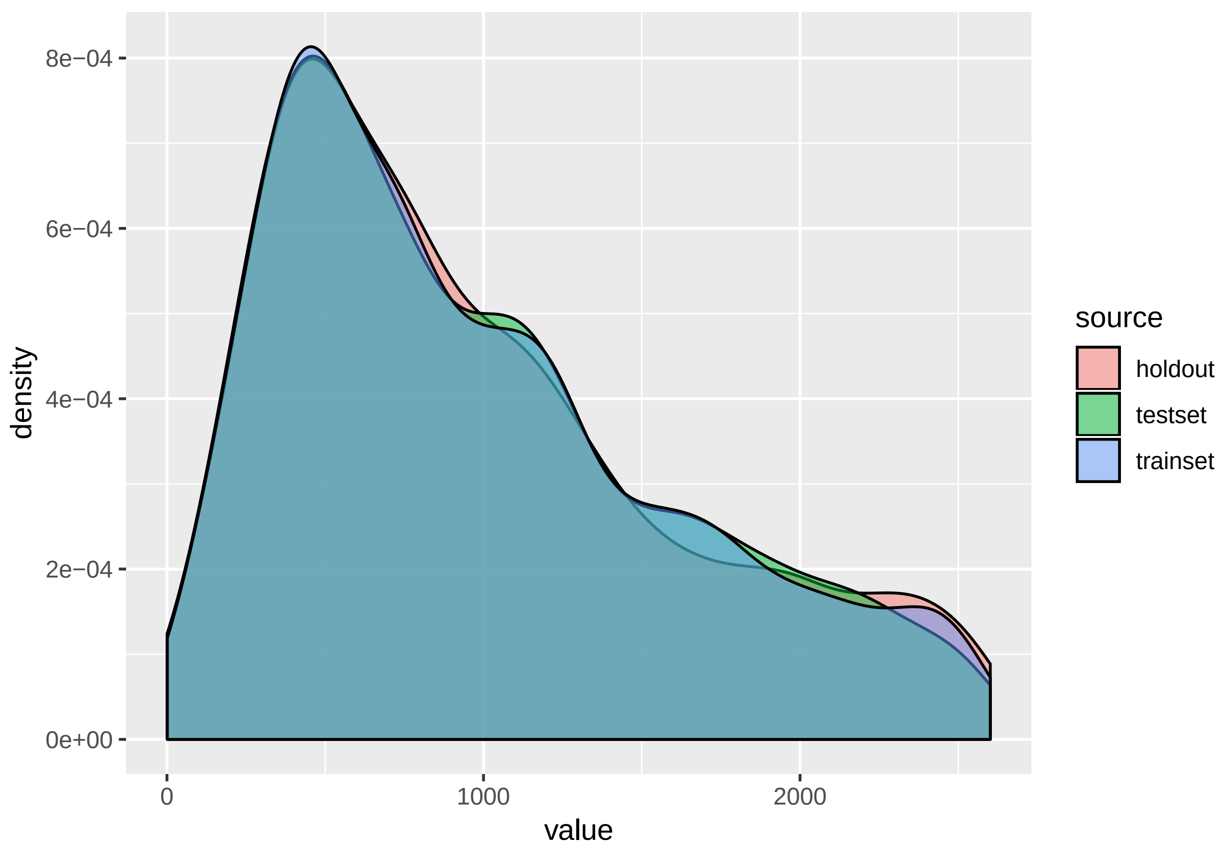
```
test_target_dist <- cbind(log(testset$HC01_VC128),
    "testset")
## holdout number of records:
nrow(holdout)
```

```
## [1] 2358
```

```r
summary(log(holdout$HC01_VC128))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   9.473  11.387  11.726  11.799  12.107  14.365
```

```r
holdout_target_dist <- cbind(log(holdout$HC01_VC128),
    "holdout")

target_dist <- as.data.frame(rbind(train_target_dist,
    test_target_dist, holdout_target_dist))
names(target_dist) <- c("value", "source")
target_dist$value <- as.numeric(target_dist$value)
ggplot(target_dist, aes(value, fill = source)) +
    geom_density(alpha = 0.5)
```



Before imputing missing values, we create missing flags and cap outliers first, as missing flags carry meaningful information, and outliers can impact missing value imputation.

####Create missing flags

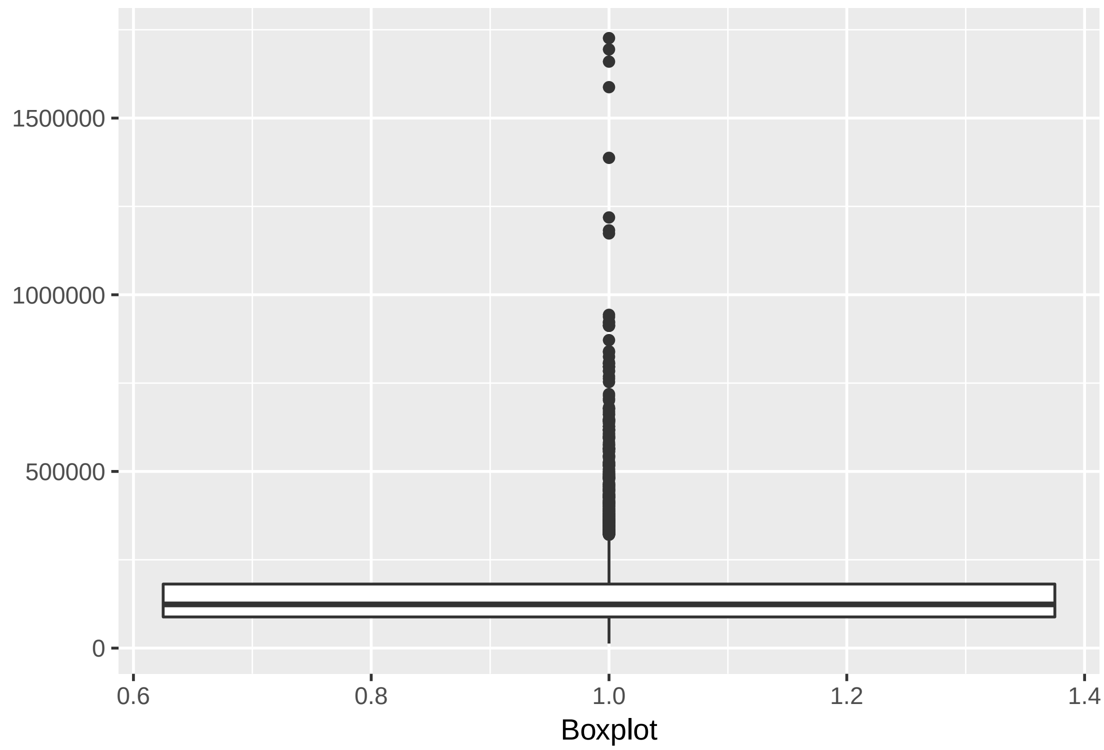
```r
for (i in 1:ncol(trainset)) {
    if (colnames(trainset)[i] != "HC01_VC128" & colnames(trainset)[i] !=
        "Target") {
        trainset[, paste0("flag_", colnames(trainset)[i])] <- as.factor((ifelse(is.na(trainset[,
            i]), 1, 0)))
```

```
    }
}
```

####Cap outliers

Due to the constrained time, here we use 5% and 95% quantiles to cap all the attributes. For future study, we can run chi-square outlier test and treat each attribute separately for outlier capping.

```r
# boxplot shows outliers for dependent variable
ggplot(data = trainset, aes(x = 1, y = trainset$HC01_VC128)) +
    geom_boxplot() + xlab("Boxplot") + ylab("")
```




```r
# tranform and cap dependent variable
trainset$Target <- pmin(log(trainset$HC01_VC128),
    quantile(log(trainset$HC01_VC128), 0.99))
trainset <- trainset %>% select(-HC01_VC128)

# cap numeric independent variable
for (i in 1:ncol(trainset)) {
    if ((substr(colnames(trainset)[i], 1, 5) == "Weat_" |
        substr(colnames(trainset)[i], 1, 5) == "HC01_") &
        (class(eval(parse(text = paste0("trainset$",
            colnames(trainset)[i])))) == "numeric" |
            class(eval(parse(text = paste0("trainset$",
                colnames(trainset)[i])))) == "integer")) {
        trainset[, paste0(colnames(trainset)[i],
```

```r
            "_cap")] <- pmin(pmax(eval(parse(text = paste0("trainset$",
            colnames(trainset)[i]))), quantile(eval(parse(text = paste0("trainset$",
            colnames(trainset)[i]))), 0.05, na.rm = TRUE)),
            quantile(eval(parse(text = paste0("trainset$",
                colnames(trainset)[i]))), 0.95, na.rm = TRUE))
    }
}

# save a capping value table for testset
# transformation later
trainset.num <- trainset[, sapply(trainset, class) ==
    "numeric" | sapply(trainset, class) == "integer"]
capping <- as.matrix(rbind(colnames(trainset.num),
    sapply(trainset.num, function(x) quantile(x,
        0.05, na.rm = TRUE)), sapply(trainset.num,
        function(x) quantile(x, 0.95, na.rm = TRUE))))
colnames(capping) <- capping[1, ]
capping <- as.data.frame(capping[2:3, ])
```

####Impute missing values

Regarding to missing values, we treat variables from different datasets differently.

1. For weather related variables, climate is highy correlated with location, and areas within the same state share similar climate, therefore one way to impute the weather data is to use the state mean, and we have fully populated state information in the dataset, which makes this approach plausible.

2. For house related variables, we can try two different approches, one is to utilize bivariate plot to go through the relationships between each indenpent variable and independent variables, and impute the missing values with user defined statistical method (mean, min, max, zero, etc.), or we can use k nearest neighor to impute the missing values. Here we use k-nearest neighbor to impute our housing variables.

Select useful columns

```r
trainset <- as.data.frame(trainset %>% select(Target,
    ends_with("_cap"), starts_with("flag_"), contains("_Type"),
    ELEVATION, STATE.x, ALAND, AWATER))
```

Seperate different data type columns

```r
colclasses <- sapply(trainset, class)
numColumns <- which(colclasses == "numeric" | colclasses ==
    "integer")
JustNumbers <- trainset %>% select(STATE.x, numColumns)
```

Create state average lookup table for weather related attributes imputation

```
weat_state <- list()
for (i in seq_along(1:(ncol(JustNumbers)))) {
    if (i == 1) {
        weat_state[[i]] <- as.vector((JustNumbers %>%
            select(STATE.x, colnames(JustNumbers)[i]) %>%
            group_by(STATE.x) %>% summarise(avg = mean(eval(parse(text = colnames(JustNumbers)[i])),
            na.rm = TRUE)))[, 1])
    } else {
        weat_state[[i]] <- as.vector((JustNumbers %>%
            select(STATE.x, colnames(JustNumbers)[i]) %>%
            group_by(STATE.x) %>% summarise(avg = mean(eval(parse(text = colnames(JustNumbers)[i])),
            na.rm = TRUE)))[, 2])
    }
}

weat_state_avg <- matrix(nrow = 51)
for (i in 1:length(weat_state)) {
    weat_state_avg <- cbind(weat_state_avg, (unlist(weat_state[i])))
}
weat_state_avg <- as_tibble((weat_state_avg))[, 2:ncol(weat_state_avg)]
weat_state_avg <- cbind(weat_state_avg[, 1], sapply(weat_state_avg[,
    2:ncol(weat_state_avg)], as.numeric))
names(weat_state_avg) = c("STATE.x", paste0("state_avg_",
    colnames(JustNumbers)[2:ncol(JustNumbers)]))

trainset <- as.data.frame(trainset %>% left_join(weat_state_avg,
    by = "STATE.x"))
```

Attributes imputation

```
# Impute house characteristics data first with k
# nearest neighbor
HC_01 <- trainset %>% select(starts_with("HC01_"))
HC_01_Imputed <- knnImputation(HC_01, k = 3)
names(HC_01_Imputed) <- paste0("Impute_", colnames(HC_01_Imputed))

# Imput weather factor data with mode and numeric
# data with state average
for (i in 1:ncol(trainset)) {
    if ((class(trainset[, i]) == "character" | class(trainset[,
        i]) == "factor") & substr(colnames(trainset)[i],
        1, 5) != "flag_") {
        # Impute Weather type data and factors
        trainset[, i][trainset[, i] == "P"] <- NA
        trainset[, paste0("F_", colnames(trainset)[i])] <- impute(as.factor(trainset[,
            i]), mode)
    } else if ((substr(colnames(trainset)[i], 1, 5) ==
        "Weat_") & (class(trainset[, i]) == "numeric" |
        class(trainset[, i]) == "integer") & (substr(colnames(trainset)[i],
        1, 10) != "state_avg_")) {
        # Impute Weather numeric data
        trainset[, i] <- as.numeric(trainset[, i])
        trainset[, paste0("Impute_", colnames(trainset)[i])] = coalesce(trainset[,
```

```
            i], eval(parse(text = paste0("trainset$state_avg_",
            colnames(trainset)[i])))) #Impute state average
    }
}

trainset <- cbind(trainset, HC_01_Imputed)
trainset <- as.data.frame(trainset %>% select(Target,
    starts_with("flag_"), starts_with("Impute_"),
    starts_with("F_"), ELEVATION, ALAND, AWATER))
```

####Create climate regions

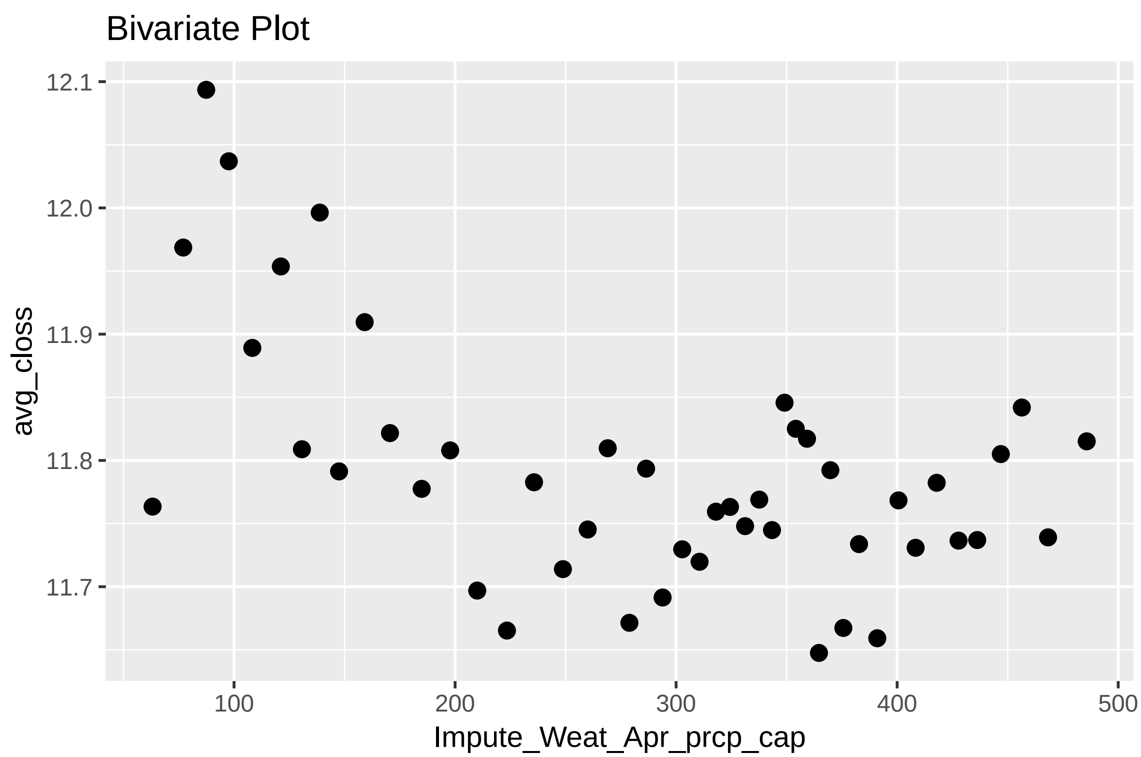
Climate is highly impacted by the geograpic feature of the location. However, how granular our data needs to be regarding to geographic feature is debatable. In this project, we include both State and Climate regions information in our model.

```
trainset$Regions <- as.factor(ifelse(trainset$F_STATE.x %in%
    c("NT", "WY", "CO", "ND", "SD", "NE", "KS"),
    "northwest", ifelse(trainset$F_STATE.x %in% c("MN",
        "IA", "IL", "WI", "MO", "OH", "IN", "MI"),
        "midwest", ifelse(trainset$F_STATE.x %in%
            c("AZ", "NM", "TX", "OK"), "southwest",
            ifelse(trainset$F_STATE.x %in% c("WA",
                "OR", "NV", "ID", "UT", "CA"), "westcoast",
                ifelse(trainset$F_STATE.x %in% c("AR",
                  "LA", "MS", "TN", "AL", "GA", "FL",
                  "SC", "NC", "VA", "WV", "KY"),
                  "south", ("northeast")))))))

# trainset<-trainset%>%select(-F_STATE.x,)
```

###3. Exploratory Analysis

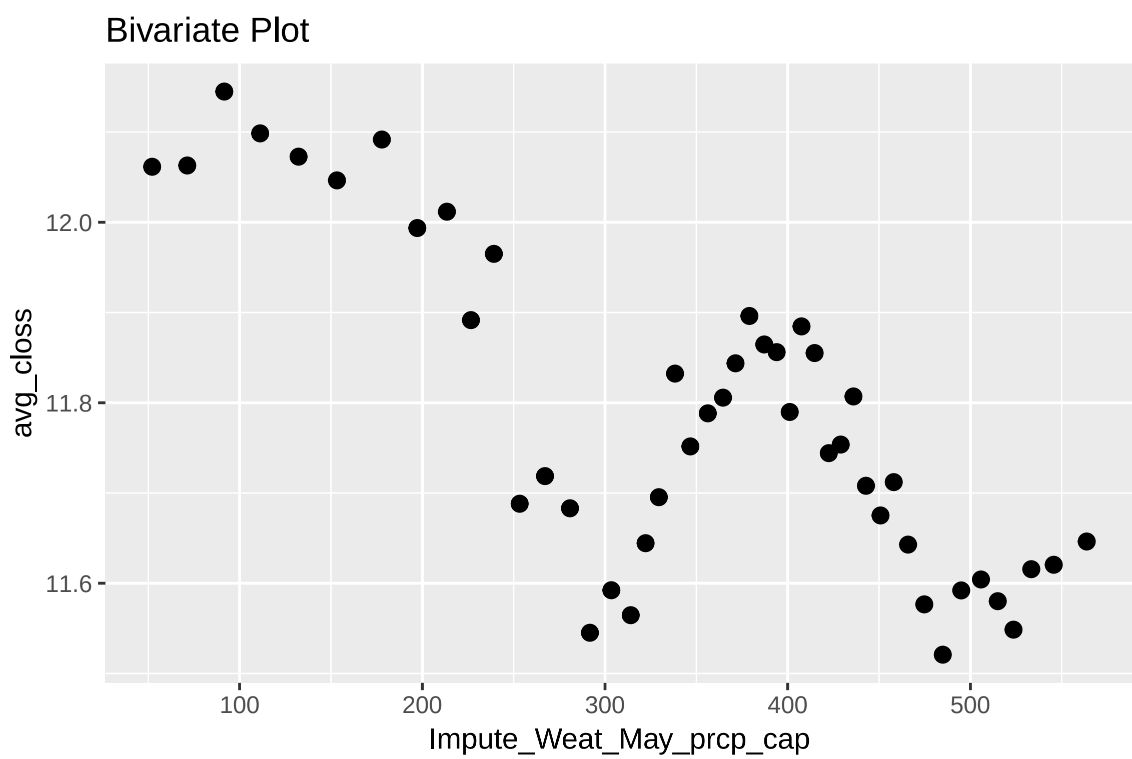
After feature engineering, now we can look at some of the important weather attributes, and examine their relationships with home price.

```
# spring months
bivar.plot(trainset, "Impute_Weat_Apr_prcp_cap",
    "Target", n.rank = 50)
```

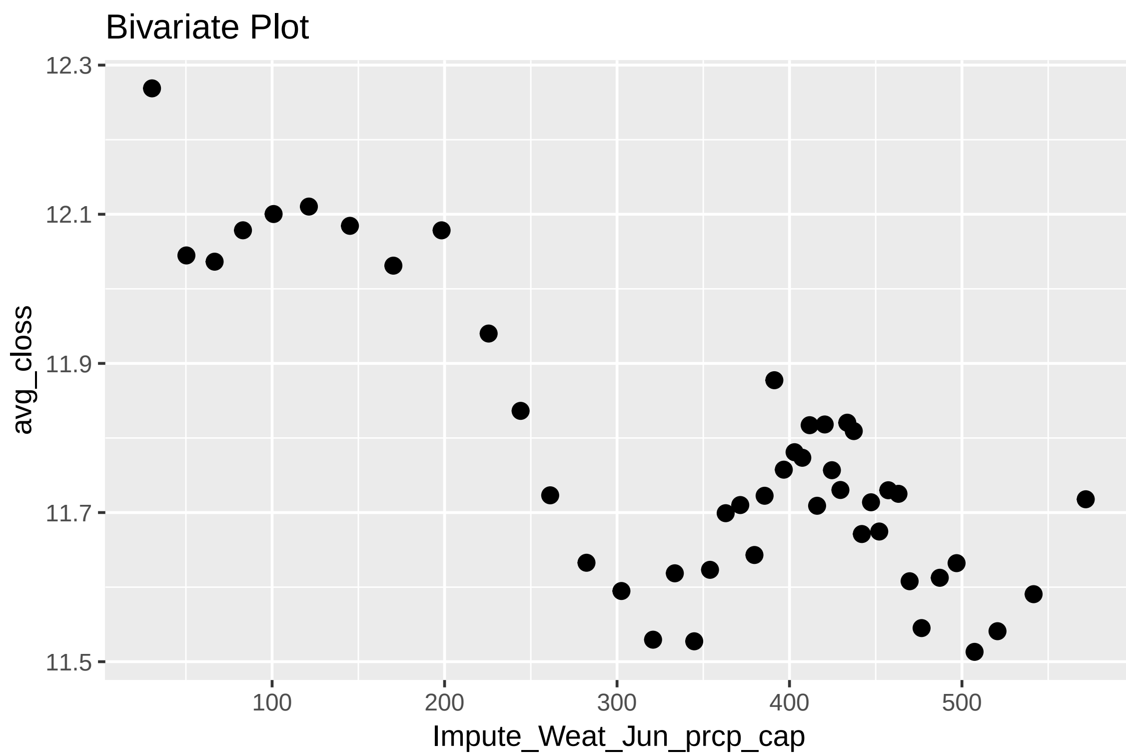


precipitation

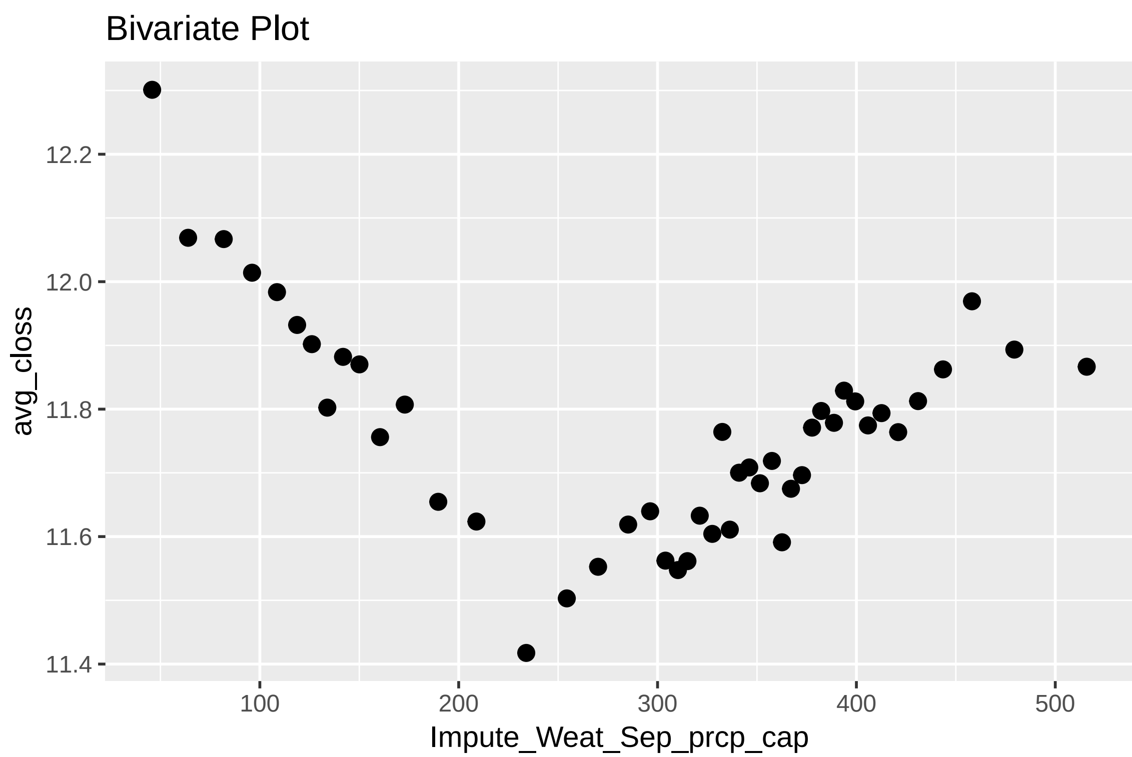
```r
bivar.plot(trainset, "Impute_Weat_May_prcp_cap",
    "Target", n.rank = 50)
```



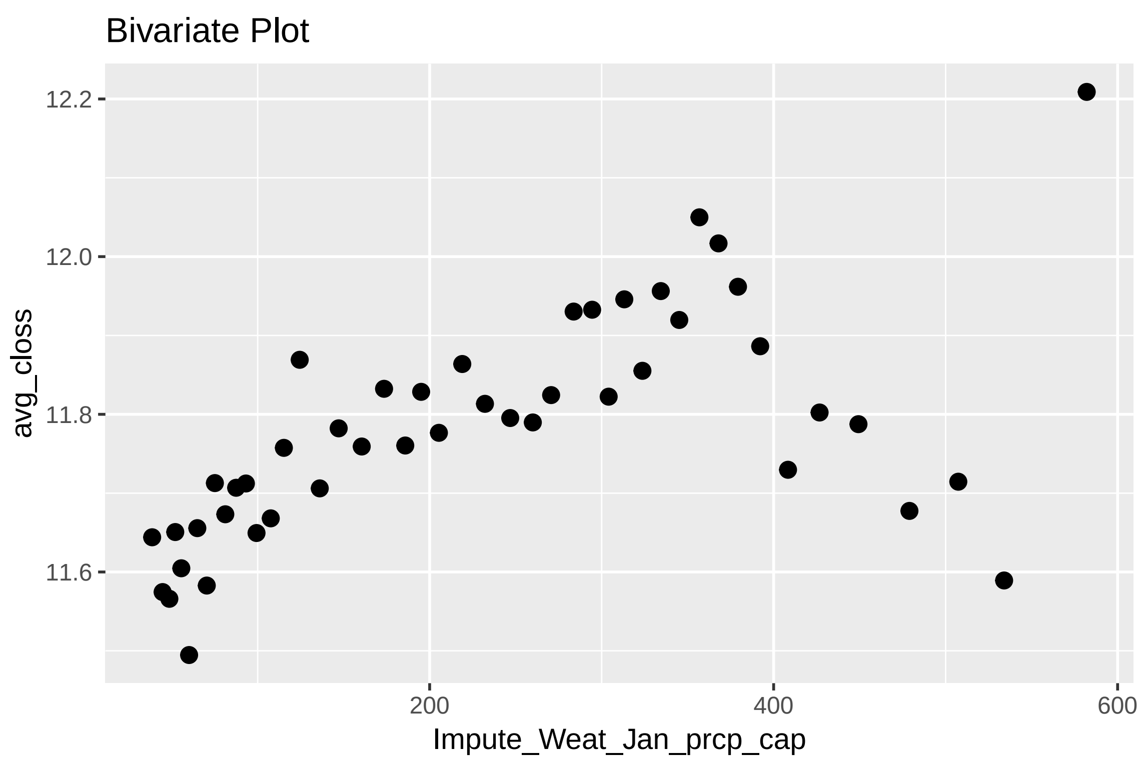
```r
# summer months
bivar.plot(trainset, "Impute_Weat_Jun_prcp_cap",
    "Target", n.rank = 50)
```



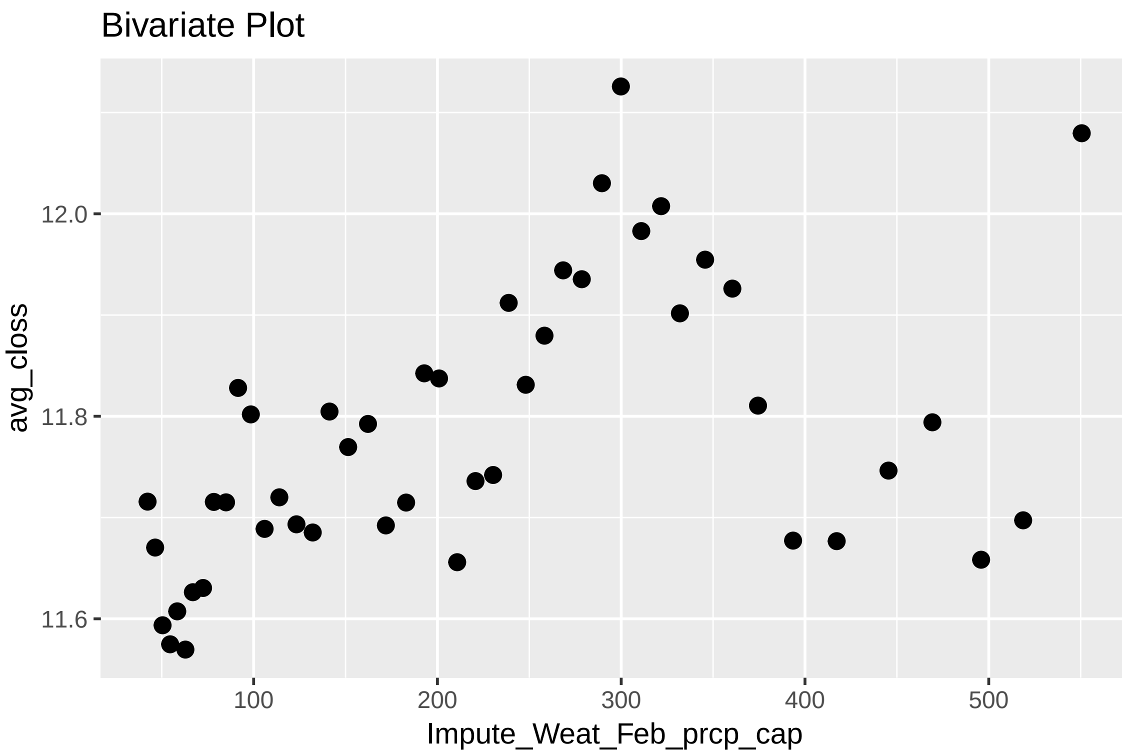

```r
# fall months
bivar.plot(trainset, "Impute_Weat_Sep_prcp_cap",
    "Target", n.rank = 50)
```




```r
# winter months
bivar.plot(trainset, "Impute_Weat_Jan_prcp_cap",
    "Target", n.rank = 50)
```



```r
bivar.plot(trainset, "Impute_Weat_Feb_prcp_cap",
    "Target", n.rank = 50)
```



From the above charts, we see that higher precipitation areas has lower home price in summer months, however higher home price in winter months. One interesting observation about this attribute is that in September, precipitation has a nonlinear relationship with home price.

To deal with this relationship, we can create two variables to represent different trends in different range. Here we use WOE to find the cutoff point.

WOE formula:

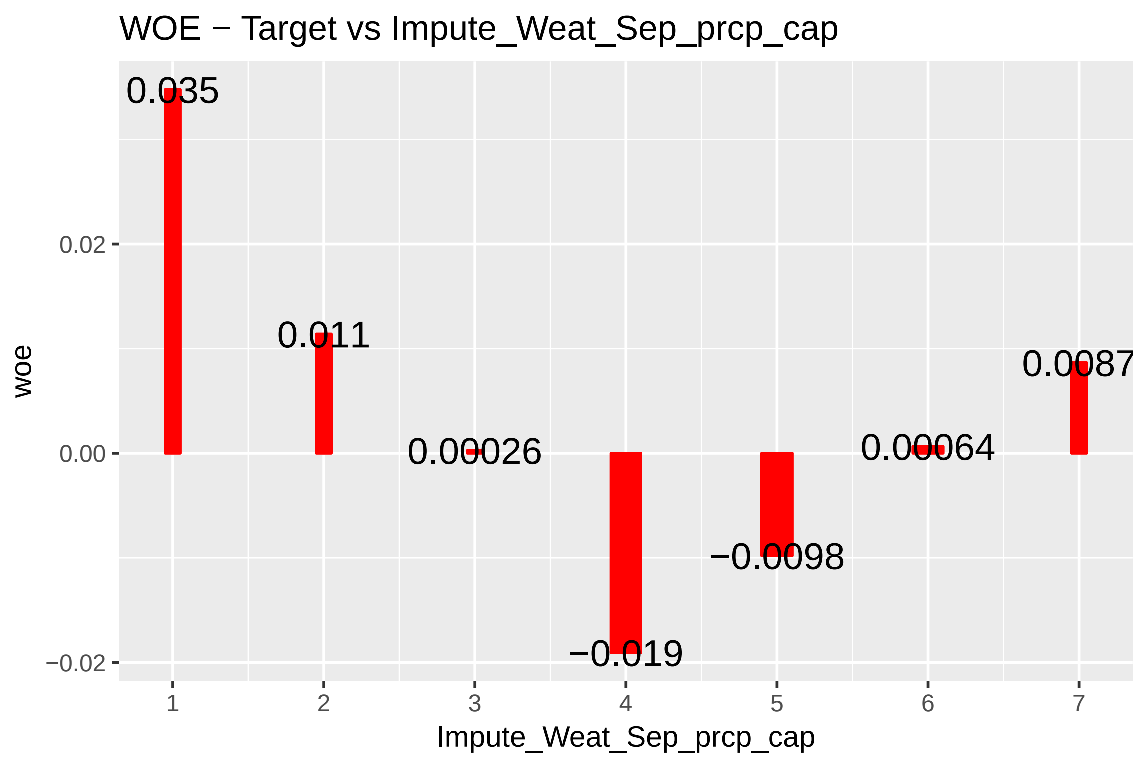
WOE=ln($\frac{p(non-event)}{p(event)}$)

IV=$\sum_{i=1}^{n}(DistributionGood - DistributionBad)$*WOE

```r
WOE_numeric_split(x = "Impute_Weat_Sep_prcp_cap",
    y1 = "Target", data = trainset, group = 10)
```

```
## $WOE
## # A tibble: 7 x 14
##     grp n_obs mean_x sum_y1 sum_y0 sum_w mean_y1 mean_y0 LowerBound UpperBound
##   <int> <int>  <dbl>  <dbl>  <dbl> <dbl>   <dbl>   <dbl>      <dbl>      <dbl>
## 1     1   470   56.2  5737.  5541.   470    12.2    11.8         45         88
## 2     2   469  116.   5593.  5530.   469    11.9    11.8         89        137
## 3     3   473  162.   5578.  5577.   473    11.8    11.8        138        198
## 4     4   928  279.  10735. 10941.   928    11.6    11.8        199        323
## 5     5   955  348.  11150. 11260.   955    11.7    11.8        324        374
## 6     6   943  405.  11125. 11118.   943    11.8    11.8        375        449
## 7     7   472  497.   5613.  5565.   472    11.9    11.8        450        518
```

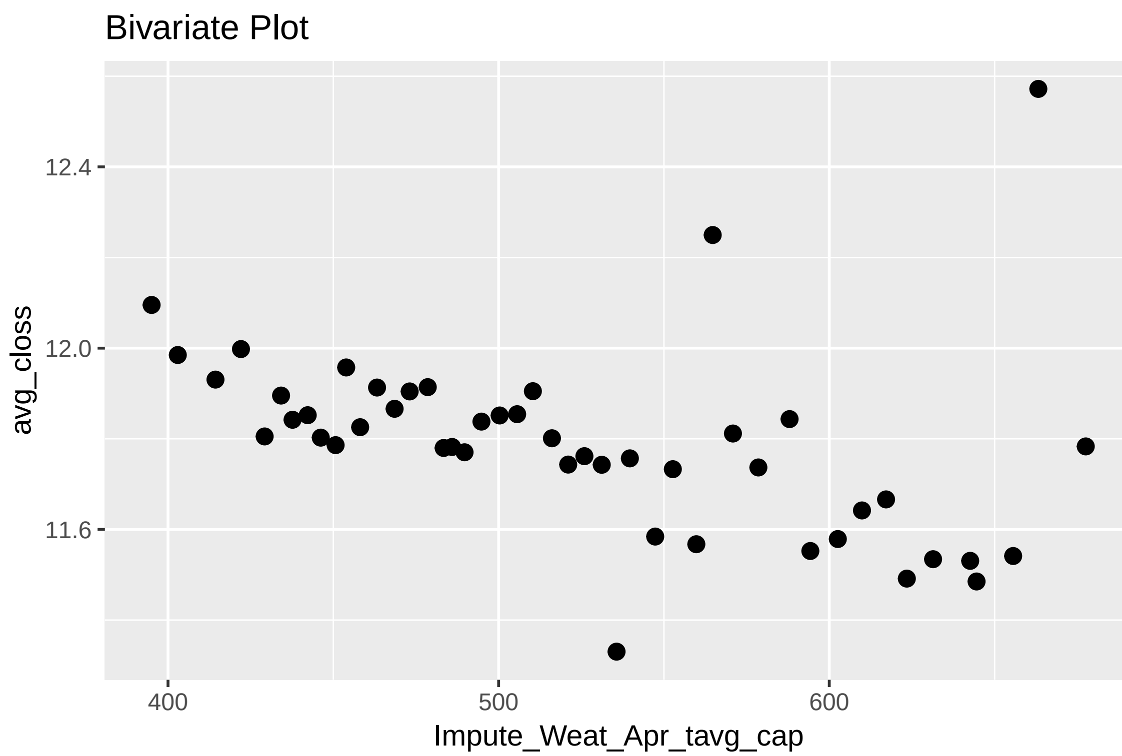
```
## # ... with 4 more variables: woe <dbl>, ks <dbl>, info <dbl>, plot.width <dbl>
##
## $Stat
##         MAX_KS     Info.Value Trend.Estimate Trend.Pr(>|t|)
##   4.698438e-01   2.337355e-04  -5.022986e-05   3.243015e-01
##
## $WOE_Code
## [1] "trainset$w_Impute_Weat_Sep_prcp_cap = trainset$Impute_Weat_Sep_prcp_cap"
## [2] "g_Impute_Weat_Sep_prcp_cap = c(45, 89, 138, 199, 324, 375, 450, 518)"
## [3] "trainset$w_Impute_Weat_Sep_prcp_cap[findInterval(trainset$Impute_Weat_Sep_prcp_cap, g_Impute_Wea
## [4] "trainset$w_Impute_Weat_Sep_prcp_cap[findInterval(trainset$Impute_Weat_Sep_prcp_cap, g_Impute_Wea
## [5] "trainset$w_Impute_Weat_Sep_prcp_cap[findInterval(trainset$Impute_Weat_Sep_prcp_cap, g_Impute_Wea
## [6] "trainset$w_Impute_Weat_Sep_prcp_cap[findInterval(trainset$Impute_Weat_Sep_prcp_cap, g_Impute_Wea
## [7] "trainset$w_Impute_Weat_Sep_prcp_cap[findInterval(trainset$Impute_Weat_Sep_prcp_cap, g_Impute_Wea
## [8] "trainset$w_Impute_Weat_Sep_prcp_cap[findInterval(trainset$Impute_Weat_Sep_prcp_cap, g_Impute_Wea
## [9] "trainset$w_Impute_Weat_Sep_prcp_cap[findInterval(trainset$Impute_Weat_Sep_prcp_cap, g_Impute_Wea
##
## $Plot
```



Seen from the above WOE graph, the trend changed from negative to positive at bin 4, and we can find the mean of the independent variable for bin 4 from the table, which is -0.08290098. So we use this value as our cutoff value.

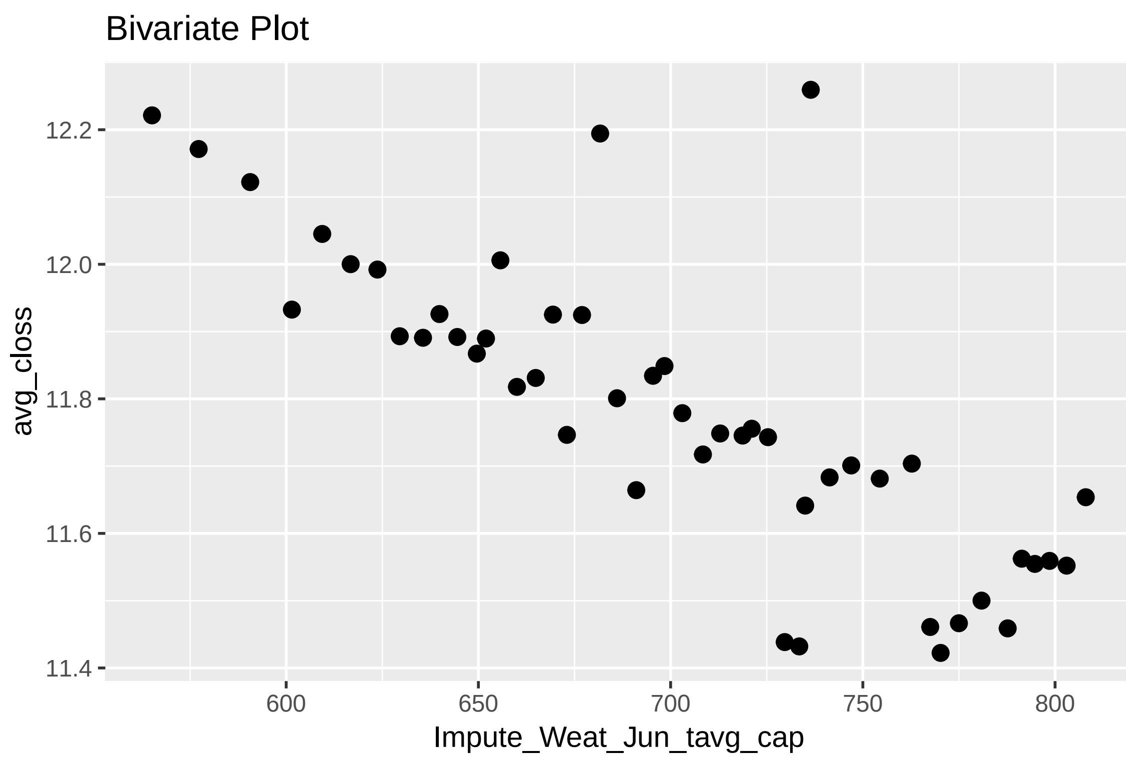
```
trainset$Impute_Weat_Sep_prcp_g1_cap <- pmax(-0.08290098 -
    trainset$Impute_Weat_Sep_prcp_cap, 0)
trainset$Impute_Weat_Sep_prcp_g2_cap <- pmax(trainset$Impute_Weat_Sep_prcp_cap +
    0.08290098, 0)

trainset <- trainset %>% select(-Impute_Weat_Sep_prcp_cap)
```

```r
# spring months
bivar.plot(trainset, "Impute_Weat_Apr_tavg_cap",
    "Target", n.rank = 50)
```

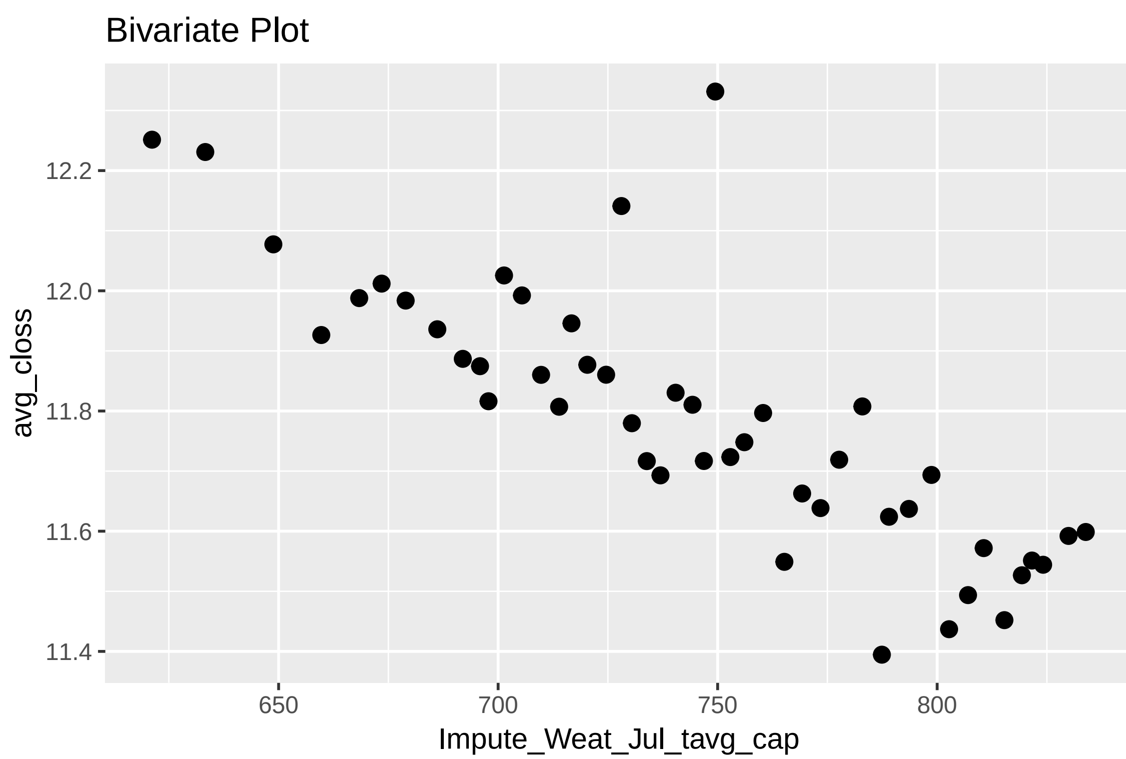


average tempurature

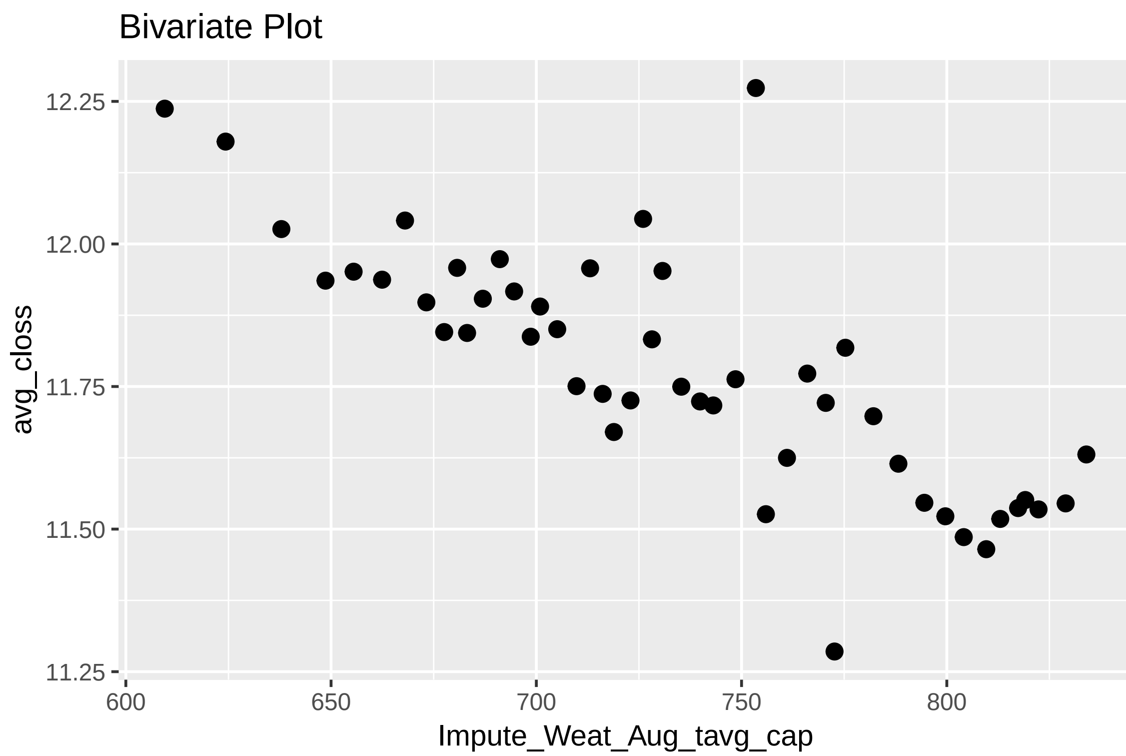
```r
# summer months
bivar.plot(trainset, "Impute_Weat_Jun_tavg_cap",
    "Target", n.rank = 50)
```



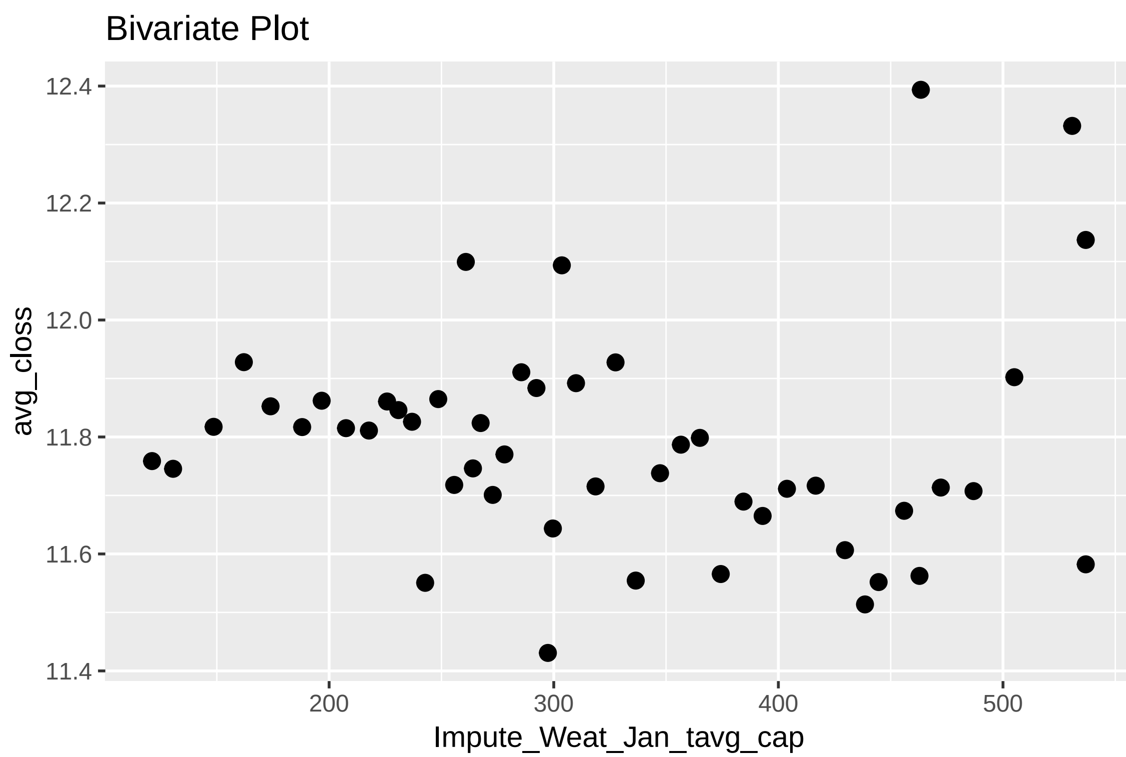

```r
bivar.plot(trainset, "Impute_Weat_Jul_tavg_cap",
    "Target", n.rank = 50)
```



```r
bivar.plot(trainset, "Impute_Weat_Aug_tavg_cap",
    "Target", n.rank = 50)
```

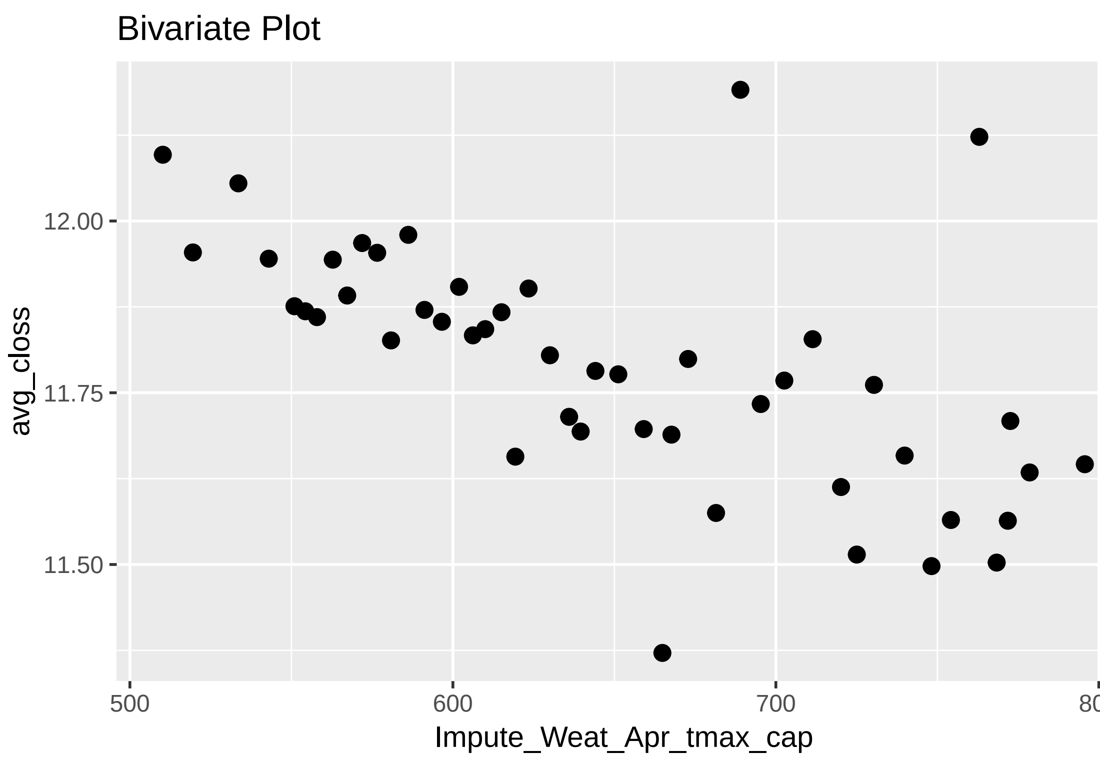



```r
# winter months
bivar.plot(trainset, "Impute_Weat_Jan_tavg_cap",
    "Target", n.rank = 50)
```



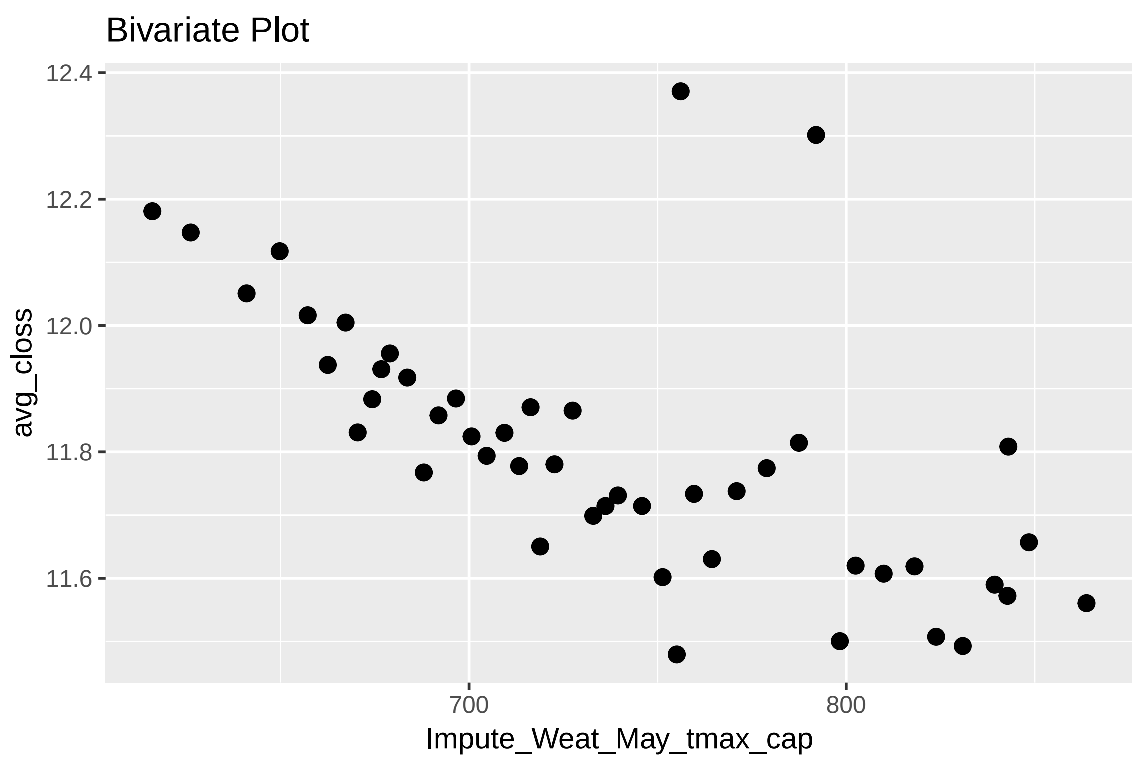
From the above charts, we see that home price is higher when summer months tempurature is lower, winter months tempurature does not have as big of an impact to the home price as summer tempurater.

```
# spring months
bivar.plot(trainset, "Impute_Weat_Apr_tmax_cap",
    "Target", n.rank = 50)
```

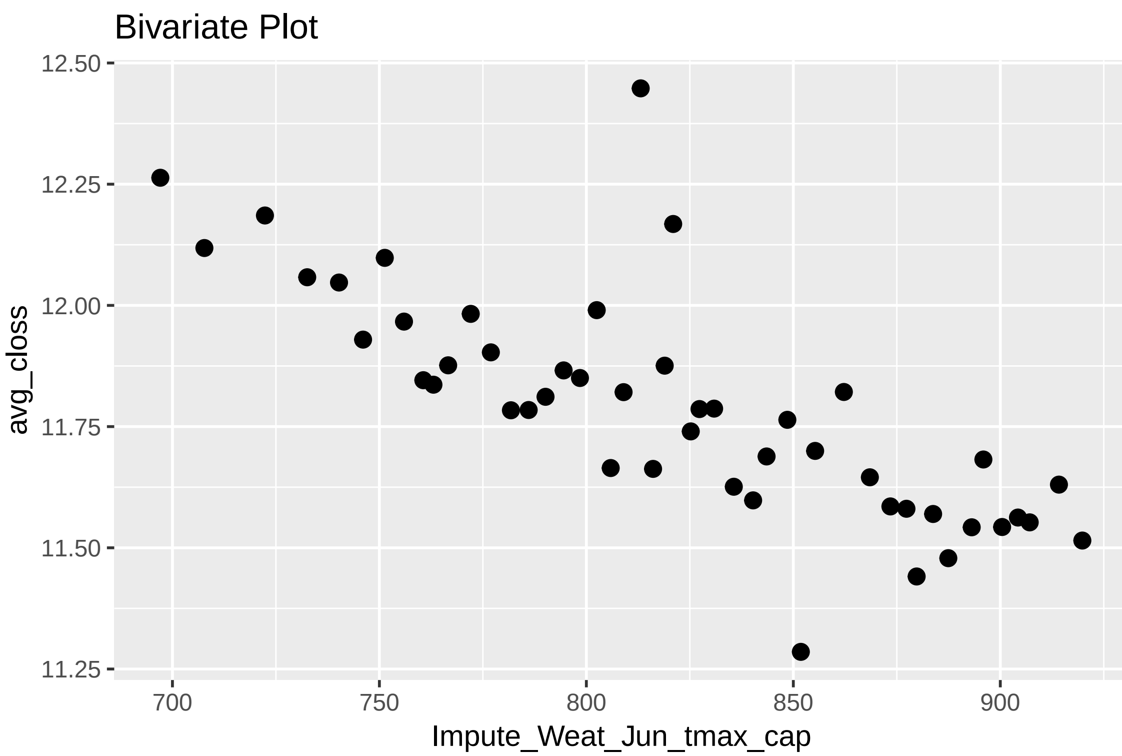


maximum tempurature

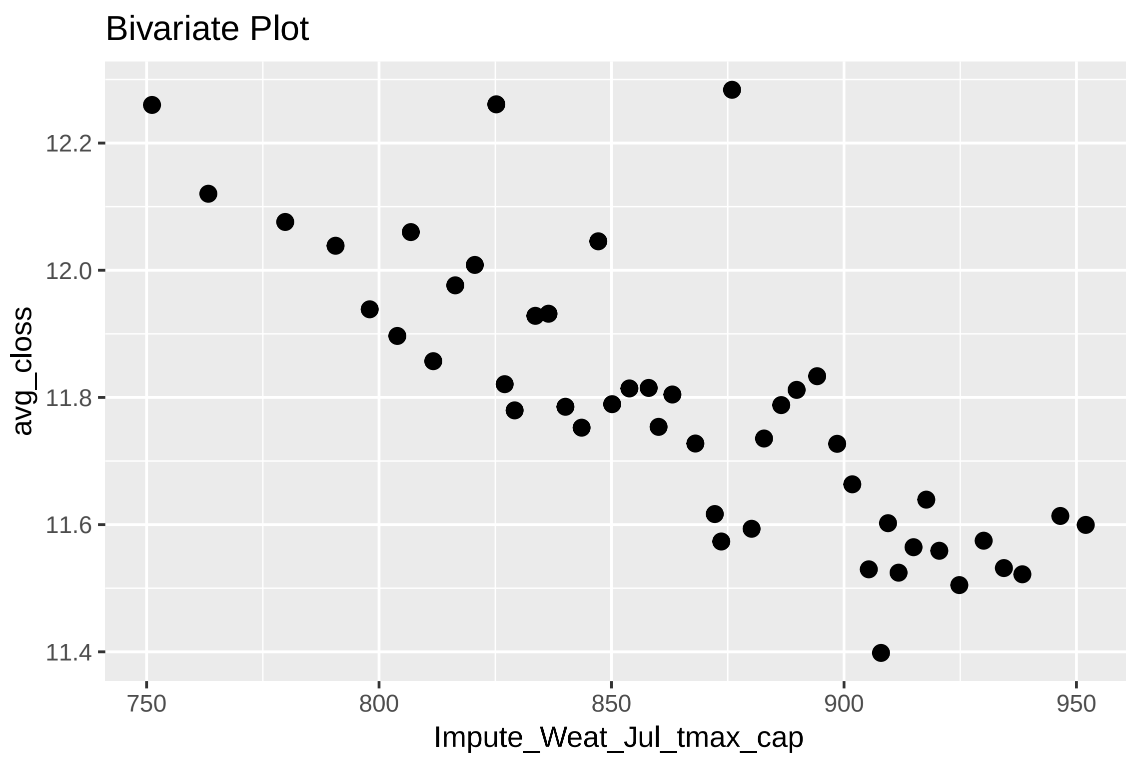
```
bivar.plot(trainset, "Impute_Weat_May_tmax_cap",
    "Target", n.rank = 50)
```



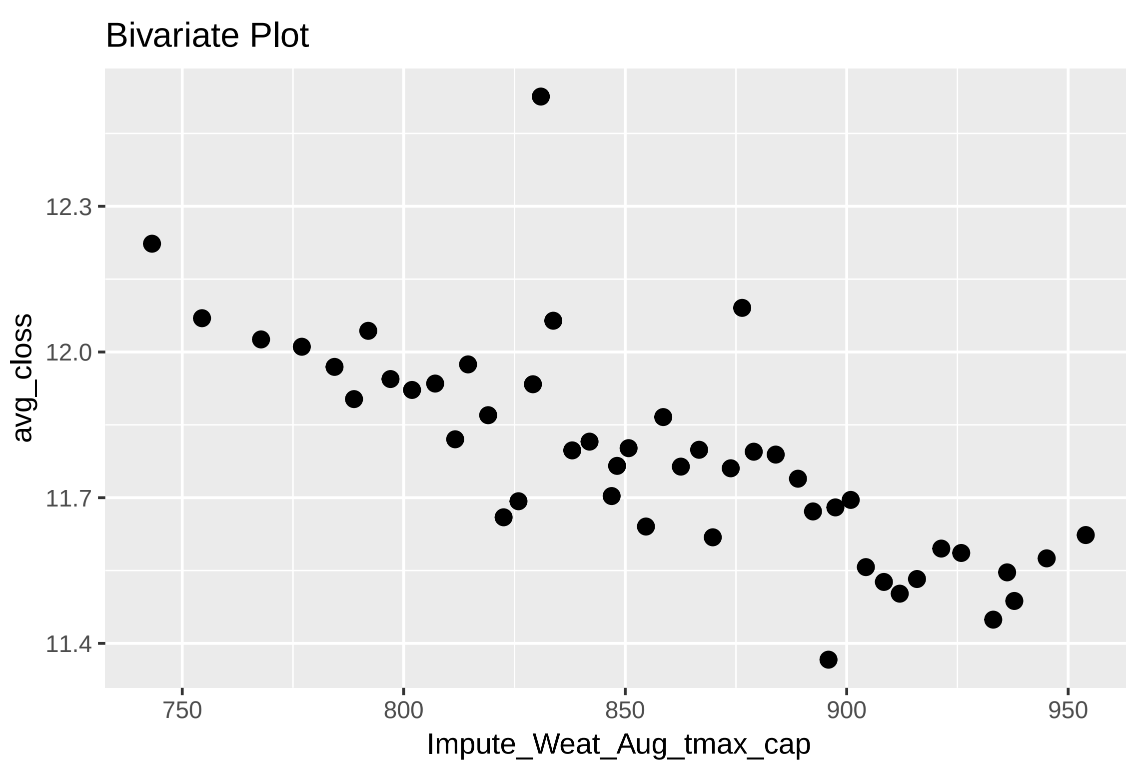
```r
# summer months
bivar.plot(trainset, "Impute_Weat_Jun_tmax_cap",
    "Target", n.rank = 50)
```



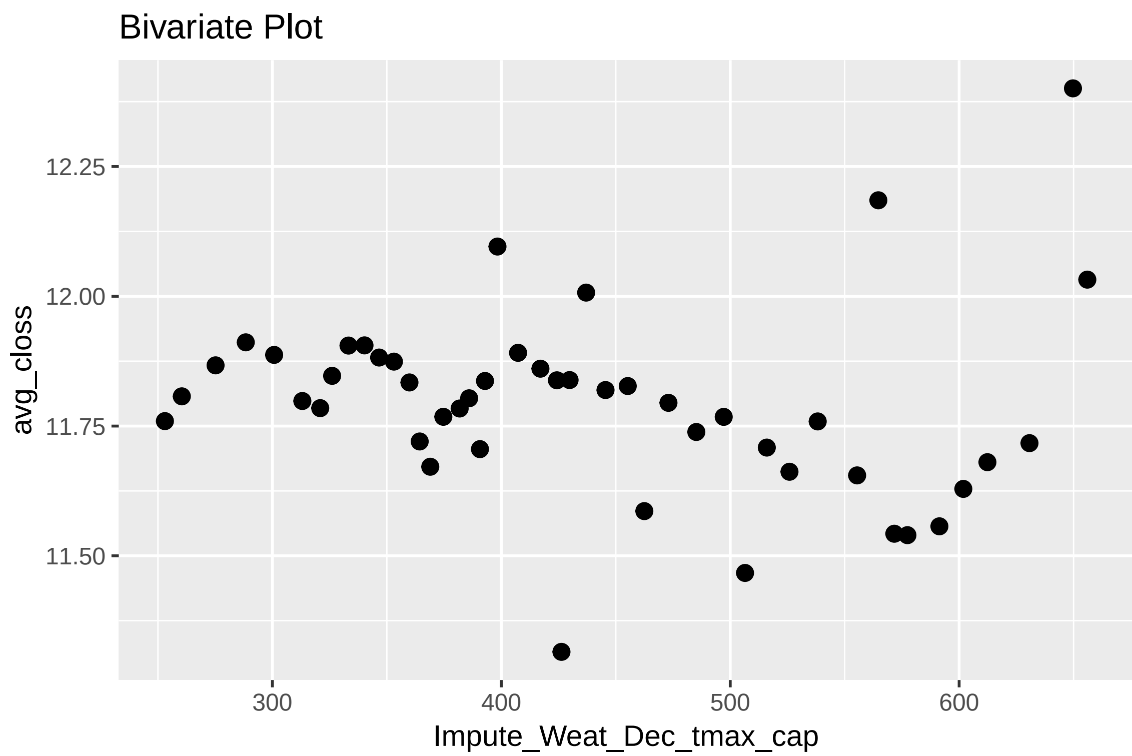

```r
bivar.plot(trainset, "Impute_Weat_Jul_tmax_cap",
    "Target", n.rank = 50)
```



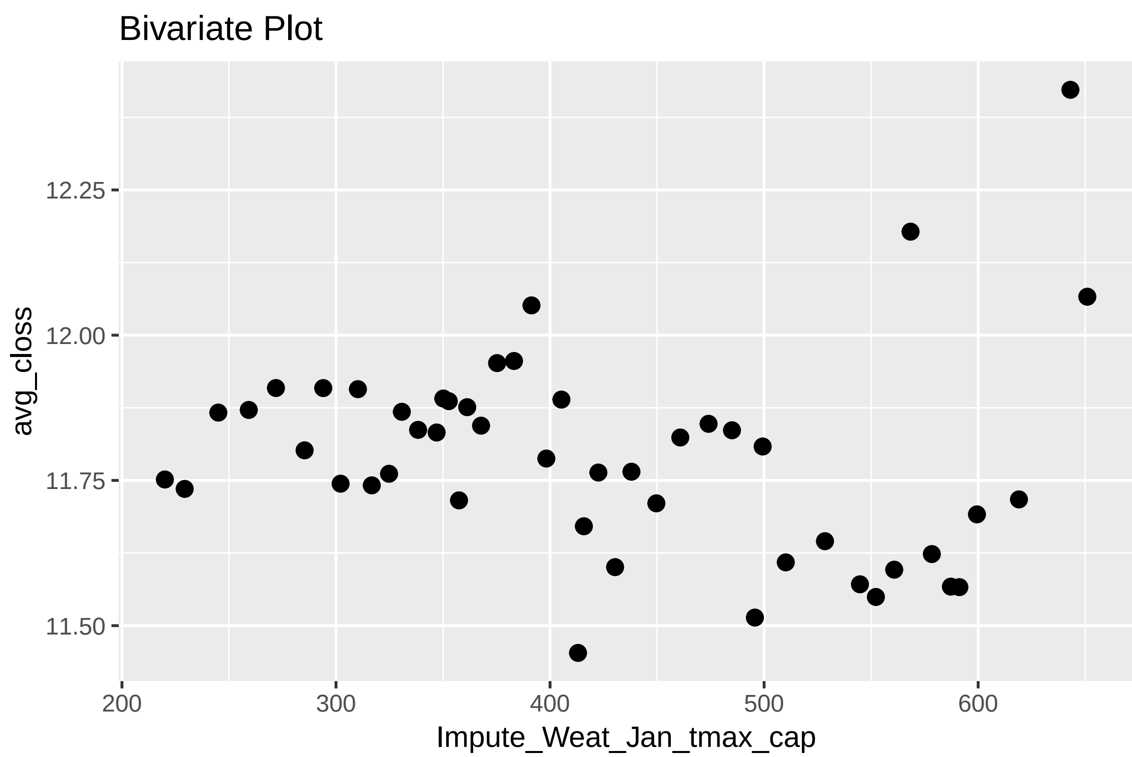

```r
bivar.plot(trainset, "Impute_Weat_Aug_tmax_cap",
    "Target", n.rank = 50)
```



```r
# winter months
bivar.plot(trainset, "Impute_Weat_Dec_tmax_cap",
    "Target", n.rank = 50)
```




```r
bivar.plot(trainset, "Impute_Weat_Jan_tmax_cap",
    "Target", n.rank = 50)
```



Bivariate Plot
avg_closs
12.25
12.00
11.75
11.50
200
300
400
500
600
Impute_Weat_Jan_tmax_cap