Two sigma rental listing inquries xgboost

Wei Xu

March 9, 2017

- **Description feature: how to explore? Generate one, two, three grams exclusively, also not appear in features.**
- **These two features: display_address and street_address have replicated informations. More detailed in street_address.**
- **Is it fine to use concatenate features and photos in both t1 and s1?**

Load the training dataset

```
# load necessary packages
packages <- c("jsonlite", "dplyr", "data.table", "purrr")
purrr::walk(packages, library, character.only = TRUE, warn.conflicts = FALSE)
# load train
t1 <- fromJSON("../input/train.json")
# extract posted features and perform nlp e.g. tolower and stemword
t1_feats <- data.table(listing_id=rep(unlist(t1$listing_id), lapply(t1$features, length)),
                       features=unlist(t1$features))
t1_feats <- t1_feats[, features:=gsub("[0-9]|[[:punct:]]", " ", trimws(tolower(features)))]
t1_feats <- t1_feats[, features:=sapply(strsplit(features, "[[:space:]]+"),
                                        function(x) {
                                              x <- char_wordstem(x); ib <- x == ""
                                              # use collapse instead of sep in paste
                                              paste(c("feature",x[!ib]), collapse = "_")}
                                        )]
# extract individual photo urls
t1_photos <- data.table(listing_id=rep(unlist(t1$listing_id), lapply(t1$photos, length)),
                        features=unlist(t1$photos))
# unlist all vars except photo and features and convert data table
vars <- setdiff(names(t1), c("photos", "features")) # set operation: difference
t1 <- map_at(t1, vars, unlist) %>% as.data.table(.)
t1 <- t1[ , filter:=0]
data.frame(sapply(t1, class))
```

```
##                 sapply.t1..class.
## bathrooms                 numeric
## bedrooms                  integer
## building_id             character
## created                 character
## description             character
## display_address         character
## features                     list
## latitude                  numeric
## listing_id                integer
## longitude                 numeric
## manager_id              character
## photos                       list
## price                     integer
```

```
## street_address          character
## interest_level          character
## filter                    numeric
```

```
##> head(t1_feats, 10)
data.frame(t1 = dim(t1), t1_feats = dim(t1_feats), t1_photos = dim(t1_photos),
           row.names = c("row", "col"))
```

```
##        t1 t1_feats t1_photos
## row 49352   267906    276714
## col    16        2         2
```

Create 5-fold CV

```
# create 5 fold CV
set.seed(321)
cvFoldsList <- createFolds(t1$interest_level, k=5, list=TRUE, returnTrain=FALSE)
```

```
# Convert classes to integers for xgboost
class <- data.table(interest_level=c("low", "medium", "high"), class=c(0,1,2))
t1 <- merge(t1, class, by="interest_level", all.x=TRUE, sort=F)
```

Load the test dataset

```
s1 <- fromJSON("../input/test.json")
s1_feats <- data.table(listing_id=rep(unlist(s1$listing_id), lapply(s1$features, length)),
                       features=unlist(s1$features))
s1_feats <- s1_feats[, features:=gsub("[0-9]|[[:punct:]]", " ", trimws(tolower(features)))]
s1_feats <- s1_feats[, features:=sapply(strsplit(features, "[[:space:]]+"),
                                        function(x) {
                                              x <- char_wordstem(x); ib <- x == ""
                                              paste(c("feature", x[!ib]), collapse = "_")}
                                        )]
s1_photos <- data.table(listing_id=rep(unlist(s1$listing_id), lapply(s1$photos, length)),
                        features=unlist(s1$photos))
vars <- setdiff(names(s1), c("photos", "features"))
s1 <- map_at(s1, vars, unlist) %>% as.data.table(.)
s1 <- s1[,":="(interest_level="pending", class=-1, filter=2)]
data.frame(s1 = dim(s1), s1_feats = dim(s1_feats), s1_photos = dim(s1_photos),
           row.names = c("row", "col"))
```

```
##        s1 s1_feats s1_photos
## row 74659   404920    419598
## col    17        2         2
```

Combine train and test data

```
ts1 <- rbind(t1, s1)
rm(t1, s1)
ts1_feats <- rbind(t1_feats, s1_feats)
rm(t1_feats, s1_feats)
```

```
ts1_photos <- rbind(t1_photos, s1_photos)
rm(t1_photos, s1_photos)
data.frame(ts1 = dim(ts1), ts1_feats = dim(ts1_feats), ts1_photos = dim(ts1_photos),
           row.names = c("row", "col"))
```

```
##        ts1 ts1_feats ts1_photos
## row 124011    672826     696312
## col     17         2          2
```

Load the time_stamp feature and merge with the train and test dataset

```
time_stamp <- fread("../input/listing_image_time.csv")
ts1 <- merge(ts1, time_stamp, by.x = "listing_id", by.y = "Listing_Id", all.x = TRUE)
```

Outliers of geographical data

This piece of code is used to justify the outliers in the geographical data, i.e. `longitude` and `latitude`. A new feature `distance_city`, which quantifies the distance to the city center is also generated.

```
outliers_addrs <- ts1[longitude == 0 | latitude == 0, ]
# addresses are supposed to be in nyc
outliers_ny <- paste(outliers_addrs$street_address, ", new york")
# search for geological location from google
coords <- sapply(outliers_ny, function(x) geocode(x, source = "google")) %>%
      t %>% data.frame
# assign data from google
ts1[longitude == 0 | latitude == 0, ]$longitude <- coords$lon
ts1[longitude == 0 | latitude == 0, ]$latitude  <- coords$lat
# add distance to city center feature
ny_center <- geocode("new york", source = "google")
ny_lat <- ny_center[1,2]; ny_lon <- ny_center[1,1]
# Add Euclidean Distance to City Center
ts1$distance_city <- mapply(
      function(lon, lat) sqrt((lon - ny_lon)^2  + (lat - ny_lat)^2),
      ts1$longitude, ts1$latitude)
```

Some feature engineering

```
ts1 <- ts1[, ":="(created=as.POSIXct(created),
                  dummy="A",
                  low=as.integer(interest_level=="low"),
                  medium=as.integer(interest_level=="medium"),
                  high=as.integer(interest_level=="high"),
                  display_address=trimws(tolower(display_address)),
                  street_address=trimws(tolower(street_address)))]
ts1 <- ts1[, ":="(created_month=month(created),
                  created_day=day(created),
                  weekofday=as.integer(as.factor(weekdays(created))),
                  created_hour=hour(created))]
ts1 <- ts1[, ":="(pred0_low=sum(interest_level=="low")/sum(filter==0),
```

```
                  pred0_medium=sum(interest_level=="medium")/sum(filter==0),
                  pred0_high=sum(interest_level=="high")/sum(filter==0))]
data.frame(sapply(ts1, function(x) class(x)[1]))
```

```
##                 sapply.ts1..function.x..class.x..1..
## listing_id                                   integer
## interest_level                             character
## bathrooms                                    numeric
## bedrooms                                     integer
## building_id                                character
## created                                      POSIXct
## description                                character
## display_address                            character
## features                                        list
## latitude                                     numeric
## longitude                                    numeric
## manager_id                                 character
## photos                                          list
## price                                        integer
## street_address                             character
## filter                                       numeric
## class                                        numeric
## time_stamp                                   integer
## distance_city                                numeric
## dummy                                      character
## low                                          integer
## medium                                       integer
## high                                         integer
## created_month                                integer
## created_day                                  integer
## weekofday                                    integer
## created_hour                                 integer
## pred0_low                                    numeric
## pred0_medium                                 numeric
## pred0_high                                   numeric
```

Merge feature column

```
# merge Feature column
feats_summ <- ts1_feats[ , .N , by=features]
# tabulate the data by rows (listing_id) and columns (features)
ts1_feats_cast <- dcast.data.table(ts1_feats[!features %in% feats_summ[N<500, features]],
                                   listing_id ~ features, value.var = "features",
                                   fun.aggregate = function(x) as.integer(length(x) > 0))
ts1 <- merge(ts1, ts1_feats_cast, by="listing_id", all.x=TRUE, sort=FALSE)
rm(ts1_feats, feats_summ, ts1_feats_cast)
```

Merge photo counts

```
ts1_photos_summ <- ts1_photos[ , .(photo_count=.N), by=listing_id]
ts1 <- merge(ts1, ts1_photos_summ, by="listing_id", all.x=TRUE, sort=FALSE)
rm(ts1_photos, ts1_photos_summ)
```

Engineer description feature

```
# load mycorpus
library(tm)
mycorpus <- VCorpus(VectorSource(ts1$description))
# clean mycorpus
toString <- content_transformer(function(x, from, to) gsub(from, to, x))
mycorpus <- tm_map(mycorpus, content_transformer(tolower))
mycorpus <- tm_map(mycorpus, toString, "[!.]([^!.]*)kagglemanager@renthop.com([^!.]*)", "")
mycorpus <- tm_map(mycorpus, toString, "[!.]([^!.]*)website(.*)redacted([^!.]*)", "")
mycorpus <- tm_map(mycorpus, toString, "<.*>", " ")
mycorpus <- tm_map(mycorpus, toString, "[[:punct:]]", " ")
mycorpus <- tm_map(mycorpus, removePunctuation)
mycorpus <- tm_map(mycorpus, removeNumbers)
mycorpus <- tm_map(mycorpus, removeWords, stopwords("english"))
mycorpus <- tm_map(mycorpus, stemDocument)
mycorpus <- tm_map(mycorpus, stripWhitespace)
mycorpus <- tm_map(mycorpus, toString,
                   "( ([a-z]){1,3})+ |^([a-z]){1,3} | ([a-z]){1,3}$", " ")
mycorpus <- tm_map(mycorpus, stripWhitespace)
# bigramtokenizer from rweka
library(RWeka)
BigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
# document-term matrix
bgram_dtm <- DocumentTermMatrix(mycorpus, control = list(tokenize = BigramTokenizer))
# remove sparse terms
bgram_dtm_sub <- removeSparseTerms(bgram_dtm, 0.97)
ts1_desc <- data.table(as.matrix(bgram_dtm_sub))
colnames(ts1_desc) <- gsub(" ", "_", paste0("descrpt_", names(ts1_desc)))
ts1 <- cbind(ts1, ts1_desc)
rm(mycorpus, bgram_dtm, bgram_dtm_sub)
```

Sentiment analysis from description feature

```
library(syuzhet)
sentiment <- get_nrc_sentiment(ts1$description)
ts1 <- cbind(ts1, sentiment)
```

Grouping entry-one observation

```
build_count <- ts1[,.(.N), by=building_id]
manag_count <- ts1[,.(.N), by=manager_id]
addre_count <- ts1[,.(.N), by=display_address]
set(ts1, i=which(ts1[["building_id"]] %in% build_count[N==1, building_id]),
```

```
    j="building_id", value="other")
set(ts1, i=which(ts1[["manager_id"]] %in% manag_count[N==1, manager_id]),
    j="manager_id", value="other")
set(ts1, i=which(ts1[["display_address"]] %in% addre_count[N==1, display_address]),
    j="display_address", value="other")
```

Mean target encoding high cardinality variables

```
# custom function for categorical encoding using target statistics
catNWayAvgCV <- function(data, varList, y, pred0, filter, k, f, g=1, lambda=NULL, r_k, cv=NULL){
      # It is probably best to sort your dataset first by filter and then by ID (or index)
      n <- length(varList)
      varNames <- paste0("v",seq(n))
      ind <- unlist(cv, use.names=FALSE)
      oof <- NULL
      if (length(cv) > 0){
            for (i in 1:length(cv)){
                  sub1 <- data.table(v1=data[,varList,with=FALSE], y=data[,y,with=FALSE],
                                     pred0=data[,pred0,with=FALSE], filt=filter)
                  sub1 <- sub1[sub1$filt==TRUE,]
                  sub1[,filt:=NULL]
                  colnames(sub1) <- c(varNames,"y","pred0")
                  sub2 <- sub1[cv[[i]],]
                  sub1 <- sub1[-cv[[i]],]
                  sum1 <- sub1[,list(sumy=sum(y), avgY=mean(y), cnt=length(y)), by=varNames]
                  tmp1 <- merge(sub2, sum1, by = varNames, all.x=TRUE, sort=FALSE)
                  set(tmp1, i=which(is.na(tmp1[,cnt])), j="cnt", value=0)
                  set(tmp1, i=which(is.na(tmp1[,sumy])), j="sumy", value=0)
                  if(!is.null(lambda)) tmp1[beta:=lambda]
                  else tmp1[,beta:= 1/(g+exp((tmp1[,cnt] - k)/f))]
                  tmp1[,adj_avg:=((1-beta)*avgY+beta*pred0)]
                  set(tmp1, i=which(is.na(tmp1[["avgY"]])), j="avgY",
                      value=tmp1[is.na(tmp1[["avgY"]]), pred0])
                  set(tmp1, i=which(is.na(tmp1[["adj_avg"]])), j="adj_avg",
                      value=tmp1[is.na(tmp1[["adj_avg"]]), pred0])
                  set(tmp1, i=NULL, j="adj_avg", value=tmp1$adj_avg*(1+(runif(nrow(sub2))-0.5)*r_k))
                  oof <- c(oof, tmp1$adj_avg)
            }
      }
      oofInd <- data.frame(ind, oof)
      oofInd <- oofInd[order(oofInd$ind),]
      sub1 <- data.table(v1=data[,varList,with=FALSE], y=data[,y,with=FALSE],
                         pred0=data[,pred0,with=FALSE], filt=filter)
      colnames(sub1) <- c(varNames,"y","pred0","filt")
      sub2 <- sub1[sub1$filt==F,]
      sub1 <- sub1[sub1$filt==T,]
      sum1 <- sub1[,list(sumy=sum(y), avgY=mean(y), cnt=length(y)), by=varNames]
      tmp1 <- merge(sub2, sum1, by = varNames, all.x=TRUE, sort=FALSE)
      tmp1$cnt[is.na(tmp1$cnt)] <- 0
      tmp1$sumy[is.na(tmp1$sumy)] <- 0
      if(!is.null(lambda)) tmp1$beta <- lambda else tmp1$beta <- 1/(g+exp((tmp1$cnt - k)/f))
      tmp1$adj_avg <- (1-tmp1$beta)*tmp1$avgY + tmp1$beta*tmp1$pred0
```

```
      tmp1$avgY[is.na(tmp1$avgY)] <- tmp1$pred0[is.na(tmp1$avgY)]
      tmp1$adj_avg[is.na(tmp1$adj_avg)] <- tmp1$pred0[is.na(tmp1$adj_avg)]
      # Combine train and test into one vector
      return(c(oofInd$oof, tmp1$adj_avg))
}
```

```
highCard <- c("building_id", "manager_id")
for (col in 1:length(highCard)){
#      ts1[,paste0(highCard[col],"_mean_low"):=
#                 catNWayAvgCV(ts1, varList=c("dummy",highCard[col]), y="low",
#                              pred0="pred0_low",
#                              filter=ts1[["filter"]]==0, k=10, f=2, r_k=0.02, cv=cvFoldsList)]
      ts1[,paste0(highCard[col],"_mean_med"):=
                catNWayAvgCV(ts1, varList=c("dummy",highCard[col]), y="medium",
                             pred0="pred0_medium",
                             filter=ts1$filter==0, k=5, f=1, r_k=0.01, cv=cvFoldsList)]
      ts1[,paste0(highCard[col],"_mean_high"):=
                catNWayAvgCV(ts1, varList=c("dummy",highCard[col]), y="high",
                             pred0="pred0_high",
                             filter=ts1$filter==0, k=5, f=1, r_k=0.01, cv=cvFoldsList)]
      }
```

More feature engineering

```
# Create some date and other features
ts1 <- ts1[,":="(
      building_id=as.integer(as.factor(building_id)),
      display_address=as.integer(as.factor(display_address)),
      manager_id=as.integer(as.factor(manager_id)),
      street_address=as.integer(as.factor(street_address)),
      desc_wordcount=str_count(description),
      pricePerBed=ifelse(!is.finite(price/bedrooms),-1, price/bedrooms),
      pricePerBath=ifelse(!is.finite(price/bathrooms),-1, price/bathrooms),
      pricePerRoom=ifelse(!is.finite(price/(bedrooms+bathrooms)),-1, price/(bedrooms+bathrooms)),
      bedPerBath=ifelse(!is.finite(bedrooms/bathrooms), -1, price/bathrooms),
      bedBathDiff=bedrooms-bathrooms,
      bedBathSum=bedrooms+bathrooms,
      bedsPerc=ifelse(!is.finite(bedrooms/(bedrooms+bathrooms)), -1, bedrooms/(bedrooms+bathrooms))
    )]
```

Missing values

In the engineered data table `ts1`, the missing values appear in `feature` and `photo_count` related predictors, we should fill these missing values with zero.

```
for (col in 1:ncol(ts1)) set(ts1, i=which(is.na(ts1[[col]])), j=col, value=0)
```

Selet predictors for training and create XGBoost training and test dataset

```
# get variable names
varnames <- setdiff(colnames(ts1),
                    c("photos", "pred0_high", "pred0_low", "pred0_medium",
                      "description", "features", "interest_level", "dummy", "filter",
                      "created", "class", "low", "medium", "high", "street_address"))
# convert dataset to sparse format
t1_sparse <- Matrix(as.matrix(ts1[filter==0, varnames, with=FALSE]), sparse=TRUE)
s1_sparse <- Matrix(as.matrix(ts1[filter==2, varnames, with=FALSE]), sparse=TRUE)
listing_id_test <- ts1[filter %in% c(2), listing_id]
labels <- ts1[filter %in% c(0), class]
# convert dataset to xgb format
dtrain <- xgb.DMatrix(data=t1_sparse, label=labels)
dtest <- xgb.DMatrix(data=s1_sparse)
```

Training and testing with XGBoost

```
# select parameters
param <- list(booster="gbtree",
              objective="multi:softprob",
              eval_metric="mlogloss",
#              nthread=13,
              num_class=3,
              eta = .02,
              gamma = 1,
              max_depth = 4,
              min_child_weight = 1,
              subsample = .7,
              colsample_bytree = .5,
              seed = 36683)
```

```
xgb2cv <- xgb.cv(data = dtrain,
                 params = param,
                 nrounds = 50000,
                 maximize=FALSE,
                 prediction = TRUE,
                 folds = cvFoldsList,
#                 nfold = 5,
                 print_every_n = 50,
                 early_stopping_round=300)
```

```
# xgb train
watch <- list(dtrain=dtrain)
xgb2 <- xgb.train(data = dtrain,
                  params = param,
                  watchlist=watch,
                  print_every_n = 50,
                  nrounds = xgb2cv$best_ntreelimit
#                  nrounds = 4452
)
```

```
# xbg predict
sPreds <- as.data.table(t(matrix(predict(xgb2, dtest), nrow=3, ncol=nrow(dtest))))
colnames(sPreds) <- class$interest_level
fwrite(data.table(listing_id=listing_id_test, sPreds[,list(high,medium,low)]),
       "submission.csv")
```

View important features

```
importance_matrix <- xgb.importance(model = xgb2)
##> xgb.plot.importance(importance_matrix = importance_matrix)
##> print(importance_matrix)
importance_matrix <-
      importance_matrix[, ":="(Varnames=varnames[as.numeric(importance_matrix$Feature)+1])]
print(importance_matrix[,c("Feature", "Varnames", "Gain")],
      nrows = dim(importance_matrix)[1])
```

```
##      Feature                    Varnames         Gain
##   1:       8                       price 9.317820e-02
##   2:     168                pricePerRoom 7.667211e-02
##   3:       3                 building_id 6.808204e-02
##   4:       9                  time_stamp 5.975269e-02
##   5:     166                pricePerBed 4.507835e-02
##   6:      10               distance_city 4.499507e-02
##   7:       5                    latitude 3.913368e-02
##   8:       0                  listing_id 3.716830e-02
##   9:       7                  manager_id 3.712943e-02
##  10:      14                created_hour 3.409262e-02
##  11:       6                   longitude 3.350197e-02
##  12:     167                pricePerBath 3.029281e-02
##  13:      59                photo_count 2.904738e-02
##  14:      43              feature_no_fee 2.589271e-02
##  15:     165              desc_wordcount 2.571771e-02
##  16:       4             display_address 2.281368e-02
##  17:     169                   bedPerBath 1.508129e-02
##  18:     161         building_id_mean_med 1.484276e-02
##  19:       1                    bathrooms 1.470483e-02
##  20:     162        building_id_mean_high 1.432055e-02
##  21:     163          manager_id_mean_med 1.429601e-02
##  22:     164         manager_id_mean_high 1.413435e-02
##  23:     172                    bedsPerc 9.268257e-03
##  24:     160                    positive 8.870563e-03
##  25:     170                 bedBathDiff 8.493919e-03
##  26:      12                created_day 8.392784e-03
##  27:      26             feature_furnish 7.630956e-03
##  28:      31       feature_hardwood_floor 7.284895e-03
##  29:       2                    bedrooms 7.146619e-03
##  30:     155                         joy 6.452011e-03
##  31:     158                       trust 6.143032e-03
##  32:     152                anticipation 5.758387e-03
##  33:     171                  bedBathSum 5.147643e-03
##  34:      35     feature_laundri_in_build 4.726874e-03
##  35:      52            feature_reduc_fee 4.495047e-03
```

```
##  36:      13                   weekofday 4.297909e-03
##  37:     159                    negative 3.919106e-03
##  38:     157                    surprise 3.058834e-03
##  39:      55             feature_simplex 2.922640e-03
##  40:     151                       anger 2.697878e-03
##  41:      19            feature_dishwash 2.562020e-03
##  42:      36     feature_laundri_in_unit 2.164603e-03
##  43:     154                        fear 2.053182e-03
##  44:     156                     sadness 2.017722e-03
##  45:      74        descrpt_central_park 1.874744e-03
##  46:     124          descrpt_queen_size 1.771569e-03
##  47:      20           feature_dog_allow 1.764718e-03
##  48:     111           descrpt_live_room 1.735245e-03
##  49:      17 feature_common_outdoor_space 1.615733e-03
##  50:      23              feature_exclus 1.612171e-03
##  51:      50 feature_privat_outdoor_space 1.598599e-03
##  52:      16           feature_cat_allow 1.568728e-03
##  53:     147        descrpt_washer_dryer 1.516460e-03
##  54:      45       feature_outdoor_space 1.483087e-03
##  55:     125          descrpt_real_estat 1.443200e-03
##  56:     153                     disgust 1.332886e-03
##  57:     113          descrpt_live_super 1.295621e-03
##  58:     136     descrpt_stainless_steel 1.272123e-03
##  59:      84          descrpt_elev_build 1.268816e-03
##  60:      21             feature_doorman 1.264396e-03
##  61:     133        descrpt_size_bedroom 1.243181e-03
##  62:      99           descrpt_high_ceil 1.218581e-03
##  63:      48             feature_pre_war 1.164146e-03
##  64:      97      descrpt_hardwood_floor 1.156295e-03
##  65:      25          feature_fit_center 1.144818e-03
##  66:     145          descrpt_upper_west 1.131973e-03
##  67:      46          feature_park_space 1.102335e-03
##  68:      56           feature_swim_pool 1.017376e-03
##  69:      58   feature_wheelchair_access 1.017105e-03
##  70:      32           feature_high_ceil 1.016455e-03
##  71:     108         descrpt_larg_window 1.012645e-03
##  72:      65       descrpt_bedroom_apart 9.994841e-04
##  73:      18          feature_dine_room 9.992910e-04
##  74:      22               feature_elev 9.903104e-04
##  75:      67          descrpt_bond_york 9.682076e-04
##  76:     110        descrpt_laundri_room 9.655828e-04
##  77:     118          descrpt_month_free 9.545980e-04
##  78:      76        descrpt_closet_space 9.500736e-04
##  79:      96      descrpt_granit_kitchen 9.451794e-04
##  80:      87        descrpt_estat_broker 9.305768e-04
##  81:      82           descrpt_east_side 9.057885e-04
##  82:      92         descrpt_full_servic 8.722421e-04
##  83:      11              created_month 8.660552e-04
##  84:      57             feature_terrac 8.631879e-04
##  85:      62         descrpt_apart_locat 8.599882e-04
##  86:     120         descrpt_newli_renov 8.596557e-04
##  87:      15             feature_balconi 8.580820e-04
##  88:     138          descrpt_step_away 8.465312e-04
##  89:      33 feature_high_speed_internet 8.376590e-04
```

```
##  90:      85         descrpt_elev_laundri 8.110884e-04
##  91:      83          descrpt_east_villag 7.914874e-04
##  92:      88          descrpt_expos_brick 7.811819e-04
##  93:     100         descrpt_hour_doorman 7.752775e-04
##  94:      72            descrpt_call_text 7.632370e-04
##  95:      54            feature_roof_deck 7.611171e-04
##  96:      61         descrpt_apart_featur 7.589508e-04
##  97:     134     descrpt_spacious_bedroom 7.568081e-04
##  98:     103            descrpt_king_size 7.252666e-04
##  99:     115         descrpt_luxuri_build 7.241103e-04
## 100:      80         descrpt_doorman_elev 7.222829e-04
## 101:      94       descrpt_granit_counter 7.183657e-04
## 102:      75            descrpt_citi_view 7.131306e-04
## 103:      30             feature_hardwood 6.940548e-04
## 104:     140          descrpt_subway_line 6.775729e-04
## 105:     112           descrpt_live_space 6.651290e-04
## 106:     117       descrpt_marbl_bathroom 6.543825e-04
## 107:     122        descrpt_pleas_contact 6.404352e-04
## 108:      39                feature_loft 6.330896e-04
## 109:      68       descrpt_broker_support 6.327913e-04
## 110:     148            descrpt_west_side 6.271282e-04
## 111:      78    descrpt_dishwash_microwav 6.244359e-04
## 112:      63    descrpt_applianc_dishwash 6.213671e-04
## 113:     119         descrpt_natur_light 6.096148e-04
## 114:      69         descrpt_build_featur 6.049477e-04
## 115:      98           descrpt_heat_water 5.982211e-04
## 116:     105    descrpt_kitchen_stainless 5.932233e-04
## 117:      49              feature_prewar 5.923681e-04
## 118:     129           descrpt_roof_deck 5.894344e-04
## 119:      90           descrpt_floor_high 5.809564e-04
## 120:      42        feature_new_construct 5.735022e-04
## 121:      81           descrpt_dryer_unit 5.674823e-04
## 122:     123         descrpt_prime_locat 5.632137e-04
## 123:      79        descrpt_doorman_build 5.600827e-04
## 124:     137       descrpt_steel_applianc 5.597505e-04
## 125:     104     descrpt_kitchen_dishwash 5.340890e-04
## 126:      70          descrpt_build_locat 5.231560e-04
## 127:      73          descrpt_ceil_window 5.198305e-04
## 128:     146          descrpt_walk_closet 5.192395e-04
## 129:     106         descrpt_larg_bedroom 5.179063e-04
## 130:      28         feature_garden_patio 5.122921e-04
## 131:     142           descrpt_text_email 5.093176e-04
## 132:     131        descrpt_separ_kitchen 5.075384e-04
## 133:     135        descrpt_spacious_live 5.060248e-04
## 134:      41          feature_multi_level 5.005200e-04
## 135:      53                feature_renov 4.985505e-04
## 136:     126        descrpt_renov_bedroom 4.967621e-04
## 137:      60          descrpt_amen_includ 4.843581e-04
## 138:     143        descrpt_tile_bathroom 4.781219e-04
## 139:      91     descrpt_floor_throughout 4.746324e-04
## 140:      40              feature_lowris 4.678273e-04
## 141:     107           descrpt_larg_live 4.647537e-04
## 142:     132         descrpt_shop_restaur 4.619903e-04
## 143:     121         descrpt_open_kitchen 4.503847e-04
```

```
## 144:      95    descrpt_granit_countertop 4.414326e-04
## 145:      93            descrpt_full_size 4.400944e-04
## 146:     128         descrpt_restaur_shop 4.393523e-04
## 147:     109        descrpt_laundri_build 4.279373e-04
## 148:      44      feature_on_site_laundri 4.271505e-04
## 149:      66     descrpt_bedroom_bathroom 4.245678e-04
## 150:      71           descrpt_call_email 4.242625e-04
## 151:     130         descrpt_schedul_view 4.214440e-04
## 152:     116           descrpt_marbl_bath 4.152336e-04
## 153:     144           descrpt_upper_east 4.121473e-04
## 154:     114         descrpt_locat_heart 3.815682e-04
## 155:      77        descrpt_conveni_locat 3.739247e-04
## 156:      89           descrpt_floor_ceil 3.551144e-04
## 157:     127        descrpt_renov_kitchen 3.517988e-04
## 158:      86           descrpt_equal_hous 2.991821e-04
## 159:     101        descrpt_hous_opportun 2.964888e-04
## 160:     102      descrpt_includ_dishwash 2.940070e-04
## 161:     139         descrpt_studio_apart 2.877485e-04
## 162:     149           descrpt_wood_floor 2.715791e-04
## 163:      64      descrpt_applianc_includ 2.533061e-04
## 164:     150            descrpt_york_real 2.516651e-04
## 165:      27                feature_garag 2.315315e-04
## 166:      24             feature_fireplac 1.829030e-04
## 167:      38        feature_live_in_super 1.616517e-04
## 168:      29          feature_green_build 1.025707e-04
## 169:      37         feature_laundri_room 1.001712e-04
## 170:      47                 feature_pool 8.354137e-05
## 171:      51        feature_publicoutdoor 8.300270e-05
## 172:     141        descrpt_support_equal 8.190068e-05
## 173:      34              feature_laundri 4.100919e-05
##      Feature                     Varnames         Gain
```