Predicting House Prices

Kaggle Competition

This Kaggle competition is about predicting house prices based on a set of around 80 predictor variables. Please read the brief description of the project and get familiar with the various predictors. We will have to do some initial cleaning to successfully work with these data. Overall, we (in teams) will use the provided training dataset to built a multiple linear regression model for predicting house prices. Once we have settled on a final model, we will use it with the predictors available in the testing dataset to predict house prices. The goal of the competition mentions that our predictions $\hat{y}_i$ for the houses in the testing data are compared to the (withheld) true selling prices y_i^{test} via $\sum_i (\log \hat{y}_i - \log y_i^{\text{test}})^2$. Because selling prices are typically right-skewed, I think as a first step we will log-transform the selling prices of the houses in the training data to obtain a more bell-shaped distribution. However, although we will built a model for the log-prices, we will still have to submit the price of a house (and not the log-price) to Kaggle, together with the ID of the house.

Loading and inspecting the train and test datasets

```
library(tidyverse)
## Load Training Data
path_traindata <- 'https://raw.githubusercontent.com/bklingen/Price-Prediction/main/train.csv'
train <- read_csv(path_traindata)
dim(train)
```

```
## [1] 1460   81
```

```
## Load Testing Data
path_testdata <- 'https://raw.githubusercontent.com/bklingen/Price-Prediction/main/test.csv'
test <- read_csv(path_testdata)
dim(test)
```

```
## [1] 1459   80
```

This makes sense: We have one less column in test data because of the missing house prices.

But, are the column names the same? Let's find the "difference" between two sets: All the column names that are in the test data but not in the train data:

```
setdiff(colnames(test), colnames(train))
```

```
## character(0)
```

OK, good, and now the other way around:

```
setdiff(colnames(train), colnames(test))
```

```
## [1] "SalePrice"
```

OK, great. So no surprises there. All predictors that exist in the train data set also appear in the test dataset.

Let's see how many quantitative and how many categorical predictors we have in the training dataset, at least at face value:

```r
train_quantPredictors = train %>% select(where(is.numeric)) %>% select(-SalePrice)
train_catPredictors = train %>% select(where(is.character))
dim(train_quantPredictors)
```

```
## [1] 1460   37
```

```r
dim(train_catPredictors)
```

```
## [1] 1460   43
```

Let's quickly do the same split for the test data:

```r
test_quantPredictors = test %>% select(where(is.numeric))
test_catPredictors = test %>% select(where(is.character))
```

Let's transform the categorical predictors into factors, which should make it easier to combine categories, create a category like "other", etc.

```r
train_catPredictors = train_catPredictors %>% transmute_all(as.factor)
test_catPredictors = test_catPredictors %>% transmute_all(as.factor)
```

First, let's see the category names and frequency for each variable:

```r
for(i in 1:ncol(train_catPredictors)) {
  print(colnames(train_catPredictors)[i])
  print("----")
  print(as.data.frame(fct_count(unlist(train_catPredictors[,i]))))
  print("---------------")
}
```

Handle Categorical Features

MSZoning (Mei)

There are no null/missing values in the training set, but there are a few in the test set

```r
sum(is.na(train$MSZoning))
```

```
## [1] 0
```

```r
sum(is.na(test$MSZoning))
```

```
## [1] 4
```

Although there are 8 potential categories for this variable, there only exist 5 unique ones in the training and test set.

```r
fct_count(train$MSZoning)
```

```
## # A tibble: 5 x 2
##   f           n
##   <fct>   <int>
## 1 C (all)    10
## 2 FV         65
## 3 RH         16
## 4 RL       1151
## 5 RM        218
```

```r
fct_count(test$MSZoning)
```

```
## # A tibble: 6 x 2
```

```
##   f           n
##   <fct>   <int>
## 1 C (all)    15
## 2 FV         74
## 3 RH         10
## 4 RL       1114
## 5 RM        242
## 6 <NA>        4
```

```
mszoning.collapse <- function(x) fct_collapse(x,
                  "FV" = c("FV"),
                  "RL" = c("RL", "RP"),
                  "RO" = c("RM", "RH"),
                  other_level = "other")

train <- train %>% mutate(MSZoning = as.factor(MSZoning), MSZoning = mszoning.collapse(MSZoning))
test <- test %>% mutate(MSZoning = as.factor(MSZoning), MSZoning = mszoning.collapse(MSZoning))
```

```
fct_count(train$MSZoning)
```

```
## # A tibble: 4 x 2
##   f         n
##   <fct> <int>
## 1 FV       65
## 2 RO      234
## 3 RL     1151
## 4 other    10
```

MSSubClass (Mei)

There are no null/missing values

```
sum(is.na(train$MSSubClass))
```

```
## [1] 0
```

```
sum(is.na(test$MSSubClass))
```

```
## [1] 0
```

Assuming the 1/2 story refers to a basement level as "(un)finished" terminology typically refers to, the categories will be split as follows (counts in parenthesis): - 1-STORY 1946 & NEWER single-family (536) - 1-STORY single-family other - 30 1-STORY 1945 & OLDER (69) - 40 1-STORY W/FINISHED ATTIC ALL AGES (4) - 45 1-1/2 STORY - UNFINISHED ALL AGES (12) - 50 1-1/2 STORY FINISHED ALL AGES (144) - multi-level single-family non PUD - 60 2-STORY 1946 & NEWER (299) - 70 2-STORY 1945 & OLDER (60) - 75 2-1/2 STORY ALL AGES (16) - 80 SPLIT OR MULTI-LEVEL (58) - 85 SPLIT FOYER (20) - other - 90 DUPLEX - ALL STYLES AND AGES (52) - 120 1-STORY PUD (Planned Unit Development) - 1946 & NEWER (87) - 150 1-1/2 STORY PUD - ALL AGES - 160 2-STORY PUD - 1946 & NEWER (63) - 180 PUD - MULTILEVEL - INCL SPLIT LEV/FOYER (10) - 190 2 FAMILY CONVERSION - ALL STYLES AND AGES (30)

```
mssubclass.collapse <- function(x) fct_collapse(x,
                  "1-story single-family 1946 & newer" = c("20"),
                  "1-story single-family other" = c("30", "40", "45", "50"),
                  "multi-level single-family non PUD" = c("60", "70", "75", "80", "85"),
                  other_level = "other")

train <- train %>% mutate(MSSubClass = as.factor(MSSubClass), MSSubClass = mssubclass.collapse(MSSubCla
```

```r
test <- test %>% mutate(MSSubClass = as.factor(MSSubClass), MSSubClass = mssubclass.collapse(MSSubClass
```

```r
fct_count(train$MSSubClass)
```

```
## # A tibble: 4 x 2
##   f                                       n
##   <fct>                               <int>
## 1 1-story single-family 1946 & newer    536
## 2 1-story single-family other           229
## 3 multi-level single-family non PUD     453
## 4 other                                 242
```

Condition1/Condition2 (Mei)

There are no null/missing values

```r
sum(is.na(train$Condition1))
```

```
## [1] 0
```

```r
sum(is.na(test$Condition1))
```

```
## [1] 0
```

```r
sum(is.na(train$Condition2))
```

```
## [1] 0
```

```r
sum(is.na(test$Condition2))
```

```
## [1] 0
```

Collapse similar locations together: - All the railroad related locations - All the park related locations - All the street related locations This results in only 4 categories: - Normal - Near railroad - Near park - Near arterial or feeder street

```r
condition.collapse <- function(x) fct_collapse(x,
                  RR = c("RRNn", "RRAn", "RRNe", "RRAe"),
                  Pos = c("PosN", "PosA"),
                  St = c("Artery", "Feedr"))

train <- train %>% mutate_at(vars(Condition1, Condition2), condition.collapse)
test <- test %>% mutate_at(vars(Condition1, Condition2), condition.collapse)
```

```r
fct_count(train$Condition1)
```

```
## # A tibble: 4 x 2
##   f         n
##   <fct> <int>
## 1 St      129
## 2 Norm   1260
## 3 Pos      27
## 4 RR       44
```

RoofStyle (Richard)

combine flat, shed as other; gambrel, mansard, gable as gable; leave others as is

```r
roof_price <- train %>% group_by(RoofStyle) %>% summarize(count=n(),
  mean(SalePrice), sd(SalePrice))
```

```r
roof_price
```

```
## # A tibble: 6 x 4
##   RoofStyle count `mean(SalePrice)` `sd(SalePrice)`
##   <chr>     <int>             <dbl>           <dbl>
## 1 Flat         13            194690          62523.
## 2 Gable      1141            171484.         66331.
## 3 Gambrel      11            148909.         67014.
## 4 Hip         286            218877.        111550.
## 5 Mansard       7            180568.         58058.
## 6 Shed          2            225000          49497.
```

```r
train$RoofStyle <- fct_collapse(train$RoofStyle, Other = c("Flat", "Shed"))
train$RoofStyle <- fct_collapse(train$RoofStyle, Gable = c("Gable", "Gambrel", "Mansard"))
```

Let's do the same on the testing dataset:

```r
test$RoofStyle <- fct_collapse(test$RoofStyle, Other = c("Flat", "Shed"))
test$RoofStyle <- fct_collapse(test$RoofStyle, Gable = c("Gable", "Gambrel", "Mansard"))
```

BldgType

Combine 2FmCon, Duplex as multifamily; leave others as is

```r
bldg_price <- train %>% group_by(BldgType) %>% summarize(count=n(),
  mean(SalePrice), sd(SalePrice))

bldg_price
```

```
## # A tibble: 5 x 4
##   BldgType count `mean(SalePrice)` `sd(SalePrice)`
##   <chr>    <int>             <dbl>           <dbl>
## 1 1Fam      1220           185764.          82649.
## 2 2fmCon      31           128432.          35459.
## 3 Duplex      52           133541.          27833.
## 4 Twnhs       43           135912.          41013.
## 5 TwnhsE     114           181959.          60626.
```

```r
train$BldgType <- fct_collapse(train$BldgType, MultiFam = c("2fmCon", "Duplex"))
```

Let's do the same on the testing dataset:

```r
test$BldgType <- fct_collapse(test$BldgType, MultiFam = c("2fmCon", "Duplex"))
```

HouseStyle

Combine 1.5Fin, 1Story, split foyer, split level as less than 2 story; 2.5fin, 2Story as two story or greater; leave 1.5Unf and 2.5Unf as is since they drag down property values

```r
style_price <- train %>% group_by(HouseStyle) %>% summarize(count=n(),
  mean(SalePrice), sd(SalePrice))

style_price
```

```
## # A tibble: 8 x 4
##   HouseStyle count `mean(SalePrice)` `sd(SalePrice)`
##   <chr>      <int>             <dbl>           <dbl>
```

```
## 1 1.5Fin       154           143117.          54278.
## 2 1.5Unf        14           110150           19036.
## 3 1Story       726           175985.          77056.
## 4 2.5Fin         8           220000          118212.
## 5 2.5Unf        11           157355.          63934.
## 6 2Story       445           210052.          87339.
## 7 SFoyer        37           135074.          30481.
## 8 SLvl          65           166703.          38305.
```

```
train$HouseStyle <- fct_collapse(train$HouseStyle, Less2story = c("1Story", "1.5Fin", "SFoyer", "SLvl"))
train$HouseStyle <- fct_collapse(train$HouseStyle, EqMore2story = c("2Story", "2.5Fin"))
```

And on the test data:

```
test$HouseStyle <- fct_collapse(test$HouseStyle, Less2story = c("1Story", "1.5Fin", "SFoyer", "SLvl"))
test$HouseStyle <- fct_collapse(test$HouseStyle, EqMore2story = c("2Story", "2.5Fin"))
```

```
## Warning: Unknown levels in `f`: 2.5Fin
```

Kyle:

```
cleanpool <- as.character(train_catPredictors$PoolQC)
cleanpool[is.na(cleanpool)] <- "none"
cleanpool <- as.factor(cleanpool)
```

```
cleanfence <- as.character(train_catPredictors$Fence)
cleanfence[is.na(cleanfence)] <- "none"
cleanfence <- as.factor(cleanfence)
```

```
cleanfunc <- as.character(train_catPredictors$Functional)
cleanfunc[cleanfunc == 'Min1' | cleanfunc == 'Min2'] <- "Minor"
cleanfunc[cleanfunc == 'Maj1' | cleanfunc == 'Maj2'] <- "Major"
cleanfunc[cleanfunc == 'Sev' | cleanfunc == 'Sal'] <- "Severe"
cleanfunc <- as.factor(cleanfunc)
```

```
train$PoolQC <- cleanpool
train$Fence <- cleanfence
train$Functional <- cleanfunc
```

We need to do the same for the test dataset, so I just copied the code block and replaced "train" by "test":

```
cleanpool <- as.character(test_catPredictors$PoolQC)
cleanpool[is.na(cleanpool)] <- "none"
cleanpool <- as.factor(cleanpool)
```

```
cleanfence <- as.character(test_catPredictors$Fence)
cleanfence[is.na(cleanfence)] <- "none"
cleanfence <- as.factor(cleanfence)
```

```
cleanfunc <- as.character(test_catPredictors$Functional)
cleanfunc[cleanfunc == 'Min1' | cleanfunc == 'Min2'] <- "Minor"
cleanfunc[cleanfunc == 'Maj1' | cleanfunc == 'Maj2'] <- "Major"
cleanfunc[cleanfunc == 'Sev' | cleanfunc == 'Sal'] <- "Severe"
cleanfunc <- as.factor(cleanfunc)
```

```
test$PoolQC <- cleanpool
test$Fence <- cleanfence
test$Functional <- cleanfunc
```

Mileva: Heating, Electrical, FireplaceQu, HeatingQC, CentralAir

The processing for the Heating, Electrical, and FireplaceQu predictors is below. The HeatingQC and CentralAir predictors did not require any additional processing.

```
# Heating: Collapsed categores with low frequencies into "other"
heating <- as.factor(train_catPredictors$Heating)
heating <- fct_other(heating, keep=c("GasA", "GasW"))
train$Heating <- heating
```

```
# Electrical: Collapsed similar categories together and handled missing values
electrical <- as.character(train_catPredictors$Electrical)

electrical <- fct_collapse(electrical, Fuse=c("FuseA", "FuseF", "FuseP"))
electrical <- fct_collapse(electrical, Other=c("Mix"))
electrical[is.na(electrical)] <- "Other"

train$Electrical <- electrical
```

```
# Fireplace: Handled missing values
fireplace <- as.character(train_catPredictors$FireplaceQu)
fireplace[is.na(fireplace)] <- "none"
train$FireplaceQu <- as.factor(fireplace)
```

Need to do the same for test dataset:

```
# Heating: Collapsed categores with low frequencies into "other"
heating <- as.factor(test_catPredictors$Heating)
heating <- fct_other(heating, keep=c("GasA", "GasW"))
test$Heating <- heating
```

```
# Electrical: Collapsed similar categories together and handled missing values
electrical <- as.character(test_catPredictors$Electrical)

electrical <- fct_collapse(electrical, Fuse=c("FuseA", "FuseF", "FuseP"))
electrical <- fct_collapse(electrical, Other=c("Mix"))
```

```
## Warning: Unknown levels in `f`: Mix
```

```
electrical[is.na(electrical)] <- "Other"
```

```
## Warning in `[<-.factor`(`*tmp*`, is.na(electrical), value = "Other"): invalid
## factor level, NA generated
```

```
test$Electrical <- electrical
```

```
# Fireplace: Handled missing values
fireplace <- as.character(test_catPredictors$FireplaceQu)
fireplace[is.na(fireplace)] <- "none"
test$FireplaceQu <- as.factor(fireplace)
```

RoofMatl - Dropped (Thomas)

1434/1460 entries in the training set are CompShg.
The off-materials aren't meaningfully different price-wise as an 'other' group. Wood Shingles ('wdshngl') does contain 2 houses in the 99th percentile sale price, but with only 6 entries I don't think it's safe to include.
I think we're better off dropping this one.

```
train <- select(train, -c(RoofMatl))
test <- select(test, -c(RoofMatl))
```

Exterior1st/2nd (Thomas)

Fixed the following label mis-matches between columns:Exterior1st - WdShing,CemntBd,BrkComm, Exterior2nd - Wd Shng,CmentBd,Brk Cmn

~90% of these two variables matched. In the ~10% that didn't match, Exterior1st is generally a better predictor of sale price than Exterior2nd. I converted Exterior2nd into a boolean, TRUE if Exterior1st!=Exterior2nd.

I combined the bottom half of Exterior1st's categories into an 'Other' category. (This leaves 7, but Brick Face/Cement Board seem to be decent categories for predicting sale price, so I didn't want to drop them.)

```
train$Exterior2nd[train$Exterior2nd=='Wd Shng'] <- 'WdShing'
train$Exterior2nd[train$Exterior2nd=='CmentBd'] <- 'CemntBd'
train$Exterior2nd[train$Exterior2nd=='Brk Cmn'] <- 'BrkComm'
train$Exterior2nd <- train$Exterior1st!=train$Exterior2nd
train$Exterior1st <- fct_collapse(train$Exterior1st, Other = c("AsbShng","AsphShn","CBlock","ImStucc","

test$Exterior2nd[test$Exterior2nd=='Wd Shng']<- 'WdShing'
test$Exterior2nd[test$Exterior2nd=='CmentBd']<- 'CemntBd'
test$Exterior2nd[test$Exterior2nd=='Brk Cmn']<- 'BrkComm'
test$Exterior2nd <- test$Exterior1st!=test$Exterior2nd
test$Exterior1st <- fct_collapse(test$Exterior1st, Other = c("AsbShng","AsphShn","CBlock","ImStucc","Br
```

```
## Warning: Unknown levels in `f`: ImStucc, Stone
```

Bernhard: I also changed `ExterCond`:

```
table(train$ExterCond)
```

```
##
##   Ex   Fa   Gd   Po   TA
##    3   28  146    1 1282
```

```
table(test$ExterCond)
```

```
##
##   Ex   Fa   Gd   Po   TA
##    9   39  153    2 1256
```

`Po` and `Ex` are rather uncommon, so we collapse them all into "other":

```
train$ExterCond = fct_collapse(train$ExterCond, other=c("Ex", "Po"))
test$ExterCond = fct_collapse(test$ExterCond, other=c("Ex", "Po"))

summary(train$ExterCond)
```

```
## other    Fa    Gd    TA
##     4    28   146  1282
```

```
summary(test$ExterCond)
```

```
## other    Fa    Gd    TA
##    11    39   153  1256
```

SaleType

WD, New, and Court deed/estate were the three most common categories, and all 3 were significant when using SaleType as sole predictor. Combined the other categories into 'Other'.

```
train$SaleType <- fct_collapse(train$SaleType, Other = c("ConLD", "ConLw", "ConLI", "CWD", "Oth", "Con"
test$SaleType <- fct_collapse(test$SaleType, Other = c("ConLD", "ConLw", "ConLI", "CWD", "Oth", "Con"))
```

Marina: Neighborhood, GarageType, GarageFinish, GarageQual, GarageCond

```
### Neighborhood ###
# Collapse categores with low frequencies into "other"

#Explore counts
train_catPredictors %>% count(Neighborhood, sort = TRUE)
```

```
## # A tibble: 25 x 2
##    Neighborhood     n
##    <fct>        <int>
##  1 NAmes          225
##  2 CollgCr        150
##  3 OldTown        113
##  4 Edwards        100
##  5 Somerst         86
##  6 Gilbert         79
##  7 NridgHt         77
##  8 Sawyer          74
##  9 NWAmes          73
## 10 SawyerW         59
## # ... with 15 more rows
```

```
#Factorize
neighborhood <- as.factor(train_catPredictors$Neighborhood)

#Convert to "Other" any category that represents less than 2% of the data
neighborhood <- fct_collapse(neighborhood, Other = c("MeadowV","BrDale", "Veenker", "NPkVill", "Blueste

levels(neighborhood) #New levels of the factor
```

```
##  [1] "Other"   "BrkSide" "CollgCr" "Crawfor" "Edwards" "Gilbert" "Mitchel"
##  [8] "NAmes"   "NoRidge" "NridgHt" "NWAmes"  "OldTown" "Sawyer"  "SawyerW"
## [15] "Somerst" "Timber"
```

```
#Update column with new values
train$Neighborhood <- neighborhood
```

Need to do the same on test data:

```
#Factorize
neighborhood <- as.factor(test_catPredictors$Neighborhood)

#Convert to "Other" any category that represents less than 2% of the data
neighborhood <- fct_collapse(neighborhood, Other = c("MeadowV","BrDale", "Veenker", "NPkVill", "Blueste
levels(neighborhood) #New levels of the factor
```

```
##  [1] "Other"   "BrkSide" "CollgCr" "Crawfor" "Edwards" "Gilbert" "Mitchel"
##  [8] "NAmes"   "NoRidge" "NridgHt" "NWAmes"  "OldTown" "Sawyer"  "SawyerW"
```

```
## [15] "Somerst" "Timber"
```

```r
#Update column with new values
test$Neighborhood <- neighborhood
```

Anyone sees the issue??

```r
table(train$Neighborhood)
```

```
##
##   Other BrkSide CollgCr Crawfor Edwards Gilbert Mitchel   NAmes NoRidge NridgHt
##     187      58     150      51     100      79      49     225      41      77
##  NWAmes OldTown  Sawyer SawyerW Somerst  Timber
##      73     113      74      59      86      38
```

```r
table(test$Neighborhood)
```

```
##
##   Other BrkSide CollgCr Crawfor Edwards Gilbert Mitchel   NAmes NoRidge NridgHt
##     201      50     117      52      94      86      65     218      30      89
##  NWAmes OldTown  Sawyer SawyerW Somerst  Timber
##      58     126      77      66      96      34
```

```r
### GarageType ###

#Explore counts
train_catPredictors %>% count(GarageType, sort = TRUE)
```

```
## # A tibble: 7 x 2
##   GarageType     n
##   <fct>      <int>
## 1 Attchd       870
## 2 Detchd       387
## 3 BuiltIn       88
## 4 <NA>          81
## 5 Basment       19
## 6 CarPort        9
## 7 2Types         6
```

```r
#Handle NAs
#According to the data description, NA means no garage.
#Change NA category to "none" to avoid issues.
garageType <- as.character(train_catPredictors$GarageType)
garageType[is.na(garageType)] <- "none"
garageType <- as.factor(garageType)
```

```r
#Collapse into "Other" categries that represent less than 5% of the data
garageType <- garageType %>%
  fct_lump(prop=0.05, other_level='Other')

#levels(garageType) #New levels of the factor
```

```r
#Update column with new values
train$GarageType <- garageType
```

Attention!! Need to do the same on the test data:

```r
garageType <- as.character(test$GarageType)
garageType[is.na(garageType)] <- "none"
```

```r
garageType <- as.factor(garageType)
garageType <- garageType %>%
  fct_lump(prop=0.05, other_level='Other')
levels(garageType)
```

```
## [1] "Attchd"  "BuiltIn" "Detchd"  "none"    "Other"
```

```r
levels(train$GarageType)
```

```
## [1] "Attchd"  "BuiltIn" "Detchd"  "none"    "Other"
```

```r
test$GarageType <- garageType
```

```r
### GarageFinish ###

#Explore counts
train_catPredictors %>% count(GarageFinish, sort = TRUE)
```

```
## # A tibble: 4 x 2
##   GarageFinish     n
##   <fct>        <int>
## 1 Unf            605
## 2 RFn            422
## 3 Fin            352
## 4 <NA>            81
```

```r
#Handle NAs
#According to the data description, NA means no garage.
#Change NA category to "none" to avoid issues.
garageFinish <- as.character(train_catPredictors$GarageFinish)
garageFinish[is.na(garageFinish)] <- "none"
garageFinish <- as.factor(garageFinish)

#No need to collapse categories
```

```r
#Update column with new values
train$GarageFinish <- garageFinish
```

Need to do the same for the test data:

```r
#Handle NAs
#According to the data description, NA means no garage.
#Change NA category to "none" to avoid issues.
garageFinish <- as.character(test_catPredictors$GarageFinish)
garageFinish[is.na(garageFinish)] <- "none"
garageFinish <- as.factor(garageFinish)
#No need to collapse categories
```

```r
#Update column with new values
test$GarageFinish <- garageFinish
```

```r
### GarageQual ###

#Explore counts
train_catPredictors %>% count(GarageQual, sort = TRUE)
```

```
## # A tibble: 6 x 2
##   GarageQual     n
```

```
##   <fct>      <int>
## 1 TA          1311
## 2 <NA>          81
## 3 Fa            48
## 4 Gd            14
## 5 Ex             3
## 6 Po             3
```

```
#Handle NAs
#According to the data description, NA means no garage.
#Change NA category to "none" to avoid issues.
garageQual <- as.character(train_catPredictors$GarageQual)
garageQual[is.na(garageQual)] <- "none"
garageQual <- as.factor(garageQual)
```

```
#Collapse categories:
# - Let's collapse Ex   (Excellent) and Gd  (Good) into 1 category: Gd
# - Let's collapse Fa   (Fair) and Po   (Poor) into 1 category: Po
# - None and TA remains the same

garageQual <- fct_collapse(garageQual, Gd = c("Ex","Gd"))
garageQual <- fct_collapse(garageQual, Po = c("Fa","Po"))
```

```
#Update column with new values
train$GarageQual <- garageQual
```

Need to do the same for test data:

```
#Handle NAs
#According to the data description, NA means no garage.
#Change NA category to "none" to avoid issues.
garageQual <- as.character(test_catPredictors$GarageQual)
garageQual[is.na(garageQual)] <- "none"
garageQual <- as.factor(garageQual)
```

```
#Collapse categories:
# - Let's collapse Ex   (Excellent) and Gd  (Good) into 1 category: Gd
# - Let's collapse Fa   (Fair) and Po   (Poor) into 1 category: Po
# - None and TA remains the same

garageQual <- fct_collapse(garageQual, Gd = c("Ex","Gd"))
```

```
## Warning: Unknown levels in `f`: Ex
```

```
garageQual <- fct_collapse(garageQual, Po = c("Fa","Po"))
```

```
#Update column with new values
test$GarageQual <- garageQual
```

```
### GarageCond ###

#Explore counts
train_catPredictors %>% count(GarageCond, sort = TRUE)
```

```
## # A tibble: 6 x 2
##   GarageCond     n
##   <fct>      <int>
## 1 TA          1326
```

```
## 2 <NA>          81
## 3 Fa            35
## 4 Gd             9
## 5 Po             7
## 6 Ex             2
```

```
#Handle NAs
#According to the data description, NA means no garage.
#Change NA category to "none" to avoid issues.
garageCond <- as.character(train_catPredictors$GarageCond)
garageCond[is.na(garageCond)] <- "none"
garageCond <- as.factor(garageCond)
```

```
#Collapse categories:
# - Let's collapse Ex   (Excellent) and Gd  (Good) into 1 category: Gd
# - Let's collapse Fa   (Fair) and Po   (Poor) into 1 category: Po
# - None and TA remains the same

garageCond <- fct_collapse(garageCond, Gd = c("Ex","Gd"))
garageCond <- fct_collapse(garageCond, Po = c("Fa","Po"))
```

```
#Update column with new values
train$GarageCond <- garageCond
```

Need to do the same with test data:

```
#Handle NAs
#According to the data description, NA means no garage.
#Change NA category to "none" to avoid issues.
garageCond <- as.character(test_catPredictors$GarageCond)
garageCond[is.na(garageCond)] <- "none"
garageCond <- as.factor(garageCond)
```

```
#Collapse categories:
# - Let's collapse Ex   (Excellent) and Gd  (Good) into 1 category: Gd
# - Let's collapse Fa   (Fair) and Po   (Poor) into 1 category: Po
# - None and TA remains the same

garageCond <- fct_collapse(garageCond, Gd = c("Ex","Gd"))
garageCond <- fct_collapse(garageCond, Po = c("Fa","Po"))
```

```
#Update column with new values
test$GarageCond <- garageCond
```

Note: We also need to discuss the NA's in the numerical variable GarageYrBlt, see later.

Paul: LotShape, LotConfig, LandContour

Fortunately there are no NA values in either the test or train sets.

```
sum(is.na(train$LotShape))
```

```
## [1] 0
```

```
sum(is.na(test$LotShape))
```

```
## [1] 0
```

```r
sum(is.na(train$LotConfig))
```

```
## [1] 0
```

```r
sum(is.na(test$LotConfig))
```

```
## [1] 0
```

```r
sum(is.na(train$LandContour))
```

```
## [1] 0
```

```r
sum(is.na(test$LandContour))
```

```
## [1] 0
```

```r
fct_count(train$LotShape)
```

```
## # A tibble: 4 x 2
##   f         n
##   <fct> <int>
## 1 IR1     484
## 2 IR2      41
## 3 IR3      10
## 4 Reg     925
```

```r
fct_count(test$LotShape)
```

```
## # A tibble: 4 x 2
##   f         n
##   <fct> <int>
## 1 IR1     484
## 2 IR2      35
## 3 IR3       6
## 4 Reg     934
```

```r
fct_count(train$LotConfig)
```

```
## # A tibble: 5 x 2
##   f           n
##   <fct>   <int>
## 1 Corner    263
## 2 CulDSac    94
## 3 FR2        47
## 4 FR3         4
## 5 Inside   1052
```

```r
fct_count(test$LotConfig)
```

```
## # A tibble: 5 x 2
##   f           n
##   <fct>   <int>
## 1 Corner    248
## 2 CulDSac    82
## 3 FR2        38
## 4 FR3        10
## 5 Inside   1081
```

```r
fct_count(train$LandContour)
```

```
## # A tibble: 4 x 2
##   f         n
##   <fct> <int>
## 1 Bnk      63
## 2 HLS      50
## 3 Low      36
## 4 Lvl    1311
```

```r
fct_count(test$LandContour)
```

```
## # A tibble: 4 x 2
##   f         n
##   <fct> <int>
## 1 Bnk      54
## 2 HLS      70
## 3 Low      24
## 4 Lvl    1311
```

All of these variables are highly imbalanced. In each there is one category that represents a “regular” shape, configuration, or land contour, which amount for ~2/3 or more of the total instances. Thus, I collapsed all of the less represented “irregular” categories into one.

```r
train$LotShape <- fct_collapse(train$LotShape, Irregular = c("IR1", "IR2", "IR3"))
train$LotConfig <- fct_collapse(train$LotConfig, Other = c("Corner","CulDSac", "FR2", "FR3"))
train$LandContour <- fct_collapse(train$LandContour, NonLvl = c("Bnk", "HLS", "Low"))
```

```r
fct_count(train$LotShape)
```

```
## # A tibble: 2 x 2
##   f             n
##   <fct>     <int>
## 1 Irregular   535
## 2 Reg         925
```

```r
fct_count(train$LotConfig)
```

```
## # A tibble: 2 x 2
##   f          n
##   <fct>  <int>
## 1 Other    408
## 2 Inside  1052
```

```r
fct_count(train$LandContour)
```

```
## # A tibble: 2 x 2
##   f          n
##   <fct>  <int>
## 1 NonLvl   149
## 2 Lvl     1311
```

Need to do the same for the test data:

```r
test$LotShape <- fct_collapse(test$LotShape, Irregular = c("IR1", "IR2", "IR3"))
test$LotConfig <- fct_collapse(test$LotConfig, Other = c("Corner","CulDSac", "FR2", "FR3"))
test$LandContour <- fct_collapse(test$LandContour, NonLvl = c("Bnk", "HLS", "Low"))
```

For the variable `MiscFeature`, almost all values are missing. However, looking in the data description file, this actually means that the house simply doesn't have any other features. So, we set the NA's to "none", in both the train and test datasets. The same applies to `Alley`, where an NA means "none":

```
train$MiscFeature = fct_explicit_na(train$MiscFeature, na_level="none")
test$MiscFeature = fct_explicit_na(test$MiscFeature, na_level="none")

train$Alley = fct_explicit_na(train$Alley, na_level="none")
test$Alley = fct_explicit_na(test$Alley, na_level="none")
```

For `LotFrontage`, the missing values are genuine. (But lets hope that the value being missing has no connection to the sales price of a house.)

Another issue with fitting a full model is the number of unique values a predictor has. If it only has **one unique value (or one unique factor level)**, then it doesn't vary, i.e., it is a constant. This causes issues because then the design matrix X is not full rank. The column for the intercept is a column of all 1's, and then each column for a predictor which is constant is also a column of a fixed number. This causes a linear dependency between these columns, and the design matrix is not full rank.

First, lets turn the character variables into factors, both in the training and testing data. This will pay off later:

```
train = train %>% mutate(across(where(is.character), as.factor))
test = test %>% mutate(across(where(is.character), as.factor))
```

Let's find the predictors which have constant values throughout:

```
train %>%
  summarize(across(everything(), ~length(unique(.x)))) %>%
  sort()
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```
## # A tibble: 1 x 80
##   Street LotShape LandContour Utilities LotConfig Exterior2nd CentralAir Alley
##    <int>    <int>       <int>     <int>     <int>       <int>      <int> <int>
## 1      2        2           2         2         2           2          2     3
## # ... with 72 more variables: LandSlope <int>, RoofStyle <int>, Heating <int>,
## #   Electrical <int>, BsmtHalfBath <int>, HalfBath <int>, PavedDrive <int>,
## #   MSSubClass <int>, MSZoning <int>, Condition1 <int>, Condition2 <int>,
## #   BldgType <int>, HouseStyle <int>, ExterQual <int>, ExterCond <int>,
## #   BsmtFullBath <int>, FullBath <int>, KitchenAbvGr <int>, KitchenQual <int>,
## #   Fireplaces <int>, GarageFinish <int>, GarageQual <int>, GarageCond <int>,
## #   PoolQC <int>, SaleType <int>, MasVnrType <int>, BsmtQual <int>, ...
```

There doesn't seem to be a variable that has only one unique value or one unique factor level. So we should be good to go.

Having done/checked all that, we are ready to fit the full model with all variables. However, using `> fit2 = lm(log(SalePrice) ~ . , data=train %>% select(-Id, -SalePrice))`, I ran into a problem, where R shows the error message `contrasts can be applied only to factors with 2 or more levels`.

With trial and error, I saw that we can fit a model with the first 8 predictors, but when we include 'Utilities', there is an issue

```
fit2 = lm(log(train$SalePrice) ~ . , data=train[,2:9])
summary(fit2)
```

```
##
## Call:
## lm(formula = log(train$SalePrice) ~ ., data = train[, 2:9])
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.36843 -0.20396 -0.03778  0.18938  1.10208
##
## Coefficients:
##                                             Estimate Std. Error t value
## (Intercept)                                1.178e+01  1.766e-01  66.695
## MSSubClass1-story single-family other     -1.999e-01  3.060e-02  -6.531
## MSSubClassmulti-level single-family non PUD  1.505e-01  2.361e-02   6.373
## MSSubClassother                            1.592e-02  3.126e-02   0.509
## MSZoningRO                                -3.892e-01  5.575e-02  -6.982
## MSZoningRL                                -2.201e-01  5.187e-02  -4.243
## MSZoningother                             -9.451e-01  1.177e-01  -8.028
## LotFrontage                                2.851e-03  4.789e-04   5.954
## LotArea                                    7.202e-06  1.346e-06   5.352
## StreetPave                                 1.959e-01  1.528e-01   1.282
## AlleyPave                                  1.152e-01  7.898e-02   1.458
## Alleynone                                  9.683e-02  5.118e-02   1.892
## LotShapeReg                               -1.762e-01  2.200e-02  -8.009
## LandContourLvl                             4.205e-02  3.304e-02   1.273
##                                           Pr(>|t|)
## (Intercept)                                < 2e-16 ***
## MSSubClass1-story single-family other     9.67e-11 ***
## MSSubClassmulti-level single-family non PUD 2.65e-10 ***
## MSSubClassother                             0.6106
## MSZoningRO                                4.82e-12 ***
## MSZoningRL                                2.37e-05 ***
## MSZoningother                             2.36e-15 ***
## LotFrontage                               3.44e-09 ***
## LotArea                                   1.05e-07 ***
## StreetPave                                  0.2002
## AlleyPave                                   0.1451
## Alleynone                                   0.0587 .
## LotShapeReg                               2.74e-15 ***
## LandContourLvl                              0.2033
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3245 on 1187 degrees of freedom
##   (259 observations deleted due to missingness)
## Multiple R-squared:  0.3984, Adjusted R-squared:  0.3918
## F-statistic: 60.47 on 13 and 1187 DF,  p-value: < 2.2e-16
```

What is going on with utilities:

```
summary(train$Utilities)
```

```
## AllPub NoSeWa
##   1459      1
```

We see that it is almost constant! There is only one observation with a different utility type. Probably, that observation has some missing values on some other variables, and hence is removed from the design matrix, making it an all constant predictor. Let's check:

```
train[train$Utilities == 'NoSeWa',]
```

```
## # A tibble: 1 x 80
##      Id MSSubClass            MSZoning LotFrontage LotArea Street Alley LotShape
##   <dbl> <fct>                 <fct>          <dbl>   <dbl> <fct>  <fct> <fct>
## 1   945 1-story single-famil~ RL                NA   14375 Pave   none  Irregul~
## # ... with 72 more variables: LandContour <fct>, Utilities <fct>,
## #   LotConfig <fct>, LandSlope <fct>, Neighborhood <fct>, Condition1 <fct>,
## #   Condition2 <fct>, BldgType <fct>, HouseStyle <fct>, OverallQual <dbl>,
## #   OverallCond <dbl>, YearBuilt <dbl>, YearRemodAdd <dbl>, RoofStyle <fct>,
## #   Exterior1st <fct>, Exterior2nd <lgl>, MasVnrType <fct>, MasVnrArea <dbl>,
## #   ExterQual <fct>, ExterCond <fct>, Foundation <fct>, BsmtQual <fct>,
## #   BsmtCond <fct>, BsmtExposure <fct>, BsmtFinType1 <fct>, ...
```

There we go, `LotFrontage` is `NA` for this particular house, so it is removed, and the remaining houses all have the same utility type.

Removing/Replacing Missing Values

We now need to handle the Basement values. We need to replace the NA's with "none":

```
train$BsmtQual = fct_explicit_na(train$BsmtQual, na_level="none")
train$BsmtCond = fct_explicit_na(train$BsmtCond, na_level="none")
train$BsmtExposure = fct_explicit_na(train$BsmtExposure, na_level="none")
train$BsmtFinType1 = fct_explicit_na(train$BsmtFinType1, na_level="none")
train$BsmtFinType2 = fct_explicit_na(train$BsmtFinType2, na_level="none")

test$BsmtQual = fct_explicit_na(test$BsmtQual, na_level="none")
test$BsmtCond = fct_explicit_na(test$BsmtCond, na_level="none")
test$BsmtExposure = fct_explicit_na(test$BsmtExposure, na_level="none")
test$BsmtFinType1 = fct_explicit_na(test$BsmtFinType1, na_level="none")
test$BsmtFinType2 = fct_explicit_na(test$BsmtFinType2, na_level="none")

train %>%
  summarize(across(everything(), ~sum(is.na(.x)))) %>%
  sort(decreasing = TRUE)
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```
## # A tibble: 1 x 80
##   LotFrontage GarageYrBlt MasVnrType MasVnrArea    Id MSSubClass MSZoning
##         <int>       <int>      <int>      <int> <int>      <int>    <int>
## 1         259          81          8          8     0          0        0
## # ... with 73 more variables: LotArea <int>, Street <int>, Alley <int>,
## #   LotShape <int>, LandContour <int>, Utilities <int>, LotConfig <int>,
## #   LandSlope <int>, Neighborhood <int>, Condition1 <int>, Condition2 <int>,
## #   BldgType <int>, HouseStyle <int>, OverallQual <int>, OverallCond <int>,
## #   YearBuilt <int>, YearRemodAdd <int>, RoofStyle <int>, Exterior1st <int>,
## #   Exterior2nd <int>, ExterQual <int>, ExterCond <int>, Foundation <int>,
## #   BsmtQual <int>, BsmtCond <int>, BsmtExposure <int>, BsmtFinType1 <int>, ...
```

For `MasVnrType`, I will introduce a new category "missing", but for `MasVnrArea` I will just imput 0 for those 8 missing areas:

```
summary(train$MasVnrType)
```

```
##  BrkCmn BrkFace    None   Stone    NA's
```

```
##      15     445     864     128       8
```

```r
train$MasVnrType = fct_explicit_na(train$MasVnrType, na_level="missing")
train$MasVnrArea[is.na(train$MasVnrArea)] = 0

test$MasVnrType = fct_explicit_na(test$MasVnrType, na_level="missing")
test$MasVnrArea[is.na(test$MasVnrArea)] = 0
```

We now have no missing predictor values in the training data:

```r
dim(train)
```

```
## [1] 1460   80
```

```r
dim(train %>% drop_na())
```

```
## [1] 1127   80
```

Let's now revisit check if any predictors are constant:

```r
train %>%
  summarize(across(everything(), ~length(unique(.x)))) %>%
  sort()
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```
## # A tibble: 1 x 80
##   Street LotShape LandContour Utilities LotConfig Exterior2nd CentralAir Alley
##    <int>    <int>       <int>     <int>     <int>       <int>      <int> <int>
## 1      2        2           2         2         2           2          2     3
## # ... with 72 more variables: LandSlope <int>, RoofStyle <int>, Heating <int>,
## #   Electrical <int>, BsmtHalfBath <int>, HalfBath <int>, PavedDrive <int>,
## #   MSSubClass <int>, MSZoning <int>, Condition1 <int>, Condition2 <int>,
## #   BldgType <int>, HouseStyle <int>, ExterQual <int>, ExterCond <int>,
## #   BsmtFullBath <int>, FullBath <int>, KitchenAbvGr <int>, KitchenQual <int>,
## #   Fireplaces <int>, GarageFinish <int>, GarageQual <int>, GarageCond <int>,
## #   PoolQC <int>, SaleType <int>, MasVnrType <int>, BsmtQual <int>, ...
```

Seems fine, although for `Utilities`:

```r
summary(train$Utilities)
```

```
## AllPub NoSeWa
##   1459      1
```

This means we also need to drop `Utilities` from the test data.

```r
train =  train %>% select(-Utilities)
test =  test %>% select(-Utilities)
```

NA's in Test Data

Just like in the training dataset, we might have some NA's in the test data:

```r
isNAtest = apply(test,1,function(x) any(is.na(x)))
sum(isNAtest)
```

```
## [1] 309
```

We still have 11 observations with at least one missing predictor. This is a problem since when we use all predictors, we will not be able to obtain a predicted sales price for these 11 houses. Which predictors have the most missing values:

```
test %>%
  summarize(across(everything(), ~sum(is.na(.x)))) %>%
  sort(decreasing = TRUE)
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames

## # A tibble: 1 x 78
##   LotFrontage GarageYrBlt MSZoning BsmtFullBath BsmtHalfBath Functional
##         <int>       <int>    <int>        <int>        <int>      <int>
## 1         227          78        4            2            2          2
## # ... with 72 more variables: Exterior1st <int>, Exterior2nd <int>,
## #   BsmtFinSF1 <int>, BsmtFinSF2 <int>, BsmtUnfSF <int>, TotalBsmtSF <int>,
## #   KitchenQual <int>, GarageCars <int>, GarageArea <int>, SaleType <int>,
## #   Id <int>, MSSubClass <int>, LotArea <int>, Street <int>, Alley <int>,
## #   LotShape <int>, LandContour <int>, LotConfig <int>, LandSlope <int>,
## #   Neighborhood <int>, Condition1 <int>, Condition2 <int>, BldgType <int>,
## #   HouseStyle <int>, OverallQual <int>, OverallCond <int>, ...
```

MSZoning:

```
summary(train$MSZoning)
```

```
##    FV    RO    RL other
##    65   234  1151    10
```

```
summary(test$MSZoning)
```

```
##    FV    RO    RL other  NA's
##    74   252  1114    15     4
```

```
test$MSZoning = fct_explicit_na(test$MSZoning, na_level="other")
summary(test$MSZoning)
```

```
##    FV    RO    RL other
##    74   252  1114    19
```

BsmtFullBath:

```
summary(train$BsmtFullBath)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.4253  1.0000  3.0000
```

```
summary(test$BsmtFullBath)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.0000  0.0000  0.0000  0.4345  1.0000  3.0000       2
```

```
test$BsmtFullBath[is.na(test$BsmtFullBath)] = 0
```

BsmtHalfBath:

```
summary(train$BsmtHalfBath)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.05753 0.00000 2.00000
```

```r
summary(test$BsmtHalfBath)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##  0.0000  0.0000  0.0000  0.0652  0.0000  2.0000       2
```

```r
test$BsmtHalfBath[is.na(test$BsmtHalfBath)] = 0
```

Functional:

```r
summary(train$Functional)
```

```
##   Major   Minor     Mod  Severe     Typ 
##      19      65      15       1    1360
```

```r
summary(test$Functional)
```

```
##   Major   Minor     Mod  Severe     Typ    NA's 
##       9      70      20       1    1357       2
```

```r
train$Functional = train$Functional == "Typ"
test$Functional = test$Functional == "Typ"
test$Functional[is.na(test$Functional)] = TRUE
summary(train$Functional)
```

```
##    Mode   FALSE    TRUE 
## logical     100    1360
```

```r
summary(test$Functional)
```

```
##    Mode   FALSE    TRUE 
## logical     100    1359
```

Exterior1st:

```r
summary(train$Exterior1st)
```

```
##   Other BrkFace CemntBd HdBoard MetalSd Plywood VinylSd Wd Sdng 
##      78      50      61     222     220     108     515     206
```

```r
summary(test$Exterior1st)
```

```
##   Other BrkFace CemntBd HdBoard MetalSd Plywood VinylSd Wd Sdng    NA's 
##      78      37      65     220     230     113     510     205       1
```

```r
test$Exterior1st <- fct_explicit_na(test$Exterior1st, na_level="Other")
summary(test$Exterior1st)
```

```
##   Other BrkFace CemntBd HdBoard MetalSd Plywood VinylSd Wd Sdng 
##      79      37      65     220     230     113     510     205
```

Exterior2nd:

```r
summary(train$Exterior2nd)
```

```
##    Mode   FALSE    TRUE 
## logical    1323     137
```

```r
summary(test$Exterior2nd)
```

```
##    Mode   FALSE    TRUE    NA's 
## logical    1327     131       1
```

```r
test$Exterior2nd[is.na(test$Exterior2nd)] = FALSE
```

BsmtFinSF1

```r
summary(train$BsmtFinSF1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0     0.0   383.5   443.6   712.2  5644.0
```

```r
summary(test$BsmtFinSF1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##     0.0     0.0   350.5   439.2   753.5  4010.0       1
```

```r
test$BsmtFinSF1[is.na(test$BsmtFinSF1)] = 0
```

BsmtFinSF2

```r
summary(train$BsmtFinSF2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    0.00    0.00    0.00   46.55    0.00 1474.00
```

```r
summary(test$BsmtFinSF2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##    0.00    0.00    0.00   52.62    0.00 1526.00       1
```

```r
test$BsmtFinSF2[is.na(test$BsmtFinSF2)] = 0
```

BsmtUnfSF

```r
summary(train$BsmtUnfSF)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0   223.0   477.5   567.2   808.0  2336.0
```

```r
summary(test$BsmtUnfSF)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##     0.0   219.2   460.0   554.3   797.8  2140.0       1
```

```r
test$BsmtUnfSF[is.na(test$BsmtUnfSF)] = 460
```

TotalBsmtSF

```r
summary(train$TotalBsmtSF)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0   795.8   991.5  1057.4  1298.2  6110.0
```

```r
summary(test$TotalBsmtSF)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       0     784     988    1046    1305    5095       1
```

```
test$TotalBsmtSF[is.na(test$TotalBsmtSF)] = 988
```

KitchenQual

```
summary(train$KitchenQual)
```

```
##  Ex  Fa  Gd  TA
## 100  39 586 735
```

```
summary(test$KitchenQual)
```

```
##   Ex   Fa   Gd   TA NA's
##  105   31  565  757    1
```

```
test$KitchenQual[is.na(test$KitchenQual)] = "TA"
```

GarageCars

```
summary(train$GarageCars)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   2.000   1.767   2.000   4.000
```

```
summary(test$GarageCars)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   1.000   2.000   1.766   2.000   5.000       1
```

```
test$GarageCars[is.na(test$GarageCars)] = 1.766
```

GarageArea

```
summary(train$GarageArea)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   334.5   480.0   473.0   576.0  1418.0
```

```
summary(test$GarageArea)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     0.0   318.0   480.0   472.8   576.0  1488.0       1
```

```
test$GarageArea[is.na(test$GarageArea)] = 480
```

SaleType

```
summary(train$SaleType)
```

```
##   COD Other   New    WD
##    43    28   122  1267
```

```
summary(test$SaleType)
```

```
##   COD Other   New    WD  NA's
##    44    39   117  1258     1
```

```
test$SaleType[is.na(test$SaleType)] = "Other"
```