Case Study: Iowa Housing Prices

Tim Lee, Taylor Pellerin, Jake Toffler, Ian Smeenk, Alex Howard

10/4/2017

Contents

EDA Summary:

Changing the sales price response field to log(sales price) for normality reasons

After checking for normality, the sales price fails the two sided K-S test, forcing us to reject the null hypothesis. Then transforming the sales price into log(sales price), we find we keep the null hypothesis and the response variable is normal.

Dropping Null Fields

A number of the fields were predominantly null. The following features were over 20% null and were excluded from the data:

- Lot Frontage
- Alley
- PoolQC
- Fence
- MiscFeature
- FireplaceQu

Converting String Quality Descriptions into numerics

Many of the fields had discrete ratings such as "excellent", "good" etc. All of these related fields were changed to a ~10 point scale listed below.

Numeric Assigned	Str Encoding	Data Dictionary Description
9	Ex	Excellent
7	Gd	Good
5	TA	Average/Typical
3	Fa	Fair
1	Po	Poor
0	NA	No basement

Transform dates into years old (continuous)

Many of the year related fields such as the year built, or the year modified, or year of the garage were transformed into continous "age" in years. The reference date to make this calculation is 2010 which is the most recent data in the fields. For example

2010 (current) - 2007 year built = house is 3 years old

Imputing missing values

Depending on the field that was missing values, either the NA was replaced with a "None" for fields such as garage type. Or the "average" value was imputed instead like "Standard Breaker". In other cases the NA was a proxy for 0. For all the qualitative fields, the NA's were considered a rating of 0 (mainly for the size of the basement)

Convert specific fields to factors

The primary field that was converted to factors was the MSSubClass. The other fields under consideration were number of bedrooms, kitchens, bathes. Should these be considered discrete fields? Technically there is no such thing as 2.35 bedrooms. The final model left these counting fields as continuous integers, but will be discussed more later.

Detecting Outliers

DFFITs was used to identify influential outliers in the data. These 59 row observations were excluded from the future analysis.

Removing Collinear Fields

These fields have been removed from tweaking and research during OLS, Lasso, or Ridge analysis. - TotalBsmtSF: this can be calculated from BsmtSF1 and BsmtSF2 added together - BldgType: this is covered more detail in the MSSubClass: Identifies the type of dwelling involved in the sale. So this is unnecessary - Combining Exterior Fields, Exterior1st,Exterior2nd,

Removing imbalanced fields - these are fields where a high % of the field is the same value

A histogram analysis was performed on all fields. The following fields had >= 90% one value so were dropped as predictors for this study:

- Condition2
- Heating
- RoofMatl
- X3SsnPorch
- LowQualFinSF
- KitchenAbvGr
- MiscVal
- LandSlope
- CentralAir
- BsmtHalfBath
- Functional
- PavedDrive
- Electrical
- ScreenPorch
- GarageQual
- LandContour

Detailed EDA Analysis, Walkthrough and Code

Load the Data

```
# load the kaggle CSV
df <- read.csv("~/Documents/MSAN/04-601-LinRegression/Project/housing.txt",
    stringsAsFactors = FALSE)
```

EDA: Examination of the Response Variable (Testing for Normality):

Boxcox Examination

```
y_resp <- df$SalePrice %>% unlist
X <- df %>% select(-SalePrice) %>% DealWithNulls
fit.model <- lm(y_resp ~ ., X)

res <- boxcox(lm(y_resp ~ ., X), plotit = T, lambda = seq(0,
    1, by = 0.5))
```

Optimal Transform - Y to the power of 0.13, close to 0, will continue to use log

```
data.frame(res) %>% arrange(desc(y)) %>% head(5) %>% kable
```

x	y
0.1313131	-1572.348
0.1212121	-1572.353
0.1414141	-1572.457
0.1111111	-1572.473
0.1515152	-1572.680

```
# make a function to do the transformation for later
bcTrans <- function(val) {
    return((val^0.131)/0.131)
}
```

Plot the histogram of Sale Price (not transformed)

```
df %>% select(SalePrice) %>% ggplot(aes(x = SalePrice)) + geom_histogram(bins = 40) +
    labs(x = "Sales Price of House", title = "Histogram of Sales Price in Iowa")
```

K-S test of Sale Price (untransformed)

From the K-S test we see we must reject the null hypothesis that hte data is normal

```r
set.seed(1)
sale_price <- df %>% select(SalePrice) %>% unlist

n_y <- length(sale_price)
sale_price_norm <- (sale_price - mean(sale_price))/sd(sale_price)
std_norm <- rnorm(n = n_y, mean = 0, sd = 1)

ks.test(sale_price_norm, std_norm)
```

```
## Warning in ks.test(sale_price_norm, std_norm): p-value will be approximate
## in the presence of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  sale_price_norm and std_norm
## D = 0.11575, p-value = 0.000000006386
## alternative hypothesis: two-sided
```

K-S test of log(Sale Price)

Since the distribution of the sale price is skewed, we will try and apply a log transformation and retest under the Kolmagorov-Test

```r
set.seed(1)
log_sale_price <- log(sale_price)
log_sale_price_norm <- (log_sale_price - mean(log_sale_price))/sd(log_sale_price)
std_norm <- rnorm(n = n_y, mean = 0, sd = 1)


ks.test(log_sale_price_norm, std_norm)
```

```
## Warning in ks.test(log_sale_price_norm, std_norm): p-value will be
## approximate in the presence of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  log_sale_price_norm and std_norm
## D = 0.043836, p-value = 0.1209
## alternative hypothesis: two-sided
```

Plot the histogram of log(Sale Price)

```r
df %>% select(SalePrice) %>% log %>% ggplot(aes(x = SalePrice)) +
    geom_histogram(bins = 40) + labs(x = "Log Sales Price of House",
    title = "Histogram of Log(Sales Price) in Iowa")
```

EDA: Cleaning up the data with a Pipeline

Function: Dropping nulls, based on histogram statistics

This will be accomplished by checking for number of nulls

```
null_columns <- colSums(is.na(df))
sort(null_columns[null_columns > 0]/n_y, decreasing = T) %>%
    head(10) %>% kable(caption = "Field Null%", col.names = c("%Null"))
```

Table 3: Field Null%

	%Null
PoolQC	0.9952055
MiscFeature	0.9630137
Alley	0.9376712
Fence	0.8075342
FireplaceQu	0.4726027
LotFrontage	0.1773973
GarageType	0.0554795
GarageYrBlt	0.0554795
GarageFinish	0.0554795
GarageQual	0.0554795

Function: Dealing with Nulls

Anything with more than 20% more nulls, the field is dropped. The fields dropped are as follows:

- Lot Frontage
- Alley
- PoolQC
- Fence
- MiscFeature
- FireplaceQu

```
# Drop fields with around 280 nulls (20%)
DealWithNulls <- function(df) {
    drop_fields = c("LotFrontage", "Alley", "PoolQC", "Fence",
        "MiscFeature", "FireplaceQu")

    df_clean <- data.frame(df) %>% select(-LotFrontage, -Alley,
        -PoolQC, -Fence, -MiscFeature, -FireplaceQu, -Utilities) %>%
        data.frame
    return(df_clean)
}
```

Function: transform dates into years old

Will assume that the current year is 2010

```
Years_to_Age <- function(df) {
    df_clean <- data.frame(df)
```

```
    df_clean$GarageYrBlt <- as.integer(2010 - df$GarageYrBlt)
    df_clean$YrSold <- as.integer(2010 - df$YrSold)
    df_clean$YearBuilt <- as.integer(2010 - df$YearBuilt)
    df_clean$YearRemodAdd <- as.integer(2010 - df$YearRemodAdd)
    return(df_clean)
}
```

Function: Fixing fields of MS Sub Class: Turning Numerics into Factors

- There is a zoning field that has numeric values that we force into a categorical factor type.
- All counting factors were switched to factors (room counts, bath counts)
- Month was changed to factor

```
SwapToFactor <- function(df) {

    # setup a copy (to prevent from looping with the same
    # variable name)
    df_clean <- data.frame(df)

    # Convert numerics into factors
    df_clean$MSSubClass <- factor(df$MSSubClass, levels = c(20,
        30, 40, 45, 50, 60, 70, 75, 80, 85, 90, 120, 150, 160,
        180, 190), labels = c("1LVL>1946", "1LVL<1945", "1LVL_W_ATTIC",
        "1.5LVL_UNF", "1.5LVL_FIN", "2LVL>1946", "2LVL<1945",
        "2.5LVL", "SPLIT", "SPLIT_FOYER", "DUPLEX", "1LVL_PUD>1946",
        "1.5LVL_PUD", "2LVL_PUD>1946", "MULTI_PUD", "2FAM_CONV"))
    return(df_clean)
}


RoomsToFactor <- function(df) {
    # setup a copy (to prevent from looping with the same
    # variable name)
    df_clean <- data.frame(df)

    df_clean$BsmtFullBath <- factor(df$BsmtFullBath)
    df_clean$BsmtHalfBath <- factor(df$BsmtHalfBath)
    df_clean$FullBath <- factor(df$FullBath)
    df_clean$HalfBath <- factor(df$HalfBath)
    df_clean$BedroomAbvGr <- factor(df$BedroomAbvGr)
    df_clean$KitchenAbvGr <- factor(df$KitchenAbvGr)
    df_clean$TotRmsAbvGrd <- factor(df$TotRmsAbvGrd)
    df_clean$Fireplaces <- factor(df$Fireplaces)
    df_clean$GarageCars <- factor(df$GarageCars)
    df_clean$MoSold <- factor(df$MoSold)

    return(df_clean)
}
```

Function: Change any of the Quality fields into numbers. Define a transformation function

For all the fields that have a word rating, we converted the answers into a numeric scale. This will both reduce the number of categorical fields and allow more interpretability in the model. The word to score translation is listed below:

Numeric Assigned	Str Encoding	Data Dictionary Description
9	Ex	Excellent
7	Gd	Good
5	TA	Average/Typical
3	Fa	Fair
1	Po	Poor
0	NA	No basement

For the few of these fields that have nulls, this is intentional, the data dictionary actually defines the NA's as no basement.

```
QualityToNumeric <- function(df) {

    # setup a copy (to prevent from looping with the same
    # variable name)
    df_clean <- data.frame(df)

    # convert words into continuous scores
    df_clean$GarageYrBlt <- as.numeric(df_clean$GarageYrBlt)
    df_clean$ExterCond <- sapply(df$ExterCond, function(x) {
        switch(x, Ex = 9, Gd = 7, TA = 5, Av = 5, Fa = 3, Po = 1,
            `NA` = 0)
    })
    df_clean$ExterQual <- sapply(df$ExterQual, function(x) {
        switch(x, Ex = 9, Gd = 7, TA = 5, Av = 5, Fa = 3, Po = 1,
            `NA` = 0)
    })
    df_clean$BsmtQual <- sapply(df$BsmtQual, function(x) {
        switch(x, Ex = 9, Gd = 7, TA = 5, Av = 5, Fa = 3, Po = 1,
            `NA` = 0)
    })
    df_clean$BsmtCond <- sapply(df$BsmtCond, function(x) {
        switch(x, Ex = 9, Gd = 7, TA = 5, Av = 5, Fa = 3, Po = 1,
            `NA` = 0)
    })
    df_clean$BsmtExposure <- sapply(df$BsmtExposure, function(x) {
        switch(x, Gd = 10, Av = 7, Mn = 4, No = 1, `NA` = 0)
    })
    df_clean$BsmtFinType1 <- sapply(df$BsmtFinType1, function(x) {
        switch(x, GLQ = 10, ALQ = 7, Rec = 7, BLQ = 4, LwQ = 1,
            Unf = 0, `NA` = 0)
    })
    df_clean$BsmtFinType2 <- sapply(df$BsmtFinType2, function(x) {
        switch(x, GLQ = 10, ALQ = 7, Rec = 7, BLQ = 4, LwQ = 1,
            Unf = 0, `NA` = 0)
    })
```

```
    df_clean$HeatingQC <- sapply(df$HeatingQC, function(x) {
        switch(x, Ex = 9, Gd = 7, TA = 5, Av = 5, Fa = 3, Po = 1,
            `NA` = 0)
    })
    df_clean$KitchenQual <- sapply(df$KitchenQual, function(x) {
        switch(x, Ex = 9, Gd = 7, TA = 5, Av = 5, Fa = 3, Po = 1,
            `NA` = 0)
    })
    df_clean$Functional <- sapply(df$Functional, function(x) {
        switch(x, Typ = 1, Min1 = 0.85, Min2 = 0.7, Mod = 0.55,
            Maj1 = 0.45, Maj2 = 0.3, Sev = 0.15, Sal = 0)
    })
    df_clean$GarageQual <- sapply(df$GarageQual, function(x) {
        switch(x, Ex = 9, Gd = 7, TA = 5, Av = 5, Fa = 3, Po = 1,
            `NA` = 0)
    })
    df_clean$GarageCond <- sapply(df$GarageQual, function(x) {
        switch(x, Ex = 9, Gd = 7, TA = 5, Av = 5, Fa = 3, Po = 1,
            `NA` = 0)
    })
    df_clean$GarageFinish <- sapply(df$GarageFinish, function(x) {
        switch(x, Fin = 1, RFn = 0.66, Unf = 0.33, `NA` = 0)
    })
    return(df_clean)
}
```

Function: Imputing some of the missing values

For the few fields that were left with NA's, the following strategy was used. Since most of the NA's showed up in Categorical all the NA's were replaced with new categories called "None". Based on the data dictionary the remainder were assigned default values.

```
ImputeValues <- function(df) {

    # setup a copy (to prevent from looping with the same
    # variable name)
    df_clean <- data.frame(df)

    df_clean[is.na(df_clean$GarageYrBlt), ]$GarageYrBlt = 0

    # add levels for none
    levels(df_clean$GarageType) = c(levels(df$GarageType), "None")
    df_clean[is.na(df_clean$GarageType), ]$GarageType <- "None"

    # addFill nones, 0 and default values
    df_clean[is.na(df_clean$MasVnrType), ]$MasVnrType <- "None"
    df_clean[is.na(df_clean$MasVnrArea), ]$MasVnrArea <- 0
    df_clean[is.na(df_clean$Electrical), ]$Electrical <- "SBrkr"
    return(df_clean)
}
```

Clean all datasets - check for nulls in the clean dataset

```
df_clean <- df %>% Years_to_Age %>% SwapToFactor %>% DealWithNulls %>%
    QualityToNumeric %>% ImputeValues

# check for nulls
sum(is.na(df_clean))
```

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

Sample of the Cleaned up table

```
t(head(df_clean)) %>% kable
```

	1	2	3	4	5	6
Id	1	2	3	4	5	6
MSSubClass	2LVL>1946	1LVL>1946	2LVL>1946	2LVL<1945	2LVL>1946	1.5LVL_FIN
MSZoning	RL	RL	RL	RL	RL	RL
LotArea	8450	9600	11250	9550	14260	14115
Street	Pave	Pave	Pave	Pave	Pave	Pave
LotShape	Reg	Reg	IR1	IR1	IR1	IR1
LandContour	Lvl	Lvl	Lvl	Lvl	Lvl	Lvl
LotConfig	Inside	FR2	Inside	Corner	FR2	Inside
LandSlope	Gtl	Gtl	Gtl	Gtl	Gtl	Gtl
Neighborhood	CollgCr	Veenker	CollgCr	Crawfor	NoRidge	Mitchel
Condition1	Norm	Feedr	Norm	Norm	Norm	Norm
Condition2	Norm	Norm	Norm	Norm	Norm	Norm
BldgType	1Fam	1Fam	1Fam	1Fam	1Fam	1Fam
HouseStyle	2Story	1Story	2Story	2Story	2Story	1.5Fin
OverallQual	7	6	7	7	8	5
OverallCond	5	8	5	5	5	5
YearBuilt	7	34	9	95	10	17
YearRemodAdd	7	34	8	40	10	15
RoofStyle	Gable	Gable	Gable	Gable	Gable	Gable
RoofMatl	CompShg	CompShg	CompShg	CompShg	CompShg	CompShg
Exterior1st	VinylSd	MetalSd	VinylSd	Wd Sdng	VinylSd	VinylSd
Exterior2nd	VinylSd	MetalSd	VinylSd	Wd Shng	VinylSd	VinylSd
MasVnrType	BrkFace	None	BrkFace	None	BrkFace	None
MasVnrArea	196	0	162	0	350	0
ExterQual	7	5	7	5	7	5
ExterCond	5	5	5	5	5	5
Foundation	PConc	CBlock	PConc	BrkTil	PConc	Wood
BsmtQual	7	7	7	5	7	7
BsmtCond	5	5	5	7	5	5
BsmtExposure	1	10	4	1	7	1
BsmtFinType1	10	7	10	7	10	10
BsmtFinSF1	706	978	486	216	655	732
BsmtFinType2	0	0	0	0	0	0
BsmtFinSF2	0	0	0	0	0	0
BsmtUnfSF	150	284	434	540	490	64
TotalBsmtSF	856	1262	920	756	1145	796
Heating	GasA	GasA	GasA	GasA	GasA	GasA
HeatingQC	9	9	9	7	9	9
CentralAir	Y	Y	Y	Y	Y	Y
Electrical	SBrkr	SBrkr	SBrkr	SBrkr	SBrkr	SBrkr

	1	2	3	4	5	6
X1stFlrSF	856	1262	920	961	1145	796
X2ndFlrSF	854	0	866	756	1053	566
LowQualFinSF	0	0	0	0	0	0
GrLivArea	1710	1262	1786	1717	2198	1362
BsmtFullBath	1	0	1	1	1	1
BsmtHalfBath	0	1	0	0	0	0
FullBath	2	2	2	1	2	1
HalfBath	1	0	1	0	1	1
BedroomAbvGr	3	3	3	3	4	1
KitchenAbvGr	1	1	1	1	1	1
KitchenQual	7	5	7	7	7	5
TotRmsAbvGrd	8	6	6	7	9	5
Functional	1	1	1	1	1	1
Fireplaces	0	1	1	1	1	0
GarageType	Attchd	Attchd	Attchd	Detchd	Attchd	Attchd
GarageYrBlt	7	34	9	12	10	17
GarageFinish	0.66	0.66	0.66	0.33	0.66	0.33
GarageCars	2	2	2	3	3	2
GarageArea	548	460	608	642	836	480
GarageQual	5	5	5	5	5	5
GarageCond	5	5	5	5	5	5
PavedDrive	Y	Y	Y	Y	Y	Y
WoodDeckSF	0	298	0	0	192	40
OpenPorchSF	61	0	42	35	84	30
EnclosedPorch	0	0	0	272	0	0
X3SsnPorch	0	0	0	0	0	320
ScreenPorch	0	0	0	0	0	0
PoolArea	0	0	0	0	0	0
MiscVal	0	0	0	0	0	700
MoSold	2	5	9	2	12	10
YrSold	2	3	2	4	2	1
SaleType	WD	WD	WD	WD	WD	WD
SaleCondition	Normal	Normal	Normal	Abnorml	Normal	Normal
SalePrice	208500	181500	223500	140000	250000	143000

EDA: Optional EDA Cleanup Functions

Optional - Function: swap roof materials with price per tile

Basically plug in the price as the indicator: http://www.hgtv.com/remodel/outdoors/top-6-roofing-materials

```
RoofMatlToDollars <- function(df) {
    # make a local copy
    df_clean <- data.frame(df)
    df_clean$RoofMatl <- sapply(df_clean$RoofMatl, function(x) switch(x,
        ClyTile = 400, CompShg = 250, Membran = 10, Metal = 500,
        Roll = 100, `Tar&Grv` = 100, WdShake = 125, WdShngl = 125))
    return(df_clean)
}
```

Optional - Function: exclude dominant features (see overview below for a table)

The following fields had 90% one value so were dropped as predictors for this study:

- Condition2
- Heating
- RoofMatl
- X3SsnPorch
- LowQualFinSF
- KitchenAbvGr
- MiscVal
- LandSlope
- CentralAir
- BsmtHalfBath
- Functional
- PavedDrive
- Electrical
- ScreenPorch
- GarageQual
- GarageCond
- LandContour

```
ExcludeFeatures_90Plus <- function(df) {
    return(df %>% select(-Condition2, -Heating, -RoofMatl, -X3SsnPorch,
        -LowQualFinSF, -KitchenAbvGr, -MiscVal, -LandSlope, -CentralAir,
        -BsmtHalfBath, -Functional, -PavedDrive, -Electrical,
        -ScreenPorch, -GarageQual, -LandContour))
}
```

Optional - Function: Remove Collinear Fields from OLS analysis

These fields have been removed from tweaking and research during OLS, Lasso, or Ridge analysis. - TotalBsmtSF: this can be calculated from BsmtSF1 and BsmtSF2 added together - BldgType: this is covered more detail in the MSSubClass: Identifies the type of dwelling involved in the sale. So this is unnecessary - Combining Exterior Fields, Exterior1st,Exterior2nd makes a single fields and deals with a Field1-NA case

```
RemoveCollinear <- function(df) {
    df_clean <- data.frame(df)
```

```
    df_clean$ExtCom <- df_clean %>% mutate(ExtCom = paste0(Exterior1st,
        "-", Exterior2nd)) %>% select(ExtCom) %>% unlist
    df_clean <- df %>% select(-TotalBsmtSF, -Exterior1st, -Exterior2nd,
        -BldgType, -HouseStyle, -GarageCond, -SaleCondition)
    return(df_clean)
}
```

Optional - Function: Convert All areas to Log Areas

Based on some of the studies below:

```
AreaToLogArea <- function(df) {
    df_clean <- data.frame(df)
    df_clean$GrLivArea <- log(df$GrLivArea + 1)
    df_clean$LotArea <- log(df$LotArea + 1)
    df_clean$GarageArea <- log(df$GarageArea + 1)
    df_clean$BsmtFinSF1 <- log(df$BsmtFinSF1 + 1)
    df_clean$BsmtFinSF2 <- log(df$BsmtFinSF2 + 1)
    df_clean$MasVnrArea <- log(df$MasVnrArea + 1)
    df_clean$X1stFlrSF <- log(df$X1stFlrSF + 1)
    df_clean$X2ndFlrSF <- log(df$X2ndFlrSF + 1)
    df_clean$BsmtUnfSF <- log(df$BsmtUnfSF + 1)

    return(df_clean)
}

AreaToLogAreaExcluded90 <- function(df) {
    df_clean <- data.frame(df)
    df_clean$X3SsnPorch <- log(df$X3SsnPorch + 1)
    df_clean$LowQualFinSF <- log(df$LowQualFinSF + 1)
    df_clean$ScreenPorch <- log(df$ScreenPorch + 1)
    df_clean$MiscVal <- log(df$MiscVal + 1)
    df_clean$PoolArea <- log(df$PoolArea + 1)
    df_clean$WoodDeckSF <- log(df$WoodDeckSF + 1)
    df_clean$OpenPorchSF <- log(df$OpenPorchSF + 1)
    df_clean$EnclosedPorch <- log(df$EnclosedPorch + 1)
    return(df_clean)
}
```

EDA: Exploration of Linear relationships to LogResponse: Check other fields manually

Compare Total Rooms Above Ground

We see from the plots below, that the rooms above ground level don't change much in comparison to their log versions. The inherent data is naturally linear already and do not need to be transformed

```
reg <- df %>% mutate(log_Saleprice = log(SalePrice)) %>% ggplot(aes(x = TotRmsAbvGrd,
    y = log_Saleprice)) + geom_point() + labs(title = "TotRmsAbvGrd")


log <- df %>% mutate(log_Saleprice = log(SalePrice)) %>% ggplot(aes(x = log(TotRmsAbvGrd),
    y = log_Saleprice)) + geom_point() + labs(title = "Log TotRmsAbvGrd")

multiplot(reg, log, cols = 2)
```

Compare How old the house is (Yr Built in years)

We see from the plots below, similarly, the years old the house is is loosely already linear to log sale price and does not need to be transformed.

```
reg <- df %>% mutate(log_Saleprice = log(SalePrice)) %>% ggplot(aes(x = YearBuilt,
    y = log_Saleprice)) + geom_point() + labs(title = "YearBuilt")


log <- df %>% mutate(log_Saleprice = log(SalePrice)) %>% ggplot(aes(x = log(YearBuilt),
    y = log_Saleprice)) + geom_point() + labs(title = "Log YearBuilt")

multiplot(reg, log, cols = 2)
```

Compare Garage Area

We look at garage area to see if log Area will be more closely linearly related to log price, and we see the shape straighten and spread out

```
reg <- df %>% mutate(log_Saleprice = log(SalePrice)) %>% ggplot(aes(x = GarageArea,
    y = log_Saleprice)) + geom_point() + labs(title = "GarageArea")


log <- df %>% mutate(log_Saleprice = log(SalePrice)) %>% ggplot(aes(x = log(GarageArea),
    y = log_Saleprice)) + geom_point() + labs(title = "Log GarageArea")

multiplot(reg, log, cols = 2)
```

Compare first floor Square Footage

As verification we look at first floor area to see if log Area will be more closely linearly related to log price, and we see the shape straighten and spread out. As a result, we will convert all area metrics into log versions of them to have a better linear relationship with the log price.

```r
reg <- df %>% mutate(log_Saleprice = log(SalePrice)) %>% ggplot(aes(x = X1stFlrSF,
    y = log_Saleprice)) + geom_point() + labs(title = "X1stFlrSF")


log <- df %>% mutate(log_Saleprice = log(SalePrice)) %>% ggplot(aes(x = log(X1stFlrSF),
    y = log_Saleprice)) + geom_point() + labs(title = "Log X1stFlrSF")

multiplot(reg, log, cols = 2)
```

EDA: Outlier Analysis

```
# setup the fit model
y_log <- df_clean$SalePrice %>% log %>% unlist
X <- df_clean %>% select(-SalePrice, -Id)
fit <- lm(y_log ~ ., X)
```

Influence Plot

From running a quick influence plot, we see a good number of outlier points with high influence (large circles). From the plot, there's approximately 30+ points that are outside of the criteria (horizontal lines)

```
# Influence Plot
influencePlot(fit, id.method = "identify", main = "Influence Plot",
    sub = "Circle size is proportial to Cook's Distance")
```

Function: Using DFFITS to Exclude Points

Will use the rule

$$2\sqrt{\frac{p}{n}}$$

< DFFITS. To exclude points

```
df_clean <- df %>% Years_to_Age %>% SwapToFactor %>% DealWithNulls %>%
    QualityToNumeric %>% ImputeValues %>% RoofMatlToDollars %>%
    RemoveCollinear %>% # ExcludeFeatures_90Plus %>% # this was to improve the score
AreaToLogArea %>% AreaToLogAreaExcluded90

Exclude_Outliers <- function(df_clean) {

    # pull out OLS variables (will use all of them at the moment)
    y_log <- df_clean$SalePrice %>% log %>% unlist
    X <- df_clean %>% select(-SalePrice, -Id)

    # fit the model
    fit <- lm(y_log ~ ., X)

    # Set a general cut off parametner 2 sqrt(p/n)
    y_log <- df_clean$SalePrice %>% log %>% unlist
    X <- df_clean %>% select(-SalePrice, -Id)
    fit_dffits <- ols_dffits_plot(lm(y_log ~ ., X))

    # conservatively leave the NA's in
    points_to_exclude_by_id <- fit_dffits$outliers$Observation

    # find hte points to keep
    points_to_keep <- setdiff(1:nrow(df_clean), points_to_exclude_by_id)

    # dplyr to filter for those points to keep
    return(df_clean %>% filter(Id %in% points_to_keep))
}
```

```r
# Sample
z <- Exclude_Outliers(df_clean)
```

EDA: Export to Disk for others to use:

```
df_clean <- df %>% Years_to_Age %>% SwapToFactor %>% DealWithNulls %>%
    QualityToNumeric %>% ImputeValues %>% RoofMatlToDollars %>%
    RemoveCollinear %>% Exclude_Outliers
```

```
df %>% Years_to_Age %>% SwapToFactor %>% DealWithNulls %>% QualityToNumeric %>%
    ImputeValues %>% RoofMatlToDollars %>% RemoveCollinear %>%
    ExcludeFeatures_90Plus %>% Exclude_Outliers %>% write.csv("~/Documents/MSAN/04-601-LinRegression/Pr
```

EDA: Function: Value counts within each field

This table is a mini histogram per field, it lists the top values, how often they show up and what % of the total field they occupy. These are used for identifying fields over 90% dominated by a single value.

```
get_df_stats <- function(df) {
    # get statistics
    rows <- nrow(df)
    colno <- ncol(df)
    colnames <- names(df)
    all_headings <- list()
    all_outputs <- list()
    null_pct <- list()

    # loop through each column
    for (i in 1:colno) {

        # store the headings
        all_headings[[i]] <- colnames[i]

        # make the histogram for one field
        histo <- df %>% select(paste(colnames[i])) %>% group_by_at(colnames[i]) %>%
            count %>% arrange(desc(n)) %>% head(3)

        output <- ""

        # then order it and concatente the info on a single line wiht
        # % and #
        for (j in 1:nrow(histo)) {

            value <- histo[[j, colnames[i]]]
            value_count <- histo[[j, "n"]]
            value_count_pct <- round(histo[j, "n"]/rows, 2) *
                100


            null_pct[[i]] <- df %>% select(paste(colnames[i])) %>%
                is.na %>% sum() * 1/rows %>% round(2) * 100

            # assemble the string output
            output <- paste0(output, "(", value_count_pct, "%, --",
                value, "--,", value_count, "# )", sep = " ",
                collapse = " ")
        }

        # add the completed string
        all_outputs[[i]] <- output
    }
    return(list(all_headings, all_outputs, null_pct))
}

clean_stats <- get_df_stats(df_clean)

# create a data table summary
```

```
summary <- data.frame(headings = unlist(clean_stats[[1]]), top_hist_values = unlist(clean_stats[[2]]),
    null_pcts = unlist(clean_stats[[3]]))
# format for print
summary %>% arrange(desc(top_hist_values)) %>% head(30) %>% kable
```

headings	top_hist_values	null_pcts
Condition2	(99%, –Norm–,1341#) (0%, –Feedr–,3#) (0%, –Artery–,2#)	0
RoofMatl	(99%, –250–,1333#) (1%, –125–,8#) (1%, –100–,7#)	0
LowQualFinSF	(99%, –0–,1332#) (0%, –80–,3#) (0%, –360–,2#)	0
X3SsnPorch	(99%, –0–,1331#) (0%, –168–,2#) (0%, –180–,2#)	0
Heating	(98%, –GasA–,1324#) (1%, –GasW–,15#) (0%, –Wall–,4#)	0
MiscVal	(97%, –0–,1306#) (1%, –400–,11#) (1%, –500–,7#)	0
KitchenAbvGr	(96%, –1–,1294#) (4%, –2–,54#) (0%, –0–,1#)	0
CentralAir	(95%, –Y–,1281#) (5%, –N–,68#)	0
LandSlope	(95%, –Gtl–,1286#) (4%, –Mod–,57#) (0%, –Sev–,6#)	0
BsmtHalfBath	(94%, –0–,1271#) (6%, –1–,76#) (0%, –2–,2#)	0
PavedDrive	(93%, –Y–,1253#) (5%, –N–,71#) (2%, –P–,25#)	0
Functional	(93%, –1–,1261#) (2%, –0.7–,31#) (2%, –0.85–,30#)	0
Electrical	(92%, –SBrkr–,1245#) (6%, –FuseA–,79#) (2%, –FuseF–,21#)	0
ScreenPorch	(92%, –0–,1240#) (0%, –192–,6#) (0%, –120–,5#)	0
LandContour	(91%, –Lvl–,1231#) (4%, –Bnk–,53#) (3%, –HLS–,39#)	0
GarageQual	(91%, –5–,1231#) (4%, –0–,60#) (3%, –3–,42#)	0
BsmtCond	(90%, –5–,1220#) (4%, –7–,60#) (3%, –3–,36#)	0
BsmtFinType2	(89%, –0–,1195#) (5%, –7–,66#) (3%, –1–,43#)	0
BsmtFinSF2	(89%, –0–,1194#) (0%, –180–,5#) (0%, –374–,3#)	0
SaleType	(88%, –WD–,1186#) (8%, –New–,112#) (3%, –COD–,38#)	0
ExterCond	(88%, –5–,1192#) (10%, –7–,136#) (1%, –3–,17#)	0
Condition1	(87%, –Norm–,1175#) (5%, –Feedr–,73#) (3%, –Artery–,43#)	0
EnclosedPorch	(86%, –0–,1166#) (1%, –112–,14#) (0%, –96–,5#)	0
MSZoning	(80%, –RL–,1077#) (15%, –RM–,196#) (5%, –FV–,62#)	0
BsmtUnfSF	(8%, –0–,102#) (1%, –728–,9#) (1%, –384–,8#)	0
RoofStyle	(79%, –Gable–,1064#) (20%, –Hip–,264#) (1%, –Gambrel–,9#)	0
LotConfig	(73%, –Inside–,980#) (18%, –Corner–,239#) (6%, –CulDSac–,82#)	0
BsmtExposure	(66%, –1–,890#) (15%, –7–,204#) (9%, –10–,116#)	0
LotShape	(64%, –Reg–,863#) (33%, –IR1–,447#) (3%, –IR2–,34#)	0
ExterQual	(62%, –5–,836#) (34%, –7–,461#) (3%, –9–,46#)	0

OLS: Linear Model Building

Load clean data

Will clean the dataframe with all the same functions outlined in the EDA section.

```
df <- read.csv("~/Documents/MSAN/04-601-LinRegression/Project/housing.txt",
    stringsAsFactors = FALSE)

df_clean <- df %>% Years_to_Age %>% SwapToFactor %>% DealWithNulls %>%
    QualityToNumeric %>% ImputeValues %>% RoofMatlToDollars %>%
    RemoveCollinear %>% # ExcludeFeatures_90Plus %>% # this was to improve the score
AreaToLogArea %>% AreaToLogAreaExcluded90 %>% Exclude_Outliers
```

Define the X by including all fields

Define our X features by all the fields, except the response variable

```
X_OLS_all <- df_clean %>% select(-SalePrice, -Id)
```

As discussed in EDA, will set our target Response Y to Log(SalePrice)

```
y_OLS <- df_clean %>% select(SalePrice) %>% unlist %>% as.numeric
y_OLS_log <- log(y_OLS)
```

OLS: Fit the OLS Model - All Fields (After EDA processing)

```
# fit model
fit_model <- lm(y_OLS_log ~ ., X_OLS_all)

# interpolate the values y_hat
y_hat_log <- predict(lm(y_OLS_log ~ ., X_OLS_all), X_OLS_all,
    se.fit = TRUE)[[1]]
```

R-squared and MSE

```
print(summary(fit_model)$r.squared)
```

```
## [1] 0.9619267
```

```
# calculate MSE
Log_MSE <- mean((y_hat_log - y_OLS_log)^2)
print(Log_MSE)
```

```
## [1] 0.005234656
```

```
MSE <- mean((exp(y_hat_log) - exp(y_OLS_log))^2)
print(MSE)
```

```
## [1] 237269622
```

```r
print(sqrt(MSE))
```

```
## [1] 15403.56
```

Plot Y_predicted by Y_hat

```r
# plot the result of y_hat vs. y_orig
p1 <- data.frame(y_hat = y_hat_log, y_OLS = y_OLS_log) %>% ggplot(aes(x = y_hat,
    y = y_OLS)) + geom_point() + geom_point(aes(x = y_hat, y = y_hat),
    color = "red") + labs(x = "Y Predicted (Log Sale Price)",
    y = "Y True (Log Sale Price)", title = "OLS Linear Regression of Log Sales Price")

p2 <- data.frame(y_hat = exp(y_hat_log), y_OLS = exp(y_OLS_log)) %>%
    ggplot(aes(x = y_hat, y = y_OLS)) + geom_point() + geom_point(aes(x = y_hat,
    y = y_hat), color = "red") + labs(x = "Y Predicted (Sale Price)",
    y = "Y True (Sale Price)", title = "OLS Linear Regression of Sales Price")

multiplot(p1, p2, cols = 2)
```

OLS: Normality of Errors

Plot: Residuals vs. Response

```
residuals <- y_hat_log - y_OLS_log

p1 <- data.frame(y_OLS_log, residuals) %>% ggplot(aes(x = y_OLS_log,
    y = residuals)) + geom_point() + labs(x = "Log Sale Price",
    y = "Residuals (y_hat - y_true)", title = "Residuals Scatter Plot (of the Logs)")

p2 <- data.frame(y_OLS = exp(y_hat_log), residuals = (exp(y_hat_log) -
    exp(y_OLS_log))) %>% ggplot(aes(x = y_OLS, y = residuals)) +
    geom_point() + labs(x = "Sale Price", y = "Residuals (y_hat - y_true)",
    title = "Residuals Scatter Plot (Non-Transformed)")

multiplot(p1, p2, cols = 2)
```

Plot: Distribution of the Residuals (Normalized)

```
residuals_norm <- (residuals - mean(residuals))/sd(residuals)
data.frame(residuals_norm = residuals_norm) %>% ggplot(aes(x = residuals_norm)) +
    geom_histogram(bins = 40) + labs(x = "residuals (Normed)",
    y = "Count", title = "Distribution of Residuals (y_hat - y_true)")
```

K-S Test of the Residuals

```
n_resid <- length(residuals_norm)
std_norm <- rnorm(n = n_resid, mean = 0, sd = 1)

print(ks.test(residuals_norm, std_norm))
```

```
## Warning in ks.test(residuals_norm, std_norm): p-value will be approximate
## in the presence of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  residuals_norm and std_norm
## D = 0.044477, p-value = 0.1386
## alternative hypothesis: two-sided
```

OLS: Significant Factors

Signficant Features Pr (<0.05), then ordered by Beta Magnitude

```
OLS_results_df <- data.frame(summary(fit_model)$coefficients)
colnames(OLS_results_df) <- c("Beta_est", "StdErr", "t_val",
    "Pr")
```

```
OLS_results_df$names <- labels(OLS_results_df)[[1]]
OLS_results_df %>% filter(Pr < 0.05) %>% arrange(desc(abs(Beta_est))) %>%
    mutate(ct = rownames(.)) %>% kable(digits = c(3, 3, 3, 3))
```

Beta_est	StdErr	t_val	Pr	names	ct
5.679	0.213	26.703	0.000	(Intercept)	1
0.410	0.041	10.030	0.000	GrLivArea	2
0.319	0.064	4.992	0.000	MSZoningFV	3
0.314	0.060	5.229	0.000	MSZoningRL	4
0.311	0.065	4.797	0.000	MSZoningRH	5
0.286	0.059	4.852	0.000	MSZoningRM	6
0.285	0.095	3.008	0.003	GarageTypeNone	7
0.260	0.109	2.391	0.017	Condition2PosA	8
0.228	0.028	8.057	0.000	Functional	9
-0.170	0.085	-1.997	0.046	FoundationWood	10
0.129	0.031	4.201	0.000	MasVnrTypeNone	11
-0.120	0.056	-2.130	0.033	LotConfigFR3	12
0.117	0.036	3.258	0.001	GarageTypeDetchd	13
0.115	0.026	4.365	0.000	Condition1PosN	14
0.109	0.009	11.747	0.000	LotArea	15
0.106	0.038	2.819	0.005	GarageTypeBuiltIn	16
0.104	0.043	2.435	0.015	GarageTypeBasment	17
0.101	0.037	2.727	0.006	SaleTypeConLD	18
0.098	0.029	3.440	0.001	NeighborhoodStoneBr	19
0.092	0.014	6.424	0.000	Condition1Norm	20
0.092	0.036	2.553	0.011	GarageTypeAttchd	21
-0.091	0.037	-2.482	0.013	LandSlopeSev	22
0.085	0.024	3.600	0.000	MasVnrTypeStone	23
-0.081	0.028	-2.917	0.004	NeighborhoodEdwards	24
0.081	0.030	2.739	0.006	NeighborhoodCrawfor	25
0.077	0.023	3.298	0.001	Condition1RRAn	26
-0.076	0.026	-2.980	0.003	MSSubClass2LVL_PUD>1946	27
0.073	0.022	3.272	0.001	MasVnrTypeBrkFace	28
0.069	0.017	4.123	0.000	SaleTypeNew	29
-0.066	0.028	-2.366	0.018	NeighborhoodNWAmes	30
0.063	0.026	2.440	0.015	NeighborhoodNridgHt	31
-0.059	0.023	-2.521	0.012	MSSubClassDUPLEX	32
-0.058	0.022	-2.606	0.009	KitchenAbvGr	33
0.051	0.017	2.996	0.003	Condition1Feedr	34
-0.048	0.013	-3.629	0.000	MSSubClassSPLIT	35
-0.048	0.018	-2.694	0.007	MSSubClass2LVL>1946	36
-0.047	0.013	-3.514	0.000	LotConfigFR2	37
0.044	0.003	14.505	0.000	OverallCond	38
0.042	0.003	12.327	0.000	OverallQual	39
-0.041	0.017	-2.489	0.013	MSSubClass1LVL<1945	40
0.038	0.018	2.096	0.036	LandContourHLS	41
0.034	0.008	4.223	0.000	GarageCars	42
0.033	0.014	2.365	0.018	SaleTypeWD	43
0.032	0.006	5.229	0.000	BsmtFullBath	44
0.029	0.008	3.801	0.000	FullBath	45
0.028	0.012	2.343	0.019	GarageFinish	46
0.027	0.007	3.767	0.000	HalfBath	47
0.027	0.012	2.251	0.025	FoundationPConc	48

Beta_est	StdErr	t_val	Pr	names	ct
0.026	0.012	2.130	0.033	PavedDriveY	49
0.019	0.005	4.173	0.000	Fireplaces	50
0.016	0.006	2.866	0.004	LotShapeReg	51
-0.013	0.005	-2.750	0.006	BedroomAbvGr	52
0.013	0.006	2.276	0.023	GarageQual	53
0.013	0.004	3.073	0.002	MasVnrArea	54
0.011	0.003	3.939	0.000	BsmtQual	55
0.011	0.003	3.895	0.000	KitchenQual	56
-0.010	0.004	-2.324	0.020	LowQualFinSF	57
-0.008	0.004	-2.291	0.022	ExterCond	58
0.007	0.002	3.985	0.000	BsmtFinSF1	59
0.007	0.003	2.199	0.028	TotRmsAbvGrd	60
0.005	0.001	5.556	0.000	BsmtExposure	61
0.005	0.002	3.234	0.001	HeatingQC	62
0.005	0.002	2.979	0.003	ScreenPorch	63
0.003	0.002	1.971	0.049	EnclosedPorch	64
-0.002	0.000	-7.833	0.000	YearBuilt	65
0.000	0.000	-2.232	0.026	YearRemodAdd	66

Discussion of Dominant OLS Factors:

The following section will look at some of the dominant features found in the OLS analysis and will do plots vs. Sale Price to verify the strong signal

Strong Beta: Comparison of Price vs. Zoning

- A Agriculture
- C Commercial
- FV Floating Village Residential
- I Industrial
- RH Residential High Density
- RL Residential Low Density
- RP Residential Low Density Park
- RM Residential Medium Density

```
p1 <- df_clean %>% select(SalePrice, MSZoning) %>% ggplot(aes(x = MSZoning,
    y = SalePrice)) + geom_violin() + labs(title = "Sale Price vs. MSZone")

p2 <- df_clean %>% select(SalePrice, MSZoning) %>% ggplot(aes(x = MSZoning,
    y = log(SalePrice))) + geom_violin() + labs(title = "Log Sale Price vs. MSZone")

multiplot(p1, p2, cols = 2)
```

Strong Beta: Comparison of Price Garage Type

- 2Types More than one type of garage
- Attchd Attached to home
- Basment Basement Garage
- BuiltIn Built-In (Garage part of house - typically has room above garage)
- CarPort Car Port
- Detchd Detached from home
- NA No Garage

```
p1 <- df_clean %>% select(SalePrice, GarageType) %>% ggplot(aes(x = GarageType,
    y = SalePrice)) + geom_violin() + labs(title = "Sale Price vs. MSZone")

p2 <- df_clean %>% select(SalePrice, GarageType) %>% ggplot(aes(x = GarageType,
    y = log(SalePrice))) + geom_violin() + labs(title = "Log Sale Price vs. MSZone")

multiplot(p1, p2, cols = 2)
```

Strong Beta: Lot Configuration

- Inside Inside lot
- Corner Corner lot
- CulDSac Cul-de-sac
- FR2 Frontage on 2 sides of property

- FR3 Frontage on 3 sides of property

```
p1 <- df_clean %>% select(LotConfig, SalePrice) %>% ggplot(aes(x = LotConfig,
    y = SalePrice)) + geom_violin() + labs(title = "Sale Price vs. LotConfig")

p2 <- df_clean %>% select(SalePrice, LotConfig) %>% ggplot(aes(x = LotConfig,
    y = log(SalePrice))) + geom_violin() + labs(title = "Log Sale Price vs. LotConfig")

multiplot(p1, p2, cols = 2)
```

Comparison of General Living Area vs. Price

```
p1 <- df_clean %>% select(SalePrice, GrLivArea) %>% ggplot(aes(x = GrLivArea,
    y = SalePrice)) + geom_point() + labs(title = "Sale Price vs. General Living Area")

p2 <- df_clean %>% select(SalePrice, GrLivArea) %>% ggplot(aes(x = log(GrLivArea),
    y = log(SalePrice))) + geom_point() + labs(title = "Log Sale Price vs. General Living Area")

multiplot(p1, p2, cols = 2)
```

Comparison of Lot Area vs. Price

```
p1 <- df_clean %>% select(SalePrice, LotArea) %>% ggplot(aes(x = LotArea,
    y = SalePrice)) + geom_point() + labs(title = "Sale Price vs. General Living Area")

p2 <- df_clean %>% select(SalePrice, LotArea) %>% ggplot(aes(x = log(LotArea),
    y = log(SalePrice))) + geom_point() + labs(title = "Log Sale Price vs. Lot Area")

multiplot(p1, p2, cols = 2)
```

Comparison of Overall Quality vs. Price

```
p1 <- df_clean %>% select(SalePrice, OverallQual) %>% ggplot(aes(x = OverallQual,
    y = SalePrice)) + geom_jitter() + labs(title = "Sale Price vs. Overall Quality")

p2 <- df_clean %>% select(SalePrice, OverallQual) %>% ggplot(aes(x = OverallQual,
    y = log(SalePrice))) + geom_jitter() + labs(title = "Sale Price vs. Overall Quality")

multiplot(p1, p2, cols = 2)
```

```
df_clean %>% select(SalePrice, LotArea, GrLivArea) %>% ggplot(aes(x = GrLivArea,
    y = LotArea, size = SalePrice)) + geom_hline(yintercept = mean(df_clean$LotArea),
    color = "red") + geom_vline(xintercept = mean(df_clean$GrLivArea),
    color = "red") + geom_point() + labs(title = "General Living Area vs. LotArea, Sale Price = Size")
```

OLS: Cosine Similiarity of Morty to Other Houses

```
# Bind together
y_OLS_hat <- predict(fit_model, X_OLS_all)

# Convert to dummy variables + model matrix
proxy_matrix <- model.matrix(SalePrice ~ ., df_clean %>% select(-Id))

# Pull morty's vector out
target_vec <- proxy_matrix[5, ]

# The remaining houses will be compared to
compar_vecs <- proxy_matrix
rows <- nrow(compar_vecs)
```

Loop through and compare against all other houses

```
# Calculate all teh scores
scores <- list()
for (i in 1:rows) {
    # scores[[i]] <- cosine.similarity(target_vec,
    # compar_vecs[i,], .do.norm=T)
    scores[[i]] <- target_vec %*% compar_vecs[i, ]/norm(target_vec,
        type = "2")/norm(compar_vecs[i, ], type = "2")
}
```

Rebuild the data frame with predicted prices + morty's house. Sort by cosine similarity

```
x_clean$Id <- 9999
x_clean$SalePrice <- 143000

close_5 <- df_clean %>% mutate(OLS_price = exp(y_OLS_hat)) %>%
    mutate(proxy_score = unlist(scores)) %>% bind_rows(x_clean) %>%
    arrange(desc(proxy_score)) %>% head(6)
```

Show Similar Houses

```
t(close_5) %>% kable
```

	1	2	3	4	5	6
Id	6	445	1085	1111	1323	140
MSSubClass	1.5LVL_FIN	2LVL>1946	2LVL>1946	2LVL>1946	2LVL>1946	2LVL>1946
MSZoning	RL	RL	RL	RL	RL	RL
LotArea	9.555064	9.076923	9.475163	8.987322	9.228868	9.643875
Street	Pave	Pave	Pave	Pave	Pave	Pave
LotShape	IR1	Reg	IR2	Reg	IR1	IR1
LandContour	Lvl	Lvl	Lvl	Lvl	Lvl	Lvl
LotConfig	Inside	Inside	Corner	Inside	Inside	Inside

	1	2	3	4	5	6
LandSlope	Gtl	Gtl	Gtl	Gtl	Gtl	Gtl
Neighborhood	Mitchel	CollgCr	Gilbert	Gilbert	NoRidge	CollgCr
Condition1	Norm	Norm	Norm	Norm	Norm	Norm
Condition2	Norm	Norm	Norm	Norm	Norm	Norm
OverallQual	5	7	6	6	7	6
OverallCond	5	5	5	5	5	5
YearBuilt	17	16	15	15	18	13
YearRemodAdd	15	15	14	14	18	13
RoofStyle	Gable	Gable	Gable	Gable	Gable	Gable
RoofMatl	250	250	250	250	250	250
MasVnrType	None	None	None	None	None	None
MasVnrArea	0	0	0	0	0	0
ExterQual	5	7	5	5	7	5
ExterCond	5	7	5	5	5	5
Foundation	Wood	PConc	PConc	PConc	PConc	PConc
BsmtQual	7	7	7	7	7	7
BsmtCond	5	5	5	5	5	5
BsmtExposure	1	1	1	1	1	1
BsmtFinType1	10	10	7	10	10	10
BsmtFinSF1	6.597146	6.466145	6.385194	5.393628	6.514713	6.711740
BsmtFinType2	0	0	0	0	0	0
BsmtFinSF2	0	0	0	0	0	0
BsmtUnfSF	4.174387	5.613128	4.605170	6.318968	4.343805	4.682131
Heating	GasA	GasA	GasA	GasA	GasA	GasA
HeatingQC	9	9	7	7	9	9
CentralAir	Y	Y	Y	Y	Y	Y
Electrical	SBrkr	SBrkr	SBrkr	SBrkr	SBrkr	SBrkr
X1stFlrSF	6.680855	6.839476	6.539586	6.651572	6.967909	6.834109
X2ndFlrSF	6.340359	6.883463	6.694562	6.786717	6.760415	6.729824
LowQualFinSF	0	0	0	0	0	0
GrLivArea	7.217443	7.554335	7.312553	7.413970	7.562162	7.475906
BsmtFullBath	1	1	0	1	1	1
BsmtHalfBath	0	0	0	0	0	0
FullBath	1	2	2	2	2	2
HalfBath	1	1	1	1	1	1
BedroomAbvGr	1	4	3	3	3	3
KitchenAbvGr	1	1	1	1	1	1
KitchenQual	5	7	5	5	7	7
TotRmsAbvGrd	5	8	6	8	8	7
Functional	1	1	1	1	1	1
Fireplaces	0	1	1	1	1	0
GarageType	Attchd	Attchd	Attchd	Attchd	Attchd	Attchd
GarageYrBlt	17	16	15	15	18	13
GarageFinish	0.33	0.33	1.00	1.00	0.66	0.66
GarageCars	2	2	2	2	2	2
GarageArea	6.175867	6.202536	6.016157	6.068426	6.336826	6.154858
GarageQual	5	5	5	5	5	5
PavedDrive	Y	Y	Y	Y	Y	Y
WoodDeckSF	3.713572	4.976734	5.755742	5.416100	5.484797	5.624018
OpenPorchSF	3.433987	4.897840	3.806662	4.442651	3.688879	4.605170
EnclosedPorch	0	0	0	0	0	0
X3SsnPorch	5.771441	0.000000	0.000000	0.000000	0.000000	0.000000
ScreenPorch	0	0	0	0	0	0
PoolArea	0	0	0	0	0	0
MiscVal	6.552508	0.000000	0.000000	0.000000	0.000000	0.000000
MoSold	10	7	7	6	6	8
YrSold	1	2	4	2	0	1

	1	2	3	4	5	6
SaleType	WD	WD	WD	WD	WD	WD
SalePrice	143000	210000	187500	188000	190000	231500
OLS_price	143000.0	211193.5	174310.5	184109.1	242765.4	207599.8
proxy_score	1.0000000	0.9989289	0.9989237	0.9989091	0.9989052	0.9988966

OLS: Explanatory Modeling: Part II

We are given a new set of features, Morty's house. We now clean this in the same way to put into our model:

```
# Make sure this pipeline is still up to date
x_new <- read.csv("Morty.txt", stringsAsFactors = F) %>% select(-X)  # Make sure pathname here is right

x_clean <- x_new %>% Years_to_Age %>% DealWithNulls %>% SwapToFactor %>%
    QualityToNumeric %>% RoofMatlToDollars %>% RemoveCollinear %>%
    # ExcludeFeatures_90Plus %>% # this was to improve the score
AreaToLogArea %>% AreaToLogAreaExcluded90 %>% select(-Id, -SalePrice)
```

Confidence interval of Morty's House Price

Our first task is to predict the value, so let's explore the 95% CI given by our model:

```
fit_model <- lm(y_OLS_log ~ ., df_clean %>% select(-Id, -SalePrice))
log_y <- predict(fit_model, x_clean, interval = "predict", level = 0.95)
(y <- exp(log_y))
```

```
##      fit      lwr      upr
## 1 143000 115553.5 176965.6
```

We find a fitted value of $ 143000, with a 95% CI of ($ 115553.5252088, $ 176965.6093404). As such, we conclude the maximise value we can reasonably expect the house to sell for is $ 176965.6093404.

Note that this analysis relies solely on unbiased OLS estimators. Our predicted range is wide, but the needed for an unbiased estimator in order to construct a CI necessitates the use of OLS for our estimation.

Key Factors

Assessing key factors to change:

First, let's just have a look at the coefficients we've got in our model:

```
summary(fit_model)
```

```
##
## Call:
## lm(formula = y_OLS_log ~ ., data = df_clean %>% select(-Id, -SalePrice))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29186 -0.04366  0.00000  0.04511  0.27851
##
## Coefficients:
##                            Estimate  Std. Error t value Pr(>|t|)
## (Intercept)              5.67902341  0.21267051  26.703  < 2e-16 ***
## MSSubClass1LVL<1945     -0.04119924  0.01655548  -2.489 0.012961 *
## MSSubClass1LVL_W_ATTIC  -0.03338649  0.08111827  -0.412 0.680722
## MSSubClass1.5LVL_UNF     0.01179347  0.02792810   0.422 0.672897
## MSSubClass1.5LVL_FIN    -0.03123179  0.01941041  -1.609 0.107875
## MSSubClass2LVL>1946     -0.04837238  0.01795817  -2.694 0.007167 **
## MSSubClass2LVL<1945     -0.01235851  0.02329197  -0.531 0.595801
## MSSubClass2.5LVL        -0.03492087  0.03447508  -1.013 0.311298
```

```
## MSSubClassSPLIT          -0.04848677  0.01336091  -3.629 0.000297 ***
## MSSubClassSPLIT_FOYER    -0.03034992  0.01936642  -1.567 0.117346
## MSSubClassDUPLEX         -0.05909032  0.02343506  -2.521 0.011816 *
## MSSubClass1LVL_PUD>1946 -0.02537820  0.01396897  -1.817 0.069505 .
## MSSubClass2LVL_PUD>1946 -0.07610030  0.02553739  -2.980 0.002941 **
## MSSubClassMULTI_PUD      -0.02237921  0.03658368  -0.612 0.540835
## MSSubClass2FAM_CONV      -0.04671662  0.02423812  -1.927 0.054166 .
## MSZoningFV                0.31913326  0.06392616   4.992 6.85e-07 ***
## MSZoningRH                0.31102212  0.06483955   4.797 1.82e-06 ***
## MSZoningRL                0.31447316  0.06013932   5.229 2.01e-07 ***
## MSZoningRM                0.28594488  0.05893397   4.852 1.38e-06 ***
## LotArea                   0.10885687  0.00926644  11.747  < 2e-16 ***
## StreetPave                0.04161926  0.05176931   0.804 0.421593
## LotShapeIR2              -0.00650045  0.01495241  -0.435 0.663827
## LotShapeIR3              -0.03268140  0.03356523  -0.974 0.330418
## LotShapeReg               0.01589216  0.00554415   2.866 0.004223 **
## LandContourHLS            0.03830265  0.01827768   2.096 0.036328 *
## LandContourLow            0.01805648  0.02282305   0.791 0.429013
## LandContourLvl            0.01610404  0.01293584   1.245 0.213406
## LotConfigCulDSac          0.02119763  0.01088290   1.948 0.051673 .
## LotConfigFR2             -0.04707823  0.01339635  -3.514 0.000457 ***
## LotConfigFR3             -0.12009952  0.05639034  -2.130 0.033393 *
## LotConfigInside          -0.00117977  0.00600441  -0.196 0.844265
## LandSlopeMod             -0.00378402  0.01388662  -0.272 0.785289
## LandSlopeSev             -0.09113569  0.03671167  -2.482 0.013184 *
## NeighborhoodBlueste       0.07234821  0.06523233   1.109 0.267616
## NeighborhoodBrDale       -0.00607910  0.03871356  -0.157 0.875249
## NeighborhoodBrkSide       0.01911914  0.03231639   0.592 0.554215
## NeighborhoodClearCr       0.03125937  0.03165915   0.987 0.323660
## NeighborhoodCollgCr      -0.03920017  0.02543934  -1.541 0.123599
## NeighborhoodCrawfor       0.08131756  0.02969217   2.739 0.006260 **
## NeighborhoodEdwards      -0.08141975  0.02790738  -2.917 0.003594 **
## NeighborhoodGilbert      -0.05157692  0.02686584  -1.920 0.055121 .
## NeighborhoodIDOTRR       -0.02197368  0.03704527  -0.593 0.553188
## NeighborhoodMeadowV      -0.06256726  0.03856161  -1.623 0.104954
## NeighborhoodMitchel      -0.03618480  0.02841163  -1.274 0.203056
## NeighborhoodNAmes        -0.03997573  0.02702821  -1.479 0.139393
## NeighborhoodNoRidge       0.05068939  0.02875786   1.763 0.078219 .
## NeighborhoodNPkVill       0.04001323  0.03680881   1.087 0.277231
## NeighborhoodNridgHt       0.06320767  0.02590267   2.440 0.014823 *
## NeighborhoodNWAmes       -0.06563587  0.02774185  -2.366 0.018142 *
## NeighborhoodOldTown      -0.01461589  0.03285742  -0.445 0.656525
## NeighborhoodSawyer       -0.01241031  0.02809059  -0.442 0.658717
## NeighborhoodSawyerW      -0.04917205  0.02683757  -1.832 0.067169 .
## NeighborhoodSomerst       0.03493329  0.03133611   1.115 0.265163
## NeighborhoodStoneBr       0.09833283  0.02858215   3.440 0.000601 ***
## NeighborhoodSWISU        -0.04224439  0.03298511  -1.281 0.200543
## NeighborhoodTimber       -0.04222390  0.02818558  -1.498 0.134379
## NeighborhoodVeenker       0.01089361  0.03586405   0.304 0.761373
## Condition1Feedr           0.05084376  0.01696994   2.996 0.002791 **
## Condition1Norm            0.09206722  0.01433101   6.424 1.91e-10 ***
## Condition1PosA            0.07717707  0.04001357   1.929 0.053995 .
## Condition1PosN            0.11499230  0.02634352   4.365 1.38e-05 ***
## Condition1RRAe           -0.03954707  0.03088068  -1.281 0.200568
## Condition1RRAn            0.07719564  0.02341028   3.298 0.001004 **
## Condition1RRNn            0.09168876  0.04971551   1.844 0.065391 .
## Condition2Feedr          -0.03231131  0.08393330  -0.385 0.700332
## Condition2Norm            0.01915507  0.06750399   0.284 0.776642
## Condition2PosA            0.25975096  0.10863798   2.391 0.016957 *
```

```
## Condition2RRAe          -0.17552103  0.16463057  -1.066 0.286570
## Condition2RRAn           0.00484233  0.10410645   0.047 0.962909
## Condition2RRNn           0.05598606  0.08988389   0.623 0.533488
## OverallQual              0.04213968  0.00341835  12.327  < 2e-16 ***
## OverallCond              0.04357824  0.00300434  14.505  < 2e-16 ***
## YearBuilt               -0.00216049  0.00027581  -7.833 1.04e-14 ***
## YearRemodAdd            -0.00042351  0.00018972  -2.232 0.025783 *
## RoofStyleGable           0.04155838  0.04857001   0.856 0.392369
## RoofStyleGambrel         0.02962773  0.05607937   0.528 0.597377
## RoofStyleHip             0.04850031  0.04884680   0.993 0.320956
## RoofStyleMansard         0.03330708  0.05892509   0.565 0.572013
## RoofStyleShed            0.10553040  0.12096336   0.872 0.383156
## RoofMatl                -0.00026578  0.00022092  -1.203 0.229191
## MasVnrTypeBrkFace        0.07307967  0.02233652   3.272 0.001099 **
## MasVnrTypeNone           0.12946978  0.03081934   4.201 2.86e-05 ***
## MasVnrTypeStone          0.08528154  0.02369227   3.600 0.000332 ***
## MasVnrArea               0.01268769  0.00412810   3.073 0.002164 **
## ExterQual                0.00586576  0.00357404   1.641 0.101016
## ExterCond               -0.00826817  0.00360903  -2.291 0.022138 *
## FoundationCBlock         0.00437485  0.01126638   0.388 0.697855
## FoundationPConc          0.02684919  0.01192644   2.251 0.024552 *
## FoundationSlab           0.03890486  0.03001520   1.296 0.195166
## FoundationStone         -0.01594753  0.05356000  -0.298 0.765945
## FoundationWood          -0.16985121  0.08507329  -1.997 0.046103 *
## BsmtQual                 0.01143684  0.00290329   3.939 8.65e-05 ***
## BsmtCond                 0.00446329  0.00380382   1.173 0.240880
## BsmtExposure             0.00535726  0.00096421   5.556 3.40e-08 ***
## BsmtFinType1             0.00097898  0.00114765   0.853 0.393812
## BsmtFinSF1               0.00693467  0.00174017   3.985 7.15e-05 ***
## BsmtFinType2            -0.00017986  0.00238049  -0.076 0.939785
## BsmtFinSF2              -0.00015072  0.00250440  -0.060 0.952022
## BsmtUnfSF               -0.00240806  0.00180325  -1.335 0.182000
## HeatingGasA              0.09005742  0.08294389   1.086 0.277802
## HeatingGasW              0.11629236  0.08543983   1.361 0.173738
## HeatingGrav             -0.02987580  0.09582311  -0.312 0.755261
## HeatingOthW             -0.00543190  0.10198524  -0.053 0.957532
## HeatingWall              0.09870332  0.09507829   1.038 0.299421
## HeatingQC                0.00517789  0.00160091   3.234 0.001253 **
## CentralAirY              0.02266100  0.01398235   1.621 0.105348
## ElectricalFuseF         -0.00368541  0.02043008  -0.180 0.856876
## ElectricalFuseP          0.00703985  0.06617633   0.106 0.915299
## ElectricalMix           -0.05699280  0.08669962  -0.657 0.511076
## ElectricalSBrkr         -0.01830787  0.01051609  -1.741 0.081951 .
## X1stFlrSF                0.06097365  0.03427775   1.779 0.075524 .
## X2ndFlrSF               -0.00181642  0.00391094  -0.464 0.642413
## LowQualFinSF            -0.00956838  0.00411756  -2.324 0.020303 *
## GrLivArea                0.40995150  0.04087363  10.030  < 2e-16 ***
## BsmtFullBath             0.03210653  0.00614043   5.229 2.01e-07 ***
## BsmtHalfBath             0.00782713  0.00991822   0.789 0.430171
## FullBath                 0.02873142  0.00755988   3.801 0.000152 ***
## HalfBath                 0.02696045  0.00715636   3.767 0.000173 ***
## BedroomAbvGr            -0.01334023  0.00485076  -2.750 0.006047 **
## KitchenAbvGr            -0.05779348  0.02217960  -2.606 0.009282 **
## KitchenQual              0.01076957  0.00276510   3.895 0.000104 ***
## TotRmsAbvGrd             0.00692299  0.00314844   2.199 0.028078 *
## Functional               0.22843799  0.02835363   8.057 1.88e-15 ***
## Fireplaces               0.01892897  0.00453630   4.173 3.23e-05 ***
## GarageTypeAttchd         0.09194351  0.03602070   2.553 0.010818 *
## GarageTypeBasment        0.10360796  0.04254792   2.435 0.015033 *
```

```
## GarageTypeBuiltIn        0.10591665  0.03756792   2.819 0.004892 **
## GarageTypeCarPort        0.07074460  0.05603556   1.262 0.207017
## GarageTypeDetchd         0.11684672  0.03586237   3.258 0.001153 **
## GarageTypeNone           0.28489544  0.09471635   3.008 0.002686 **
## GarageYrBlt              0.00028401  0.00022229   1.278 0.201630
## GarageFinish             0.02812405  0.01200443   2.343 0.019302 *
## GarageCars               0.03429219  0.00812126   4.223 2.60e-05 ***
## GarageArea               0.01672145  0.01421441   1.176 0.239680
## GarageQual               0.01327432  0.00583120   2.276 0.022996 *
## PavedDriveP              0.00209184  0.01920773   0.109 0.913295
## PavedDriveY              0.02630023  0.01234712   2.130 0.033369 *
## WoodDeckSF               0.00159039  0.00098837   1.609 0.107857
## OpenPorchSF              0.00172693  0.00129953   1.329 0.184140
## EnclosedPorch            0.00306299  0.00155403   1.971 0.048955 *
## X3SsnPorch               0.00359377  0.00382751   0.939 0.347956
## ScreenPorch              0.00496082  0.00166547   2.979 0.002954 **
## PoolArea                 0.00254878  0.00781816   0.326 0.744475
## MiscVal                  0.00131740  0.00196576   0.670 0.502877
## MoSold                   0.00007815  0.00084247   0.093 0.926103
## YrSold                   0.00044743  0.00172032   0.260 0.794842
## SaleTypeConLD            0.10110486  0.03708099   2.727 0.006493 **
## SaleTypeConLI           -0.00687841  0.05949725  -0.116 0.907982
## SaleTypeConLw            0.02463696  0.04880607   0.505 0.613797
## SaleTypeCWD             -0.01195777  0.08145538  -0.147 0.883313
## SaleTypeNew              0.06918022  0.01678048   4.123 4.00e-05 ***
## SaleTypeWD               0.03293800  0.01392670   2.365 0.018184 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07681 on 1197 degrees of freedom
## Multiple R-squared:  0.9619, Adjusted R-squared:  0.9571
## F-statistic: 200.3 on 151 and 1197 DF,  p-value: < 2.2e-16
```

Now let's make some ground rules for how we're going to search:
- A lot of features will be excluded immediately because they're not under our control (e.g. district of the house)
- Our OLS analysis deals with correlation, and so we have no information as to causation. As such, we will try to safeguard our suggestions with the highest of significance levels and cross check this with common sense.
- For the reason above, we'll look for fields that will add maximum value for minimum expense, to avoid unnecessary costs for Morty.

Given that we're most concerned by the significance of our coefficients, let's rank by the SL and exclude any not significant at 95% level:

```
significant_fields <- NULL
coeffs <- NULL
SL <- NULL
for (i in 1:135) {
    if (summary(fit_model)$coefficients[i, 4] < 0.05) {
        significant_fields <- c(significant_fields, row.names(summary(fit_model)$coefficients)[i])
        coeffs <- c(coeffs, summary(fit_model)$coefficients[i,
            1])
        SL <- c(SL, summary(fit_model)$coefficients[i, 4])
    }
}
(sig_fields_df <- data.frame(field = significant_fields, beta = coeffs,
    SL = SL) %>% arrange(SL))
```

```
##                     field          beta            SL
## 1             (Intercept)  5.6790234100 1.280471e-123
## 2             OverallCond  0.0435782388  4.843863e-44
## 3             OverallQual  0.0421396835  5.857323e-33
## 4                 LotArea  0.1088568681  3.096875e-30
## 5               GrLivArea  0.4099515024  8.696371e-23
## 6              Functional  0.2284379903  1.880778e-15
## 7               YearBuilt -0.0021604880  1.043081e-14
## 8          Condition1Norm  0.0920672187  1.907061e-10
## 9            BsmtExposure  0.0053572606  3.396840e-08
## 10             MSZoningRL  0.3144731642  2.009076e-07
## 11           BsmtFullBath  0.0321065279  2.012984e-07
## 12             MSZoningFV  0.3191332622  6.852952e-07
## 13             MSZoningRM  0.2859448781  1.383676e-06
## 14             MSZoningRH  0.3110221183  1.815139e-06
## 15         Condition1PosN  0.1149923003  1.380258e-05
## 16             GarageCars  0.0342921898  2.598775e-05
## 17         MasVnrTypeNone  0.1294697828  2.855526e-05
## 18             Fireplaces  0.0189289670  3.226565e-05
## 19             BsmtFinSF1  0.0069346657  7.153871e-05
## 20               BsmtQual  0.0114368362  8.645106e-05
## 21            KitchenQual  0.0107695734  1.037041e-04
## 22               FullBath  0.0287314187  1.516610e-04
## 23               HalfBath  0.0269604470  1.730215e-04
## 24        MSSubClassSPLIT -0.0484867725  2.965178e-04
## 25        MasVnrTypeStone  0.0852815351  3.317981e-04
## 26           LotConfigFR2 -0.0470782350  4.574459e-04
## 27    NeighborhoodStoneBr  0.0983328313  6.009566e-04
## 28          Condition1RRAn  0.0771956354  1.004116e-03
## 29      MasVnrTypeBrkFace  0.0730796721  1.099302e-03
## 30       GarageTypeDetchd  0.1168467221  1.152708e-03
## 31              HeatingQC  0.0051778856  1.252575e-03
## 32             MasVnrArea  0.0126876942  2.163506e-03
## 33         GarageTypeNone  0.2848954400  2.685707e-03
## 34        Condition1Feedr  0.0508437574  2.790799e-03
## 35 MSSubClass2LVL_PUD>1946 -0.0761003029  2.941078e-03
## 36    NeighborhoodEdwards -0.0814197508  3.594395e-03
## 37            LotShapeReg  0.0158921623  4.223417e-03
## 38      GarageTypeBuiltIn  0.1059166463  4.891610e-03
## 39           BedroomAbvGr -0.0133402296  6.046816e-03
## 40    NeighborhoodCrawfor  0.0813175643  6.260039e-03
## 41    MSSubClass2LVL>1946 -0.0483723846  7.166981e-03
## 42           KitchenAbvGr -0.0577934759  9.282322e-03
## 43       GarageTypeAttchd  0.0919435052  1.081801e-02
## 44       MSSubClassDUPLEX -0.0590903245  1.181622e-02
## 45    MSSubClass1LVL<1945 -0.0411992431  1.296139e-02
## 46           LandSlopeSev -0.0911356908  1.318369e-02
## 47    NeighborhoodNridgHt  0.0632076681  1.482347e-02
## 48      GarageTypeBasment  0.1036079585  1.503341e-02
## 49         Condition2PosA  0.2597509630  1.695733e-02
## 50     NeighborhoodNWAmes -0.0656358731  1.814237e-02
## 51           GarageFinish  0.0281240505  1.930245e-02
## 52           LowQualFinSF -0.0095683847  2.030302e-02
## 53              ExterCond -0.0082681750  2.213829e-02
```

```
## 54              GarageQual  0.0132743230  2.299618e-02
## 55         FoundationPConc  0.0268491939  2.455157e-02
## 56            YearRemodAdd -0.0004235114  2.578304e-02
## 57            TotRmsAbvGrd  0.0069229934  2.807813e-02
## 58            LotConfigFR3 -0.1200995221  3.339258e-02
## 59          LandContourHLS  0.0383026532  3.632755e-02
## 60          FoundationWood -0.1698512101  4.610262e-02
```

Our process now will be to start with the most significant variables - search to find ones which we may be able to change on short notice, and then see how Morty's property compares to average values:

Evaluating categories we think we may be able to change:

1. Fireplaces

```
(Morty_Fireplaces <- x_clean$Fireplaces)
```

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

```
(avg_Fireplaces <- mean(df_clean$Fireplaces))
```

```
## [1] 0.618977
```

Morty has a below average number of fireplaces (0), so adding an electrical fireplace may give his house an added boost if appropriate.

Adding an electrical fireplace may boost the appeal of Morty's house.

2. Half-Bathrooms:

```
(Morty_HalfBath <- x_clean$HalfBath)
```

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

```
(avg_HalfBath <- mean(df_clean$HalfBath))
```

```
## [1] 0.3884359
```

Morty's already above average on this metric though, so likely isn't going to be helped by creating more half-bathrooms.

3. Garage Cars:

```
(Morty_GarageCars <- x_clean$GarageCars)
```

```
## [1] 2
```

```
(avg_GarageCars <- mean(df_clean$GarageCars))
```

```
## [1] 1.784285
```

Above average on this one too so unlikely to help by adding garage capacity.

4. Heating Quality and Condition:

```
(Morty_HeatingQC <- x_clean$HeatingQC)
```

```
## [1] 9
```

```
(avg_HeatingQC <- mean(df_clean$HeatingQC))
```

```
## [1] 7.32765
```

Likewise, Morty already has excellent heating quality and condition.

5. Kitchen Quality:

```
(Morty_Kitchen <- x_clean$KitchenQual)
```

```
## [1] 5
```

```
(avg_Kitchen <- mean(df_clean$KitchenQual))
```

```
## [1] 6.039288
```

Kitchen Quality - could be improved on to bring up to average.

6. Garage Finish:

```
(Morty_GarageFinish <- x_clean$GarageFinish)
```

```
## [1] 0.33
```

```
(avg_GarageFinish <- mean(df_clean$GarageFinish))
```

```
## [1] 0.5783173
```

Morty has below average garage finish, and we would think this is an easy thing to improve on.

OLS: Summary

These are the only variables we can spot that are significant at 95% level that can be easily adapted. As such, we only have a list of 3 which we think can reasonably be improved upon:
1. Adding electrical fireplace
2. Improving kitchen quality
3. Improving garage finish.

If, we impute the mean values for these metrics (and add a single fireplace) and recalculate the prediction interval:

```
x_update <- x_clean
x_update$KitchenQual <- mean(df_clean$KitchenQual)
x_update$GarageFinish <- mean(df_clean$GarageFinish)
x_update$Fireplaces <- 1
log_y_update <- predict(fit_model, x_update, interval = "predict",
    level = 0.95)
(y <- exp(log_y_update))
```

```
##        fit      lwr      upr
## 1 148405.7 119883.3 183714.2
```

Finally, we summarise all of the above information in one table:

```
model_params <- sig_fields_df %>% filter(field == "Fireplaces" |
    field == "KitchenQual" | field == "GarageFinish")
Morty_values <- data.frame(Morty = as.numeric(c(x_clean["Fireplaces"],
    x_clean["KitchenQual"], x_clean["GarageFinish"])))
Mean_values <- data.frame(Mean = as.numeric(c(mean(df_clean$Fireplaces),
    mean(df_clean$KitchenQual), mean(df_clean$GarageFinish))))
summary_data <- cbind(model_params, Morty_values, Mean_values)
summary_data$exp_beta <- exp(summary_data$beta)
kable(summary_data)
```

field	beta	SL	Morty	Mean	exp_beta
Fireplaces	0.0189290	0.0000323	0.00	0.6189770	1.019109
KitchenQual	0.0107696	0.0001037	5.00	6.0392884	1.010828
GarageFinish	0.0281241	0.0193024	0.33	0.5783173	1.028523

Note that the final column in this table, the exponent of the coefficient, represents the approximate factor that value would increase if we were to increment the corresponding feature value by 1 unit.

Pred: Predictive Analysis

Predictive modeling - Data Prep

```
clean <- function(df) {
    housing <- df

    pct_na <- sapply(housing, function(x) round((sum(is.na(x))/length(x)) *
        100))
    housing <- housing[, pct_na < 80]

    max_pct <- sapply(housing, function(x) round(max((table(x)/length(x)) *
        100)))
    housing <- housing[, max_pct < 90]

    housing$GarageYrBlt <- sapply(housing$GarageYrBlt, function(x) (as.integer(x)%/%10) *
        10)
    housing$GarageYrBlt <- paste0(as.character(housing$GarageYrBlt),
        "s")

    housing$GarageYrBlt[is.na(housing$GarageYrBlt)] <- "None"
    housing$GarageType[is.na(housing$GarageType)] <- "None"
    housing$GarageFinish[is.na(housing$GarageFinish)] <- "None"
    housing$MasVnrType[is.na(housing$MasVnrType)] <- "None"
    housing$BsmtQual[is.na(housing$BsmtQual)] <- "None"
    housing$BsmtExposure[is.na(housing$BsmtExposure)] <- "None"
    housing$BsmtFinType1[is.na(housing$BsmtFinType1)] <- "None"
    housing$BsmtFinType2[is.na(housing$BsmtFinType2)] <- "None"
    housing$FireplaceQu[is.na(housing$FireplaceQu)] <- "None"

    housing$MasVnrArea[is.na(housing$MasVnrArea)] <- 0
    housing$LotFrontage[is.na(housing$LotFrontage)] <- 0

    housing$MSSubClass <- factor(housing$MSSubClass, levels = c(20,
        30, 40, 45, 50, 60, 70, 75, 80, 85, 90, 120, 150, 160,
        180, 190), labels = c("1-STORY 1946 & NEWER ALL STYLES",
        "1-STORY 1945 & OLDER", "1-STORY W/FINISHED ATTIC ALL AGES",
        "1-1/2 STORY - UNFINISHED ALL AGES", "1-1/2 STORY FINISHED ALL AGES",
        "2-STORY 1946 & NEWER", "2-STORY 1945 & OLDER", "2-1/2 STORY ALL AGES",
        "SPLIT OR MULTI-LEVEL", "SPLIT FOYER", "DUPLEX - ALL STYLES AND AGES",
        "1-STORY PUD (Planned Unit Development) - 1946 & NEWER",
        "1-1/2 STORY PUD - ALL AGES", "2-STORY PUD - 1946 & NEWER",
        "PUD - MULTILEVEL - INCLSPLIT LEV/FOYER", "2 FAMILY CONVERSION - ALL STYLES AND AGES"))

    cleaner <- housing %>% select(-Id)

    return(cleaner)
}
```

Predictive modeling - Test Train Split for Model Fitting

20% of the entire data was held out for MSPE purposes. This is accomplished with the sample function that will randomly pull id's used for filtering.

```
housing <- read.csv("housing.txt", stringsAsFactors = F)

housing <- clean(housing)

train <- sample(1:nrow(housing), nrow(housing) * (4/5))
test <- (-train)

x <- model.matrix(SalePrice ~ ., data = housing)
y <- housing$SalePrice
grid.lambda <- 10^seq(10, -2, length.out = 100)

y.train <- y[train]
y.test <- y[test]
x.train <- x[train, ]
x.test <- x[test, ]
```

Ridge Modeling

The ridge regularized regression was used as a possible model. Due to the high number of features and often redundant data, there could be a high level of collinearity between different fields. As mentioned previously, total area in square feet is redundant if all the parts are found in the data. For example, the MSSubClass is really a mashup of the year, and the number of levels of the house. The ridge should automatically minimize the impact of these redundant features on the predictive model.

```
# do a grid search of lambdas for optimal value
model.ridge.train <- glmnet(x.train, y.train, alpha = 0, lambda = grid.lambda)
set.seed(101)
cv.ridge.out <- cv.glmnet(x.train, y.train, alpha = 0, type.measure = "mse")
best.lambda.ridge <- cv.ridge.out$lambda.min

# do predictions
ridge.pred <- predict(model.ridge.train, s = best.lambda.ridge,
    newx = x.test)

# calculate error
mspe.ridge <- mean((ridge.pred - y.test)^2)
model.ridge.final <- glmnet(x, y, alpha = 0, lambda = best.lambda.ridge)
ridge.coefs <- coef(model.ridge.final)[-2, ]
r.squared.ridge <- max(model.ridge.final$dev.ratio)
```

Lasso Modeling

Lasso modeling was also applied against the dataset. This was ideal because we suspect that over 80+ features, there are probably a large number of features that do not have an affect on sales price. Things such as paved roads, or electrical breakers will most likely have little effect. Instead of plotting each of these fields against the main response variable SalePrice, we will apply a Lasso Regularized model which will drop some of these unnecessary variables.

```
# do a grid search of lambdas for optimal value
model.lasso.train <- glmnet(x.train, y.train, alpha = 1, lambda = grid.lambda)
set.seed(101)
cv.lasso.out <- cv.glmnet(x.train, y.train, alpha = 1, type.measure = "mse")
best.lambda.lasso <- cv.lasso.out$lambda.min

# do predictions
lasso.pred <- predict(model.lasso.train, s = best.lambda.lasso,
    newx = x.test)

# calculate error
mspe.lasso <- mean((lasso.pred - y.test)^2)
model.lasso.final <- glmnet(x, y, alpha = 1, lambda = best.lambda.lasso)
lasso.coefs <- coef(model.lasso.final)[-2, ]
r.squared.lasso <- max(model.lasso.final$dev.ratio)
```

ElasticNet Modeling

ElasticNet was also used, which has both regularization terms from both Ridge and Lasso, with an alpha blend factor. The model that was tried here was with alpha = 0.5

```
# do a grid search of lambdas for optimal value
model.en.train <- glmnet(x.train, y.train, alpha = 0.5, lambda = grid.lambda)
set.seed(101)
cv.en.out <- cv.glmnet(x.train, y.train, alpha = 0.5, type.measure = "mse")
best.lambda.en <- cv.en.out$lambda.min

# do predictions
en.pred <- predict(model.en.train, s = best.lambda.en, newx = x.test)

# calculate error
mspe.en <- mean((en.pred - y.test)^2)
model.en.final <- glmnet(x, y, alpha = 0.5, lambda = best.lambda.en)
en.coefs <- coef(model.en.final)[-2, ]
r.squared.en <- max(model.en.final$dev.ratio)
```

OLS Modeling informed by lasso model

OLS was re-run again, only pulling features that were selected after running the Lasso model. This was used as a baseline score to compare the other models against.

```
# ols defined by lasso variables
ols.vars <- names(abs(lasso.coefs) > 0)

# ols defined by lasso variables
x.ols <- x[, abs(lasso.coefs) > 0]

# ols test train split
x.ols.train <- x.ols[train, ]
x.ols.test <- x.ols[test, ]

# run the model
```

```
model.ols.train <- lm(y.train ~ x.ols.train)
set.seed(101)
ols.pred <- predict(model.ols.train, newx = x.ols.test)
mspe.ols <- mean((ols.pred - y.test)^2)
r.squared.ols <- summary(model.ols.train)$r.squared
```

```
Coefficients <- data.frame(Ridge = ridge.coefs, Lasso = lasso.coefs,
    Elastic.Net = en.coefs)

MSPE_frame <- data.frame(model = c("Ridge", "Lasso", "Elastic.net",
    "OLS"), MSPEscores = c(mspe.ridge, mspe.lasso, mspe.en, mspe.ols),
    r.squared = c(r.squared.ridge, r.squared.lasso, r.squared.en,
        r.squared.ols), best.lambda = c(best.lambda.ridge, best.lambda.lasso,
        best.lambda.en, 0)) %>% mutate(RMSPE = sqrt(MSPEscores))
MSPE_frame %>% kable
```

model	MSPEscores	r.squared	best.lambda	RMSPE
Ridge	1821798399	0.8825165	10978.2598	42682.53
Lasso	1895415185	0.8772348	658.1294	43536.37
Elastic.net	1880313264	0.8809334	995.7015	43362.58
OLS	11874370973	0.9067736	0.0000	108969.59

Regularized Prediction Discussion:

Task 2: Predictive Modeling

Here, we employed a handful of modeling techniques, iteratively testing out how well they performed as measured by the mean squared prediction error on a set of hold out data. Parameters for the models considered were generated via OLS, Ridge, LASSO, and Elastic Net algorithms. Besides sharing the same model type as explanatory modeling, extrapolation was markedly different in the following ways:

Normality Conditions: When optimizing our model for explanation, we used a number of functions to clean and alter the data in order to reduce bias and meet several assumptions when performing regression. When optimizing our model for prediction, however, we relaxed these assumptions to focus our concerns on prediction performance. Since we were only focused on the closeness of our predictions, our main criterion for selecting our best regression model was minimizing MSPE.

Variable Selection: To do this, we took raw data and tested out different subsets of our earlier data cleaning functions. Some of the steps taken in the variable selection for interpolation could not be omitted, such as cleaning up any null values, imputing values where necessary, and changing qualitative variables to numeric variables. For the other cleaning procedures, such as translating the year a house was built to age or removing collinear variables, we exhausted all combinations to create "partially clean" datasets.

Parameter Selection: With the data cleaned, we then ran several iterations of OLS, Ridge, LASSO, and Elastic Net on these variables to find the parameter estimates whose model had the lowest MSPE. After much consideration, we finally settled on a model with parameter estimates generated by L1 norm penalized regression (Lasso). Despite the fact that the table below has Lasso listed as having the highest MSPE, when averaged out, Lasso actually had the lowest mean MSPE. It also proved to be useful for decisive variable selection. The model in all its glory is detailed in this report

```
lasso_plot <- data.frame(y_hat = lasso.pred, true_sale_price = y.test)
colnames(lasso_plot) <- c("lasso_predictions", "true_sale_price")
lasso_plot %>% ggplot(aes(x = lasso_predictions, y = true_sale_price)) +
    geom_point() + geom_point(aes(x = lasso_predictions, y = lasso_predictions),
```

```
    color = "red") + labs(x = "Y Predicted (Sale Price)", y = "Y True (Sale Price)",
    title = "Lasso Regression of Sales Price") + theme(axis.text.x = element_text(size = 9),
    axis.text.y = element_text(size = 9)) + theme(plot.title = element_text(size = 11)) +
    theme(axis.text = element_text(size = 11))
```

Appendix: Optimal Lasso Beta Values

```
lasso.coefs.ordered <- lasso.coefs[order(abs(lasso.coefs), decreasing = T)]
lasso.coefs.ordered <- lasso.coefs.ordered[lasso.coefs.ordered !=
    0]
lasso.coefs.ordered %>% data.frame %>% kable
```

	.
(Intercept)	-608517.2922300
NeighborhoodStoneBr	47431.3267089
NeighborhoodNoRidge	45662.9346698
NeighborhoodNridgHt	38228.7027895
LotShapeIR3	-21326.0610045
NeighborhoodCrawfor	20220.5255680
Exterior2ndImStucc	19149.7317117
BsmtExposureGd	18994.9189199
KitchenQualTA	-17091.1605832
KitchenQualGd	-16904.5192380
Exterior1stBrkFace	15608.3439417
SaleTypeNew	15194.4736734
BsmtQualGd	-14789.6451610
MSSubClass2-STORY PUD - 1946 & NEWER	-13268.7095679
NeighborhoodSomerst	12956.6679942
OverallQual	11997.5284207
BldgTypeDuplex	-11957.9942409
NeighborhoodVeenker	11687.3682026
KitchenQualFa	-11676.8786343
Exterior1stImStucc	-11000.2924904
BsmtQualTA	-10994.4298744
BldgTypeTwnhs	-10912.2123246
MSSubClass1-STORY PUD (Planned Unit Development) - 1946 & NEWER	-10728.4427941
GarageCars	10334.0821401
Exterior2ndStucco	-9688.0819065
MSSubClass2 FAMILY CONVERSION - ALL STYLES AND AGES	-9584.4600230
LotConfigCulDSac	8577.5790590
NeighborhoodEdwards	-8343.7403026
BsmtExposureNone	-8132.6645434
Condition1Norm	8057.2898354
LotConfigFR3	-7691.2662172
Condition1RRAe	-7416.3293302
LotShapeIR2	7171.6469383
HouseStyle1Story	7048.9886371
BldgTypeTwnhsE	-7013.8794273
BsmtQualFa	-6281.0371253
SaleTypeCon	6084.9829943
GarageYrBlt1910s	-5960.9346584
Exterior2ndStone	-5867.7598498

	.
BsmtFullBath	5426.9795573
OverallCond	4963.9690270
BsmtFinType1GLQ	4863.1251505
FullBath	4775.9770943
GarageTypeBasment	-4521.7806986
Fireplaces	4490.7137590
NeighborhoodMitchel	-4432.8253243
BsmtExposureNo	-4408.3645407
BsmtFinType1Unf	-4309.5571282
MSSubClassDUPLEX - ALL STYLES AND AGES	-4212.2473818
NeighborhoodOldTown	-3892.9382746
ExterQualTA	-3739.3442280
Exterior1stCemntBd	3737.5038629
NeighborhoodBrkSide	3686.5866015
NeighborhoodSWISU	-3460.9139771
BsmtFinType1None	-3330.5525355
GarageTypeNone	2972.2852847
HalfBath	2939.2005229
FoundationSlab	-2905.9501064
MSZoningRM	-2675.1618959
LotConfigFR2	-2654.1150304
GarageTypeCarPort	-2434.2535928
Exterior2ndWd Shng	-2431.0981568
Condition1PosN	-2307.0991163
GarageFinishRFn	-1830.9201043
NeighborhoodNPkVill	1805.0152796
Condition1Feedr	-1787.0847342
GarageYrBlt1930s	1778.2730712
HeatingQCGd	-1690.3883730
SaleConditionFamily	-1660.8817522
HeatingQCTA	-1605.2593287
FoundationPConc	1509.0014737
NeighborhoodNAmes	-1493.3519450
Exterior2ndVinylSd	1274.2832602
TotRmsAbvGrd	1265.3122703
BsmtFinType1LwQ	-1155.6600294
Exterior1stHdBoard	-1153.5782777
RoofStyleHip	1093.2569034
SaleConditionNormal	1082.1859690
GarageYrBlt1970s	-1049.6995138
BsmtFinType2None	-997.6626058
GarageTypeBuiltIn	794.1792132
NeighborhoodNWAmes	-780.7074408
MSZoningRL	515.8076487
BedroomAbvGr	-463.6988810
GarageYrBlt1960s	-426.9536472
RoofStyleGable	-413.6108089
MasVnrTypeNone	245.2471296
NeighborhoodIDOTRR	-225.3068869
YearBuilt	218.4159159
MasVnrTypeBrkFace	-210.2018408
SaleTypeCWD	131.3577196

	.
MoSold	-119.2441860
BsmtQualNone	-117.2509235
Condition1RRAn	85.2278350
YearRemodAdd	80.0861523
GarageYrBltNAs	62.4396116
GrLivArea	42.2476886
GarageFinishUnf	-35.8073003
WoodDeckSF	17.2095504
MasVnrArea	13.0330982
GarageFinishNone	11.6058167
TotalBsmtSF	4.9624702
BsmtFinSF1	3.6150310
LotArea	0.3080657