Capstone Project - Housing Prices

Renz Asprec

7/12/2021

Contents

Executive Summary **2**

Introduction **3**

Methodology **5**

Exploratory Data Analysis 5

Feature Engineering 13

Model and Evaluation Metrics 14

Results **15**

Linear Regression Model 15

Random Forest 16

Conclusions **20**

Limitations **20**

Appendix **21**

Code by user: 21

Executive Summary

This project aims to predict the house prices in residential houses in Ames, Iowa, given several house features. The data set used was obtained from a competition on the prediction of house prices in Kaggle.com.

Through exploratory data analysis, certain features were dropped based on the percentages of NAs and single weight values. Feature engineering was also performed to reduce the number of similar features.

There were two models explored in this project. The first model is a linear regression model. The second model is based on the random forest algorithm, specifically from the `Rborist` package in R. For both models, initial modeling were performed to identify the 15 most important features. These features where then used to train the final models.

The model which produced the better RMSE is the random forest model, with an RMSE of $29167 as compared to the $42371 RMSE of the linear regression model. The random forest model was able to produce an RMSE close to its training RMSE compared to the linear regression model.

Introduction

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. This dataset is used to prove that there are much more influences to price negotiations than the number of bedrooms or a white-picket fence.

This data set contains 79 exploratory features describing every aspect of residential homes in Ames, Iowa. The description of each feature is shown below.

- MSSubClass : Identifies the type of dwelling involved in the sale
- MSZoning : Identifies the general zoning classification of the sale.
- LotFrontage : Linear feet of street connected to property
- LotArea: Lot size in square feet
- Street: Type of road access to property
- Alley: Type of alley access to property
- LotShape: General shape of property
- LandContour: Flatness of the property
- Utilities: Type of utilities available
- LotConfig: Lot configuration
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits
- Condition1: Proximity to various conditions
- Condition2: Proximity to various conditions (if more than one is present)
- BldgType: Type of dwelling
- HouseStyle: Style of dwelling
- OverallQual: Rates the overall material and finish of the house
- OverallCond: Rates the overall condition of the house
- YearBuilt: Original construction date
- YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)
- RoofStyle: Type of roof
- RoofMatl: Roof material
- Exterior1st: Exterior covering on house
- Exterior2nd: Exterior covering on house (if more than one material)
- MasVnrType: Masonry veneer type
- MasVnrArea: Masonry veneer area in square feet
- ExterQual: Evaluates the quality of the material on the exterior
- ExterCond: Evaluates the present condition of the material on the exterior
- Foundation: Type of foundation
- BsmtQual: Evaluates the height of the basement
- BsmtCond: Evaluates the general condition of the basement
- BsmtExposure: Refers to walkout or garden level walls
- BsmtFinType1: Rating of basement finished area
- BsmtFinSF1: Type 1 finished square feet
- BsmtFinType2: Rating of basement finished area (if multiple types)
- BsmtFinSF2: Type 2 finished square feet
- BsmtUnfSF: Unfinished square feet of basement area
- TotalBsmtSF: Total square feet of basement area
- Heating: Type of heating
- HeatingQC: Heating quality and condition
- CentralAir: Central air conditioning
- Electrical: Electrical system
- 1stFlrSF: First Floor square feet
- 2ndFlrSF: Second floor square feet
- LowQualFinSF: Low quality finished square feet (all floors)

- GrLivArea: Above grade (ground) living area square feet
- BsmtFullBath: Basement full bathrooms
- BsmtHalfBath: Basement half bathrooms
- FullBath: Full bathrooms above grade
- HalfBath: Half baths above grade
- Bedroom: Bedrooms above grade (does NOT include basement bedrooms)
- Kitchen: Kitchens above grade
- KitchenQual: Kitchen quality
- TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
- Functional: Home functionality (Assume typical unless deductions are warranted)
- Fireplaces: Number of fireplaces
- FireplaceQu: Fireplace quality
- GarageType: Garage location
- GarageYrBlt: Year garage was built
- GarageFinish: Interior finish of the garage
- GarageCars: Size of garage in car capacity
- GarageArea: Size of garage in square feet
- GarageQual: Garage quality
- GarageCond: Garage condition
- PavedDrive: Paved driveway
- WoodDeckSF: Wood deck area in square feet
- OpenPorchSF: Open porch area in square feet
- EnclosedPorch: Enclosed porch area in square feet
- 3SsnPorch: Three season porch area in square feet
- ScreenPorch: Screen porch area in square feet
- PoolArea: Pool area in square feet
- PoolQC: Pool quality
- Fence: Fence quality
- MiscFeature: Miscellaneous feature not covered in other categories
- MiscVal: $Value of miscellaneous feature
- MoSold: Month Sold (MM)
- YrSold: Year Sold (YYYY)
- SaleType: Type of sale
- SaleCondition: Condition of sale

Methodology

Exploratory Data Analysis

We'll start with exploring our data. We can see the structure of our data below:

```
## spec_tbl_df [1,460 x 81] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Id           : num [1:1460] 1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass   : num [1:1460] 60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning     : chr [1:1460] "RL" "RL" "RL" "RL" ...
##  $ LotFrontage  : num [1:1460] 65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : num [1:1460] 8450 9600 11250 9550 14260 ...
##  $ Street       : chr [1:1460] "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley        : chr [1:1460] NA NA NA NA ...
##  $ LotShape     : chr [1:1460] "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour  : chr [1:1460] "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities    : chr [1:1460] "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ LotConfig    : chr [1:1460] "Inside" "FR2" "Inside" "Corner" ...
##  $ LandSlope    : chr [1:1460] "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood : chr [1:1460] "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
##  $ Condition1   : chr [1:1460] "Norm" "Feedr" "Norm" "Norm" ...
##  $ Condition2   : chr [1:1460] "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType     : chr [1:1460] "1Fam" "1Fam" "1Fam" "1Fam" ...
##  $ HouseStyle   : chr [1:1460] "2Story" "1Story" "2Story" "2Story" ...
##  $ OverallQual  : num [1:1460] 7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : num [1:1460] 5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : num [1:1460] 2003 1976 2001 1915 2000 ...
##  $ YearRemodAdd : num [1:1460] 2003 1976 2002 1970 2000 ...
##  $ RoofStyle    : chr [1:1460] "Gable" "Gable" "Gable" "Gable" ...
##  $ RoofMatl     : chr [1:1460] "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st  : chr [1:1460] "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
##  $ Exterior2nd  : chr [1:1460] "VinylSd" "MetalSd" "VinylSd" "Wd Shng" ...
##  $ MasVnrType   : chr [1:1460] "BrkFace" "None" "BrkFace" "None" ...
##  $ MasVnrArea   : num [1:1460] 196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : chr [1:1460] "Gd" "TA" "Gd" "TA" ...
##  $ ExterCond    : chr [1:1460] "TA" "TA" "TA" "TA" ...
##  $ Foundation   : chr [1:1460] "PConc" "CBlock" "PConc" "BrkTil" ...
##  $ BsmtQual     : chr [1:1460] "Gd" "Gd" "Gd" "TA" ...
##  $ BsmtCond     : chr [1:1460] "TA" "TA" "TA" "Gd" ...
##  $ BsmtExposure : chr [1:1460] "No" "Gd" "Mn" "No" ...
##  $ BsmtFinType1 : chr [1:1460] "GLQ" "ALQ" "GLQ" "ALQ" ...
##  $ BsmtFinSF1   : num [1:1460] 706 978 486 216 655 ...
##  $ BsmtFinType2 : chr [1:1460] "Unf" "Unf" "Unf" "Unf" ...
##  $ BsmtFinSF2   : num [1:1460] 0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : num [1:1460] 150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : num [1:1460] 856 1262 920 756 1145 ...
##  $ Heating      : chr [1:1460] "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC    : chr [1:1460] "Ex" "Ex" "Ex" "Gd" ...
##  $ CentralAir   : chr [1:1460] "Y" "Y" "Y" "Y" ...
##  $ Electrical   : chr [1:1460] "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
##  $ 1stFlrSF     : num [1:1460] 856 1262 920 961 1145 ...
##  $ 2ndFlrSF     : num [1:1460] 854 0 866 756 1053 ...
##  $ LowQualFinSF : num [1:1460] 0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ GrLivArea    : num [1:1460] 1710 1262 1786 1717 2198 ...
##  $ BsmtFullBath : num [1:1460] 1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : num [1:1460] 0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : num [1:1460] 2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : num [1:1460] 1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : num [1:1460] 3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : num [1:1460] 1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : chr [1:1460] "Gd" "TA" "Gd" "Gd" ...
##  $ TotRmsAbvGrd : num [1:1460] 8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : chr [1:1460] "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces   : num [1:1460] 0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : chr [1:1460] NA "TA" "TA" "Gd" ...
##  $ GarageType   : chr [1:1460] "Attchd" "Attchd" "Attchd" "Detchd" ...
##  $ GarageYrBlt  : num [1:1460] 2003 1976 2001 1998 2000 ...
##  $ GarageFinish : chr [1:1460] "RFn" "RFn" "RFn" "Unf" ...
##  $ GarageCars   : num [1:1460] 2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : num [1:1460] 548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : chr [1:1460] "TA" "TA" "TA" "TA" ...
##  $ GarageCond   : chr [1:1460] "TA" "TA" "TA" "TA" ...
##  $ PavedDrive   : chr [1:1460] "Y" "Y" "Y" "Y" ...
##  $ WoodDeckSF   : num [1:1460] 0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : num [1:1460] 61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: num [1:1460] 0 0 0 272 0 0 0 228 205 0 ...
##  $ 3SsnPorch    : num [1:1460] 0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : num [1:1460] 0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : num [1:1460] 0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : chr [1:1460] NA NA NA NA ...
##  $ Fence        : chr [1:1460] NA NA NA NA ...
##  $ MiscFeature  : chr [1:1460] NA NA NA NA ...
##  $ MiscVal      : num [1:1460] 0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : num [1:1460] 2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : num [1:1460] 2008 2007 2008 2006 2008 ...
##  $ SaleType     : chr [1:1460] "WD" "WD" "WD" "WD" ...
##  $ SaleCondition: chr [1:1460] "Normal" "Normal" "Normal" "Abnorml" ...
##  $ SalePrice    : num [1:1460] 208500 181500 223500 140000 250000 ...
```

We can see the corresponding class of each feature. We can see that there are features of variable type `character` that we know should be `factors` based on the information we know regarding our data set. Similarly, there are also features that are `numeric` that should be `factors`. We transform them to their correct variable type. Consequently, we partition our data to train and test sets.

Let's see how many observations we have as our training data, and how many features are present.

```
## [1] 1312   81
```

As shown above, there are 1312 observations and 79 features (excluding `Id` and the outcome `SalePrice`).

Checking data for NAs

We will first check our data for missing values. The table below shows the features with NAs with their corresponding total number and proportion of NAs.

feature	total_na	prop_na
PoolQC	1305	0.99466
MiscFeature	1266	0.96494
Alley	1229	0.93674
Fence	1061	0.80869
FireplaceQu	630	0.48018
LotFrontage	226	0.17226
GarageType	75	0.05716
GarageYrBlt	75	0.05716
GarageFinish	75	0.05716
GarageQual	75	0.05716
GarageCond	75	0.05716
BsmtExposure	34	0.02591
BsmtFinType2	34	0.02591
BsmtQual	33	0.02515
BsmtCond	33	0.02515
BsmtFinType1	33	0.02515
MasVnrType	7	0.00534
MasVnrArea	7	0.00534
Electrical	1	0.00076

We see that there are features which mostly contain NAs. We'll drop the features which contain more than 80% of NAs as these features won't be useful to us. Shown below are those features:

```
## [1] "PoolQC"      "MiscFeature" "Alley"       "Fence"
```

We'll also drop features related to them such as `PoolArea` and `MiscVal`.

Investigate maximum single weight values for each feature

Here, we'll investigate the maximum single weight values for each feature. Based on the results, we'll drop features which have single value weight of more than 0.80. The table below shows each feature and their corresponding single value weight.

feature	single_val_wt
Utilities	0.99924
Street	0.99619
Condition2	0.98857
RoofMatl	0.98323
3SsnPorch	0.98323
LowQualFinSF	0.98095
Heating	0.97790
KitchenAbvGr	0.95274
LandSlope	0.94817
BsmtHalfBath	0.94436
CentralAir	0.93216
Functional	0.93140
ScreenPorch	0.92378
PavedDrive	0.91692
Electrical	0.91540

feature	single_val_wt
GarageCond	0.90549
LandContour	0.90091
GarageQual	0.89710
BsmtCond	0.89634
BsmtFinSF2	0.88948
ExterCond	0.88034
SaleType	0.86509
BsmtFinType2	0.86433
Condition1	0.85976
EnclosedPorch	0.85442
BldgType	0.83384
SaleCondition	0.82088
RoofStyle	0.78582
MSZoning	0.78201
LotConfig	0.71951
BsmtExposure	0.65854
LotShape	0.63491
HalfBath	0.62348
ExterQual	0.61738
MasVnrType	0.59451
MasVnrArea	0.59299
BsmtFullBath	0.58994
GarageType	0.58765
OverallCond	0.56250
GarageCars	0.55945
2ndFlrSF	0.55793
BedroomAbvGr	0.54345
FullBath	0.52820
WoodDeckSF	0.52287
HeatingQC	0.50610
KitchenQual	0.49924
HouseStyle	0.48780
Fireplaces	0.48018
FireplaceQu	0.48018
OpenPorchSF	0.44893
BsmtQual	0.44512
Foundation	0.44360
GarageFinish	0.41616
MSSubClass	0.35366
Exterior1st	0.35366
Exterior2nd	0.34527
BsmtFinSF1	0.32774
BsmtFinType1	0.30259
TotRmsAbvGrd	0.27210
OverallQual	0.26905
YrSold	0.23018
LotFrontage	0.17226
MoSold	0.17149
Neighborhood	0.14787
YearRemodAdd	0.12576
BsmtUnfSF	0.08079
GarageYrBlt	0.05716

feature	single_val_wt
GarageArea	0.05716
YearBuilt	0.04649
TotalBsmtSF	0.02515
1stFlrSF	0.01829
LotArea	0.01753
GrLivArea	0.01601
SalePrice	0.01296
Id	0.00076

Below are the features that we will drop.

x
Utilities
Street
Condition2
RoofMatl
3SsnPorch
LowQualFinSF
Heating
KitchenAbvGr
LandSlope
BsmtHalfBath
CentralAir
Functional
ScreenPorch
PavedDrive
Electrical
GarageCond
LandContour
GarageQual
BsmtCond
BsmtFinSF2
ExterCond
SaleType
BsmtFinType2
Condition1
EnclosedPorch
BldgType
SaleCondition

Filling NA values

We've already dropped features based on the single weight values. However, we still have not dealt with the NAs in our data. The features below are the features which still contain NAs.

feature
LotFrontage
MasVnrType
MasVnrArea

feature
BsmtQual
BsmtExposure
BsmtFinType1
FireplaceQu
GarageType
GarageYrBlt
GarageFinish

Based on the information on our data, we'll replace NAs with `0` (for numerical features where NAs mean 0 value), `none` (for features where NAs mean no presence of such feature), and with the median of the variables (for factor features were there are no inputted data)

List of numerical features where NAs mean 0 value:

feature
LotFrontage
MasVnrArea
LotArea
BsmtFinSF1
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
GrLivArea
BsmtFullBath
FullBath
HalfBath
BedroomAbvGr
TotRmsAbvGrd
Fireplaces
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF

List of features where NAs mean no presence of such feature:

feature
MasVnrType
BsmtQual
BsmtExposure
BsmtFinType1
FireplaceQu
GarageType
GarageFinish

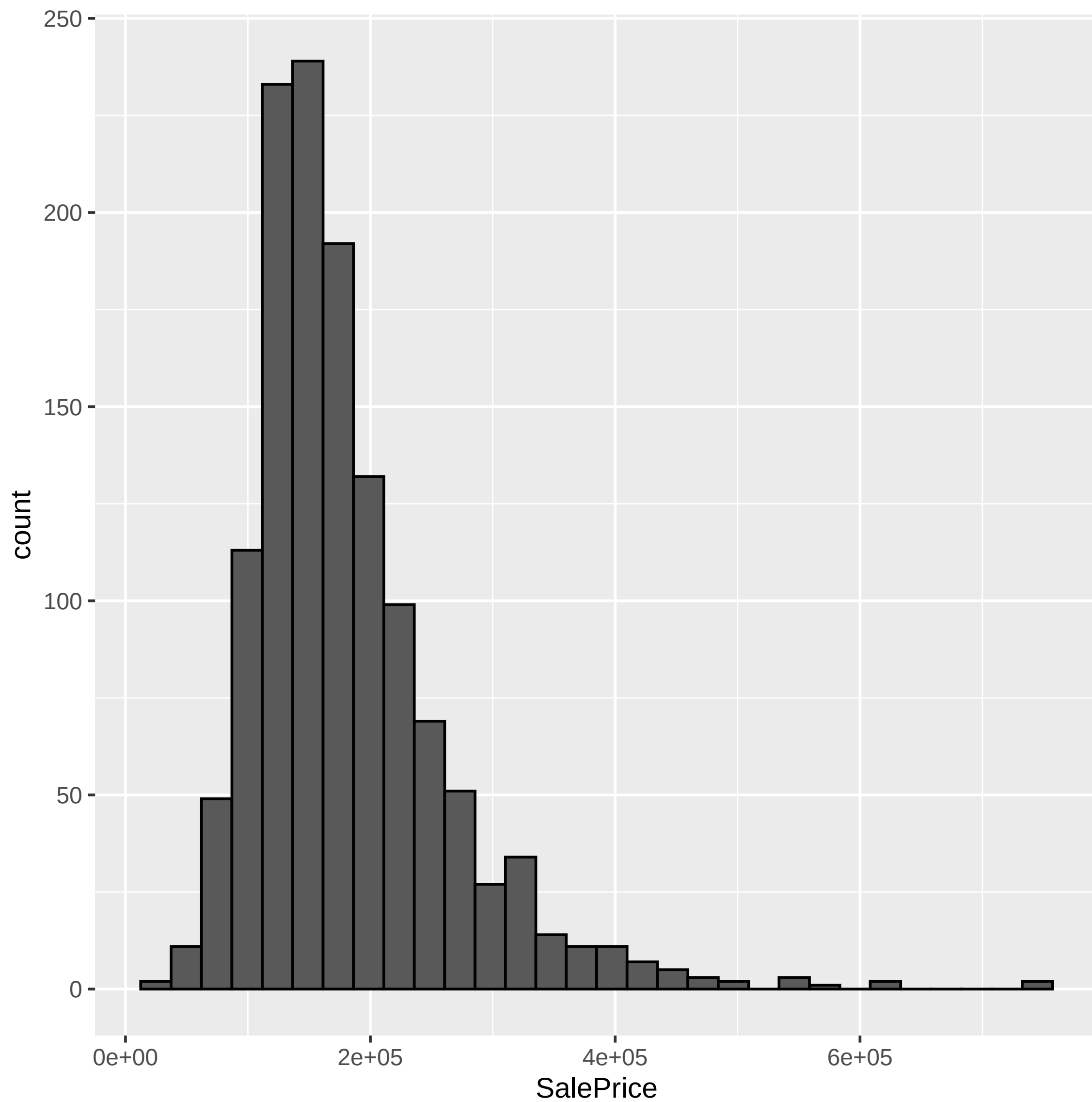
List of factor features were there are no inputted data:

feature
MSSubClass
MSZoning
LotShape
LotConfig
Neighborhood
HouseStyle
OverallQual
OverallCond
YearBuilt
YearRemodAdd
RoofStyle
Exterior1st
Exterior2nd
ExterQual
Foundation
HeatingQC
KitchenQual

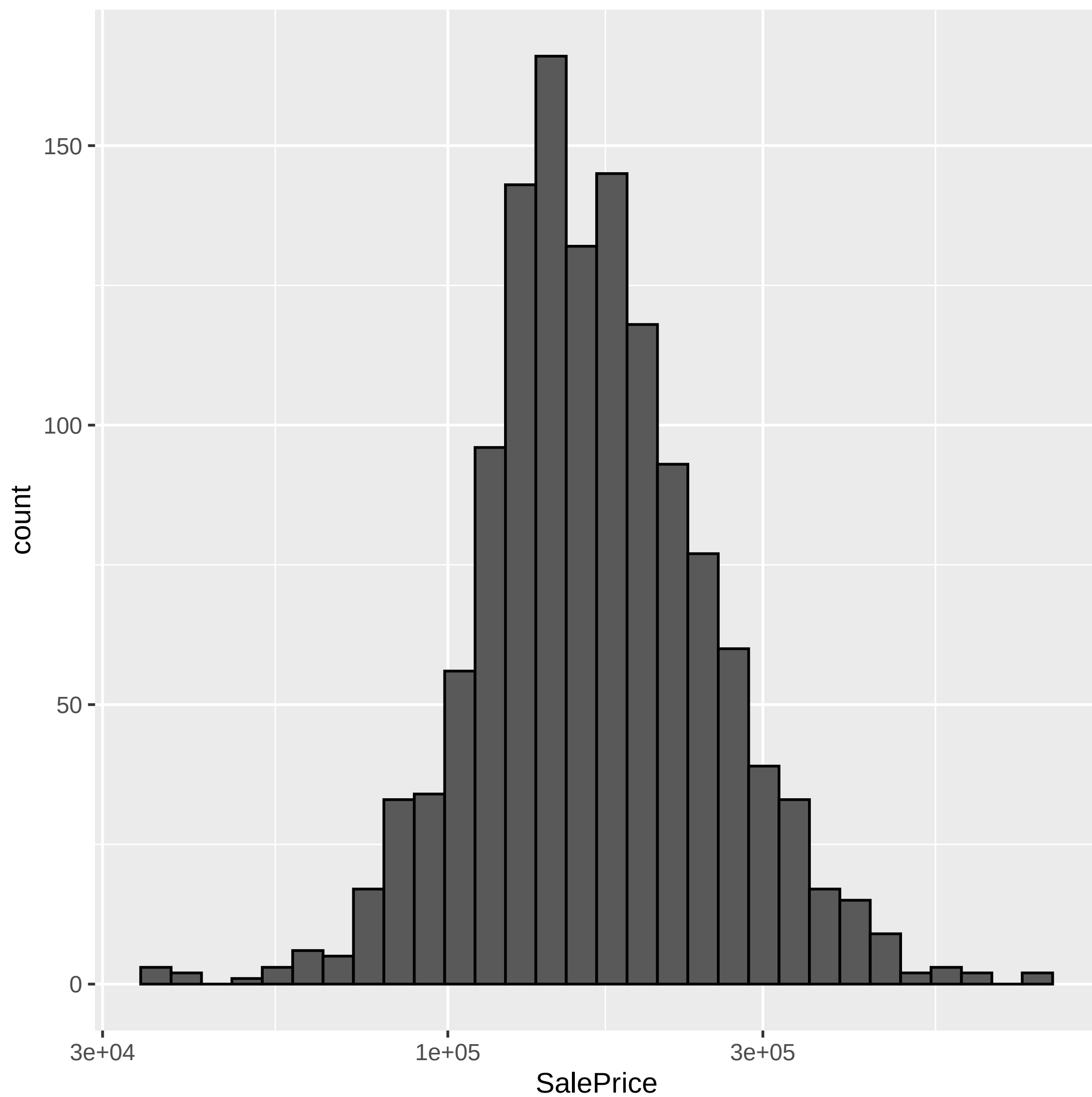
There are still other features with NAs such as `GarageYrBlt`. Here, we'll assume that it is the same as the year the house was built, `YearBuilt`.

Investigation of outcome

Now, we'll investigate our outcome, which is `SalePrice`. Below shows the distribution of the sale price.



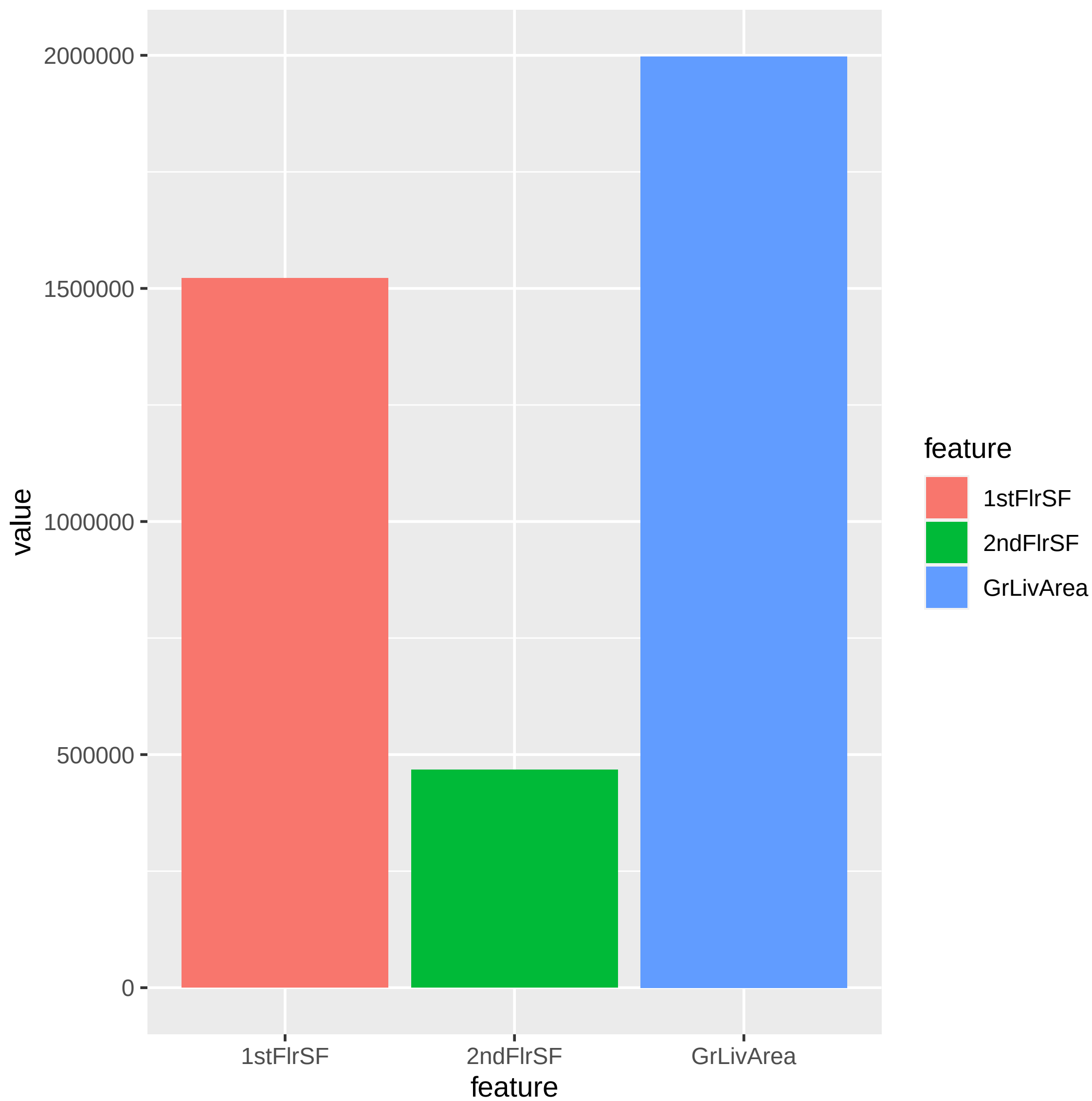
We can see that the distribution is skewed to the right. To fix this, we'll perform a log transformation of the `SalePrice`. The distribution of the log transformed `SalePrice` is shown below:



Now, we can see that it is normally distributed. We will use the log transformed values of `SalePrice` in our model.

Feature Engineering

Further inspection of our data shows that there are similar features. Based on our information on the data, `GrLivArea`, `1stFlrSF`, and `2ndFlrSF` seem to be related. Let's investigate if we'll find a relation between them. The plot of the sum of each of those features is shown below:



We can see that `GrLivArea` is just similar to the sum of `1stFlrSF` and `2ndFlrSF`. So we'll just keep `GrLivArea` and drop the other 2 features.

Going back to our data, we know that the number of garage cars that can fit inside the garage is just proportional to its area. Thus, we only need to select 1 feature between `GarageCars` and `GarageArea`. Here, we'll drop `GarageCars` and use `GarageArea`.

Prior to proceeding with training our data, let's drop the column for `Id` since it won't be used for training.

The number of features we will be using are now reduced from the initial number of 79 down to 44.

Model and Evaluation Metrics

For this project, we would be exploring two models, a linear model, and a random forest model, specifically the `Rborist` package in R. For the random forest model we'll perform bagging, sampling 0.90 of the training data, for 20 iterations. In relation, we will evaluate our predicted results using the root mean squared error (RMSE) loss function.

Results

Linear Regression Model

First, we'll fit our data using a linear regression model.

The resulting RMSE for the train set is $26429.

Now, let's investigate the most important features based on the linear regression model. The importance of the features are shown below:

	Overall
GrLivArea	7.8117
MSZoningRL	6.3757
MSZoningRM	6.1832
MSZoningFV	5.6006
OverallQual9	5.4071
GarageArea	5.2638
FullBath	5.0912
MSZoningRH	5.0402
OverallQual8	4.8909
OverallQual10	4.7298
OverallQual7	4.5683
BsmtFinType1Unf	4.3295
HalfBath	4.2688
OverallQual6	4.1731
BsmtExposureGd	4.0009
MSSubClass20	3.9719
NeighborhoodStoneBr	3.9152
OverallQual5	3.8812
OverallQual4	3.7622
BsmtFullBath	3.5070
LotArea	3.4621
NeighborhoodNridgHt	3.4575
KitchenQualGd	3.2986
YearRemodAdd	3.1878
KitchenQualTA	3.1214
OverallQual3	3.0905
NeighborhoodCrawfor	2.9440
BsmtQualGd	2.8525
NeighborhoodNoRidge	2.7990
BsmtQualTA	2.6973
GarageTypeAttchd	2.6158
WoodDeckSF	2.5704
LotShapeIR2	2.4825
KitchenQualFa	2.4814
MSSubClass50	2.4694

Now, we'll redefine our data and use only the 15 most important features. Then, we'll create a model using the redefined data.

The RMSE of the redefined training data is $30060.

Prior to predicting the sale price of the test set, we need to perform data cleaning and feature engineering similar to the one we did with the train set.

We'll only use the 15 most important features as we previously defined from the lm model.

The resulting RMSE of our model is $42372.

Random Forest

Now, we will be using `Rborist` to create our model. But first, we need to train our data and optimize the tuning parameters that will give us the best RMSE. For `Rborist` there are 2 tuning parameters, `predFixed` and `minNode`. We'll define the range of tuning parameters for training as show below:

predFixed	minNode
15	2
18	2
21	2
24	2
27	2
30	2
33	2
36	2
39	2
42	2
15	4
18	4
21	4
24	4
27	4
30	4
33	4
36	4
39	4
42	4
15	6
18	6
21	6
24	6
27	6
30	6
33	6
36	6
39	6
42	6
15	8
18	8
21	8
24	8
27	8
30	8
33	8
36	8
39	8
42	8
15	10
18	10

predFixed	minNode
21	10
24	10
27	10
30	10
33	10
36	10
39	10
42	10
15	12
18	12
21	12
24	12
27	12
30	12
33	12
36	12
39	12
42	12
15	14
18	14
21	14
24	14
27	14
30	14
33	14
36	14
39	14
42	14
15	16
18	16
21	16
24	16
27	16
30	16
33	16
36	16
39	16
42	16
15	18
18	18
21	18
24	18
27	18
30	18
33	18
36	18
39	18
42	18
15	20
18	20
21	20
24	20

predFixed	minNode
27	20
30	20
33	20
36	20
39	20
42	20

We can now train our model and obtain the best tuning parameters. The resulting best tuning parameters are shown below:

	predFixed	minNode
20	42	4

Now, we can run `Rborist` based on the best tuning parameters.

The resulting RMSE of the training set is $29332.

Now, we'll investigate the importance of each feature based on the result of the random forest model. We can see the importance of each feature below:

```
## Rborist variable importance
##
##   only 20 most important variables shown (out of 172)
##
##                 Overall
## GrLivArea        100.00
## YearBuilt         75.42
## TotalBsmtSF       48.58
## GarageArea        45.78
## GarageYrBlt       42.95
## ExterQualTA       29.27
## KitchenQualTA     26.78
## LotArea           24.36
## Fireplaces        23.49
## FullBath          21.29
## FireplaceQunone   20.83
## BsmtFinSF1        16.07
## YearRemodAdd      13.16
## TotRmsAbvGrd      13.01
## OpenPorchSF        9.66
## FoundationPConc    9.38
## LotFrontage        5.95
## BsmtUnfSF          4.73
## HalfBath           4.34
## BsmtQualGd         3.94
```

From the results above, we'll take the 15 most important features and use this to retrain our data.

```
## [1] 30747
```

The resulting RMSE of the redefined training set is $30747.

Now, we'll redefine our test set based on the 15 most important features from the random forest model. Then, we can proceed to predicting the sale prices.

```
## [1] 29166
```

The resulting RMSE of the model is $29166.

Below is the summary of the RMSEs for the two models.

model	training_RMSE	test_RMSE
linear regression	30060	42372
random forest	30747	29166

Conclusions

There were two methods explored in this project for modeling the sale price of houses in Ames, Iowa given several house features. This project was able to reduce the number of features from 79 to 15 through data cleaning, feature engineering, and selection of features based on importance. The model which produced the better RMSE is the random forest model. It's resulting RMSE is $29166. This resulting RMSE is close to our training RMSE of $30747, which is a good indication that we did not over train our model.

Limitations

The data available for training the model in this project only contains 1312 observations. The model would further be improved with more observations available for training.

Appendix

Code by user:

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(readr)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
if(!require(Rborist)) install.packages("knitr", repos = "http://cran.us.r-project.org")

#load required libraries
library(tidyverse)
library(caret)
library(readr)
library(knitr)
library(Rborist)

# set options
options(digits=5)

url<- "https://raw.githubusercontent.com/renzasprec/House-Prices/main/train.csv"
dat<- read_csv(url)

# Exploratory Data Analysis -----
str(dat, give.attr=FALSE)

# split into a training and testing set
set.seed(1, sample.kind = "Rounding")
test_ind<- createDataPartition(dat$SalePrice, times=1, p=0.1, list=FALSE)
test_set<- dat %>% slice(test_ind)
train_set<- dat %>% slice(-test_ind)

# match class of feature based on information on data
num_to_factor<- c("MSSubClass","OverallQual","OverallCond")
train_set[num_to_factor]<- lapply(train_set[num_to_factor], FUN = factor)

# convert characters to factors
char_to_factor<-lapply(train_set,class)
char_to_factor_list<- names(char_to_factor[which(char_to_factor=="character")])

train_set[char_to_factor_list]<- lapply(train_set[char_to_factor_list], FUN = factor)

# convert dates to num
date_to_num<- c("YearBuilt","GarageYrBlt","YearRemodAdd","MoSold","YrSold")
train_set[date_to_num]<- apply(train_set[date_to_num],
                               MARGIN = 2,
                               FUN = as.numeric)

# dimensions of train set
init_dim<- dim(train_set)
init_dim
```

```r
## check data for NAs -----
nas<- apply(X = train_set, MARGIN = 2, FUN = function(x){sum(is.na(x))})
n_rows<- nrow(train_set)
nas<- rownames_to_column(data.frame(total_na = nas),var = "feature") %>%
  mutate(prop_na = total_na/n_rows) %>%
  arrange(desc(prop_na))

# output table of NAs
nas %>% filter(prop_na>0) %>% kable()

# drop features with proportion of NAs greater than 0.80
drop_cols_na<- with(nas, feature[prop_na>=0.80])
train_set<- train_set %>% select(-drop_cols_na)
drop_cols_na

# drop related features
train_set<- train_set %>% select(-c("PoolArea","MiscVal"))

## check maximum weight of a single value ------
single_val_wt_dat<- data.frame(single_val_wt= apply(train_set, MARGIN = 2, FUN = function(x){
  tab<- table(x, useNA = "ifany")
  sort(tab, decreasing = TRUE)[1]/n_rows
})) %>%
  arrange(desc(single_val_wt)) %>%
  rownames_to_column(var = "feature")

single_val_wt_dat %>% kable()

# remove features with single value weight more than 0.8 as these features won't be helpful
drop_cols_swt<- with(single_val_wt_dat, feature[single_val_wt>0.8])
drop_cols_swt %>% kable()

train_set<- train_set %>% select(-drop_cols_swt)

## Filling NA values------

# define indexes for NAs
ind_na<- apply(X = train_set,MARGIN = 2,FUN = function(x){
  any(is.na(x))
})

# features with NAs
na_cols<- data.frame(feature= names(train_set[,ind_na]))
na_cols %>% kable()

# select features where NAs mean 0 (numerical)
num_feat_0<- c("LotFrontage","MasVnrArea", "LotArea","BsmtFinSF1","BsmtUnfSF","TotalBsmtSF","1stFlrSF",

data.frame(feature = num_feat_0) %>% kable()

# replace NAs which corresponds to 0
train_set[,num_feat_0]<- apply(train_set[,num_feat_0], MARGIN = 2, FUN = function(x){
  replace(x, is.na(x), 0)
```

```r
})

# select features where NAs mean none (characters)
char_feat_none<- c("MasVnrType","BsmtQual","BsmtExposure","BsmtFinType1","FireplaceQu","GarageType","Ga

data.frame(feature = char_feat_none) %>% kable()

# replace NAs which corresponds to none
train_set[,char_feat_none]<- apply(train_set[,char_feat_none], MARGIN = 2,FUN = function(x){
  replace(x, is.na(x),"none")
})

# select features where NAs mean that there are no inputted data
char_feat_mode<- c("MSSubClass","MSZoning","LotShape","LotConfig","Neighborhood","HouseStyle","OverallQu

data.frame(feature = char_feat_mode) %>% kable()

# replace NAs for factor with the most common value
train_set[,char_feat_mode]<- apply(train_set[,char_feat_mode], MARGIN = 2,FUN = function(x){
  replace(x, is.na(x), names(which.max(table(x))))
})

# other NAs
train_set<- train_set %>% mutate(GarageYrBlt = ifelse(is.na(GarageYrBlt),YearBuilt,GarageYrBlt))

## Investigation of outcome ------

# distribution of sale price
train_set %>%
  ggplot(aes(SalePrice)) +
  geom_histogram(bins=30, color="black")

# distribution of log of sale price
train_set %>%
  ggplot(aes(SalePrice)) +
  geom_histogram(bins=30, color="black")+
  scale_x_log10()

# transform sale price to log10
train_set<- train_set %>%
  mutate(SalePrice = log10(SalePrice))

# Feature Engineering------

# investigate similar features

# plot GrLivArea, 1stFlrSF, and 2ndFlrSF
train_set %>%
  select(GrLivArea,`1stFlrSF`,`2ndFlrSF`) %>%
  apply(X = .,MARGIN = 2,FUN=sum) %>%
  as.data.frame() %>% rownames_to_column(var="feature") %>%
  rename(value=".") %>%
  ggplot(aes(feature,value, fill=feature)) +
```

```
  geom_col()

# drop 1stFlrSF and 2ndFlrSF
train_set<- train_set %>% select(-c(`1stFlrSF`,`2ndFlrSF`))

# drop GarageCars since it is just similar to GarageArea
train_set<- train_set %>% select(-GarageCars)

# remove Id
train_set<- train_set %>% select(-Id)

# match class of feature based on information on data
num_to_factor<- c("MSSubClass","OverallQual","OverallCond")
train_set[num_to_factor]<- lapply(train_set[num_to_factor], FUN = factor)

# convert characters to factors
char_to_factor<-lapply(train_set,class)
char_to_factor_list<- names(char_to_factor[which(char_to_factor=="character")])

train_set[char_to_factor_list]<- lapply(train_set[char_to_factor_list], FUN = factor)

# convert dates to num
date_to_num<- c("YearBuilt","GarageYrBlt","YearRemodAdd","MoSold","YrSold")
train_set[date_to_num]<- apply(train_set[date_to_num],
                               MARGIN = 2,
                               FUN = as.numeric)

# training set dimensions
final_dim<- dim(train_set)

# Model Evaluation and Metrics ------

# RMSE function
RMSE<- function(SalePrice_predicted, SalePrice_true){
  sqrt(mean((SalePrice_predicted-SalePrice_true)^2))
}

# Model Training ------

## train model - lm -----
train_set_lm<- train_set

fit_lm<- lm(SalePrice~., data = train_set_lm)

rmse_train_lm<- RMSE(10^fit_lm$fitted.values, 10^train_set_lm$SalePrice)

# obtain 15 most important features
varImp(fit_lm) %>% arrange(desc(Overall)) %>% slice_head(n=35) %>% kable()

imp_lm<- c("GrLivArea","MSZoning","OverallQual","GarageArea","FullBath","BsmtFinType1","HalfBath","Bsmt

# redefine train_set with only the 15 most important features
train_set_lm<- train_set %>% select(imp_lm,SalePrice)
```

```r
# new training set dimensions
final_dim_lm<- dim(train_set_lm)

# new model based on redefined data
fit_lm<- lm(SalePrice~., data = train_set_lm)

# rmse of the redefined training data
rmse_train_imp_lm<- RMSE(10^fit_lm$fitted.values,10^train_set_lm$SalePrice)

## TEST SET
# perform data cleaning and feature engineering for the test set

# drop features with proportion of NAs greater than 0.80 (based on train_set data)
test_set<- test_set %>% select(-drop_cols_na)

# drop related features
test_set<- test_set %>% select(-c("PoolArea","MiscVal"))

# remove features with single value weight more than 0.8 as removed from the train_set
test_set<- test_set %>% select(-drop_cols_swt)

# Filling NA values

# define values for NAs
ind_na<- apply(X = test_set,MARGIN = 2,FUN = function(x){
  any(is.na(x))
})

# features with NAs
na_cols<- names(test_set[,ind_na])

# replace NAs which corresponds to 0
test_set[,num_feat_0]<- apply(test_set[,num_feat_0], MARGIN = 2, FUN = function(x){
  replace(x, is.na(x), 0)
})

# replace NAs which corresponds to none
test_set[,char_feat_none]<- apply(test_set[,char_feat_none], MARGIN = 2,FUN = function(x){
  replace(x, is.na(x),"none")
})

# replace NAs for factors with the most common value
test_set[,char_feat_mode]<- apply(test_set[,char_feat_mode], MARGIN = 2,FUN = function(x){
  replace(x, is.na(x), names(which.max(table(x))))
})

# other NAs
test_set<- test_set %>% mutate(GarageYrBlt = ifelse(is.na(GarageYrBlt),YearBuilt,GarageYrBlt))

# drop 1stFlrSF and 2ndFlrSF
test_set<- test_set %>% select(-c(`1stFlrSF`,`2ndFlrSF`))

# drop GarageCars since it is just similar to GarageArea
```

```r
test_set<- test_set %>% select(-GarageCars)

# remove Id
test_set<- test_set %>% select(-Id)

# match class of feature based on information on data
num_to_factor<- c("MSSubClass","OverallQual","OverallCond")
test_set[num_to_factor]<- lapply(test_set[num_to_factor], FUN = factor)

# convert characters to factors
char_to_factor<-lapply(test_set,class)
char_to_factor_list<- names(char_to_factor[which(char_to_factor=="character")])

test_set[char_to_factor_list]<- lapply(test_set[char_to_factor_list], FUN = factor)

# convert dates to num
date_to_num<- c("YearBuilt","GarageYrBlt","YearRemodAdd","MoSold","YrSold")
test_set[date_to_num]<- apply(test_set[date_to_num],
                              MARGIN = 2,
                              FUN = as.numeric)

# transform sale price to log10
test_set<- test_set %>%
  mutate(SalePrice = log10(SalePrice))

# predict SalePrice with the test_set containing only the 15 most imporant features (based on lm model)
test_set_lm<- test_set %>% select(imp_lm,SalePrice)
levels(test_set_lm$OverallQual)<- levels(train_set_lm$OverallQual)
levels(test_set_lm$MSSubClass)<- levels(train_set_lm$MSSubClass)

SalePrice_predicted_lm<- predict(fit_lm,newdata = test_set_lm %>% select(-SalePrice))

rmse_lm<- RMSE(10^SalePrice_predicted_lm,10^test_set_lm$SalePrice)

## train model - Random Forest ----
train_set_rf<- train_set
tune_grid<- expand.grid(predFixed= seq(round(final_dim[2]/3),42,3),
                        minNode = seq(2,20,2))

control<- trainControl(method="oob",
                number=20,
                p=0.9)

train_rf<- train(SalePrice~.,
                 method="Rborist",
                 data = train_set_rf,
                 tuneGrid= tune_grid,
                 trControl= control,
                 nTree=500)

# best tune
train_rf$bestTunetTune
```

```r
# create model using best tuning parameters
fit_rf<- Rborist(x = train_set_rf %>% select(-SalePrice),
                 y = train_set_rf$SalePrice,
                 predFixed=train_rf$bestTune$predFixed,
                 minNode=train_rf$bestTune$minNode)

# compute RMSE of the model based on the train data
rmse_train_rf<- RMSE(10^fit_rf$validation$yPred,10^train_set_rf$SalePrice)
rmse_train_rf

# Obtain importance of each feature
varImp(train_rf)

# Select 15 most important variables
imp_rf<- c("GrLivArea","TotalBsmtSF","GarageArea","YearBuilt","Foundation","FireplaceQu","KitchenQual",

# redefine train set based on the 15 most important features
train_set<- train_set %>% select(imp_rf, SalePrice)

# training set dimensions
final_dim_imp<- dim(train_set)

# train model
tune_grid<- expand.grid(predFixed= seq(round(15/3),14,1),
                        minNode = seq(2,20,2))

control<- trainControl(method="oob",
                       number=20,
                       p=0.9)

train_rf<- train(SalePrice~.,
                 method="Rborist",
                 data = train_set,
                 tuneGrid= tune_grid,
                 trControl= control,
                 nTree=500)

# create model using best tuning parameters
fit_rf<- Rborist(x = train_set %>% select(-SalePrice),
                 y = train_set$SalePrice,
                 predFixed=train_rf$bestTune$predFixed,
                 minNode=train_rf$bestTune$minNode)

# compute RMSE of the model based on the train data
rmse_train_imp_rf<- RMSE(10^fit_rf$validation$yPred,10^train_set$SalePrice)
rmse_train_imp_rf

# redefine test set with the 15 most important features (based on rf)
test_set_rf<- test_set %>% select(imp_rf, SalePrice)

# predict sale price of test set using train_rf with the optimized parameters
SalePrice_predicted_rf<- predict(fit_rf, newdata = test_set_rf %>% select(-SalePrice))
```

```r
# compute RMSE of the antilogs of the predicted and true sale prices
rmse_rf<- RMSE(10^SalePrice_predicted_rf$yPred, 10^test_set$SalePrice)
rmse_rf

# summary of RMSEs for the two models -----
data.frame(model = c("linear regression","random forest"), training_RMSE = c(rmse_train_imp_lm, rmse_tra
```