Durga Siva Lokesh Telaprolu
Student ID: 923065881
5/17/2023

Math-448 Term Project Report

House Price Prediction

Contents

Introduction/Objective:

Real estate investments often appear to be profitable because the values of the properties do not drop off quickly. Many individuals are interested in real estate investments and the demand for the sale of real estate has dramatically increased since 2011. Identifying the exact formula for predicting price has been a concern for sellers and buyers which will involve many factors such as the number of bedrooms, bathrooms, square feet living, number of floors, year built, and city. To aid buyers and sellers in making decisions, it is crucial to predict property values without bias. My objective for this project is Prediction, where I wish to estimate the price based on the predictor variables. The study's conclusions can be helpful to both developers and property buyers.

Description of the Data:

The title of the dataset is "House Price Prediction" from the Kaggle data set repository. It contains information of 4600 house prices along with their characteristics. It has 17 predictor variables which includes date, the number of bedrooms, bathrooms, square foot living, square feet lot, floors, waterfront, view, condition, square feet above, square feet basement, year built, year renovated, street, city, state zip, and country.
In my project, I will be using 3220 (70% of the total records) for training the model and the remaining 1380 (30% of the total records) for testing the model. Each observation has all 18 columns of information and there are no missing data entries. Of these predictor variables, I plan to focus on the predicting the response variable (Price). A detailed explanation of all the variables can be found in the next section.

Explanation of Variables and Feature Selection:

The characteristics include the following: date (probably the time the when the data about the house was gathered), house price (expressed in dollars), number of bedrooms, number of bathrooms, living area, plot size, number of floors, location by the sea, view (on a scale from 0 to 4), condition of the house (on a scale from 1 to 5), ground area of the house, basement area (if any), year of construction, year of renovation and area is given in square feet. As there are no missing values in my data set, I'll go ahead with data transformation and feature selection.

Summary:

There is no way that all the elements will be helpful for modeling. As a result, we go on to feature reduction. Like the address, the time feature doesn't add anything to the research, thus we delete it from our database. We don't use the information about the number of floors in our future investigation because it appears to be rare (not integer numbers) and somewhat overlaps the information about the area of the house. Similar circumstances apply to the features "view" and "waterfront" (almost all cases have the value equal to 0). Also, it makes no sense to compare the basement area when more than half of the houses doesn't have basement. The variable 'year of construction' can be transformed into age.

Output:

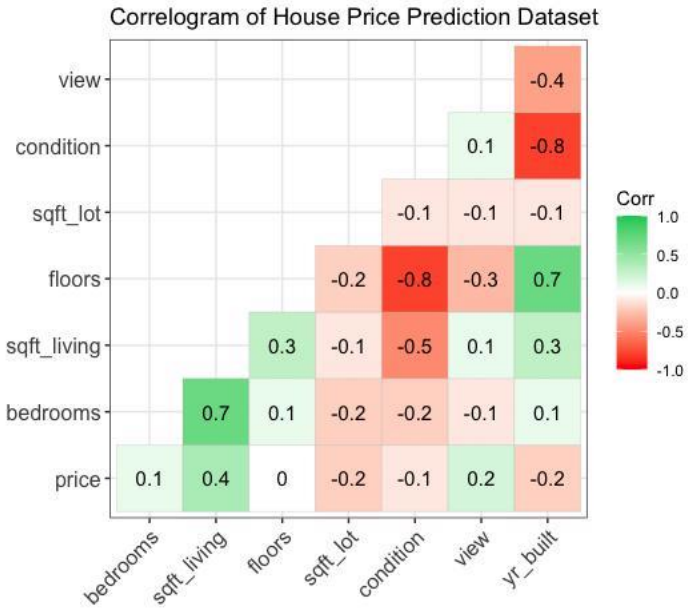
```
     price              bedrooms      bathrooms      sqft_living
 Min.   :       0   Min.   :0.000   Min.   :0.000   Min.   :  370
 1st Qu.:  322875   1st Qu.:3.000   1st Qu.:1.750   1st Qu.: 1460
 Median :  460943   Median :3.000   Median :2.250   Median : 1980
 Mean   :  551963   Mean   :3.401   Mean   :2.161   Mean   : 2139
 3rd Qu.:  654962   3rd Qu.:4.000   3rd Qu.:2.500   3rd Qu.: 2620
 Max.   :26590000   Max.   :9.000   Max.   :8.000   Max.   :13540
    sqft_lot           floors        waterfront           view
 Min.   :    638   Min.   :1.000   Min.   :0.000000   Min.   :0.0000
 1st Qu.:   5001   1st Qu.:1.000   1st Qu.:0.000000   1st Qu.:0.0000
 Median :   7683   Median :1.500   Median :0.000000   Median :0.0000
 Mean   :  14852   Mean   :1.512   Mean   :0.007174   Mean   :0.2407
 3rd Qu.:  11001   3rd Qu.:2.000   3rd Qu.:0.000000   3rd Qu.:0.0000
 Max.   :1074218   Max.   :3.500   Max.   :1.000000   Max.   :4.0000
   condition       sqft_above    sqft_basement       yr_built     yr_renovated
 Min.   :1.000   Min.   : 370   Min.   :   0.0   Min.   :1900   Min.   :   0.0
 1st Qu.:3.000   1st Qu.:1190   1st Qu.:   0.0   1st Qu.:1951   1st Qu.:   0.0
 Median :3.000   Median :1590   Median :   0.0   Median :1976   Median :   0.0
 Mean   :3.452   Mean   :1827   Mean   : 312.1   Mean   :1971   Mean   : 808.6
 3rd Qu.:4.000   3rd Qu.:2300   3rd Qu.: 610.0   3rd Qu.:1997   3rd Qu.:1999.0
 Max.   :5.000   Max.   :9410   Max.   :4820.0   Max.   :2014   Max.   :2014.0
```

The features selected for modeling:

- price, bedrooms, sqft_living, floors, sqft_lot, condition, yr_built_age.

Data Visualization:

Correlation plot of selected Features:

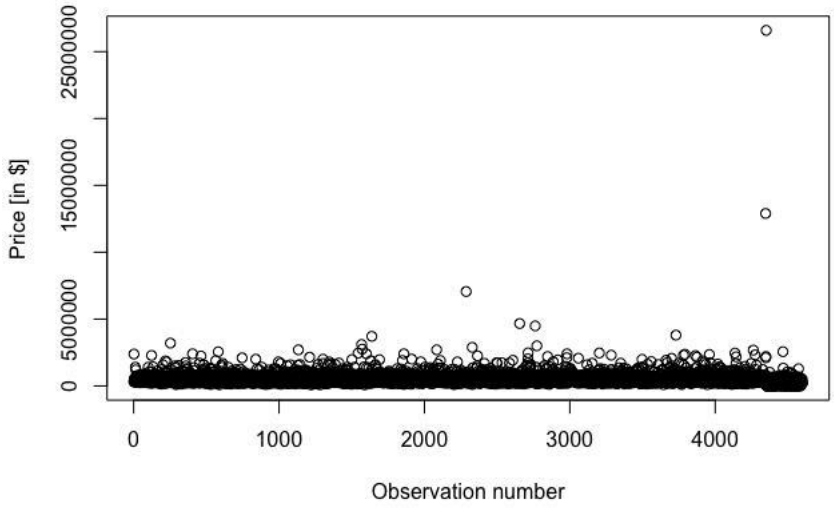


Observations from correlation plot:

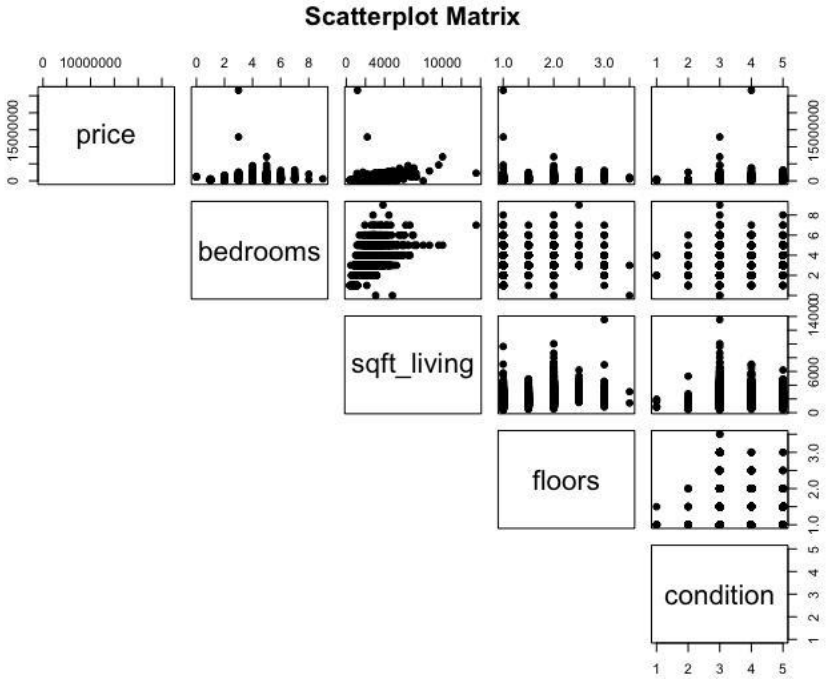
From the correlogram it's clearly visible that the variables ('bedrooms', 'sqft_living') , ('floors','yr_bulit')are positively correlated strongly. And the variables ('condition', 'yr_built'),

('condition',' floors') are negatively correlated strongly. The remaining variables have no significant correlation among them.

Value of Dependent Variable Price vs no: of observations:



As per the graph, it's clearly evident that majority of the observations are under 1 million dollars. Few of the values stand out from the rest having higher price. As a result, we decide to discard the cases where the cost of the home exceeds $1 million because they are extremely uncommon in the context of the study in the issue (the possible seller or buyer will not be interested in such expensive housing). There are 389 such houses and after eliminating them, we have 4,211 cases.
Scatterplot Matrix:

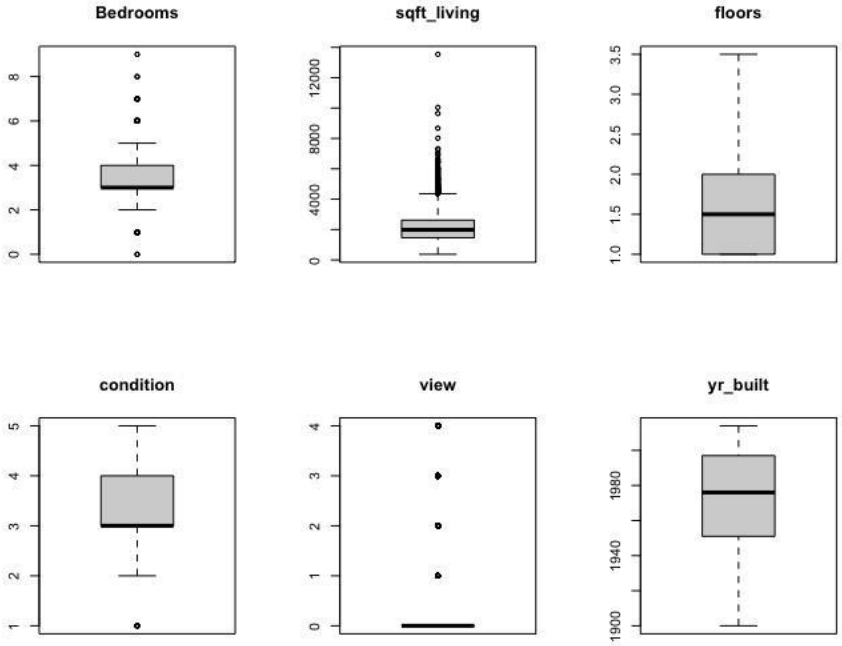


Observations:

Scatterplot Matrix helps us in identifying the relation between the response variable, Price and the other explanatory models. Also, it depicts the correlation of the predictor variables as well. If we take a close look at scatterplot, response variable doesn't correlates strongly with any of the predictors.

However, variables like (‘bedrooms’, ‘sqft_living’) and (‘floors’,’yr_bulit’) have strongly correlated. This gives us a basic intuition like which variables to be included in model building.

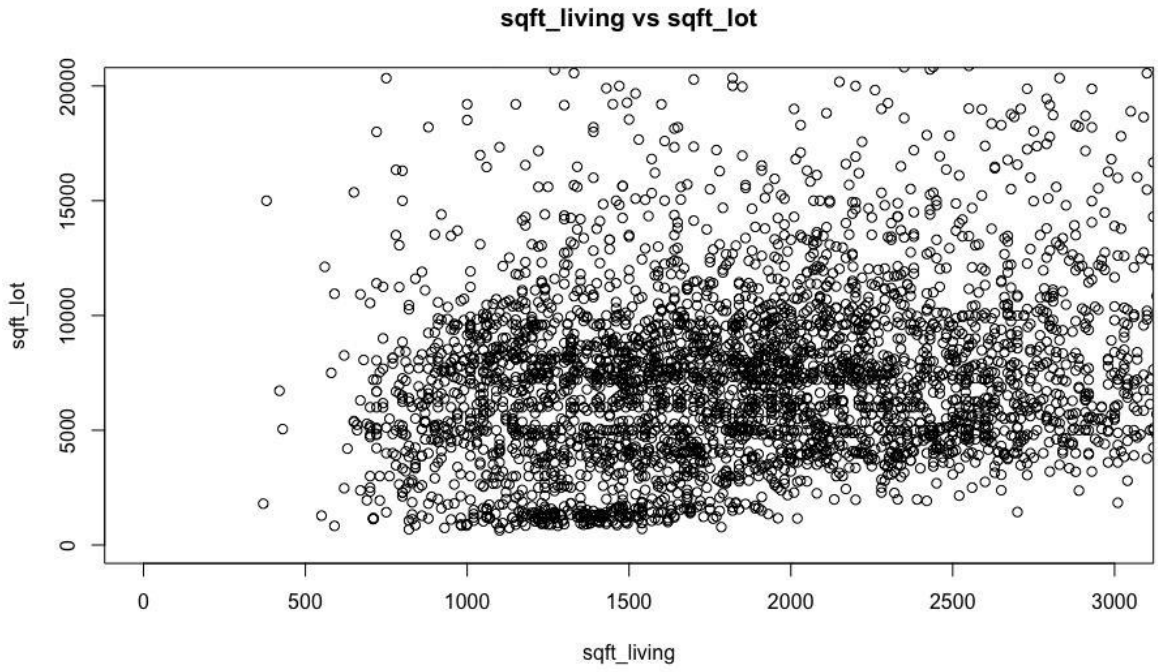
Boxplot for checking outliers:



Observations:

The variable ‘view’ is no longer significant as it mostly contains outliers(with 0 values). The variables, ‘bedrooms’ and ‘sqft_living’ have some unrealistic data and rows of such can be removed.

Relation between Square foot Living and Square foot Lot:

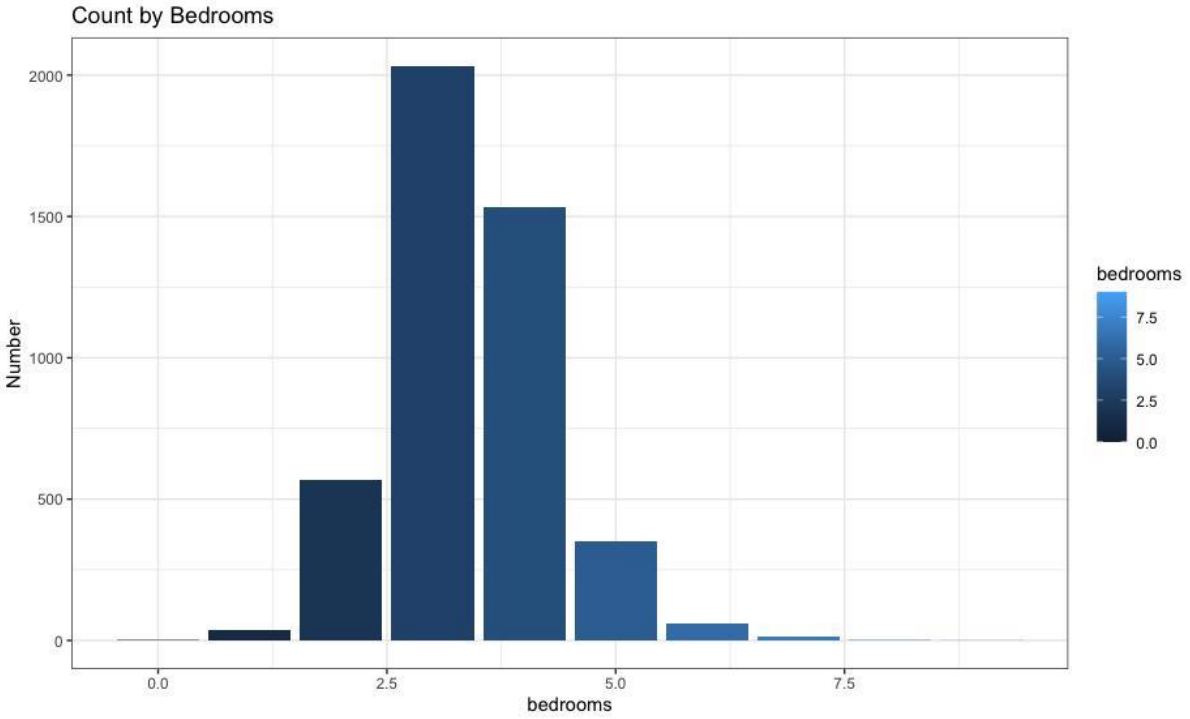


Observation:

Plot has strong negative correlation. As the square foot lot increases square foot living decreases. So, both the variables must be considered while modelling.

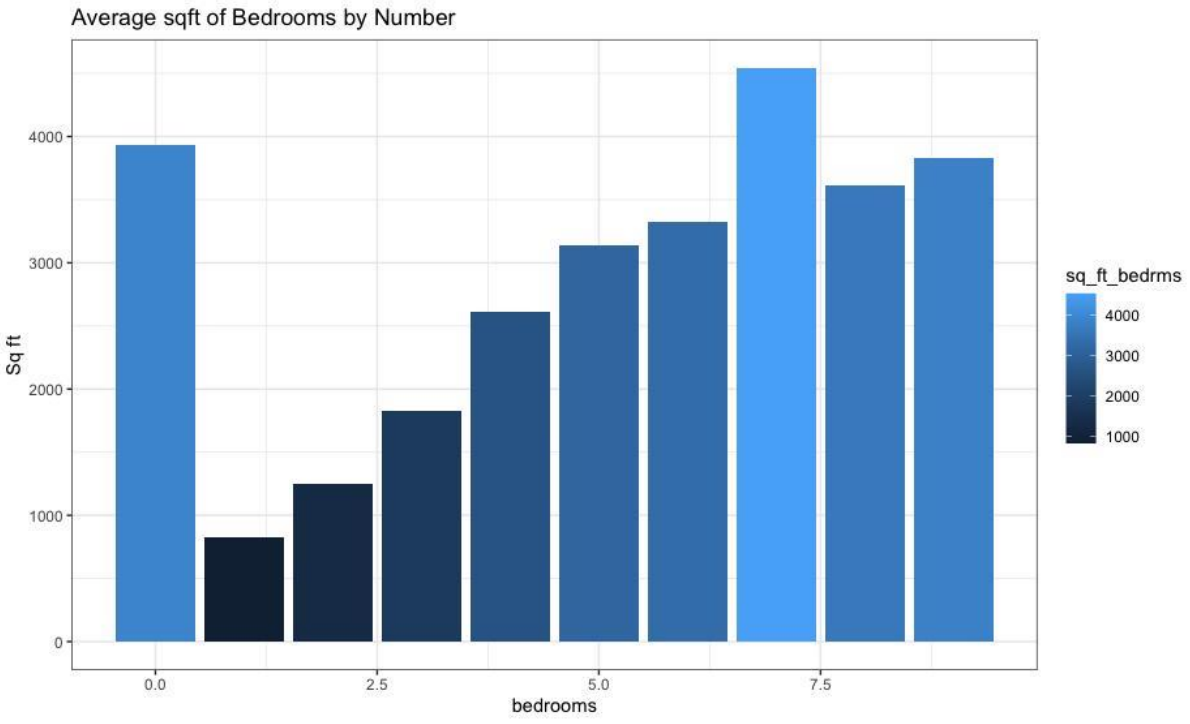
Bar Graph Representation:

Count by Bedrooms:

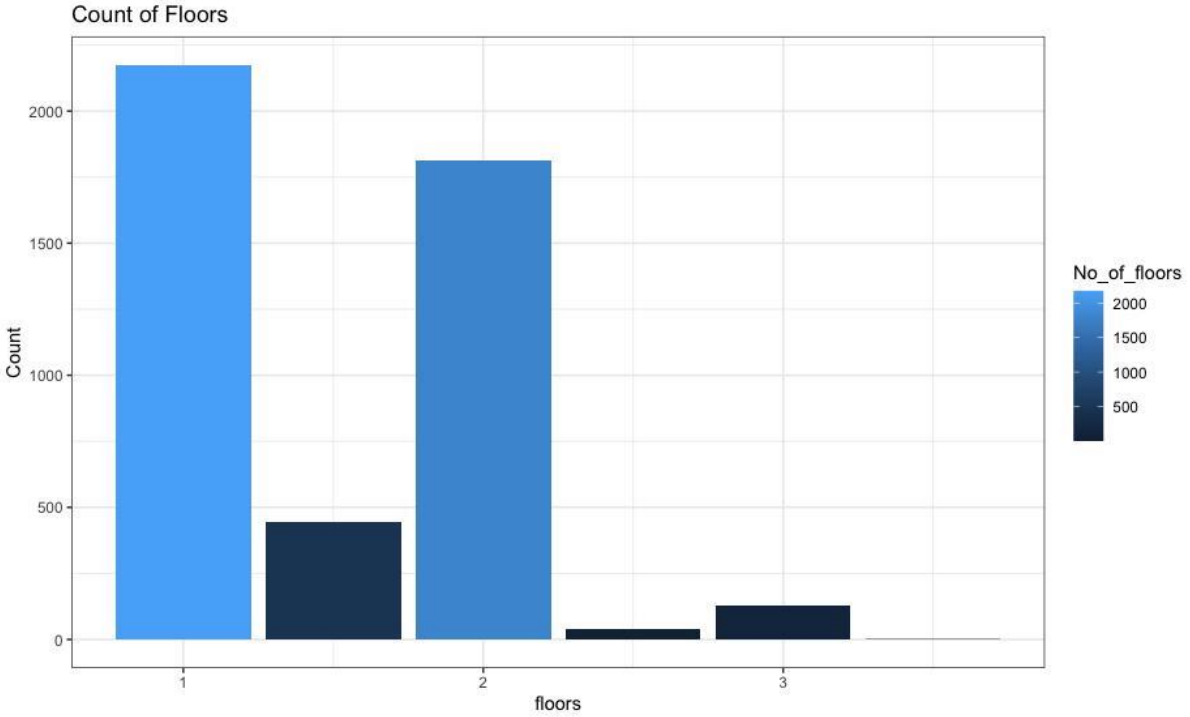


Majority of the houses have bedrooms in the range of 2 to 5.

Average Sqft by number of bedrooms:

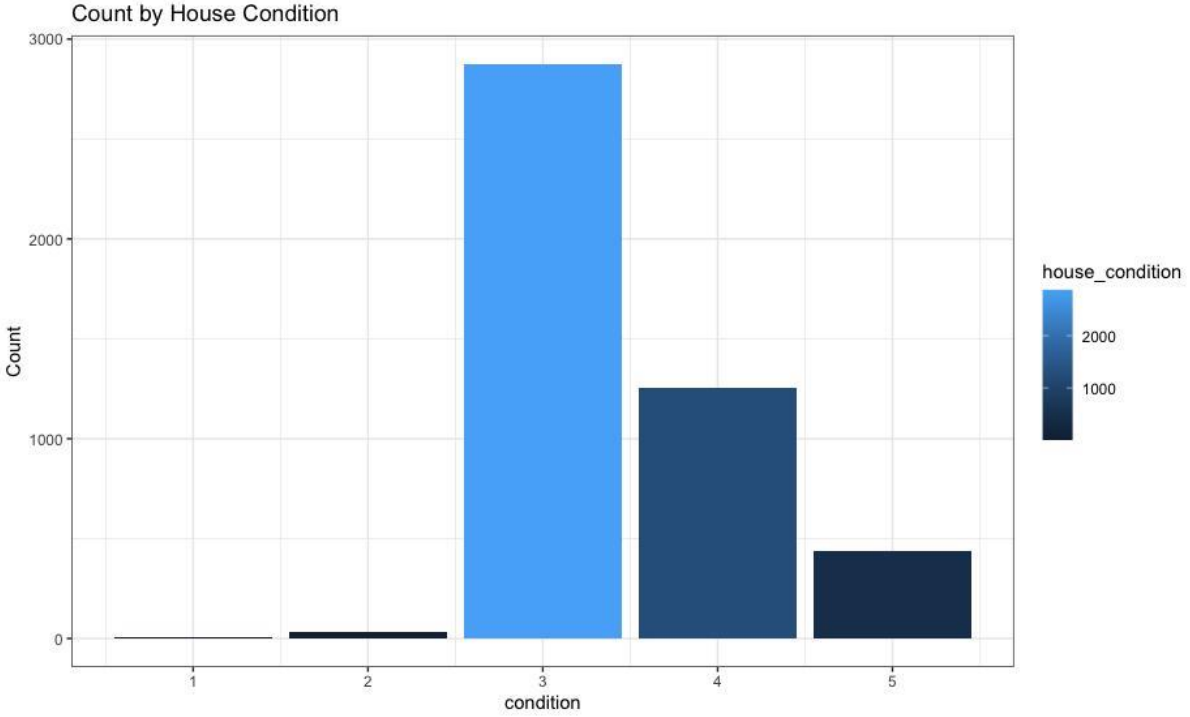


Count of floors:



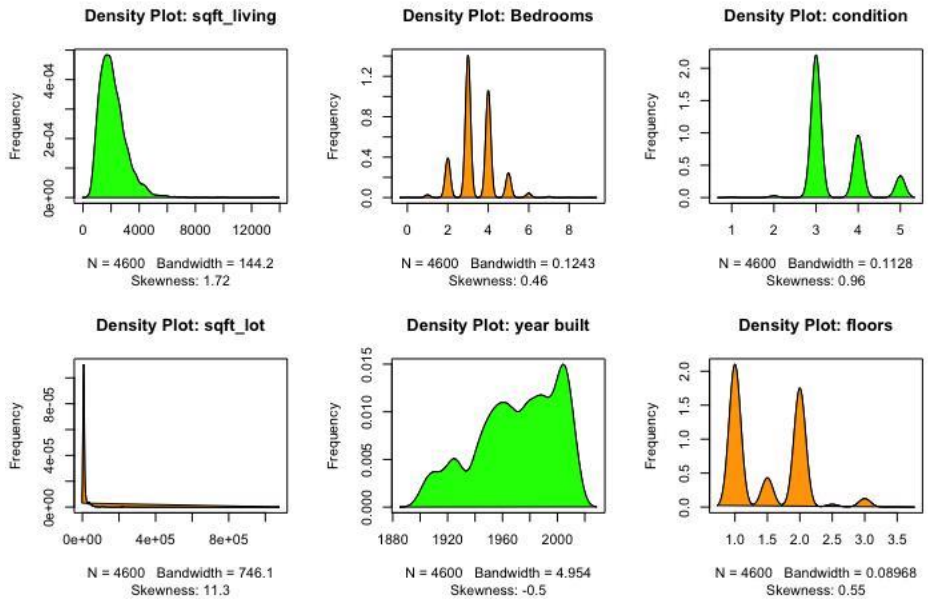
It is evident that 2174 houses have only one floor, while 1811 houses have two floors. Very few houses have 3 floors which approximates to around 130.

Count by condition of the house:



2875 houses were rated 3 out of 5. Similarly, 1252 and 455 houses were given 4 and 5 respectively.

Density Plots of selected features:



The variables we have in the data set are not normally distributed. However, its quite evident that most of the density plots are continuously distributed. The plot of square foot living has a bell-shaped curve which is slightly normally distributed.

Model Selection:

In my project, I will be using 3220 (70% of the total records) for training the model and the remaining 1380 (30% of the total records) for testing the model. Lastly, I'll compare all the models by evaluating R squared, RMSE.

Linear Regression:

Linear regression is a statistical method for finding/ modelling the relationship between a dependent variable and one or more independent variables. This method assumes a linear relationship between the independent and dependent variables, and tries to find the best-fit line that reduces the difference between the observed and predicted values. The resulting linear regression model can then be used to make predictions about the dependent variable given new values of the independent variables.

RMSE value of Linear Regression model is 0.384.

Regularized Linear Models:

Regularized linear models are a class of linear regression models that incorporate a penalty term into the loss function in order to prevent overfitting. The penalty term works to prevent the model from fitting the training data too closely, which can increase the model's capacity to generalize to new, unexplored data.

L1 regularization, also known as Lasso regularization, and L2 regularization, also known as Ridge regularization, are the two main methods of regularization frequently applied to linear regression models. The penalty term introduced by L1 regularization, which is proportional to the absolute value of the model's coefficients, can cause some coefficients to be set to absolutely zero. This effectively removes pointless features from the model, which can be valuable for feature selection.

L2 regularization adds a penalty term proportional to the square of the model's coefficients, and it tends to shrink the coefficients towards zero without setting them exactly to zero. This prevents overfitting by reducing the model's reliance on any particular set of features. Elastic Net regularization, which enables the combination of feature selection and coefficient shrinking, combines both L1 and L2 regularization.

After comparing these three regularization methods, each predicted the almost the same RMSE and R squared values which are around 0.39 and 0.49 respectively.

Decision Tree:

A decision tree is a popular method for both classification and regression tasks. It works by repeatedly partitioning the data into subsets according to the values of the input features until a stopping requirement is met. The result is a structure resembling a tree where each leaf node is a class label or a predicted value and each internal node is a decision based on the value of a feature.

The decision tree method has capacity to handle both numerical and categorical data and its interpretability. On datasets with high dimensionality, however, it could not perform effectively due to overfitting. To prevent overfitting when using decision trees, it's crucial to carefully select the splitting criterion and pruning method. In order to enhance the performance of decision trees, ensemble techniques like random forests and gradient boosting are frequently used.

RMSE value of decision tree model is 0.404.

Random Forest:

Random Forest is a popular ensemble learning technique which is widely used for both classification and regression tasks. It is made up of a number of decision trees that were trained on various data subsets using random feature subsets. In order to reduce overfitting and boost the robustness of the model, the method chooses a random subset of features at each split in the tree during training.

Additionally, they provide insight into the relative significance of various features in forecasting the target variable. The algorithm can be tuned using hyperparameters such as the

number of trees, the maximum depth of the trees, and the minimum number of samples required to split a node, among others.

Compared to Regularized linear methods, Random forests fits the data well with a R squared value of 0.482. And RMSE value of Random Forest(0.404) is better compared to Decision Tree (0.450).

Support vector Machines:

Support Vector Machines (SVMs) are a powerful class of machine learning models widely used for classification tasks. They have proven to be highly effective in handling complex datasets with both linear and non-linear decision boundaries. SVMs aim to find the optimal hyperplane that can separate data points belonging to different classes with the maximum possible margin.

The key idea behind SVMs is to identify the support vectors, which are the data points closest to the decision boundary. By focusing on these critical points, SVMs are able to construct a decision boundary that is resilient to noise and can generalize well to unseen data. This characteristic makes SVMs particularly useful in real-world scenarios where data can be noisy or exhibit complex relationships.

One of the notable advantages of SVMs is their ability to handle high-dimensional feature spaces, making them suitable for datasets with a large number of features. Additionally, SVMs employ kernel functions to implicitly transform the input data into higher-dimensional spaces, allowing them to capture intricate relationships that may not be easily discernible in the original feature space.

Training an SVM involves finding the optimal hyperplane by solving an optimization problem. The objective is to minimize a cost function that penalizes misclassifications while maximizing the margin. Various optimization algorithms, such as Sequential Minimal Optimization (SMO) or quadratic programming, can be used to efficiently solve this problem and obtain the optimal solution.

It is worth noting that SVMs offer robustness against overfitting, thanks to the margin maximization objective. However, the performance of SVMs heavily relies on appropriate parameter tuning, such as the choice of the kernel function and regularization parameter. Careful consideration and fine-tuning of these hyperparameters are necessary to achieve optimal results with SVMs.

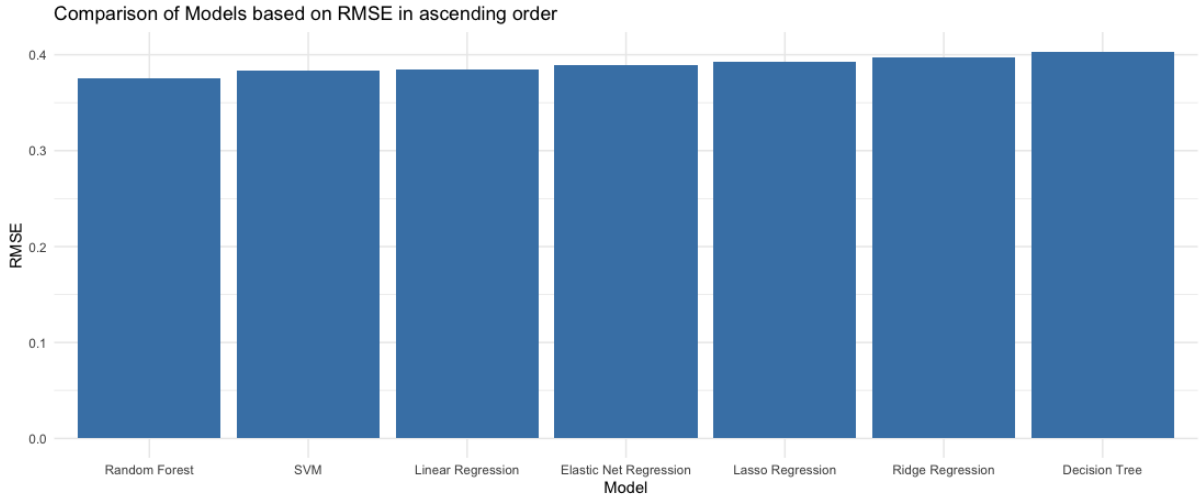
RMSE and R squared values for the SVM model are 0.384 and 0.505 respectively.

Evaluation Metrics Comparison:

Algorithm	Metrics	Algorithm with Default Setting
Linear Regression	RMSE	0.384
	R squared	0.501
Ridge Regression	RMSE	0.397
	R squared	0.489
Lasso Regression	RMSE	0.392
	R squared	0.490
Elastic Net	RMSE	0.390
	R squared	0.496
Decision Tree	RMSE	0.404
	R squared	0.404
Random Forest	RMSE	0.376
	R squared	0.523
SVM	RMSE	0.384
	R squared	0.505

	Model	RMSE	R_squared
	<chr>	<dbl>	<dbl>
1	Random Forest	0.376	0.523
2	SVM	0.384	0.505
3	Linear Regression	0.384	0.501
4	Elastic Net Regression	0.390	0.496
5	Lasso Regression	0.392	0.490
6	Ridge Regression	0.397	0.489
7	Decision Tree	0.404	0.450

Plot Representation:



Conclusion:

Based on the analysis conducted on various algorithms for house price prediction, we can draw several conclusions. Firstly, the performance of the linear regression algorithm with default settings yielded a root mean squared error (RMSE) of 0.384 and an R-squared value of 0.501.

Among the regularization techniques, both Ridge and Lasso regression exhibited similar performance, with slightly higher RMSE values of 0.397 and 0.392, respectively, and R-squared values of 0.489 and 0.490. These techniques proved to be effective in reducing overfitting and improving generalization.

The Elastic Net algorithm demonstrated slightly better results compared to Ridge and Lasso regression, with an RMSE of 0.390 and an R-squared value of 0.496. This hybrid

approach combining L1 and L2 regularization effectively balanced the benefits of both techniques.

In terms of decision tree-based algorithms, the Decision Tree model had an RMSE of 0.404 and an R-squared value of 0.404. While this algorithm can capture complex relationships within the data, it may struggle with overfitting.

The Random Forest algorithm, on the other hand, outperformed other algorithms with an RMSE of 0.376 and an impressive R-squared value of 0.523. By aggregating multiple decision trees, this ensemble method achieved better accuracy and robustness.

Lastly, the Support Vector Machine (SVM) algorithm produced results similar to linear regression, with an RMSE of 0.384 and an R-squared value of 0.505. SVM is known for its ability to handle high-dimensional data and nonlinear relationships.

In conclusion, the Random Forest algorithm demonstrated the best predictive performance among the models evaluated, exhibiting the lowest RMSE and highest R-squared value. Therefore, it is recommended to utilize the Random Forest algorithm for house price prediction tasks, as it provides accurate and reliable estimations.

Appendix: R Code Used for Analysis:

```
library(glmnet)
library(tidyverse)
library(tidyr)
library(tidyselect)
library(plotly)
library(dplyr)
library(corrplot)
library(ggcorrplot)
library(htmlwidgets)
library('IRdisplay')
library(caTools)
library(rpart)
library(rpart.plot)
library(caret)
library(e1071)
library(caret)
library(ggplot2)

# Train random forest model

install.packages("randomForest")
library(randomForest)

house_data=read.csv('~/Desktop/Syllabus/448- Stat/data.csv')
head(house_data)
tail(house_data)
```

```r
dim(house_data)
colnames(house_data)
str(house_data)
summary(house_data)
unique(house_data$city)

house_data$yr_built_age <- as.integer(format(Sys.Date(), "%Y")) - house_data$yr_built

drops <- c("yr_built")
house_data = house_data[ , !(names(house_data) %in% drops)]
head(house_data)

house_data$price=log(house_data$price)

selected_features=house_data[,c("price","bedrooms","sqft_living","floors",
                                "sqft_lot", "condition","yr_built_age")]

summary(selected_features)
sum(is.na(house_data))
correlation_matrix=cor(selected_features)
correlation_matrix
#write.csv(correlation_matrix, file="correlation_matrix.csv",row.names = FALSE)
corrplot(correlation_matrix)
corr=round(cor(correlation_matrix),1)
ggcorrplot(corr,
           type = "lower",
           lab = TRUE,
           lab_size = 4,
           colors = c("red", "white", "springgreen3"),
           title="Correlogram of House Price Prediction Dataset",
           ggtheme=theme_bw)
options(scipen = 999)
plot(house_data$price, xlab = "Observation number", ylab = "Price [in $]")
pairs(price~bedrooms + sqft_living + floors + condition, data =house_data,
      main = "Scatterplot Matrix",lower.panel=NULL,pch=19)
par(mfrow=c(2, 3))
boxplot(house_data$bedrooms, main="Bedrooms")
boxplot(house_data$sqft_living, main="sqft_living")
boxplot(house_data$floors, main="floors")
boxplot(house_data$condition, main="condition")
boxplot(house_data$view, main="view")
boxplot(house_data$yr_built, main="yr_built")

plot(x = house_data$sqft_living, y = house_data$sqft_lot,
     xlab = "sqft_living",
     ylab = "sqft_lot",
     xlim = c(0, 3000),
     ylim = c(0, 20000),
     main = "sqft_living vs sqft_lot"
)
bedrooms_count=house_data%>%group_by(bedrooms)%>%summarise(Number=length(bedrooms))
```

```
head(bedrooms_count)

v1<-ggplot(bedrooms_count, aes(bedrooms, Number, fill=bedrooms)) + geom_bar(stat='Identity') +
labs(title='Count by Bedrooms') + theme_bw()
v1

avg_sqft_living<-house_data%>%group_by(bedrooms)%>%summarise(sq_ft_bedrms=mean(sqft_living))

head(avg_sqft_living)

v3<-ggplot(avg_sqft_living, aes(bedrooms, sq_ft_bedrms, fill=sq_ft_bedrms)) + geom_bar(stat='Identity') +
labs(title='Average sqft of Bedrooms by Number', y='Sq ft') + theme_bw()

v3
no_of_floors<-house_data%>%group_by(floors)%>%summarise(No_of_floors=length(floors))

no_of_floors

v5<-ggplot(no_of_floors, aes(floors, No_of_floors, fill=No_of_floors)) + geom_bar(stat='Identity') +
labs(title='Count of Floors', y='Count') +  theme_bw()
v5

condition<-house_data%>%group_by(condition)%>%summarise(house_condition=length(condition))

condition

v8<-ggplot(condition, aes(condition, house_condition, fill=house_condition)) + geom_bar(stat='Identity') +
labs(title='Count by House Condition', y='Count') + theme_bw()
v8

library(e1071)

par(mfrow=c(2, 3))

plot(density(house_data$sqft_living), main="Density Plot: sqft_living", ylab="Frequency",
    sub=paste("Skewness:", round(e1071::skewness(house_data$sqft_living), 2)))
polygon(density(house_data$sqft_living), col="green")

plot(density(house_data$bedrooms), main="Density Plot: Bedrooms", ylab="Frequency",
    sub=paste("Skewness:", round(e1071::skewness(house_data$bedrooms), 2)))
polygon(density(house_data$bedrooms), col="orange")


plot(density(house_data$condition), main="Density Plot: condition", ylab="Frequency",
    sub=paste("Skewness:", round(e1071::skewness(house_data$condition), 2)))
polygon(density(house_data$condition), col="green")


plot(density(house_data$sqft_lot), main="Density Plot: sqft_lot", ylab="Frequency",
    sub=paste("Skewness:", round(e1071::skewness(house_data$sqft_lot), 2)))
```

```r
polygon(density(house_data$sqft_lot), col="orange")

plot(density(house_data$yr_built), main="Density Plot: year built", ylab="Frequency",
    sub=paste("Skewness:", round(e1071::skewness(house_data$yr_built), 2)))
polygon(density(house_data$yr_built), col="green")

plot(density(house_data$floors), main="Density Plot: floors", ylab="Frequency",
    sub=paste("Skewness:", round(e1071::skewness(house_data$floors), 2)))
polygon(density(house_data$floors), col="orange")


# Split data into training and validation sets
smp_size <- floor(0.7 * nrow(selected_features))
set.seed(123)
train_ind <- sample(seq_len(nrow(selected_features)), size = smp_size)
training_data <- selected_features[train_ind, ]
validation_data <- selected_features[-train_ind, ]

training_data= subset(training_data, price> 0)

validation_data= subset(validation_data, price> 0)


# Train linear regression model
linear_model <- lm(price ~ bedrooms + sqft_living + floors + sqft_lot +
                   condition+ yr_built_age, data = training_data)
plot(linear_model)

# Train decision tree model
tree_model <- rpart(price ~ bedrooms + sqft_living + floors + sqft_lot +
                    condition  + yr_built_age, data = training_data, method = "anova")

rpart.plot(tree_model)

random_forest_model <-randomForest(price ~ bedrooms + sqft_living + floors + sqft_lot +
                                   condition + yr_built_age, data = training_data, ntree = 100)

plot(random_forest_model)

# Fit Ridge regression model
ridge_model <- cv.glmnet(
  x = as.matrix(training_data[, -1]),
  y = training_data$price,
  alpha = 0,
  family = "gaussian"
)

plot(ridge_model)

# Fit Lasso regression model
lasso_model <- cv.glmnet(
```

```
  x = as.matrix(training_data[, -1]),
  y = training_data$price,
  alpha = 1,
  family = "gaussian"
)

plot(lasso_model)

# Fit Elastic Net regression model
elastic_net_model <- cv.glmnet(
  x = as.matrix(training_data[, -1]),
  y = training_data$price,
  alpha = 0.5,
  family = "gaussian"
)

plot(elastic_net_model)
# Train the SVM model

svm_model <- svm(price ~ ., data = training_data, kernel = "radial")


#Step 5: Evaluate the models using RMSE and R-squared

# Make predictions
linear_preds <- predict(linear_model, validation_data)
tree_preds <- predict(tree_model, validation_data)
random_forest_preds <- predict(random_forest_model, validation_data)

# Make predictions for Ridge, Lasso,  Elastic Net, and SVM models
ridge_preds <- predict(ridge_model, newx = as.matrix(validation_data[, -1]))
lasso_preds <- predict(lasso_model, newx = as.matrix(validation_data[, -1]))
elastic_net_preds <- predict(elastic_net_model, newx = as.matrix(validation_data[, -1]))
svm_preds <- predict(svm_model, validation_data)


# Calculate RMSE
linear_rmse <- sqrt(mean((validation_data$price - linear_preds)^2))
tree_rmse <- sqrt(mean((validation_data$price - tree_preds)^2))
random_forest_rmse <- sqrt(mean((validation_data$price - random_forest_preds)^2))


# Calculate RMSE for Ridge, Lasso,  Elastic Net, and SVM models
ridge_rmse <- sqrt(mean((validation_data$price - ridge_preds)^2))
lasso_rmse <- sqrt(mean((validation_data$price - lasso_preds)^2))
elastic_net_rmse <- sqrt(mean((validation_data$price - elastic_net_preds)^2))
svm_rmse <- sqrt(mean((validation_data$price - svm_preds)^2))


# Calculate R-squared
linear_r2 <- cor(validation_data$price, linear_preds)^2
tree_r2 <- cor(validation_data$price, tree_preds)^2
```

```r
random_forest_r2 <- cor(validation_data$price, random_forest_preds)^2

# Calculate R-squared for Ridge, Lasso, Elastic Net, and SVM models
ridge_r2 <- cor(validation_data$price, ridge_preds)^2
lasso_r2 <- cor(validation_data$price, lasso_preds)^2
elastic_net_r2 <- cor(validation_data$price, elastic_net_preds)^2
svm_r2 <- cor(validation_data$price, svm_preds)^2


# Create a tibble (dataframe) to store model performance metrics
model_comparison <- tibble(
  Model = c("Linear Regression", "Decision Tree", "Random Forest", "Ridge Regression", "Lasso Regression",
"Elastic Net Regression","SVM"),
  RMSE = c(linear_rmse, tree_rmse, random_forest_rmse, ridge_rmse, lasso_rmse, elastic_net_rmse,svm_rmse),
  R_squared = c(linear_r2, tree_r2, random_forest_r2, ridge_r2, lasso_r2, elastic_net_r2,svm_r2)
)


# Display model comparison
model_comparison_sorted <- model_comparison %>%
  arrange(RMSE)

print(model_comparison_sorted)


ggplot(model_comparison_sorted, aes(x = reorder(Model, RMSE), y = RMSE)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(x = "Model", y = "RMSE") +
  ggtitle("Comparison of Models based on RMSE in ascending order") +
  theme_minimal()


model_comparison_sorted <- model_comparison %>%
  arrange(desc(R_squared))

# Plot the R squared values for each model in descending order
ggplot(model_comparison_sorted, aes(x = reorder(Model, -R_squared), y = R_squared)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(x = "Model", y = "R squared") +
  ggtitle("Comparison of Models based on R squared in descending order") +
  theme_minimal()
```