Housing - Price Prediction

By:

Chandana K

CONTENT:

- Problem Statement
- What is Housing Price Prediction
- Steps followed
- Importing data
- Exploratory data analysis
- Data Processing
- Visualizations
- Data cleaning
- Model Building
- Hyper Parameter Tunning
- Conclusion.

PROBLEM STATEMENT:

Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world’s economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company

WHAT IS HOUSING PRICE PREDICTION

Prediction of house prices in simple terms is predicting the prices of homes as close to selling price as possible. House Price prediction, is important to drive Real Estate efficiency. As earlier, House prices were determined by calculating the acquiring and selling price in a locality.

STEPS FOLLOWED

- Importing the dataset
- Exploratory Data Analysis (EDA)
- Visualization
- Checking for Outliers and skewness
- Removing outliers and skewness
- Pre – processing
- Model building
- Hyper parameter tuning
- Model saving

IMPORTING THE DATASET

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

1168 rows × 81 columns

Train dataset has the 1168 rows and 81 columns

Test dataset has the 292 rows and 80 columns

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC	Fence	MiscFea
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	...	0	0	NaN	NaN	
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	...	0	0	NaN	NaN	
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
287	83	20	RL	78.0	10206	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
288	1048	20	RL	57.0	9245	Pave	NaN	IR2	Lvl	AllPub	...	0	0	NaN	NaN	
289	17	20	RL	NaN	11241	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	[illegible]
290	523	50	RM	50.0	5000	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
291	1379	160	RM	21.0	1953	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	

292 rows × 80 columns

EXPLORATORY DATA ANALYSIS

- Checking for missing or null values.

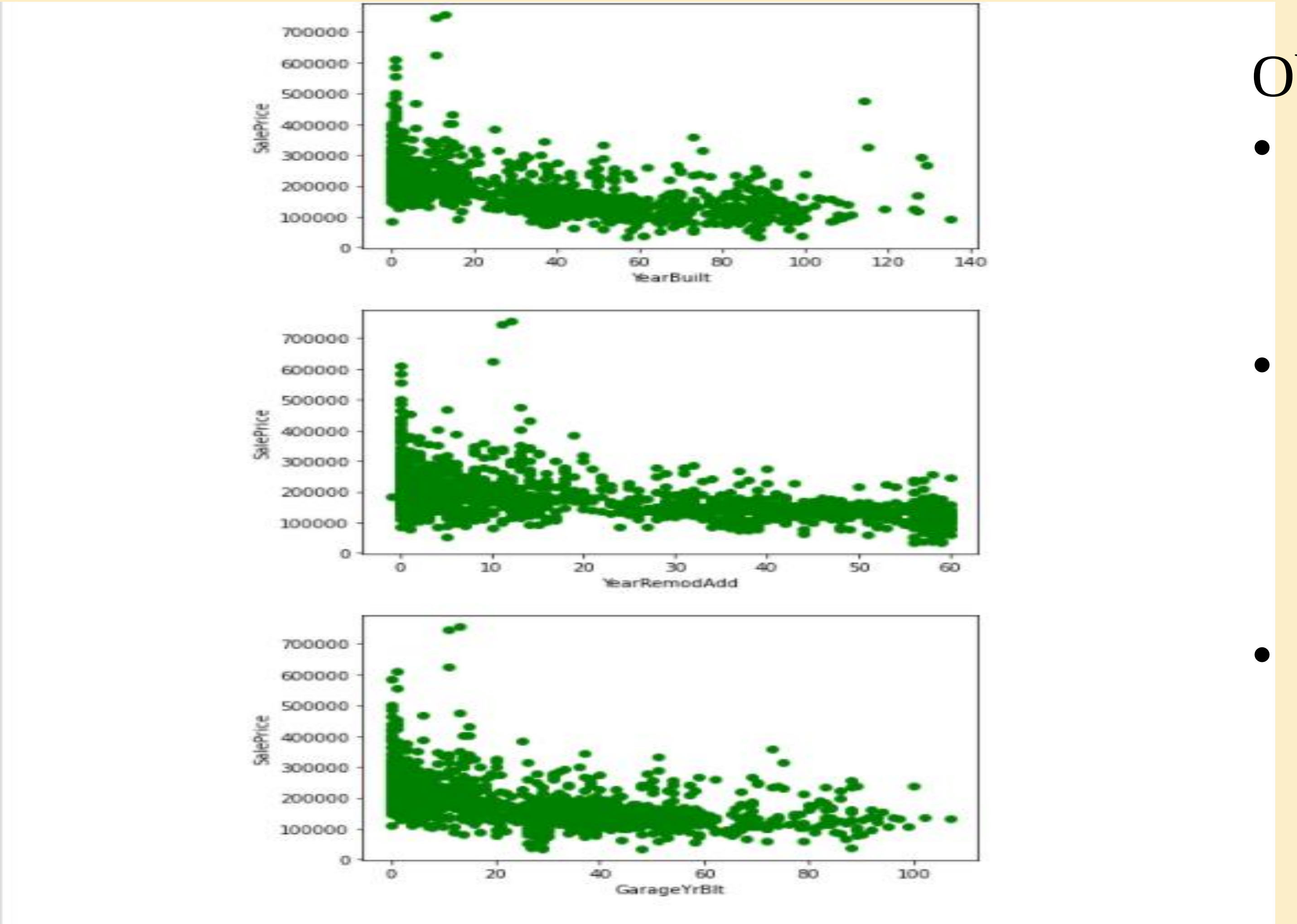
- Understanding the data type and classifying them.

- Simultaneously performing the analysis to both test and train data.

DATA PROCESSING

- NaN values in “PoolQC” were assumed to be houses without Pool.
- NaN values in “MiscFeature” were assumed to have no miscellaneous features.
- NaN values in “Alley” were assumed to have no alley access.
- NaN values in “Fence” were assumed to have No fence.
- NaN values in “FireplaceQu” were assumed to have no fire place.
- NaN values all basement related features were assumed to have no basement.
- The same assumtions were followed for both test and train datasets.
- While for NaN values in the columns “Electrical”,”MasVnrArea” and “MasVnrType” in the test dataset were filled by mode, mean and mode methods respectively.
- Since the ID column has all unique values, this feature wouldn’t help in prediction.

VISUALIZATION



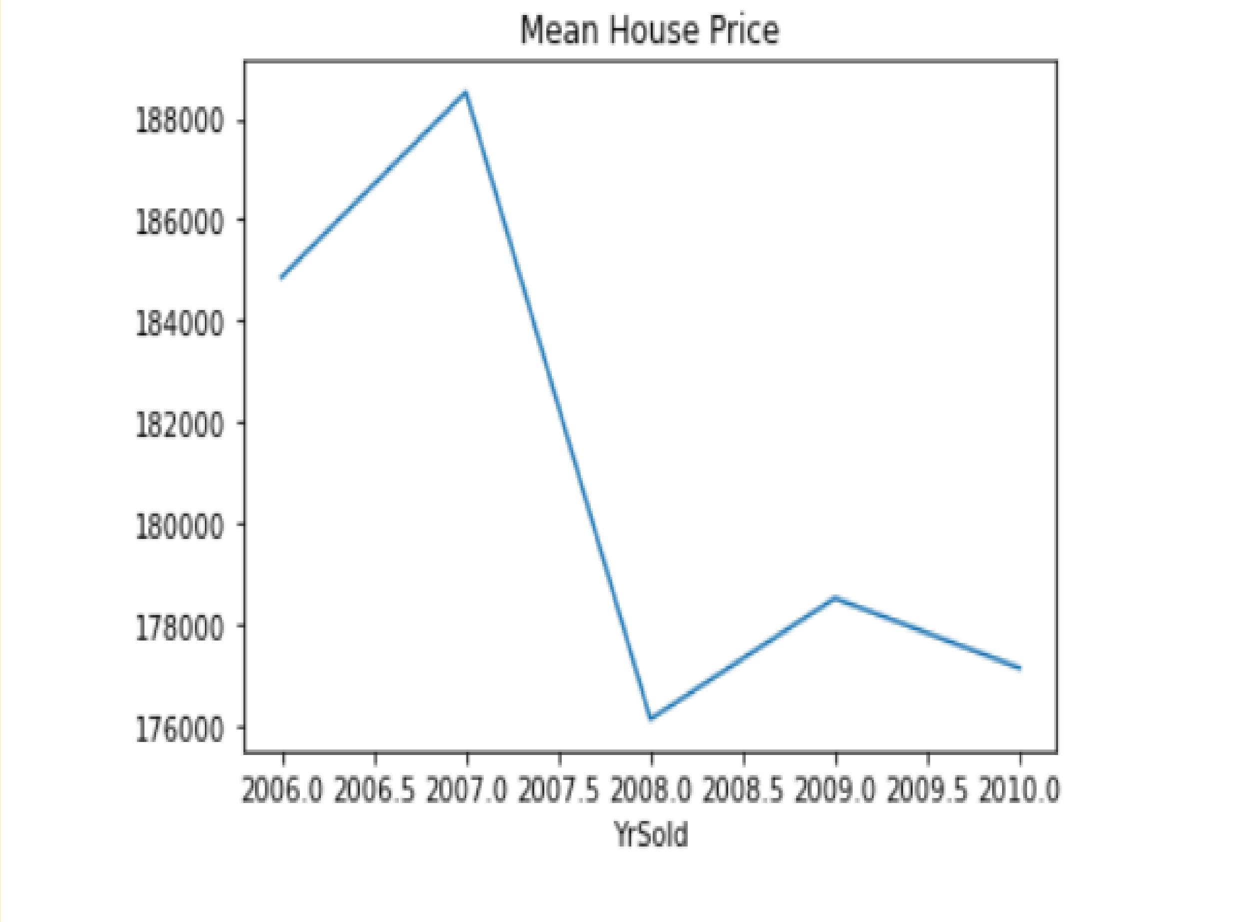
Observation:

- *Newer houses have higher price.*

- *Newer house remodeling has higher prices.*

- *Newer garage construction has higher price.*

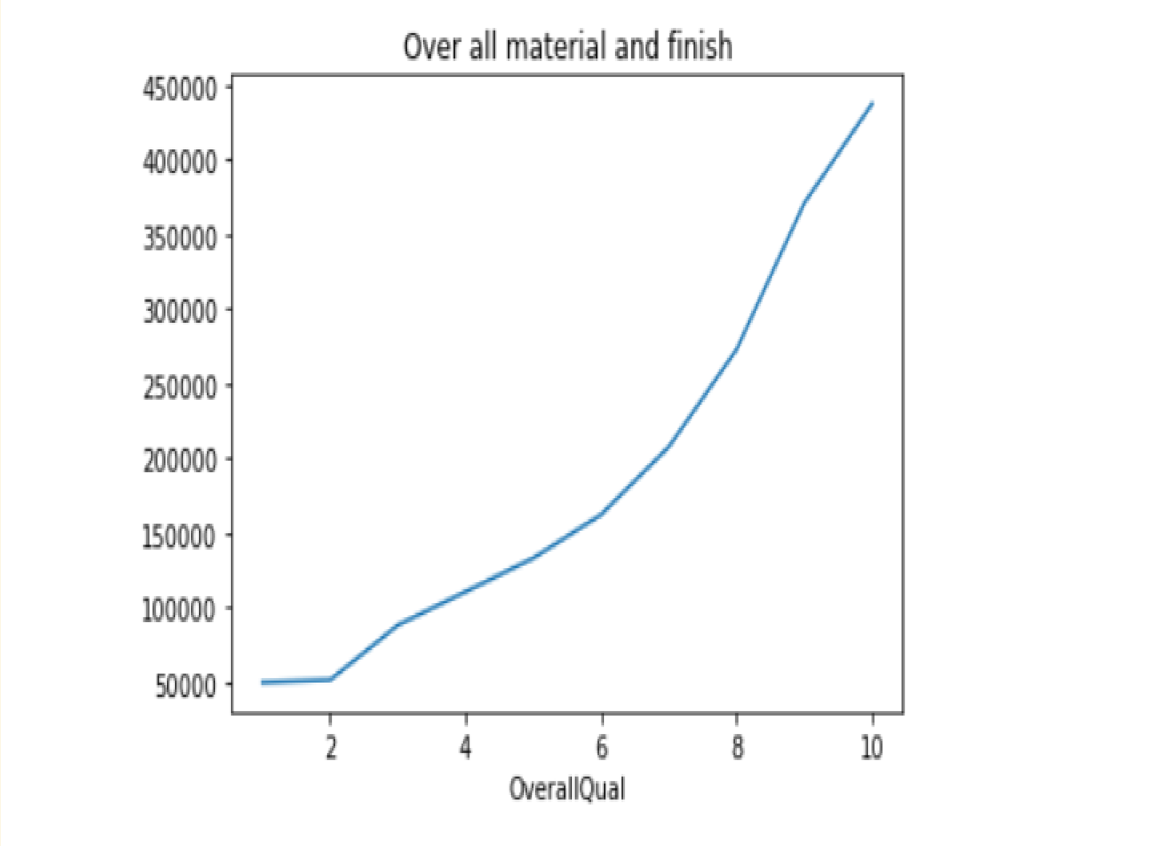
VISUALIZATION



Observation:

evidently there was a crash in the real estate towards the end of 2007 and begin of 2008. This could be a result of economic crisis during the period. While a peak was noticed between 2006 and 2007.

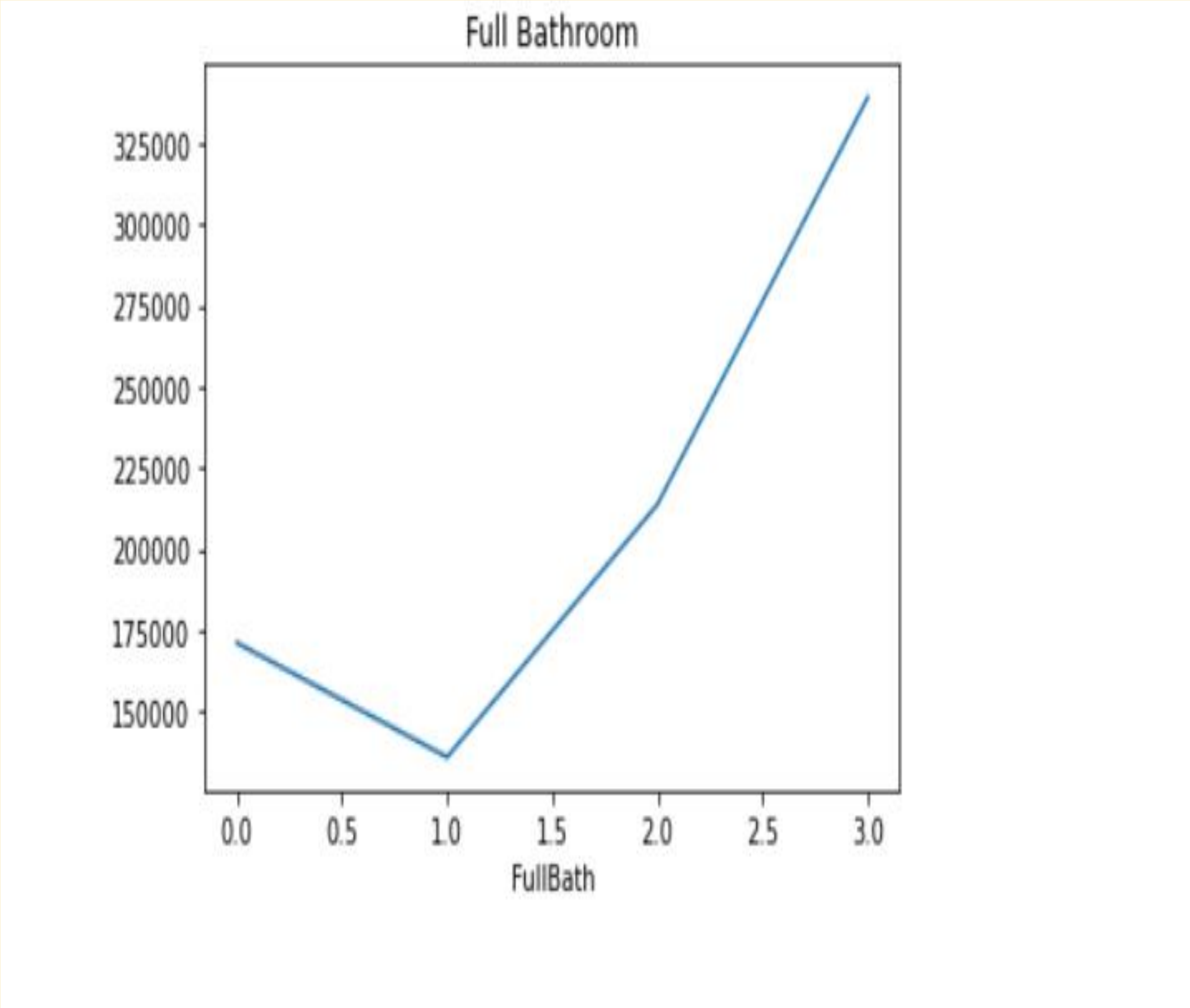
VISUALIZATION



Observation:

As the Over quality of the house increases the price also increases.

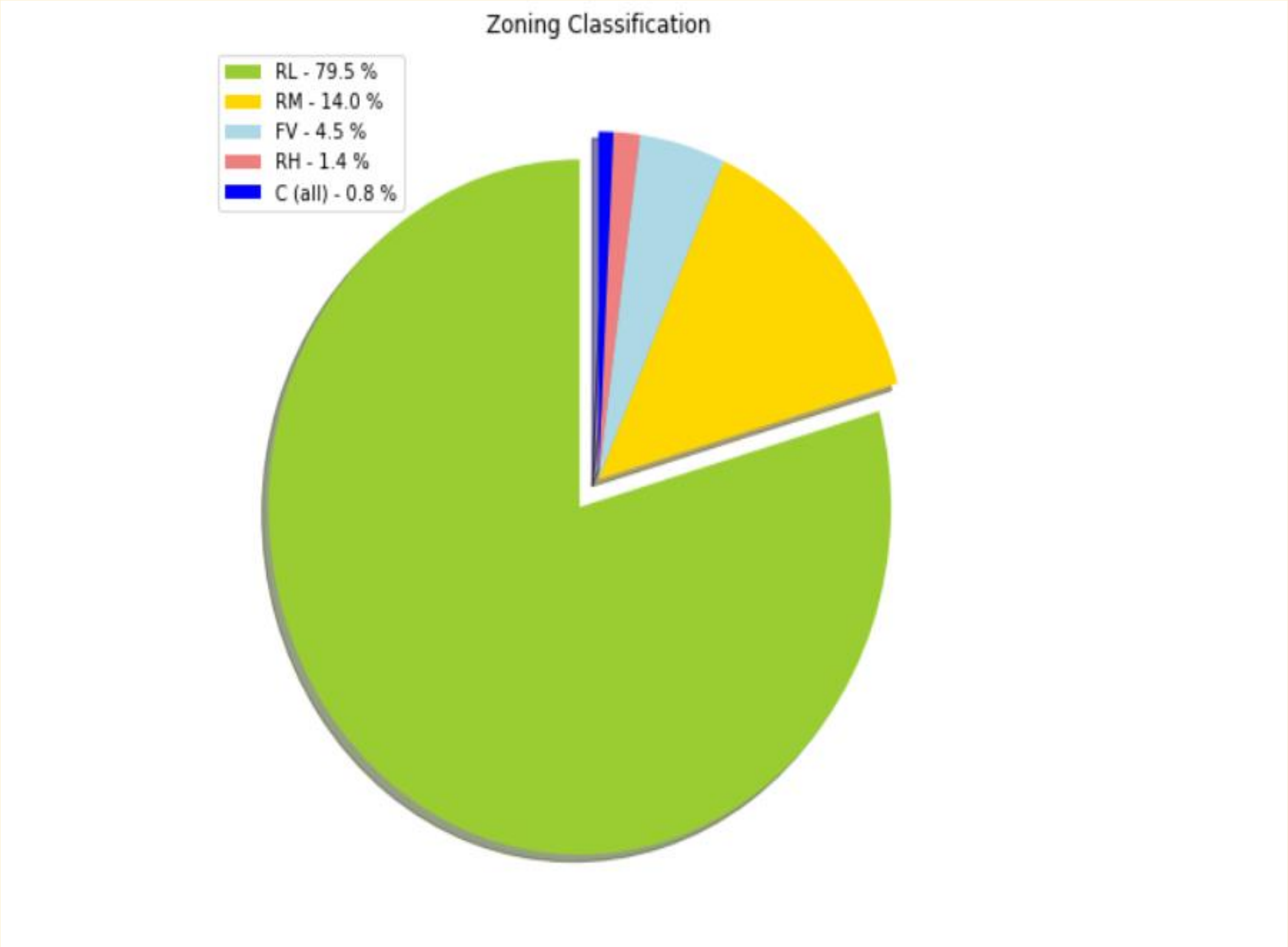
VISUALIZATION



Observation:

•*Full bathroom over 1.0 contributes in linear increase in price.*

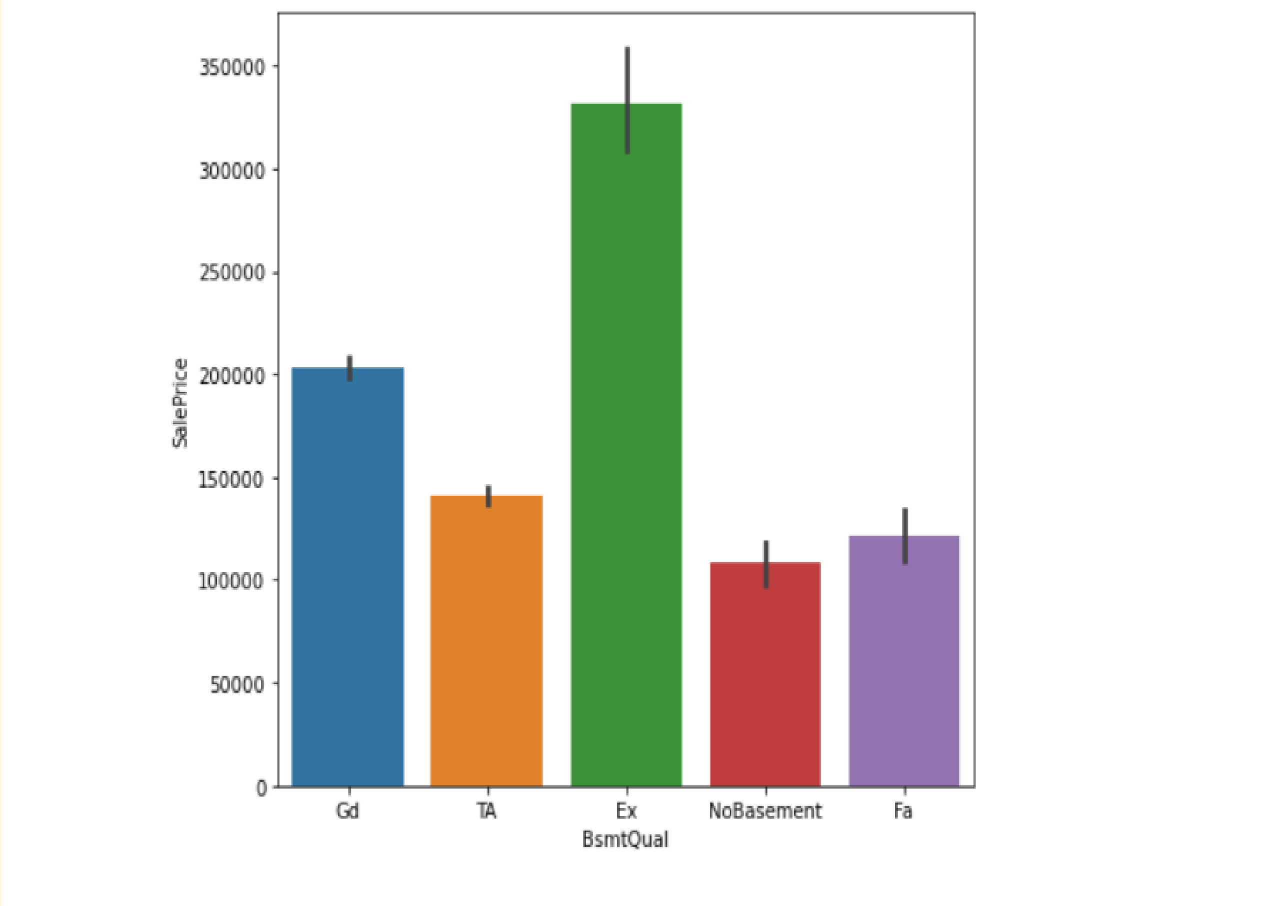
VISUALIZATION



Observation:

- *RL consumes over 79.5% of the total.*
- *C only contributes 0.8%*

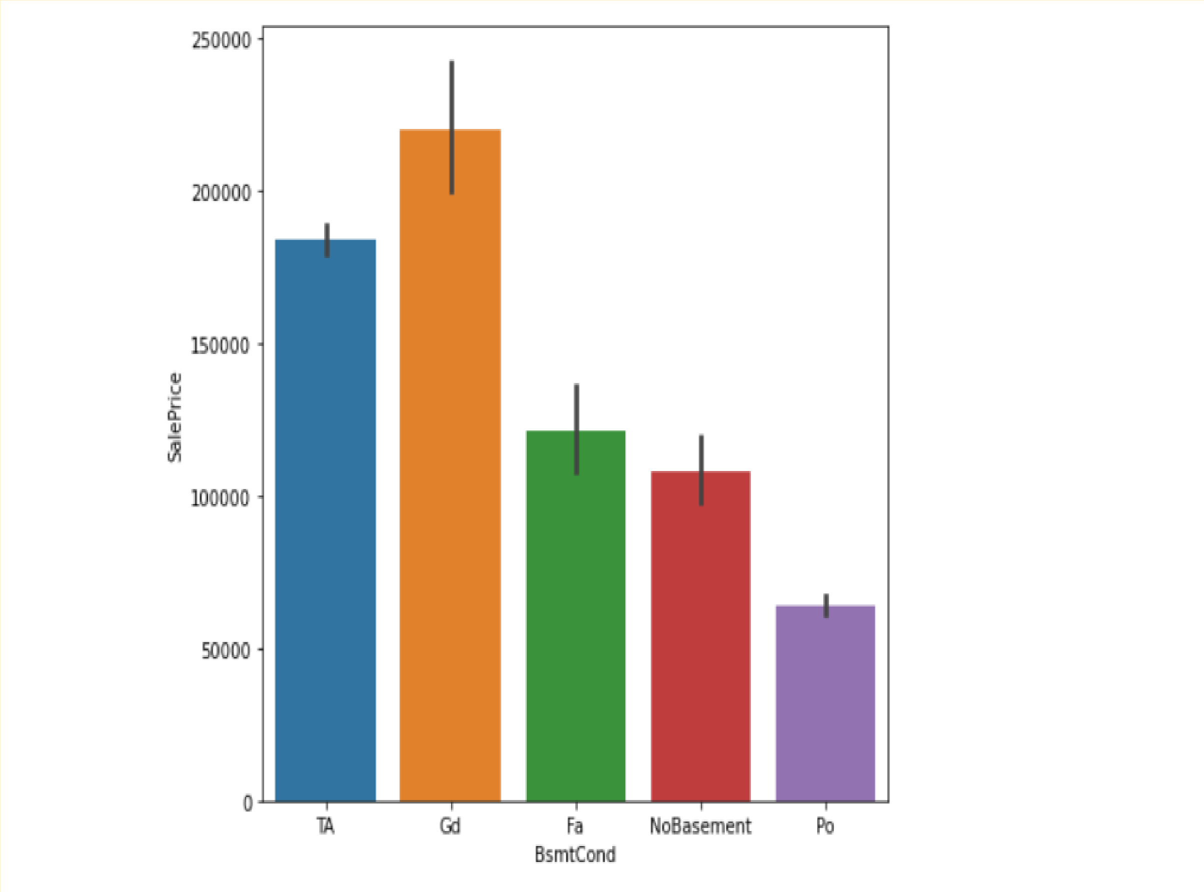
VISUALIZATION



Observation:

•*Most houses having excellent and good basement have higher prices.*

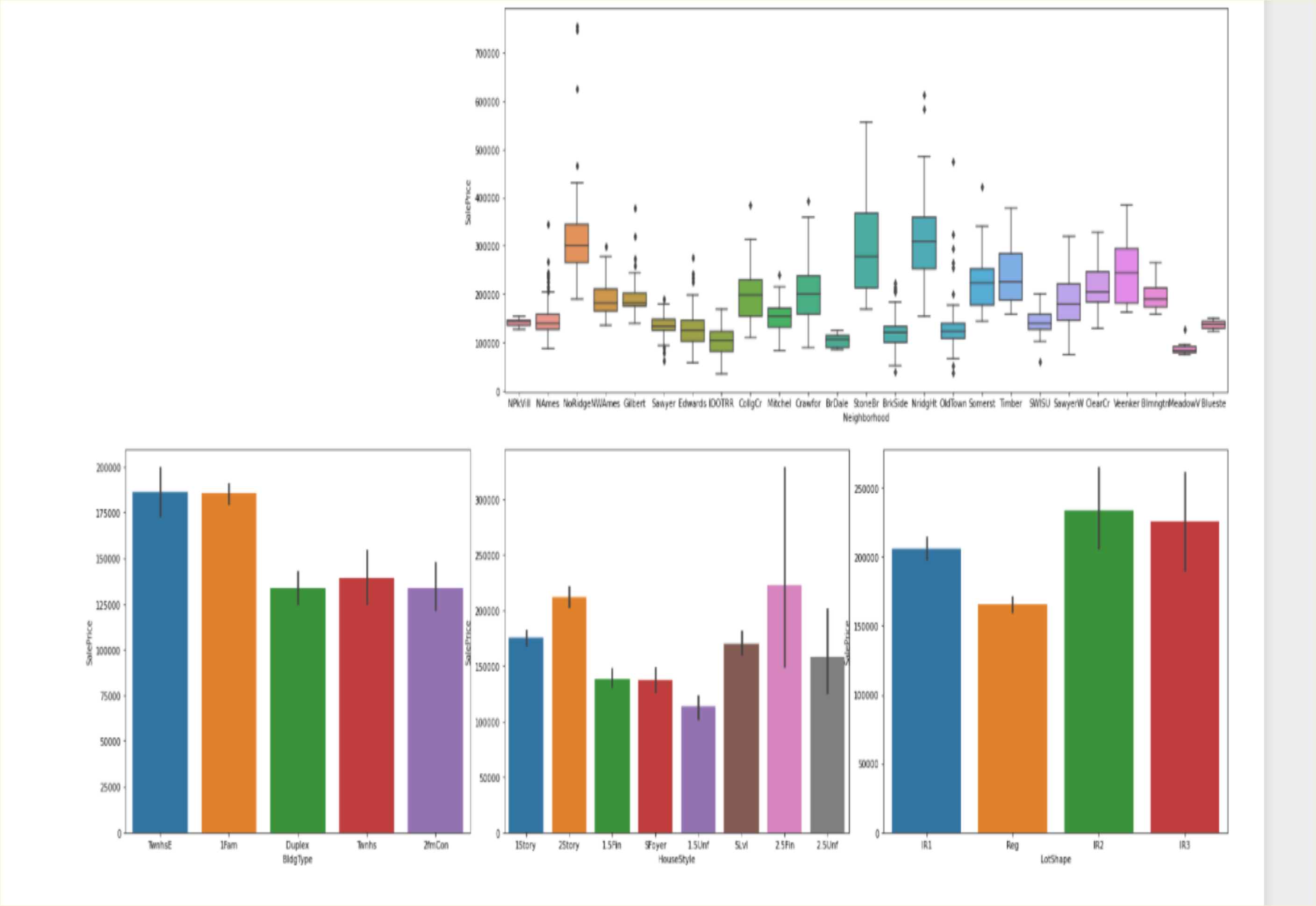
VISUALIZATION



Observation:

Most basements are good/typical that contribute to higher price.

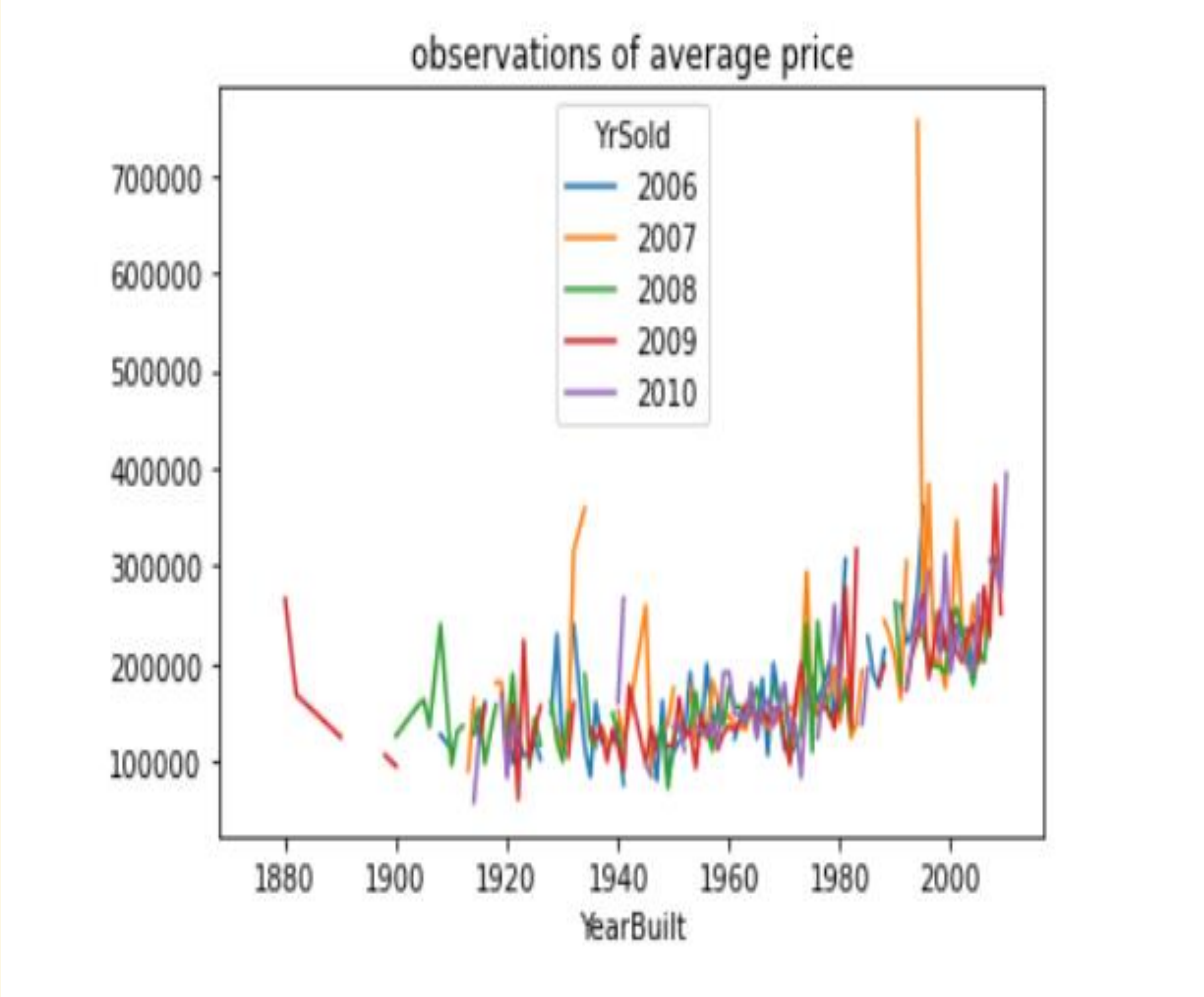
VISUALIZATION



Observation:

1. *2.5 storey dwelling are priced higher.*
2. *Townhouse end unit are priced higher.*
3. *slightly irregular properties are priced higher.*

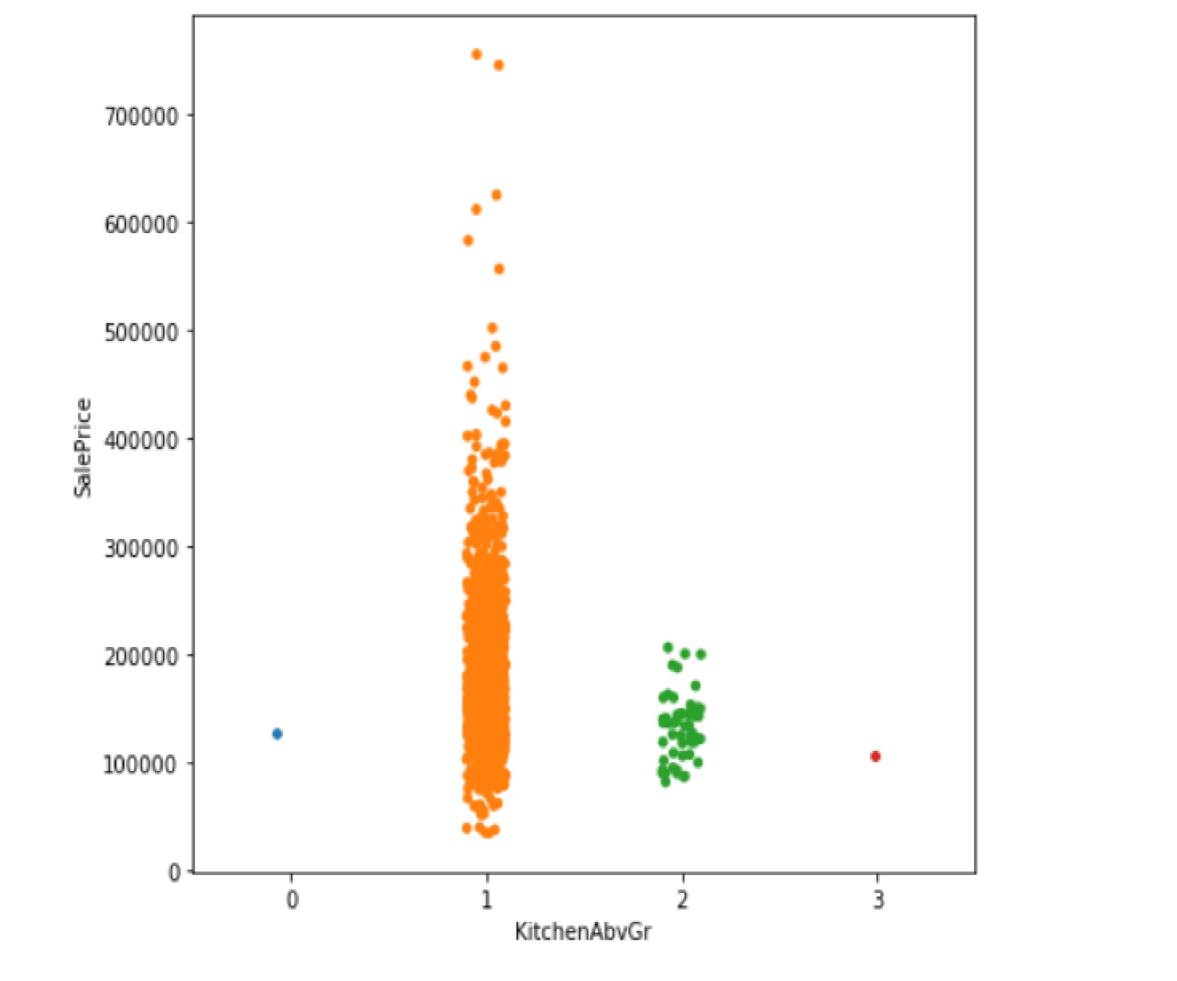
VISUALIZATION



Observation:

1. *The year 2007 has seen a sharp peak.*

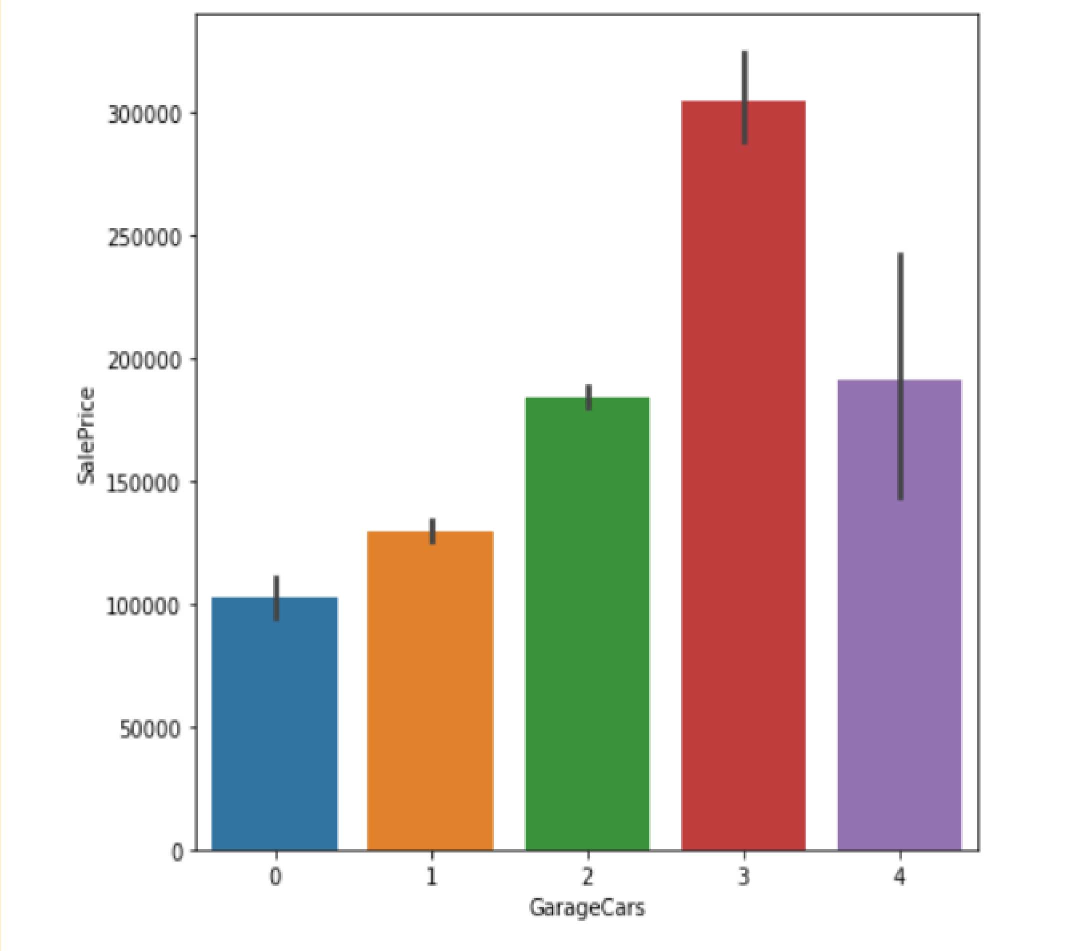
VISUALIZATION



Observation

- *Most kitchens have 1 kitchen above grade.*

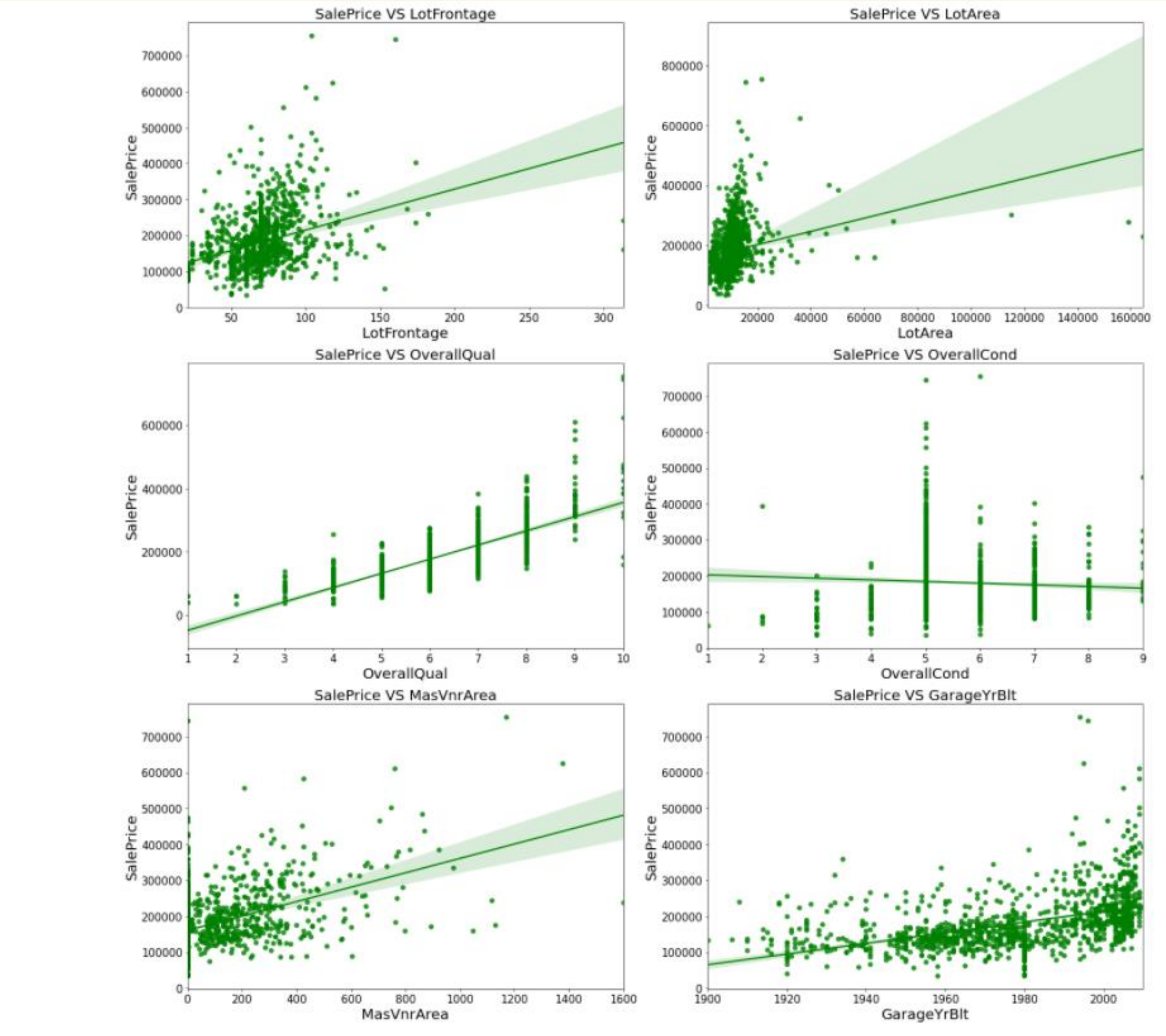
VISUALIZATION



Observation:

A notable increase in the price as the garage capac ity increases. At 4 cars th ere seems to be a dip in p rice.

VISUALIZATION



Observation:

1. *As linear feet of street increses the price increases.*
2. *As lot area increses there is an increse in the sale price.*
3. *As overall quality increses the sale prices increses.*
4. *With increse in garage built year there isn't a sharp increase in the price.*
5. *At overall condition 5 (average) has the highest sale price while the consecutive condition see a declined trend.*

DATA CLEANING

- The datasets had null values, outliers and skewness.
- Imputation method was used to replace null values.
- Yeo-Johnson method to remove outliers.
- Label encoder was used to encode all categorical data.
- Made use of Pearson's correlation coefficient to check the correlation between dependent and independent features.

MODEL BUILDING

Given my target column is continuous in nature the model building is made using regression. My final model is selected based on the r2 score and cross validation score by determining their difference. To assess the best fit model, I've run multiple regression models, the models are enlisted below.

- RandomForestRegressor
- GradientBoostingRegressor
- DecisionTreeRegressor

1. RANDOM FOREST REGRESSOR

RandomForest Regressor

In [125]:
```
RFR=RandomForestRegressor()
RFR.fit(X_train,y_train)
pred=RFR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))

#cross validation score
scores = cross_val_score(RFR, X, y, cv = 10).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 90.04671127256715
mean_squared_error: 599093012.4047372
mean_absolute_error: 16793.267464387463
root_mean_squared_error: 24476.376619196257

Cross validation score : 82.72414647013342

R2_Score - Cross Validation Score : 7.322564802433732
```

2. DECISION TREE REGRESSOR

DecisionTree Regressor

In [124]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
from sklearn.model_selection import cross_val_score
DTR=DecisionTreeRegressor()
DTR.fit(X_train,y_train)
pred=DTR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))

#cross validation score
scores = cross_val_score(DTR, X, y, cv = 10).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 70.98521219681808
mean_squared_error: 1746413382.079772
mean_absolute_error: 29288.803418803418
root_mean_squared_error: 41790.11105608325

Cross validation score : 60.07149948862425

R2_Score - Cross Validation Score : 10.913712708193835
```

3. GRADIENT BOOST REGRESSOR

GradientBoosting Regressor

In [126]:
```
from sklearn.ensemble import GradientBoostingRegressor
GBR=GradientBoostingRegressor()
GBR.fit(X_train,y_train)
pred=GBR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))

#cross validation score
scores = cross_val_score(GBR, X, y, cv = 10).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 91.11573192554091
mean_squared_error: 534748169.12215793
mean_absolute_error: 15592.048095145745
root_mean_squared_error: 23124.622572534194

Cross validation score : 83.58436444216586

R2_Score - Cross Validation Score : 7.53136748337505
```

HYPER PARAMETER TUNING

GridSearch Cv for RandomForest Regressor

```
from sklearn.model_selection import GridSearchCV
param = {"criterion":["squared_error","absolute_error","poisson"],"max_features":["auto","sqrt","log2"],"bootstrap":[True,False],
clf = GridSearchCV(RFR,param_grid=param)
```

```
clf.fit(X_train,y_train)
```

```
GridSearchCV(estimator=RandomForestRegressor(),
             param_grid={'bootstrap': [True, False],
                         'criterion': ['squared_error', 'absolute_error',
                                       'poisson'],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [50, 100, 150, 200]})
```

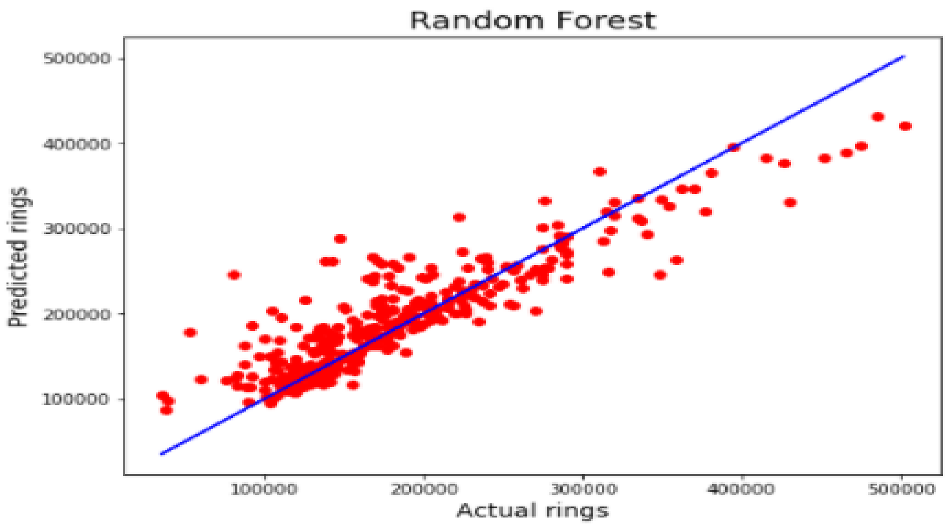
```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'bootstrap': False, 'criterion': 'poisson', 'max_features': 'log2', 'n_estimators': 100}
0.7076408619318525
```

```
RFR=RandomForestRegressor(bootstrap=False,criterion='poisson',max_features='log2',n_estimators=100,random_state=42)
RFR.fit(X_train,y_train)
pred=RFR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))

#cross validation score
scores = cross_val_score(RFR, X, y, cv = 10).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 78.42112167331264
mean_squared_error: 1298842581.0877426
mean_absolute_error: 24767.58894586894
root_mean_squared_error: 36039.458668073006

Cross validation score : 75.38534929947181

R2_Score - Cross Validation Score : 3.035772373840828
```

```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred,color="r")
plt.plot(y_test,y_test,color="b")
plt.xlabel("Actual rings",fontsize=14)
plt.ylabel("Predicted rings",fontsize=14)
plt.title("Random Forest",fontsize=18)
plt.show()
```



HYPER PARAMETER TUNING

GridSearch Cv for GradientBoosting Regressor

```
param = {"loss":["squared_error", "absolute_error", "huber", "quantile"],"criterion":["friedman_mse","squared_error","mse","mae"]
clf = GridSearchCV(GBR,param_grid=param)
```

```
clf.fit(X_train,y_train)
```

```
GridSearchCV(estimator=GradientBoostingRegressor(),
             param_grid={'criterion': ['friedman_mse', 'squared_error', 'mse',
                                       'mae'],
                         'loss': ['squared_error', 'absolute_error', 'huber',
                                  'quantile'],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [50, 100, 150, 200]})
```

```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'criterion': 'mae', 'loss': 'huber', 'max_features': 'sqrt', 'n_estimators': 200}
0.8245020684372859
```

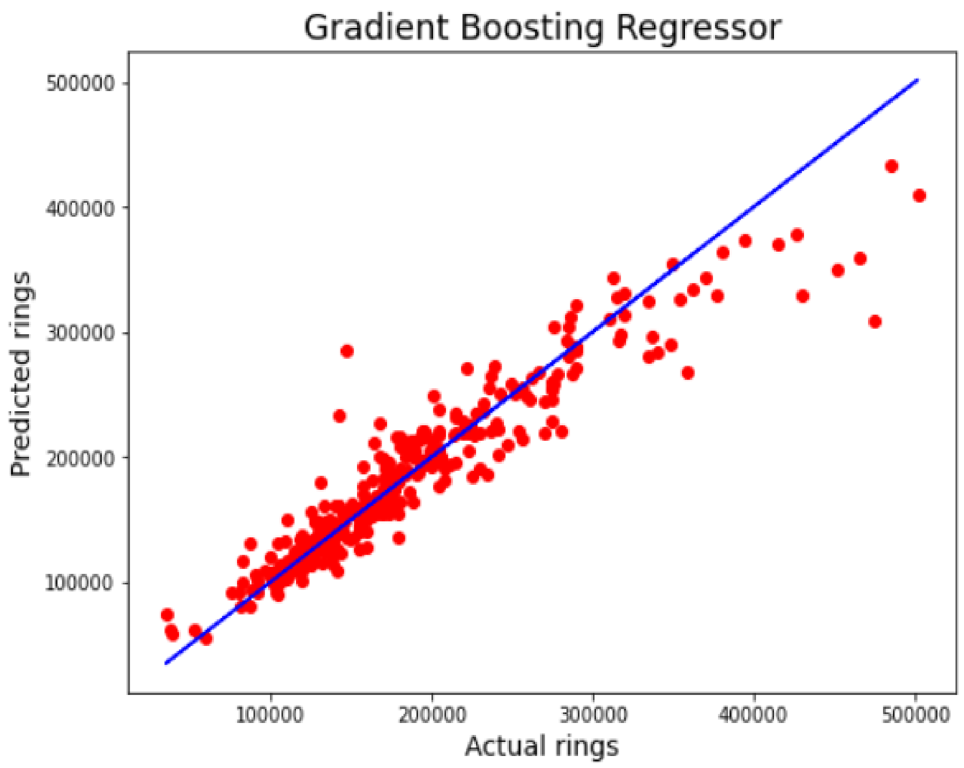
```
GBR=GradientBoostingRegressor(criterion="mse",loss="huber",max_features="log2",n_estimators=100,random_state=9)
GBR.fit(X_train,y_train)
pred_gbr=GBR.predict(X_test)
R2_score = r2_score(y_test,pred_gbr)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred_gbr))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred_gbr))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred_gbr)))
```

```
R2_score: 89.23696348789251
mean_squared_error: 647832102.8595073
mean_absolute_error: 16224.063932714307
root_mean_squared_error: 25452.546097777868
```

```python
#difference of accuracy and cv score
scores = cross_val_score(GBR, X, y, cv = 9)
scores=scores.mean()
diff = R2_score - (scores)*100
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_Score - Cross Validation Score : 4.695520142467998
```

```python
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_gbr,color="r")
plt.plot(y_test,y_test,color="b")
plt.xlabel("Actual rings",fontsize=14)
plt.ylabel("Predicted rings",fontsize=14)
plt.title("Gradient Boosting Regressor",fontsize=18)
plt.show()
```



CONCLUSION

The study throws light on how volatile the real estate market can be and how complicated the intangible features associated them can be. Understanding and deal with such data is complex on its own, with prediction this adds a new dimension to the complexity. Given the inconsistency of the dataset in terms on NaN values the project posed new complexes. By identifying the correlation between the target feature and the input features we can establish the features that are more relevant to us. Hence, making building a model easier with definitive accuracy. To conclude, although the application of this model is only in the early stage with research in progress I'm certain we can achieve higher accuracy for most geographical locations with more features.

Thank You