HOUSE PRICE PREDICTION

Submitted by:

Atul Mahajan

INTRODUCTION

Business Problem Framing

The objective was to model the price of houses with the available independent variables. This model can then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

Conceptual Background of the Domain Problem

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.

A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below. The company is looking at prospective properties to buy houses to enter the market. I was required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

- Which variables are important to predict the price of variable?
- How do these variables describe the price of the house?

Technical Requirements:

- Data contains 1460 entries each having 81 variables.
- Data contains Null values. You need to treat them using the domain knowledge and your own understanding.
- Extensive EDA has to be performed to gain relationships of important variable and price.
- Data contains numerical as well as categorical variable. You need to handle them accordingly.
- Need to build Machine Learning models, apply regularization and determine the optimal values of Hyper Parameters.
- Need to find important features which affect the price positively or negatively.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  This is a Regression problem, where our end goal is to predict the Prices of House based on given data. I will be dividing my data into **Training** and **Testing** parts. A Regression Model will be built and trained using the Training data and the Test data will be used to predict the outcomes. This will be compared with available test results to find how well the model has performed.
  The 'r2' score will be used to determine the best model among,
  - Linear Regression with Lasso, Ridge
  - Random Forest Regression
  - XGBoost
- The best results were obtained using Lasso Regression. So, let's understand a little about it.

In a simple regression problem (a single x and a single y), the form of the model would be:

y = B0 + B1*x,

where B0 —

intercept

B1 —coefficient

x —independent variable

y —output or the dependent variable

In higher dimensions when we have more than one input (x),

The General equation for a Multiple linear regression with p — independent

variables: Y=B0 + B1 * X1 + B2 * X2 + + Bp * Xp + E(Random Error or

Noise)

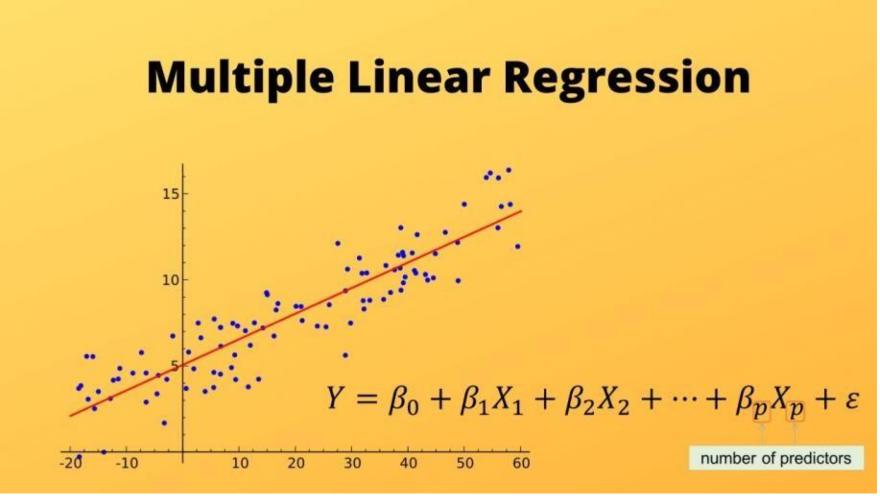


Image Source: https://morioh.com/p/0d9b2bedf683

Let's consider a regression scenario where 'y' is the predicted vector and 'x' is the feature matrix. Basically in any regression problem, we try to minimize the squared error. Let 'β' be the vector of parameters (weights of importance of features) and 'p' be the number of features

Now, let's discuss the case of **lasso regression**, which is also called L1 regression since it uses the L1 norm for regularization. In lasso regression, we try to solve the below minimization problem:

$$Min_\beta \; L_1 = (y - x\beta)^2 + \lambda \sum_{i=1}^{p} |\beta_i|$$

For simplicity, let p=1 and $\beta_i = \beta$. Now,

$$L_1 = (y - x\beta)^2 + \lambda|\beta|$$
$$= \mathrm{y}^2 - 2\mathrm{xy}\beta + x^2\beta^2 + \lambda|\beta|$$

Example: Suppose we are building a linear model out of two features, we'll have two coefficients (β_1 and β_2). For better understanding let β_1 = 10 and β_2 = 1000.

In lasso regression, the L1 penalty would look like,

$L_{1p} = |\beta_1| + |\beta_2|$

Shrinking β_1 to 8 and β_2 to 100 would minimize the penalty to 108 from 1010, which means in this case the change is not so significant just by shrinking the larger quantity. So, in the case of the L_1 penalty, both the coefficients have to be shrunk to extremely small values, in order to achieve regularization. And in this whole process, some coefficients may shrink to zero. [1] [Ref: URL for the above explanation in the foot note]

Assumptions:

There are four assumptions associated with a linear regression model:

1. **Linearity**: The relationship between X and the mean of Y is linear.
2. **Homoscedasticity**: The variance of residual is the same for any value of X.
3. **Independence**: Observations are independent of each other.
4. **Normality**: For any fixed value of X, Y is normally distributed.

[1] https://www.analyticsvidhya.com/blog/2020/11/lasso-regression-causes-sparsity-while-ridge-regression- doesnt-unfolding-the-math/

Data Sources and their formats

A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file

Here's how the top 10 rows of the data looks like:

```
# Rows, Columns
df.shape
```

```
(1460, 81)
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm
5	6	50	RL	85.0	14115	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Mitchel	Norm
6	7	20	RL	75.0	10084	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Somerst	Norm
7	8	60	RL	NaN	10382	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	NWAmes	PosN
8	9	50	RM	51.0	6120	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	OldTown	Artery
9	10	190	RL	50.0	7420	Pave	NaN	Reg	Lvl	AllPub	Corner	Gtl	BrkSide	Artery

Condition2	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	YearRemodAdd	RoofStyle	RoofMatl	Exterior1st	Exterior2nd	MasVnrType
Norm	1Fam	2Story	7	5	2003	2003	Gable	CompShg	VinylSd	VinylSd	BrkFace
Norm	1Fam	1Story	6	8	1976	1976	Gable	CompShg	MetalSd	MetalSd	None
Norm	1Fam	2Story	7	5	2001	2002	Gable	CompShg	VinylSd	VinylSd	BrkFace
Norm	1Fam	2Story	7	5	1915	1970	Gable	CompShg	Wd Sdng	Wd Shng	None
Norm	1Fam	2Story	8	5	2000	2000	Gable	CompShg	VinylSd	VinylSd	BrkFace
Norm	1Fam	1.5Fin	5	5	1993	1995	Gable	CompShg	VinylSd	VinylSd	None
Norm	1Fam	1Story	8	5	2004	2005	Gable	CompShg	VinylSd	VinylSd	Stone
Norm	1Fam	2Story	7	6	1973	1973	Gable	CompShg	HdBoard	HdBoard	Stone
Norm	1Fam	1.5Fin	7	5	1931	1950	Gable	CompShg	BrkFace	Wd Shng	None
Artery	2fmCon	1.5Unf	5	6	1939	1950	Gable	CompShg	MetalSd	MetalSd	None

ExterQual	ExterCond	Foundation	BsmtQual	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinSF1	BsmtFinType2	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF
Gd	TA	PConc	Gd	TA	No	GLQ	706	Unf	0	150	856
TA	TA	CBlock	Gd	TA	Gd	ALQ	978	Unf	0	284	1262
Gd	TA	PConc	Gd	TA	Mn	GLQ	486	Unf	0	434	920
TA	TA	BrkTil	TA	Gd	No	ALQ	216	Unf	0	540	756
Gd	TA	PConc	Gd	TA	Av	GLQ	655	Unf	0	490	1145
TA	TA	Wood	Gd	TA	No	GLQ	732	Unf	0	64	796
Gd	TA	PConc	Ex	TA	Av	GLQ	1369	Unf	0	317	1686
TA	TA	CBlock	Gd	TA	Mn	ALQ	859	BLQ	32	216	1107
TA	TA	BrkTil	TA	TA	No	Unf	0	Unf	0	952	952
TA	TA	BrkTil	TA	TA	No	GLQ	851	Unf	0	140	991

Heating	HeatingQC	CentralAir	Electrical	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	BedroomAbvGr
GasA	Ex	Y	SBrkr	856	854	0	1710	1	0	2	1	3
GasA	Ex	Y	SBrkr	1262	0	0	1262	0	1	2	0	3
GasA	Ex	Y	SBrkr	920	866	0	1786	1	0	2	1	3
GasA	Gd	Y	SBrkr	961	756	0	1717	1	0	1	0	3
GasA	Ex	Y	SBrkr	1145	1053	0	2198	1	0	2	1	4
GasA	Ex	Y	SBrkr	796	566	0	1362	1	0	1	1	1
GasA	Ex	Y	SBrkr	1694	0	0	1694	1	0	2	0	3
GasA	Ex	Y	SBrkr	1107	983	0	2090	1	0	2	1	3
GasA	Gd	Y	FuseF	1022	752	0	1774	0	0	2	0	2
GasA	Ex	Y	SBrkr	1077	0	0	1077	1	0	1	0	2

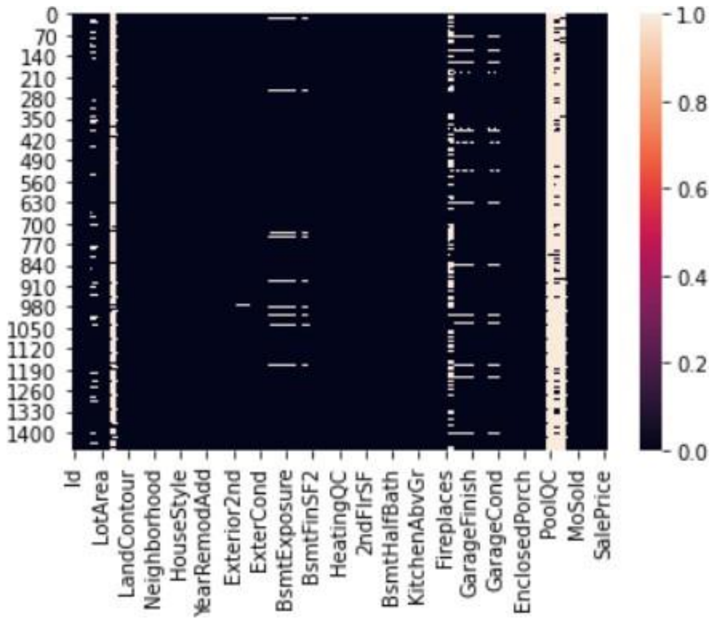
KitchenAbvGr	KitchenQual	TotRmsAbvGrd	Functional	Fireplaces	FireplaceQu	GarageType	GarageYrBlt	GarageFinish	GarageCars	GarageArea	GarageQual
1	Gd	8	Typ	0	NaN	Attchd	2003.0	RFn	2	548	TA
1	TA	6	Typ	1	TA	Attchd	1976.0	RFn	2	460	TA
1	Gd	6	Typ	1	TA	Attchd	2001.0	RFn	2	608	TA
1	Gd	7	Typ	1	Gd	Detchd	1998.0	Unf	3	642	TA
1	Gd	9	Typ	1	TA	Attchd	2000.0	RFn	3	836	TA
1	TA	5	Typ	0	NaN	Attchd	1993.0	Unf	2	480	TA
1	Gd	7	Typ	1	Gd	Attchd	2004.0	RFn	2	636	TA
1	TA	7	Typ	2	TA	Attchd	1973.0	RFn	2	484	TA
2	TA	8	Min1	2	TA	Detchd	1931.0	Unf	2	468	Fa
2	TA	5	Typ	2	TA	Attchd	1939.0	RFn	1	205	Gd

GarageCond	PavedDrive	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold
TA	Y	0	61	0	0	0	0	NaN	NaN	NaN	0	2
TA	Y	298	0	0	0	0	0	NaN	NaN	NaN	0	5
TA	Y	0	42	0	0	0	0	NaN	NaN	NaN	0	9
TA	Y	0	35	272	0	0	0	NaN	NaN	NaN	0	2
TA	Y	192	84	0	0	0	0	NaN	NaN	NaN	0	12
TA	Y	40	30	0	320	0	0	NaN	MnPrv	Shed	700	10
TA	Y	255	57	0	0	0	0	NaN	NaN	NaN	0	8
TA	Y	235	204	228	0	0	0	NaN	NaN	Shed	350	11
TA	Y	90	0	205	0	0	0	NaN	NaN	NaN	0	4
TA	Y	0	4	0	0	0	0	NaN	NaN	NaN	0	1

YrSold	SaleType	SaleCondition	SalePrice
2008	WD	Normal	208500
2007	WD	Normal	181500
2008	WD	Normal	223500
2006	WD	Abnorml	140000
2008	WD	Normal	250000
2009	WD	Normal	143000
2007	WD	Normal	307000
2009	WD	Normal	200000
2008	WD	Abnorml	129900
2008	WD	Normal	118000

The last Feature: SalePrice is the target variable. The above Snapshots show all the features and the top 10 rows. As mentioned earlier, there are 1460 rows and 81 columns.

Data Preprocessing



The above heatmap shows there are many Null Values, which can't be processed. One Observation here is that a lot of variables have been labelled at NaN, but they are actually not null values and have certain meaning.

For Example,

- NA in feature 'Alley' means No_Alley
- in case of PoolQC, NA means 'No Pool' (* Refer Data Description at the end of the notebook)

I've replaced them with actual variables before going further.

First let us handle Categorical features which are missing; based on domain knowledge and given explanation. The percentage of Null values in Categorical features:

```
Alley: 0.9377% missing values
MasVnrType: 0.0055% missing values
BsmtQual: 0.0253% missing values
BsmtCond: 0.0253% missing values
BsmtExposure: 0.026% missing values
BsmtFinType1: 0.0253% missing values
BsmtFinType2: 0.026% missing values
FireplaceQu: 0.4726% missing values
GarageType: 0.0555% missing values
GarageFinish: 0.0555% missing values
GarageQual: 0.0555% missing values
GarageCond: 0.0555% missing values
PoolQC: 0.9952% missing values
Fence: 0.8075% missing values
MiscFeature: 0.963% missing values
```

Then I replaced all other categorical missing values with a new label 'Missing'. The numerical missing values will be imputed during feature engineering.

Numerical variables

```
# list of numerical variables
numerical_features = [feature for feature in df.columns if df[feature].dtypes != 'O']

print('Number of numerical variables: ', len(numerical_features))

# visualise the numerical variables
df[numerical_features].head()
```

```
Number of numerical variables:  37
```

Identified all features that were numerical

Year Features

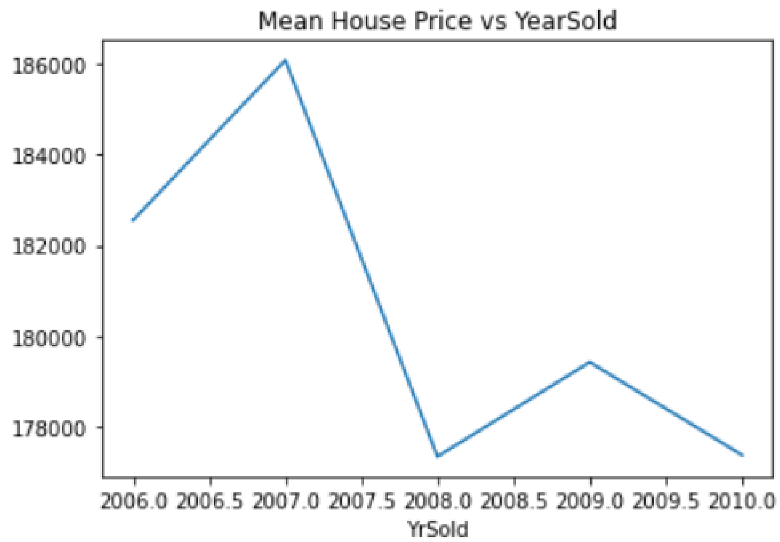
```
# (identified features with Year using key words 'year' or 'yr' in column headers)
year_feature = [feature for feature in numerical_features if 'Yr' in feature or 'Year' in feature]

year_feature
```

```
['YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'YrSold']
```

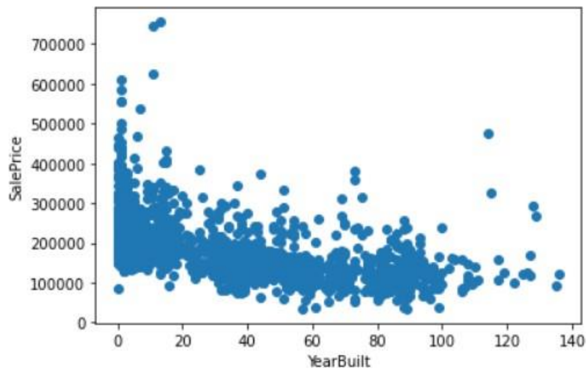
```
# Analyzing Prices of House vs Year Built
df.groupby('YrSold')['SalePrice'].mean().plot()
plt.title("Mean House Price vs YearSold")
```

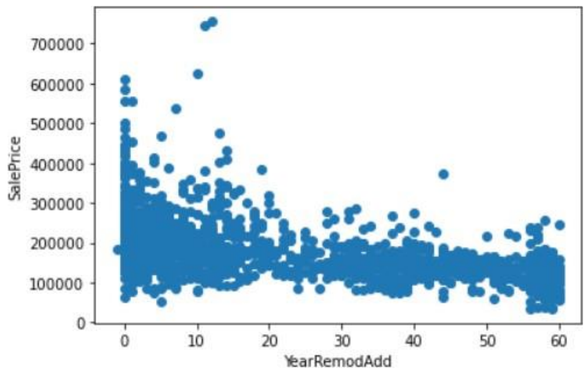
```
Text(0.5, 1.0, 'Mean House Price vs YearSold')
```

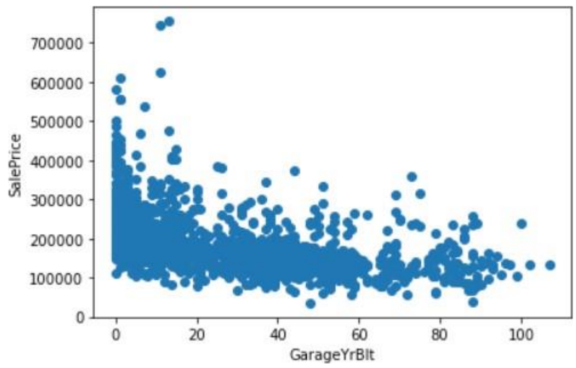


There seems to be a peak in House Prices, but a sharp drop in between 2007 to 2008. This can be due to Economic Crash. "Economies worldwide slowed during this period since credit tightened and international trade declined. Housing markets suffered and unemployment soared, resulting in evictions and foreclosures."

Let's see the scatterplot between All years features with SalePrice







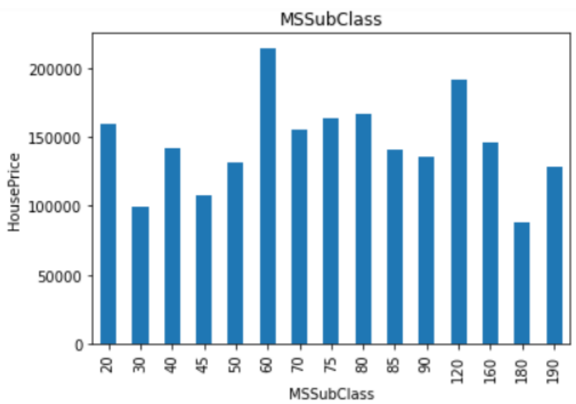
- Obs 1: The Houses built recently have Higher Sales Price.
- Obs 2: The Houses remodelled recently have Higher Sales Price.
- Obs 3: The Houses whose Garages were built recently have Higher Sales Price.

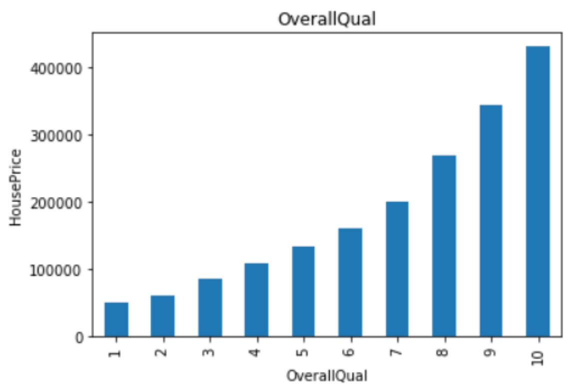
Identifying Discrete Variables

The following 17 features were identified as discrete variables:

```
['MSSubClass', 'OverallQual', 'OverallCond', 'LowQualFinSF', 'BsmtFullB
ath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenA
bvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', '3SsnPorch', 'PoolAr
ea', 'MiscVal', 'MoSold']
```

Plotted Bar Plots like these to understand relations with Sale Price





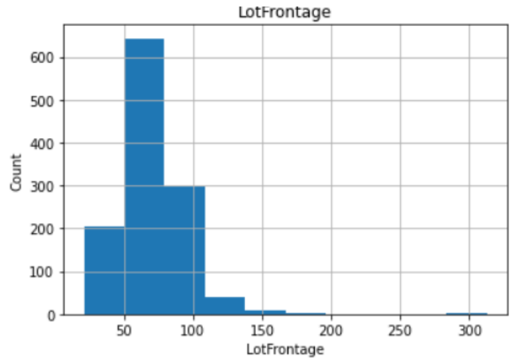
Similarly, plotted for all discrete values, and observed features.

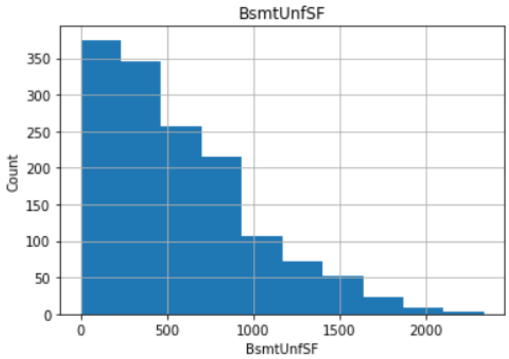
Identifying Continuous Features

```
continuous_feature=[feature for feature in numerical_features if feature not in discrete_feature+year_feature+['Id']]
print("Continuous feature Count",len(continuous_feature))
```

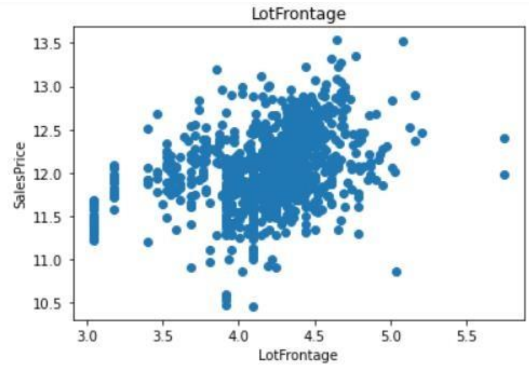
```
Continuous feature Count 16
```

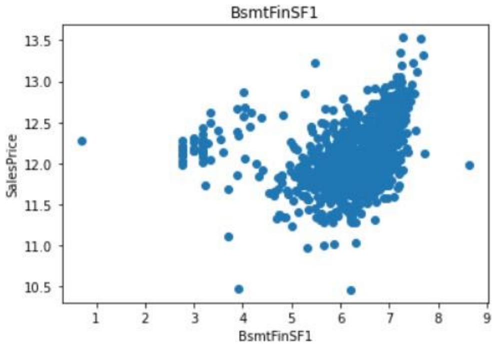
I've plotted Histograms for all 16 features like the following





As clear from above a lot of features were not normally distributed. Let's I did log transformation, plotted the scatterplots to see the trends.





Categorical Features

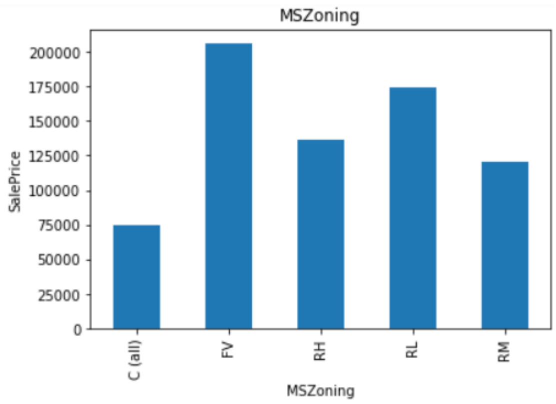
```
categorical_features=[feature for feature in df.columns if df[feature].dtypes=='O']
```

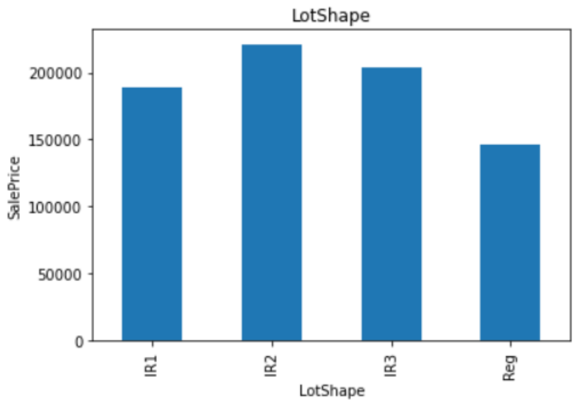
```
Identified total unique categories in each feature:
```

MSZoning has 5
categories Street has 2
categories Alley has 3
categories LotShape
has 4 categories
LandContour has 4
categories Utilities has 2
categories LotConfig has 5
categories LandSlope has 3
categories Neighborhood has
25 categories Condition1 has
9 categories Condition2 has 8
categories BldgType has 5
categories HouseStyle has 8
categories RoofStyle has 6
categories RoofMatl has 8
categories Exterior1st has 15
categories Exterior2nd has
16 categories

MasVnrType has 5
categories ExterQual has 4
categories ExterCond has 5
categories Foundation has 6
categories BsmtQual has 5
categories BsmtCond has 5
categories BsmtExposure
has 5 categories
BsmtFinType1 has 7
categories BsmtFinType2
has 7 categories Heating
has 6 categories HeatingQC
has 5 categories CentralAir
has 2 categories Electrical
has 6 categories
KitchenQual has 4
categories Functional has 7
categories FireplaceQu has
6 categories GarageType
has 7 categories
GarageFinish has 4
categories GarageQual has
6 categories GarageCond
has 6 categories
PavedDrive has 3
categories PoolQC has 4
categories
Fence has 5 categories
MiscFeature has 5
categories SaleType has 9
categories SaleCondition
has 6 categories

Plotted all Categorical variables vs SalesPrice as shown below





Feature Engineering

I had already treated all Null Values in categorical Features, Now I will check for numerical variables. Imputed the numerical null values with medians.

Now, as there were some features(Temporal) which contained year values. Differences:

	YearBuilt	YearRemodAdd	GarageYrBlt
0	5	5	5.0
1	31	31	31.0
2	7	6	7.0
3	91	36	8.0
4	8	8	8.0

Handling Rare Categorical Feature

We will remove categorical variables that are present less than 1% of the observations

```
for feature in categorical_features:
    temp=df.groupby(feature)['SalePrice'].count()/len(df)
    temp_df=temp[temp>0.01].index
    df[feature]=np.where(df[feature].isin(temp_df),df[feature],'Rare_var')
```

Label Encoding the Categorical Features For Machine to understand

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in categorical_features:
    df[i]=le.fit_transform(df[i])
```

Skewness in some Continuous Variables

There are a lot of skewed variables. I have treated them with log1 transformation.

Before Treating Skewness, Splitting into train and test set to avoid data leakage

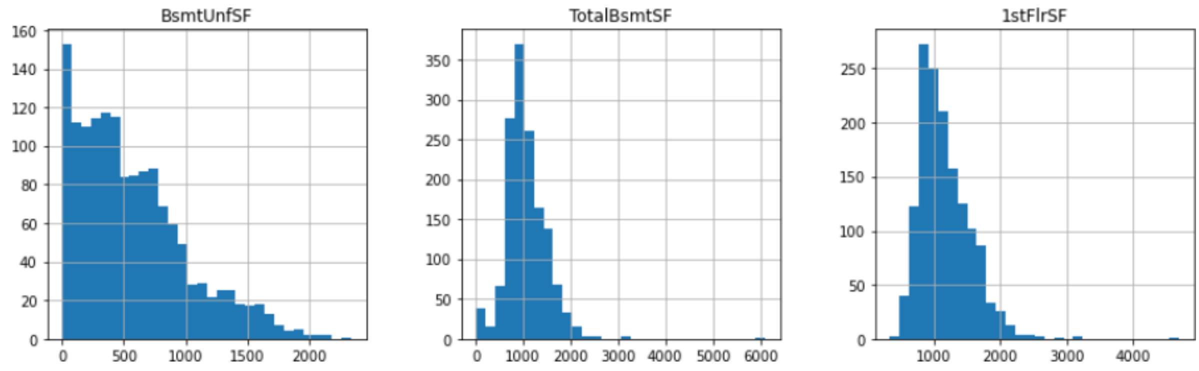
```
from sklearn.model_selection import train_test_split
df_train,df_test = train_test_split(df,train_size=0.8,test_size=0.2,random_state=42)
```

80% data will be used for training and 20% for Testing

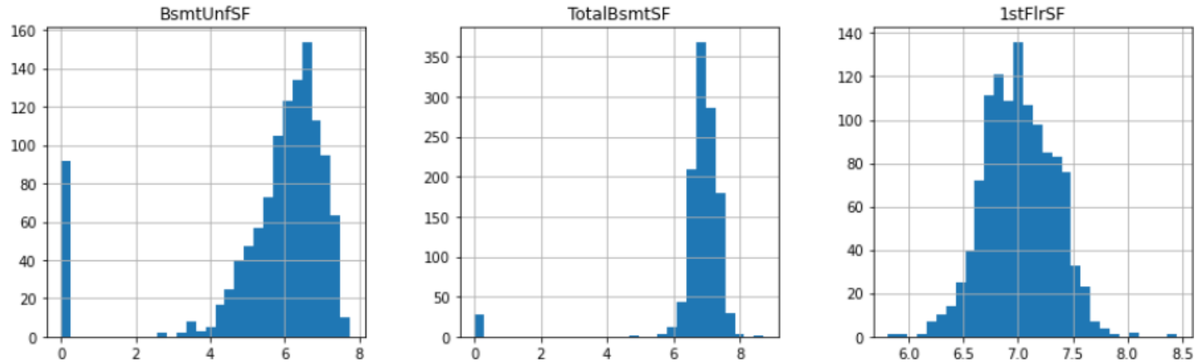
Reducing Skewness

```
for col in df_train[continuous_feature].columns:
    if df_train.skew().loc[col]>0.55 and col!='SalePrice':
        df_train[col]=np.log1p(df_train[col])
```

As seen in the below examples, I've treated all the features.



Before Treating for Skewness



After Treating for Skewness

Scaling the dataset

Splitting Dependent and Independent Features

```
y_train = df_train.pop('SalePrice')
X_train = df_train
```

```
y_test = df_test.pop('SalePrice')
X_test = df_test
```

```
#Lets scale the parameters
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_train=pd.DataFrame(X_train,columns=df_train.columns)
X_train.head()
```

```
#Lets scale the test parameters
X_test=sc.fit_transform(X_test)
X_test=pd.DataFrame(X_test,columns=df_test.columns)
X_test.head()
```

I've used Standard Scalar to make all the data comparable.

Modelling

1. Random Forest Regressor with PCA

```
# Selecting 70 features, as it explains 99% of data
```

```
pca = PCA(n_components=70)
x=pca.fit_transform(x)
x_t=X_test.copy()
x_t=pca.fit_transform(x_t)
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
parameters={'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}
rfr=RandomForestRegressor()
rand = RandomizedSearchCV(estimator = rfr, param_distributions = parameters,
                          n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1,scoring='r2')
rand.fit(x,y_train)
rand.best_params_
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:   54.6s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:  6.9min
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed: 18.9min finished
```

```
rfr=RandomForestRegressor(n_estimators =1800,
                          min_samples_split= 5,
                          min_samples_leaf= 4,
                          max_features= 'auto',
                          max_depth= 80,
                          bootstrap= True)
```

```
rfr.fit(x,y_train)
y_pred = rfr.predict(x_t)
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("RMSE is: ",np.sqrt(mean_squared_error(y_test,y_pred)))
print("r2_score is: ",r2_score(y_test,y_pred))
```

Results: Top 10 Features and R2 Score

	Features	Gini-Importance
0	MSSubClass	0.801095
1	LotFrontage	0.058321
2	LotShape	0.005449
3	Alley	0.005347
4	LotArea	0.005258
5	Utilities	0.002516
6	MSZoning	0.002034
7	Street	0.001795
8	LandContour	0.001484
9	LotConfig	0.001342

```
RMSE is:  41621.690289391634
r2_score is:  0.7741471411055758
```

2. XGBoost Regressor with PCA

```
params = {
        'min_child_weight': [1, 5, 10],
        'gamma': [0.5, 1, 1.5, 2, 5],
        'subsample': [0.6, 0.8, 1.0],
        'colsample_bytree': [0.6, 0.8, 1.0],
        'max_depth': [3, 4, 5]
        }
```

```
xg = XGBRegressor(learning_rate=0.02, n_estimators=600,
                   silent=True, nthread=1)
```

```
skf = StratifiedKFold(n_splits=5, shuffle = True, random_state = 1001)

random_search = RandomizedSearchCV(xg, param_distributions=params, n_iter=5, scoring='r2',
                                   n_jobs=4, cv=skf.split(x,y_train), verbose=3, random_state=1001 )
```

```
random_search.fit(x,y_train)
random_search.best_params_
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
```

```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done  25 out of  25 | elapsed:   45.1s finished
```

```
xg = XGBRegressor(learning_rate=0.02, n_estimators=600,
                   silent=True, nthread=1,subsample = 0.8,
                   min_child_weight= 1, max_depth = 4, gamma = 1,
                   colsample_bytree = 1.0)
```

```
xg.fit(x,y_train)
y_pred = xg.predict(x_t)
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("RMSE is: ",np.sqrt(mean_squared_error(y_test,y_pred)))
print("r2_score is: ",r2_score(y_test,y_pred))
```

Results:

```
RMSE is:  43960.40768938855
r2_score is:  0.7480527695932312
```

The score was way less than Random Forest, so I've rejected this model. Then I checked with the following models

3. Linear Regression with RFE

a. Lasso b. Ridge

Preparing the Data by reducing features using RFE

```
# Eliminate features at a step 0.05*n_featurees
from sklearn.feature_selection import RFECV
from sklearn.model_selection import KFold
def feature_RFE(model,train_data,y_data):
    support = []
    n_features = []
    scores = []
    rfecv = RFECV(estimator=model, step=0.05, cv=KFold(5,random_state=0,shuffle=True))
    rfecv.fit(train_data, y_train)
    return rfecv
```

```
# Now we run RFE for linear regression
from sklearn.linear_model import LinearRegression
lm  = LinearRegression()
rfecv = feature_RFE(lm,X_train,y_train)
```

```
print("Optimal RFE number of features : %d" % rfecv.n_features_)
print("Feature Ranking: ")
print(rfecv.ranking_)
```

```
Optimal RFE number of features : 49
Feature Ranking:
[ 1  8  1  1  1  1  2  1  1  4  1  1  7  1  1  1  1  1  1 10  1  1  1  1
  2 11  1 11  1  1  1  1  1  1  6  3  4  5  5 11  9 10  1  1 10  1  1  9
  1  1  1  1  1  1  1  1  3  1  6  1  1  1  6  1  2  1  8  7  3  1  1  1
  8  5  7  4  9  1  1]
```

```
from sklearn.feature_selection import RFE
lm.fit(X_train,y_train)
rfe = RFE(lm,49)
rfe.fit(X_train,y_train)
```

```
RFE(estimator=LinearRegression(), n_features_to_select=49)
```

```
rfe_scores = pd.DataFrame(list(zip(X_train.columns,rfe.support_,rfe.ranking_)))
rfe_scores.columns = ['Column_Names','Status','Rank']
rfe_sel_columns = list(rfe_scores[rfe_scores.Status==True].Column_Names)
```

Lets filter the train and test set for the RFE selected columns

```
X_train_lm = X_train[rfe_sel_columns]
X_test_lm = X_test[rfe_sel_columns]
```

```
X_train_lm.shape
```

```
(1168, 49)
```

3 a) Lasso regression model with Grid search CV

```
GridSearchCV(cv=KFold(n_splits=10, random_state=42, shuffle=True),
             estimator=Lasso(),
             param_grid={'alpha': [0.001, 0.01, 0.1, 1.0, 5.0, 10.0, 20.0]},
             return_train_score=True, scoring='r2', verbose=1)
```

```
lasso = Lasso(alpha=20)
lasso.fit(X_train_lm,y_train)

y_train_pred = lasso.predict(X_train_lm)
y_test_pred = lasso.predict(X_test_lm)

print(r2_score(y_true=y_train,y_pred=y_train_pred))
print(r2_score(y_true=y_test,y_pred=y_test_pred))
```

R2 Scores for Train and Test Data

```
0.8413407167403752
0.8115457630494485
```

3 b) Now lets use the ridge regression

```
GridSearchCV(cv=KFold(n_splits=10, random_state=42, shuffle=True),
             estimator=Ridge(),
             param_grid={'alpha': [0.001, 0.01, 0.1, 0.2, 0.5, 0.9, 1.0, 5.0,
                                   10.0, 20.0]},
             return_train_score=True, scoring='r2', verbose=1)
```

```
# Checking the best parameter(Alpha value)
model_cv.best_params_
```

```
{'alpha': 20.0}
```

```
ridge = Ridge(alpha=20)
ridge.fit(X_train_lm,y_train)

y_train_pred = ridge.predict(X_train_lm)
print(r2_score(y_train,y_train_pred))
y_test_pred = ridge.predict(X_test_lm)
print(r2_score(y_test,y_test_pred))
```

R2 Scores for Train and Test Data

```
0.8399787386121278
0.8112957990384801
```

Finally, after all the model testing, I’ve found Lasso Ridge to be the best performing model. Building final Model.

Final Model

```
lasso = Lasso(alpha=20)
las so. lit ( X_t ra1 n_l m, y_t ra in)

y train pred : lasso.predict(X train lm)
y te st p red = Las so . predlet (X te st La)

print(r2 score(y true:y train,y pred:y train pred))
print(r2 score(y true=y test,y pred=y test pred))
```

```
0.8413407167403752
0. 81154576 30494485
```

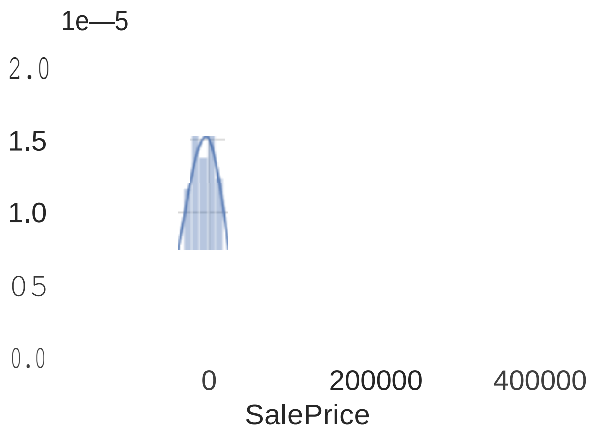
The R2 score is almost equal for both training and test data.

```
p ri nt ( "Rf'\S E is : " , np . sqrt (mean squa red e r ror (y test,y pred ) ) )
```

```
RMSE is:  4Z960.40768938855
```

```
sns.distplot(y test-y test pred)
```

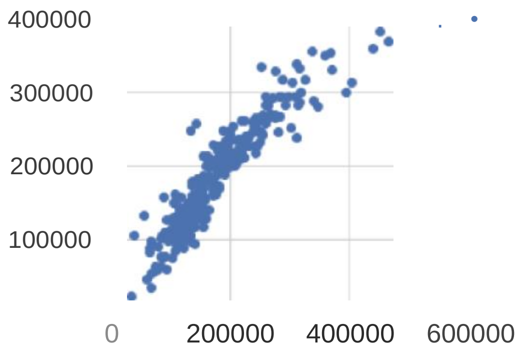
```
‹AxesSubplot:xlabel='SalePrice', ylabel='Density'>
```




```
# We are getting an almost normal distribution in our predicted values
```

```
plt.scatter(y test,y test pred)
```

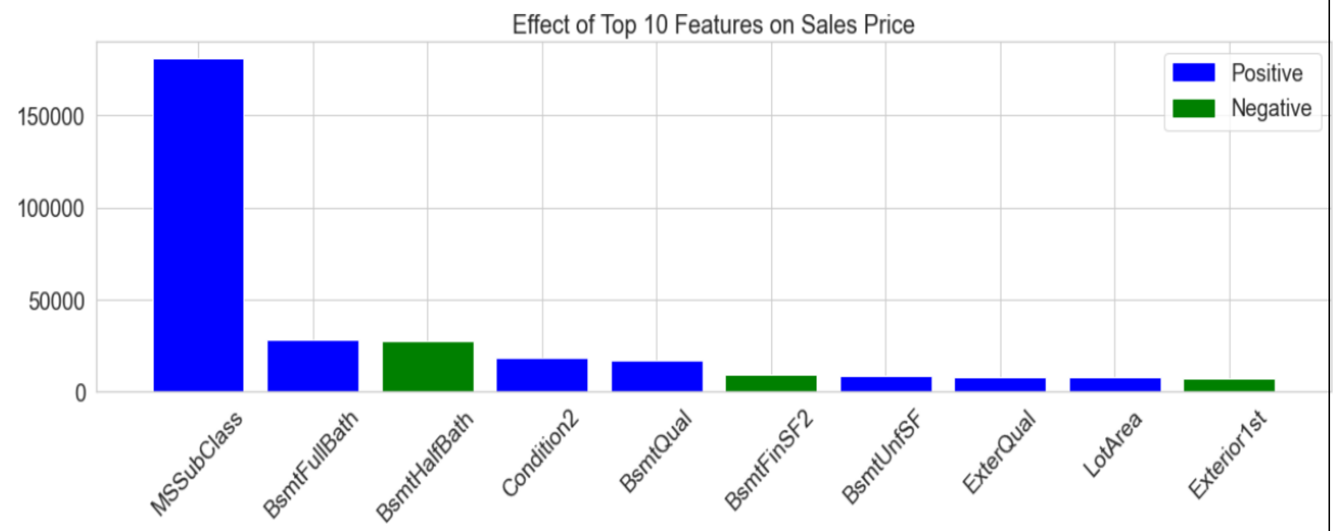
```
<matplotlib.collections.PathCollection at 0x1a95d102130›
```




```
# The model is also almost a straight line
```

Top 10 Features Based on effect on Sales Price of House

	Feature	Coef	Coef_Absolute
0	MSSubClass	181441.541952	181441.541952
46	BsmtFullBath	28383.907099	28383.907099
47	BsmtHalfBath	-27781.415681	27781.415681
13	Condition2	18222.129811	18222.129811
29	BsmtQual	16988.777175	16988.777175
35	BsmtFinSF2	-9269.760718	9269.760718
36	BsmtUnfSF	8861.579874	8861.579874
26	ExterQual	8201.874033	8201.874033
3	LotArea	8057.779549	8057.779549
22	Exterior1st	-7578.851452	7578.851452



CONCLUSION

- Key Findings and Conclusions of the Study:

  - MS Sub Class seems to have the biggest impact on House Prices, followed by Basement Full Bath and Basement Half Bath

  - Other than the Basement related features, Condition 2, Exterior Quality and Lot Area are some of the other important features.

- Learning Outcomes of the Study in respect of Data Science
  - Got to understand about the concept of Data Leakage. All transformation must be done after splitting the data to test and train, otherwise the parameters are affected.
  - Used RFE for the first time. It is a great technique for Feature Selection.
  - Learned about the usage of Lasso and Ridge Regression.