HOUSING: PRICE PREDICTION

Submitted by:

Aaron D'souza.

ACKNOWLEDGMENT

Link:
https://github.com/krishnaik06/Gaussian-Trnasformaion/blob/master/Untitled1.ipynb

Link:

https://github.com/krishnaik06/Feature-Engineering-Live-sessions/blob/master/Outliers.ipynb

Link:

https://github.com/krishnaik06/Complete-Feature-Selection/blob/master/2-Feature%20Selection-%20Correlation.ipynb

Link:

https://github.com/ashishpatel26/500-AI-Machine-learning-Deep-learning-Computer-vision-NLP-Projects-with-code

Link:

https://www.youtube.com/user/krishnaik06

INTRODUCTION

• Business Problem Framing

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.

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.

Real estate is the least transparent industry in our ecosystem. Housing prices keep changing day in and day out and sometimes are hype rather than being based on valuation. Predicting housing prices with real factors is the main crux of our project.

• Conceptual Background of the Domain Problem

The real estate market is a standout amongst the most focused regarding pricing and keeps fluctuating.

It is one of the prime fields to apply the ideas of machine learning on how to enhance and foresee the costs with high accuracy.

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.

- ## Review of Literature

Every single organization in today's real estate business is operating fruitfully to achieve a competitive edge over alternative competitors.

There is a need to simplify the process for a normal human being while providing the best results.

The value of a particular property depends on the infrastructure amenities surrounding the property.

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.

- ## Motivation for the Problem Undertaken

The three Essential for a human to nurture in today's world are Food, Cloth, and House. With the availability of these resources, human productivity can increase.

Housing plays a vital role in one's growth. From investment, to providing refuge it plays an indispensable role in one's life.

Many Real Estate companies are facing breakdown in the market due to lack of such analysis and this could also lead to disruption in countries economy as Real Estate sector contributes significantly.

The market is evolving day by day, today a lot of software giants are shifting towards Artificial intelligence for better decision-making and resolving some complicated difficulties in real-world using the data accessible in ample quantity.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

As an aspiring Data Scientist the goal is to create a model that will predict the house prices with the available independent variables.

This model will then be used by the management to understand how exactly the prices vary with the variables.

The company 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.

Based on all the independent variables the model needs to predict dependent variable (Sale Price)

A total of 8 regression models were used in order to predict the target variable Sale Price.

Linear Regression.

Random Forest Regression.

Bagging Regressor.

XGB Regressor.

ADA Boost Regressor.

Regularization(Lasso).

Regularization(Ridge)

Gradient Boosting Regressor.

- ## Data Sources and their formats

The sample data is provided to us from our client database.

A total of three data types in the data set Float, Int and Object.

The data provided was well organized structured data in .CSV (comma-separated values) format.

The data set has a total of 37 columns:

Ms-subclass: Identifies the type of dwelling involved in the sale.

20	**1-STORY 1946 & NEWER ALL STYLES**
30	**1-STORY 1945 & OLDER**
40	**1-STORY W/FINISHED ATTIC ALL AGES**
45	**1-1/2 STORY - UNFINISHED ALL AGES**
50	**1-1/2 STORY FINISHED ALL AGES**
60	**2-STORY 1946 & NEWER**
70	**2-STORY 1945 & OLDER**
75	**2-1/2 STORY ALL AGES**
80	**SPLIT OR MULTI-LEVEL**
85	**SPLIT FOYER**
90	**DUPLEX - ALL STYLES AND AGES**
120	**1-STORY PUD (Planned Unit Development) - 1946 & NEWER**
150	**1-1/2 STORY PUD - ALL AGES**
160	**2-STORY PUD - 1946 & NEWER**
180	**PUD - MULTILEVEL - INCL SPLIT LEV/FOYER**

190 2 FAMILY CONVERSION - ALL STYLES AND AGES

Ms Zoning: Identifies the general zoning classification of the sale.

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

Lot Frontage: Linear feet of street connected to property

Lot Area: Lot size in square feet

Street: Type of road access to property

Grvl	Gravel
Pave	Paved

Alley: Type of alley access to property

Grvl Gravel

Pave Paved

NA No alley access

Lot-shape: General shape of property

Reg Regular

IR1 Slightly irregular

IR2 Moderately Irregular

IR3 Irregular

Land Contour: Flatness of the property

Lvl Near Flat/Level

Bnk Banked - Quick and significant rise from street grade to building

HLS Hillside - Significant slope from side to side

Low Depression

Utilities: Type of utilities available

AllPub All public Utilities (E,G,W,& S)

NoSewr Electricity, Gas, and Water (Septic Tank)

NoSeWa Electricity and Gas Only

ELO Electricity only

LotConfig: Lot configuration

Inside	Inside lot
Corner	Corner lot
CulDSac	Cul-de-sac
FR2	Frontage on 2 sides of property
FR3	Frontage on 3 sides of property

Land-slope: Slope of property

Gtl	Gentle slope
Mod	Moderate Slope
Sev	Severe Slope

Neighborhood: Physical locations within Ames city limits

Blmngtn	Bloomington Heights

Blueste	**Bluestem**
BrDale	**Briardale**
BrkSide	**Brookside**
ClearCr	**Clear Creek**
CollgCr	**College Creek**
Crawfor	**Crawford**
Edwards	**Edwards**
Gilbert	**Gilbert**
IDOTRR	**Iowa DOT and Rail Road**
MeadowV	**Meadow Village**
Mitchel	**Mitchell**
Names	**North Ames**
NoRidge	**Northridge**
NPkVill	**Northpark Villa**
NridgHt	**Northridge Heights**
NWAmes	**Northwest Ames**
OldTown	**Old Town**
SWISU	**South & West of Iowa State University**
Sawyer	**Sawyer**
SawyerW	**Sawyer West**
Somerst	**Somerset**
StoneBr	**Stone Brook**
Timber	**Timberland**
Veenker	**Veenker**

Condition1: **Proximity to various conditions**

Artery	**Adjacent to arterial street**
Feedr	**Adjacent to feeder street**
Norm	**Normal**
RRNn	**Within 200' of North-South Railroad**
RRAn	**Adjacent to North-South Railroad**
PosN	**Near positive off-site feature--park, greenbelt, etc.**
PosA	**Adjacent to postive off-site feature**
RRNe	**Within 200' of East-West Railroad**
RRAe	**Adjacent to East-West Railroad**

Condition2: **Proximity to various conditions (if more than one is present)**

Artery	**Adjacent to arterial street**
Feedr	**Adjacent to feeder street**
Norm	**Normal**
RRNn	**Within 200' of North-South Railroad**
RRAn	**Adjacent to North-South Railroad**
PosN	**Near positive off-site feature--park, greenbelt, etc.**
PosA	**Adjacent to postive off-site feature**
RRNe	**Within 200' of East-West Railroad**
RRAe	**Adjacent to East-West Railroad**

BldgType: Type of dwelling

1Fam Single-family Detached

2FmCon Two-family Conversion; originally built as one-family dwelling

Duplx Duplex

TwnhsE Townhouse End Unit

TwnhsI Townhouse Inside Unit

House-style: Style of dwelling

1Story One story

1.5Fin One and one-half story: 2nd level finished

1.5Unf One and one-half story: 2nd level unfinished

2Story Two story

2.5Fin Two and one-half story: 2nd level finished

2.5Unf Two and one-half story: 2nd level unfinished

SFoyer Split Foyer

SLvl Split Level

OverallQual: **Rates the overall material and finish of the house**

10 Very Excellent

9 Excellent

8 Very Good

7 Good

6 Above Average

5 Average

4 Below Average

3 Fair

2 Poor

1 Very Poor

OverallCond: **Rates the overall condition of the house**

10 Very Excellent

9 Excellent

8 Very Good

7 Good

6 Above Average

5 Average

4 Below Average

3 Fair

2 Poor

1 Very Poor

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

Flat	Flat
Gable	Gable
Gambrel	Gabrel (Barn)
Hip	Hip
Mansard	Mansard
Shed	Shed

RoofMatl: Roof material

ClyTile	Clay or Tile
CompShg	Standard (Composite) Shingle
Membran	Membrane
Metal	Metal
Roll	Roll

Tar&Grv	Gravel & Tar
WdShak	Wood Shakes
WdShngl	Wood Shingles

Exterior1st: Exterior covering on house

AsbShng	Asbestos Shingles
AsphShn	Asphalt Shingles
BrkComm	Brick Common
BrkFace	Brick Face
CBlock	Cinder Block
CemntBd	Cement Board
HdBoard	Hard Board
ImStucc	Imitation Stucco
MetalSd	Metal Siding
Other	Other
Plywood	Plywood
PreCast	PreCast
Stone	Stone
Stucco	Stucco
VinylSd	Vinyl Siding
Wd Sdng	Wood Siding
WdShing	Wood Shingles

Exterior2nd: Exterior covering on house (if more than one material)

AsbShng	Asbestos Shingles
AsphShn	Asphalt Shingles
BrkComm	Brick Common
BrkFace	Brick Face
CBlock	Cinder Block
CemntBd	Cement Board
HdBoard	Hard Board
ImStucc	Imitation Stucco
MetalSd	Metal Siding
Other	Other
Plywood	Plywood
PreCast	PreCast
Stone	Stone
Stucco	Stucco
VinylSd	Vinyl Siding
Wd Sdng	Wood Siding
WdShing	Wood Shingles

MasVnrType: Masonry veneer type

BrkCmn Brick Common

BrkFace Brick Face

CBlock	Cinder Block
None	None
Stone	Stone

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

Ex	Excellent
Gd	Good
TA	Average/Typical FaFair
Po	Poor

ExterCond: Evaluates the present condition of the material on the exterior

Ex	Excellent
Gd	Good
TA	Average/Typical FaFair
Po	Poor

Foundation: Type of foundation

BrkTil Brick & Tile

CBlock Cinder Block

PConc Poured Contrete

Slab Slab

Stone Stone

Wood Wood

BsmtQual: Evaluates the height of the basement

ExExcellent (100+ inches)

Gd Good (90-99 inches)

TA Typical (80-89 inches)

FaFair (70-79 inches)

Po Poor (<70 inches

NA No Basement

BsmtCond: Evaluates the general condition of the basement

Ex Excellent

Gd Good

TA Typical - slight dampness allowed

FaFair dampness or some cracking or settling

Po Poor - Severe cracking, settling, or wetness

NA No Basement

BsmtExposure: Refers to walkout or garden level walls

Gd	Good Exposure
Av	Average Exposure (split levels or foyers typically sc average or above)
Mn	Minimum Exposure
No	No Exposure
NA	No Basement

BsmtFinType1: Rating of basement finished area

GLQ	Good Living Quarters
ALQ	Average Living Quarters
BLQ	Below Average Living Quarters
Rec	Average Rec Room
LwQ	Low Quality
Unf	Unfinshed
NA	No Basement

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

GLQ	Good Living Quarters
ALQ	Average Living Quarters

BLQ	Below Average Living Quarters
Rec	Average Rec Room
LwQ	Low Quality
Unf	Unfinshed
NA	No Basement

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

Floor	Floor Furnace
GasA	Gas forced warm air furnace
GasW	Gas hot water or steam heat
Grav	Gravity furnace
OthW	Hot water or steam heat other than gas
Wall	Wall furnace

HeatingQC: Heating quality and condition

Ex	Excellent
Gd	Good

TA Average/Typical Fa-Fair

Po Poor

CentralAir: Central air conditioning

N No

Y Yes

Electrical: Electrical system

SBrkr	Standard Circuit Breakers & Romex
FuseA	Fuse Box over 60 AMP and all Romex wiring (Average)
FuseF	60 AMP Fuse Box and mostly Romex wiring (Fair)
FuseP	60 AMP Fuse Box and mostly knob & tube wiring (poor)
Mix	Mixed

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

Ex	**Excellent**
Gd	**Good**
TA	**Typical/Average FaFair**
Po	**Poor**

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality (Assume typical unless deductions are warranted)

Typ	Typical Functionality
Min1	Minor Deductions 1
Min2	Minor Deductions 2
Mod	Moderate Deductions
Maj1	Major Deductions 1
Maj2	Major Deductions 2
Sev	Severely Damaged
Sal	Salvage only

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

Ex	Excellent - Exceptional Masonry Fireplace
Gd	Good - Masonry Fireplace in main level
TA	Average - Prefabricated Fireplace in main living area or Masonry Fireplace in basement
Fa	Fair - Prefabricated Fireplace in basement
Po	Poor - Ben Franklin Stove
NA	No Fireplace

GarageType: Garage location

2Types	More than one type of garage

Attchd Attached to home

Basment Basement Garage

BuiltIn Built-In (Garage part of house - typically has room above garage)

CarPort Car Port

Detchd Detached from home

NA No Garage

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

Fin Finished

RFn Rough Finished

Unf Unfinished

NA No Garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

NA No Garage

GarageCond: Garage condition

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

NA No Garage

PavedDrive: Paved driveway

Y Paved

P Partial Pavement

N Dirt/Gravel

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**

Ex	**Excellent**
Gd	**Good**
TA	**Average/Typical**
Fa	**Fair**
NA	**No Pool**

Fence: **Fence quality**

GdPrv	**Good Privacy**
MnPrv	**Minimum Privacy**
GdWo	**Good Wood**
MnWw	**Minimum Wood/Wire**
NA	**No Fence**

MiscFeature: **Miscellaneous feature not covered in other categories**

Elev	**Elevator**
Gar2	**2nd Garage (if not described in garage section)**
Othr	**Other**
Shed	**Shed (over 100 SF)**
TenC	**Tennis Court**
NA	**None**

MiscVal: **$Value of miscellaneous feature**

MoSold: **Month Sold (MM)**

YrSold: **Year Sold (YYYY)**

SaleType: **Type of sale**

WD	**Warranty Deed - Conventional**
CWD	**Warranty Deed - Cash**

VWD	Warranty Deed - VA Loan
New	Home just constructed and sold
COD	Court Officer Deed/Estate
Con	Contract 15% Down payment regular terms
ConLw	Contract Low Down payment and low interest
ConLI	Contract Low Interest
ConLD	Contract Low Down
Oth	Other

SaleCondition: Condition of sale

Normal	Normal Sale
Abnorml	Abnormal Sale - trade, foreclosure, short sale
AdjLand	Adjoining Land Purchase
Alloca	Allocation - two linked properties with separate deeds, typically condo with a garage unit
Family	Sale between family members
Partial	Home was not completed when last assessed (associated with New Homes)

```
1 df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Uti
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	A
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	A
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	A
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	A
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	A

Fig.1 (Data frame Sample)

- # Data Preprocessing Done

Some Null values in the data set.

```
Your selected dataframe has 81 columns.
There are 18 columns that have missing values.
```

Out[14]:

	Missing Values	% of Total Values
PoolQC	1161	99.4
MiscFeature	1124	96.2
Alley	1091	93.4
Fence	931	79.7
FireplaceQu	551	47.2
LotFrontage	214	18.3
GarageType	64	5.5
GarageYrBlt	64	5.5

Fig.2 (Null Values)

There were no duplicated values in the dataset.

Lets check for duplicate values

```
In [9]:  1  df.duplicated().sum()
Out[9]: 0
```

Fig.3 (No Duplicated values)

Dropping PoolQC, Fence, Alley and MiscFeature columns as they have more then 75 % of missing data.

- We can drop the Id column as it has no special significance in predicting the house prices.
- We can drop PoolQC, Fence, Alley and MiscFeature columns as they have more then 75 % of missing data.

```
In [21]:
# Dropping some columns
df = df.drop(["Id","PoolQC","Fence","Alley","MiscFeature"],axis=1)

In [22]:
# Similarly for dropping the values for test data set
df_test = df_test.drop(["Id","PoolQC","Fence","Alley","MiscFeature"],axis=1)
```

Fig.4 (Dropping some columns)

Most of the missing values were replaced with mean , median and mode of the data if the variable was continuous in nature then it was replaced by (Mean or Median) and if the data was categorical it was replaced by mode of the data. Similar steps were followed for test data.

```
In [40]:
# Handling missing values in GarageCond
# Replacing null values in GarageCond with mode of data

df['BsmtExposure'] = df['BsmtExposure'].fillna("No")
df["BsmtExposure"].isnull().sum()

Out[40]: 0

In [41]:
# Similarly for test data

df_test['BsmtExposure'] = df_test['BsmtExposure'].fillna("No")
df_test['BsmtExposure'].isnull().sum()

Out[41]: 0
```

Fig.5 (Replacing null values)

Finally there were no missing values in training and testing database

```
In [55]:
# No more missing values in train data
df.isnull().sum()

Out[55]: MSSubClass       0
         MSZoning         0
         LotFrontage      0
         LotArea          0
         Street           0
         LotShape         0
         LandContour      0
         Utilities        0
         LotConfig        0
         LandSlope        0
         Neighborhood     0
         Condition1       0
```

Fig.6 (No more null values)

The train and test data was combined in order to perform further preprocessing.

Combining train and test data

In [115]:
```
df_combine = pd.concat([df,df_test])
```

In [116]:
```
df_combine.head()
```

Out[116]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	Lan
0	120	RL	70.98847	4928	Pave	IR1	Lvl	Inside	
1	20	RL	95.00000	15865	Pave	IR1	Lvl	Inside	
2	60	RL	92.00000	9920	Pave	IR1	Lvl	CulDSac	
3	20	RL	105.00000	11751	Pave	IR1	Lvl	Inside	
4	20	RL	70.98847	16635	Pave	IR1	Lvl	FR2	

Fig.7 (combining Train and test data)

Three functions were created to handle outliers in data

```
# Outlier detection for features which seems normally distributed

def normal(col):
    # Lower outlier
    low = df_combine[col].mean() - 3*df_combine[col].std()
    # Upper outlier
    up = df_combine[col].mean() + 3*df_combine[col].std()

    return (low,up)
```

Fig.8 (Normal distribution Outlier Detection)

```
# features with skewness

def outSkew(col):
    # Calculating IQR
    Iqr = df_combine[col].quantile(0.75)-df_combine[col].quantile(0.25)

    # Lower outlier
    low = df_combine[col].quantile(0.25)-(Iqr*1.5)

    # upper outlier
    up = df_combine[col].quantile(0.75)+(Iqr*1.5)

    return (low,up)
```

Fig.9 (Some Skewness Outlier Detection)

```
# features with very high skewness

def highSKew(col):

    # Calculating IQR
    Iqr = df_combine[col].quantile(0.75)-df_combine[col].quantile(0.25)

    # First Quartile
    low = df_combine[col].quantile(0.25)-(Iqr*3)

    # third Quartile
    up = df_combine[col].quantile(0.75)+(Iqr*3)

    return (low,up)
```

Fig.10 (Very High Skewness Outlier Detection)

Using Histograms, QQ Plots and Box Plots to understand the distribution of data, to know if the data was Normal , Skewed or Highly Skewed.

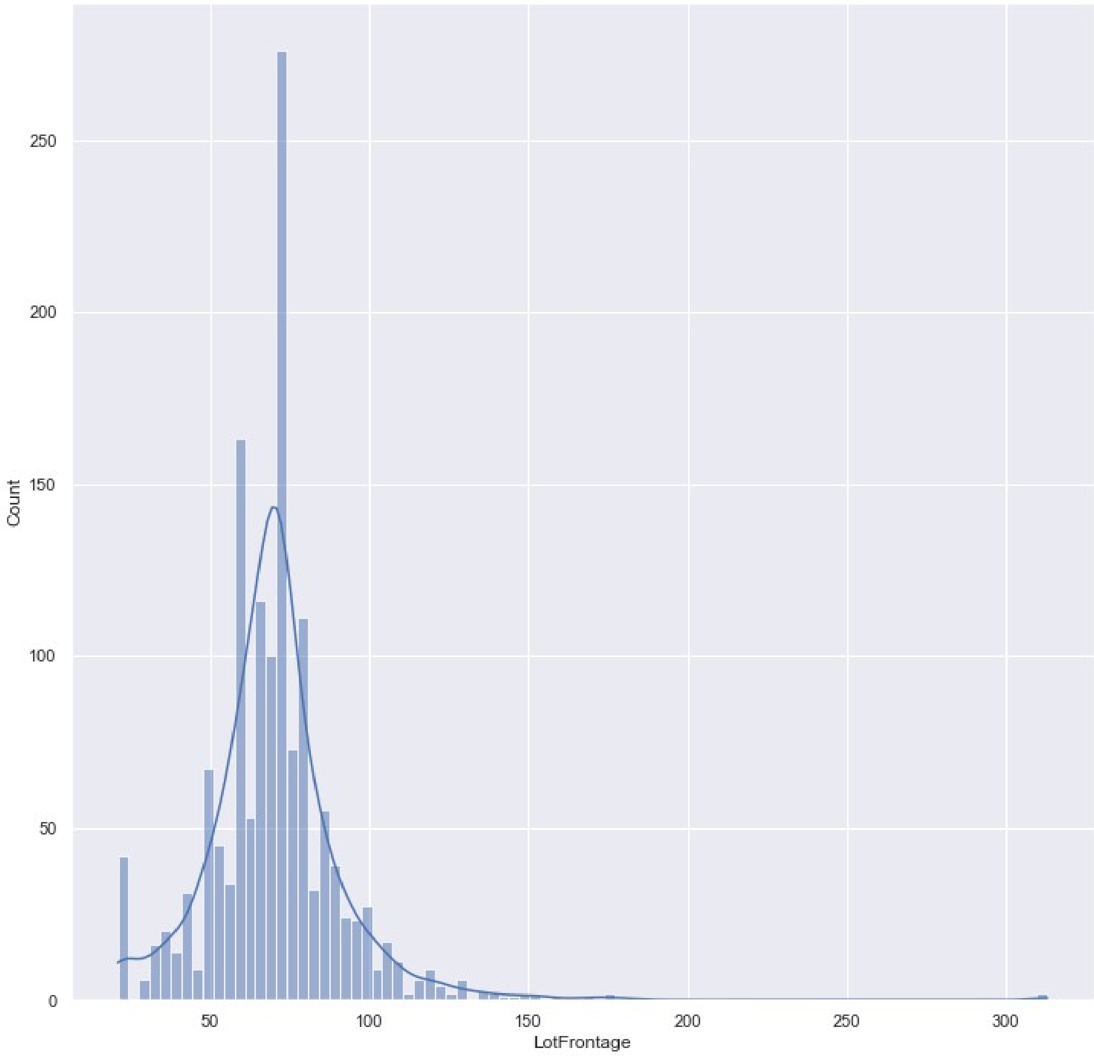

Fig.11 (The feature looks normally distributed)

```
QQ_plots("LotFrontage")
```

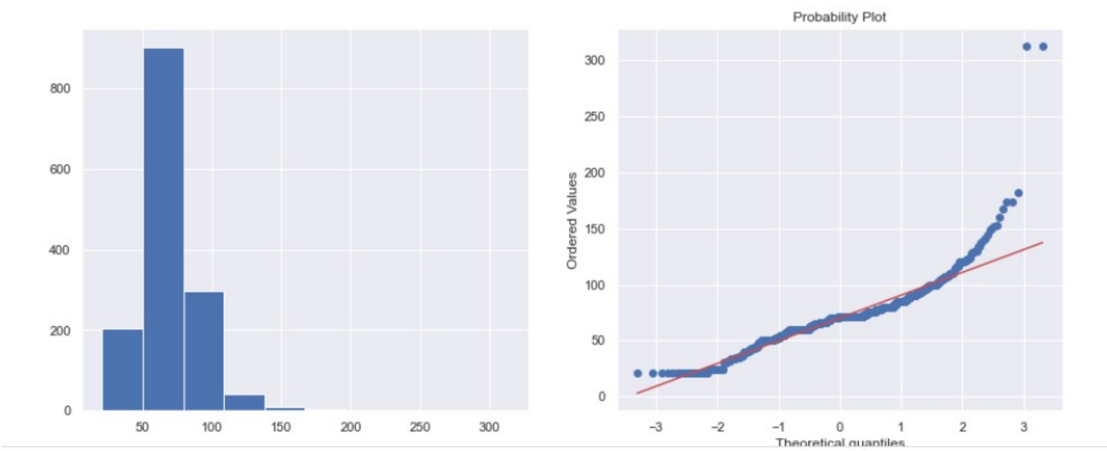

Fig.12 (Understanding Distribution)

```
In [133]:  # Outlier detection
           normal("LotFrontage")

Out[133]:  (3.967382859776265, 136.18420909227802)

In [134]:  # Anything above 136.18 is considered as an outlier

           df_combine.loc[df_combine["LotFrontage"]>=136.18,"LotFrontage"] = 136.18
```

Fig.13 (Handling Outliers)

```
In [136]:  # Checking Skewness
           df_combine["LotFrontage"].skew()

Out[136]:  0.4191088792007457
```

Fig.14 (Checking Skewness)

If the skewness of the data is still high after outlier removal then various transformation methods are used to reduce skewness like Log Transformation, reciprocal Transformation, Square root and exponential Transformation.

```
In [124]:
# function for Log transformation

def logt(col):
    df_combine[col] = np.log(df_combine[col]+1)
    return df_combine[col].skew()
```

```
In [125]:
# function for resiprocal transformation

def resiprocal(col):
    df_combine[col]=1/(df_combine[col]+1)
    return df_combine[col].skew()
```

Fig.15 (Functions for Transforming skewed data)

After Handling transformations and outliers Continuous and categorical data is separated in order to perform encoding on categorical data.

```
In [279]:
# Label ENcoding most of the ordinal variables

In [280]:
# Label encoding Street feature
df_categorical["Street"] = df_categorical["Street"].replace({"Grvl":1,"Pave"

In [281]:
# Label encoding LotShape feature
df_categorical["LotShape"] = df_categorical["LotShape"].replace({'IR3': 1, '

In [282]:
# Label encoding LandContour feature
df_categorical["LandContour"] = df_categorical["LandContour"].replace({'Low'
```

Fig.16 (Applying Label encoding on ordinal data)

```
In [305]:
# Using pandas .get_dummies()
df_final = pd.get_dummies(df_final,drop_first=True)

In [306]:
df_final.head()
```

Out[306]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	Mas\
0	120	70.98847	8.502891	6	5	1976	1976	0
1	20	95.00000	9.671934	8	6	1970	1970	0
2	60	92.00000	9.202409	7	5	1996	1997	0
3	20	105.00000	9.371779	6	6	1977	1977	6

Fig.17 (Applying pandas get dummies on rest of the categorical data)

The data preprocessing part is done now we need to separate both the train and test data which we combined earlier.

Using Sk learn's Standard Scaler to scale the data

```
In [320]:  # using fit transform of test data
           scale_x = sc.fit_transform(x)

In [321]:  x = pd.DataFrame(scale_x,columns=x.columns)

In [322]:  x.head()
Out[322]:
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	Mas
0	1.508301	0.026142	-1.258118	-0.075169	-0.530217	0.168236	-0.421565	-(
1	-0.877042	1.263197	1.153847	1.364138	0.359572	-0.030885	-0.710356	-(

Fig.18 (Scaling the data)

Now we need to use Feature selection in order to reduce the number of input variables when developing a predictive model.

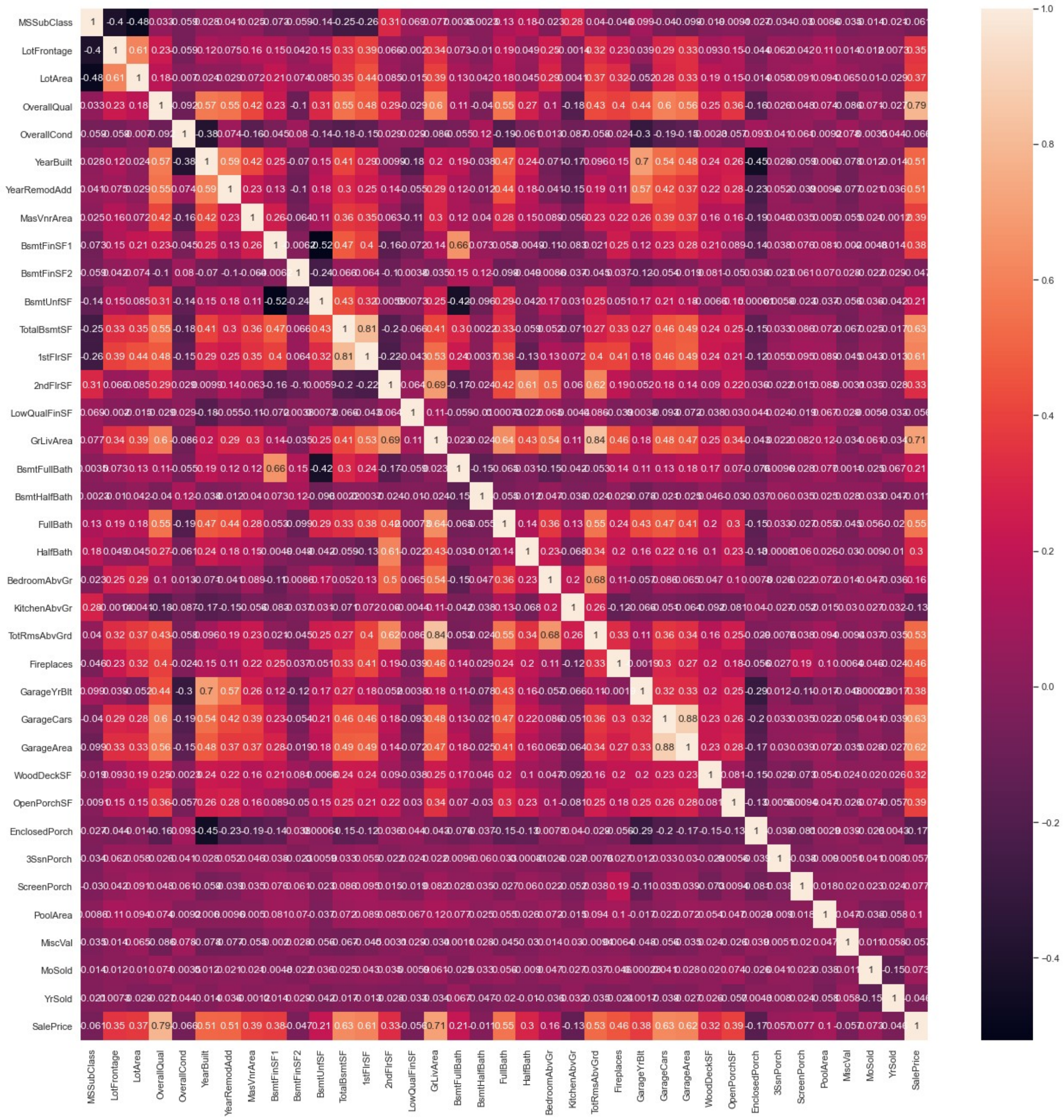

Fig.19 (Checking correlation in continuous data)

```
In [329]:
# We can clearly see the features which are highly correlated with the t
cor["SalePrice"].sort_values(ascending=False)
```

```
Out[329]: SalePrice       1.000000
          OverallQual     0.789185
          GrLivArea       0.714038
          TotalBsmtSF     0.634836
          GarageCars      0.628329
          GarageArea      0.619000
          1stFlrSF        0.610390
          FullBath        0.554988
          TotRmsAbvGrd    0.528363
          YearBuilt       0.514408
          YearRemodAdd    0.507831
          Fireplaces      0.459611
          MasVnrArea      0.392208
```

Fig.20 (Some highly correlated features with target variable)

No we need to remove some features which are highly correlated to each other, removing features which have more than 80 % correlation between them.

```
def correlation(dataset, threshold):
    # Set of all the names of correlated columns
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            # we are interested in absolute coeff value
            if abs(corr_matrix.iloc[i, j]) > threshold:
                # getting the name of column
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr
```

Fig.21(Function to remove highly correlated features)

We have three features which have more than 80% correlation between them we can drop all 3 of them

```
In [331]:
# creating a variable to store the function results
corr_feat = correlation(df_continuous,0.8)
```

```
In [332]:
# we have 3 features which are highly correlated so we can drop them
corr_feat

Out[332]: {'1stFlrSF', 'GarageArea', 'TotRmsAbvGrd'}
```

```
In [333]:
len(set(corr_feat))

Out[333]: 3
```

```
In [334]:
# dropping the features
x = x.drop(corr_feat,axis=1)
```

Fig.22(dropping highly correlated features)

Using mutual_info_regression from sk-learn's feature selection library to find out the features with high weight-age

```
In [339]:
# using mutual_info_regression to select the best features
from sklearn.feature_selection import mutual_info_regression
```

```
In [340]:
# determine the mutual information

mutual_info = mutual_info_regression(x,y)
```

```
In [341]:
# mutual information of each individual feature
mutual_info

Out[341]: array([2.60475592e-01, 1.90790294e-01, 1.61036487e-01, 5.94155940e-01,
           1.12123667e-01, 3.62350744e-01, 2.52402413e-01, 1.26287800e-01,
           1.46143829e-01, 2.87463908e-03, 1.33239989e-01, 3.64739056e-01,
           2.08052336e-01, 2.15646395e-02, 4.49550560e-01, 2.82653861e-02,
           1.64979723e-02, 2.64107933e-01, 9.32396682e-02, 7.23820236e-02,
           2.05720066e-02, 1.59543090e-01, 2.28303714e-01, 3.57926172e-01,
```

Fig.22(Most important features)

Using Select Percentile from sk-learn's feature selection library to find find top 50% of the most important features which impact the Sale Price of the house

```
In [345]:  selected_top_columns.fit(x,y)

Out[345]: SelectPercentile(percentile=50,
                           score_func=<function mutual_info_regression at 0x0000025C126E0
          280>)

In [346]:  imp_columns = x.columns[selected_top_columns.get_support()]

In [347]:  # total of 92 features
           len(imp_columns)

Out[347]: 92
```

Fig.23(A total of 92 columns)

Sklearn's Train Test split was used. And the initial random state was taken as zero.

```
In [360]:  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, ra

In [361]:  X_train.shape

Out[361]: (817, 92)

In [362]:  X_test.shape

Out[362]: (351, 92)

In [363]:  y_train.shape

Out[363]: (817,)
```

Fig.24(Performing train test split)

- # Data Inputs- Logic- Output Relationships

```
In [342]:
# Sorting in desending order from most important feature to the least import
mutual_info = pd.Series(mutual_info)
mutual_info.index = x.columns
mutual_info.sort_values(ascending=False)
```

```
Out[342]:
OverallQual        0.594156
GrLivArea          0.449551
TotalBsmtSF        0.364739
YearBuilt          0.362351
GarageCars         0.357926
KitchenQual        0.335972
BsmtQual           0.327164
ExterQual          0.326494
FullBath           0.264108
MSSubClass         0.260476
YearRemodAdd       0.252402
GarageYrBlt        0.228304
```

Fig.25(Features having high impact on the prices)

Based on the analysis in fig 25 features like Overall Quality GrLivArea(Above grade (ground) living area square feet), TotalBsmtSF(Total square feet of basement area) and Year Built are some of the very important independent features which are having a very high impact on the dependent variable (Sales Price) of the house.

- ## Hardware and Software Requirements and Tools Used

Hardware used:

- **OS: Windows 10 Home Single Language 64 bit**
- **Ram: 8 GB**
- **Processor: Intel I5**

Software used:

- **Jupyter Notebook**

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods)**

Given the number of inputs the Machine learning Algorithm has to predict the Sale Price column.

Target Variable is Sale Price. The target variable is continuous in nature.

The best approach is to solve this as a regression problem using various regression algorithms and find out which algorithm gives the best results

- Testing of Identified Approaches (Algorithms)

Linear Regression:

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

Random Forest Regression:

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.

Bagging Regressor:

A Bagging regressor is an ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

XGB Regressor:

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks.

ADA Boost Regressor:

AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique that is used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights to incorrectly classified instances.

Regularization(Lasso):

Lasso regression is a regularization technique. It is used over regression methods for a more accurate prediction. This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters).

Regularization(Ridge):

Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values to be far away from the actual values.

Gradient Boosting Regressor:

Gradient boosting is a machine learning technique
for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models,
typically decision trees.

- Run and Evaluate selected models

Linear Regression.

In [369]:
```
# Taking best random state as 29
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, ra
mod_1 = LinearRegression()
mod_1.fit(X_train,y_train)
train_score_1 = mod_1.score(X_train,y_train)
pred_1 = mod_1.predict(X_test)
test_score_1 = r2_score(y_test,pred_1)

print("The training accuracy is :",train_score_1)
print("The testing accuracy is :",test_score_1)
print("\n")
```

```
The training accuracy is : 0.8603345250474552
The testing accuracy is : 0.8640956918662577
```

Fig.26(Linear Regression Results)

Random Forest Regression.

In [382]:
```
# Taking the best random state as 50
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, ra
mod_2 = RandomForestRegressor()
mod_2.fit(X_train,y_train)
train_score_2 = mod_2.score(X_train,y_train)
pred_2 = mod_2.predict(X_test)
test_score_2 = r2_score(y_test,pred_2)

print("The training accuracy is :",train_score_2)
print("The testing accuracy is :",test_score_2)
print("\n")
```

```
The training accuracy is : 0.9758901604378921
The testing accuracy is : 0.9043512854759762
```

Fig.27(Random Forest Regression Results)

Bagging Regressor.

In [412]:
```
# The best random state is 6
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, ra
mod_6 = BaggingRegressor()
mod_6.fit(X_train,y_train)
train_score_6 = mod_6.score(X_train,y_train)
pred_6 = mod_6.predict(X_test)
test_score_6 = r2_score(y_test,pred_6)

print("The training accuracy is :",train_score_6)
print("The testing accuracy is :",test_score_6)
print("\n")
```

```
The training accuracy is : 0.9667131459350262
The testing accuracy is : 0.872545711557754
```

Fig.28(Bagging Regressor Results)

XGB Regressor.

In [423]:
```
# Taking the best random state as 4
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, ra
mod_7 = XGBRegressor()
mod_7.fit(X_train,y_train)
train_score_7 = mod_7.score(X_train,y_train)
pred_7 = mod_7.predict(X_test)
test_score_7 = r2_score(y_test,pred_7)

print("The training accuracy is :",train_score_7)
print("The testing accuracy is :",test_score_7)
print("\n")
```

```
The training accuracy is : 0.9999478566670498
The testing accuracy is : 0.8808550733599081
```

Fig.29(XGB Regressor Results)

ADA Boost Regressor.

In [402]:
```
# Taking the best random state as 12
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, ra
mod_4 = AdaBoostRegressor()
mod_4.fit(X_train,y_train)
train_score_4 = mod_4.score(X_train,y_train)
pred_4 = mod_4.predict(X_test)
test_score_4 = r2_score(y_test,pred_4)

print("The training accuracy is :",train_score_4)
print("The testing accuracy is :",test_score_4)
print("\n")
```

```
The training accuracy is : 0.8636459558602826
The testing accuracy is : 0.8428731363608536
```

Fig.30(ADA Boost Regressor Results)

Regularization(Lasso).

In [438]:

```python
# Taking the best random state as 24
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, ra
mod_8 = Lasso()
mod_8.fit(X_train,y_train)
train_score_8 = mod_8.score(X_train,y_train)
pred_8 = mod_8.predict(X_test)
test_score_8 = r2_score(y_test,pred_8)

print("The training accuracy is :",train_score_8)
print("The testing accuracy is :",test_score_8)
print("\n")
```

```
The training accuracy is : 0.8675876871784429
The testing accuracy is : 0.8648448307926586
```

Fig.31(Lasso Regressor Results)

Regularization(Ridge)

In [448]:

```python
# taking the best random state as 3
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, ra
mod_9 = Ridge()
mod_9.fit(X_train,y_train)
train_score_9 = mod_9.score(X_train,y_train)
pred_9 = mod_9.predict(X_test)
test_score_9 = r2_score(y_test,pred_9)

print("The training accuracy is :",train_score_9)
print("The testing accuracy is :",test_score_9)
print("\n")
```

```
The training accuracy is : 0.8700719118897562
The testing accuracy is : 0.851440716531349
```

Fig.32(Ridge Regressor Results)

Gradient Boosting Regressor.

```
In [392]:
# using the best random state as 6
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, ra
mod_3 = GradientBoostingRegressor()
mod_3.fit(X_train,y_train)
train_score_3 = mod_3.score(X_train,y_train)
pred_3 = mod_3.predict(X_test)
test_score_3 = r2_score(y_test,pred_3)

print("The training accuracy is :",train_score_3)
print("The testing accuracy is :",test_score_3)
print("\n")
```

```
The training accuracy is : 0.9708149351319942
The testing accuracy is : 0.906691220987045
```

Fig.33(Gradient Boosting Regressor Results)

- ## Key Metrics for success in solving problem under consideration

Median:

It is the middle value in the sorted(ordered) data. Median is a better measure of centre than mean as it is not affected by outliers. Most of the data had enormous non-realistic values, those values were replaced with the medians of the data in order to reduce the outliers.

Mean:

The Average of the most common value in the collection of numbers

Mode:

The most repeated value in the data.

Measures of spread:

Quartiles were used in order to detect the outliers. The lower quartile (Q1) is the value between the lowest 25% of values and the highest 75% of values. It is also called the 25th percentile.
The second quartile (Q2) is the middle value of the data set. It is also called the 50th percentile, or the median.
The upper quartile (Q3) is the value between the lowest 75% and the highest 25% of values. It is also called the 75th percentile.

Correlation:

Correlation is simply, a metric that measures the extent to which variables (or features or samples or any groups) are associated with one another.
Pearson Correlation Coefficient is a statistic that also measures the linear correlation between two features.

K Fold Cross Validation:

Cross-validation is a statistical method used to estimate the skill of machine learning models.
It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

Kfold Crrossvalidation

In [396]:
```
kfold = KFold(n_splits=10,random_state=24)

K_results = cross_val_score(mod_3,X,y,cv=kfold)

kfold_accuracy_GB = np.mean(abs(K_results))
```

In [397]:
```
kfold_accuracy_GB
```

Out[397]:
```
0.8777162852851246
```

Fig.34(K fold cross validation)

- Visualizations

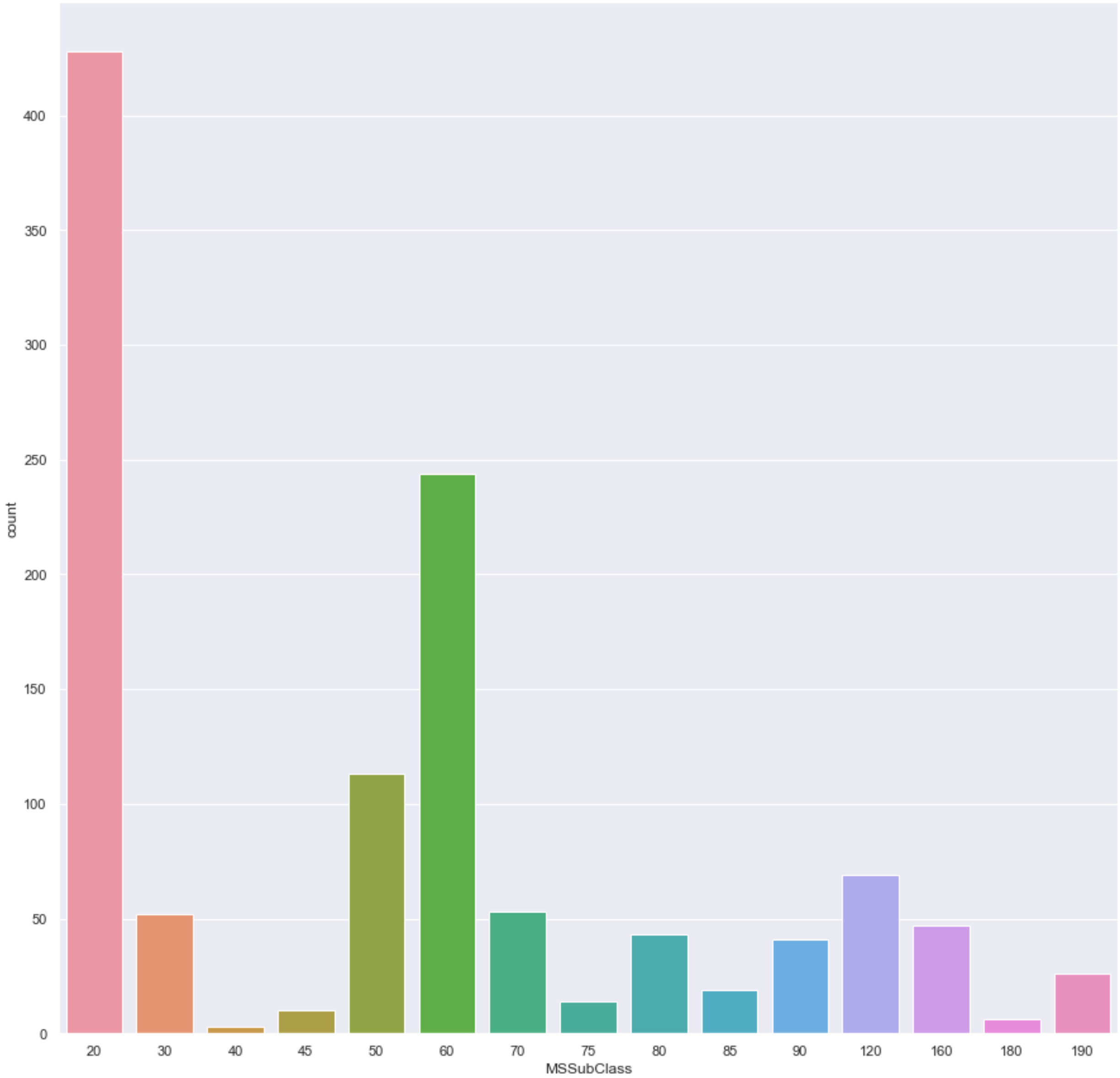

Fig.35(MSSubClass count plot)

MSSubClass Identifies the type of dwelling(a house, flat, or other place of residence) involved in the sale.
20 (1-STORY 1946 & NEWER ALL STYLES) is the most acquainted dwelling.
40 (1-STORY W/FINISHED ATTIC ALL AGES) is the least acquainted dwelling.

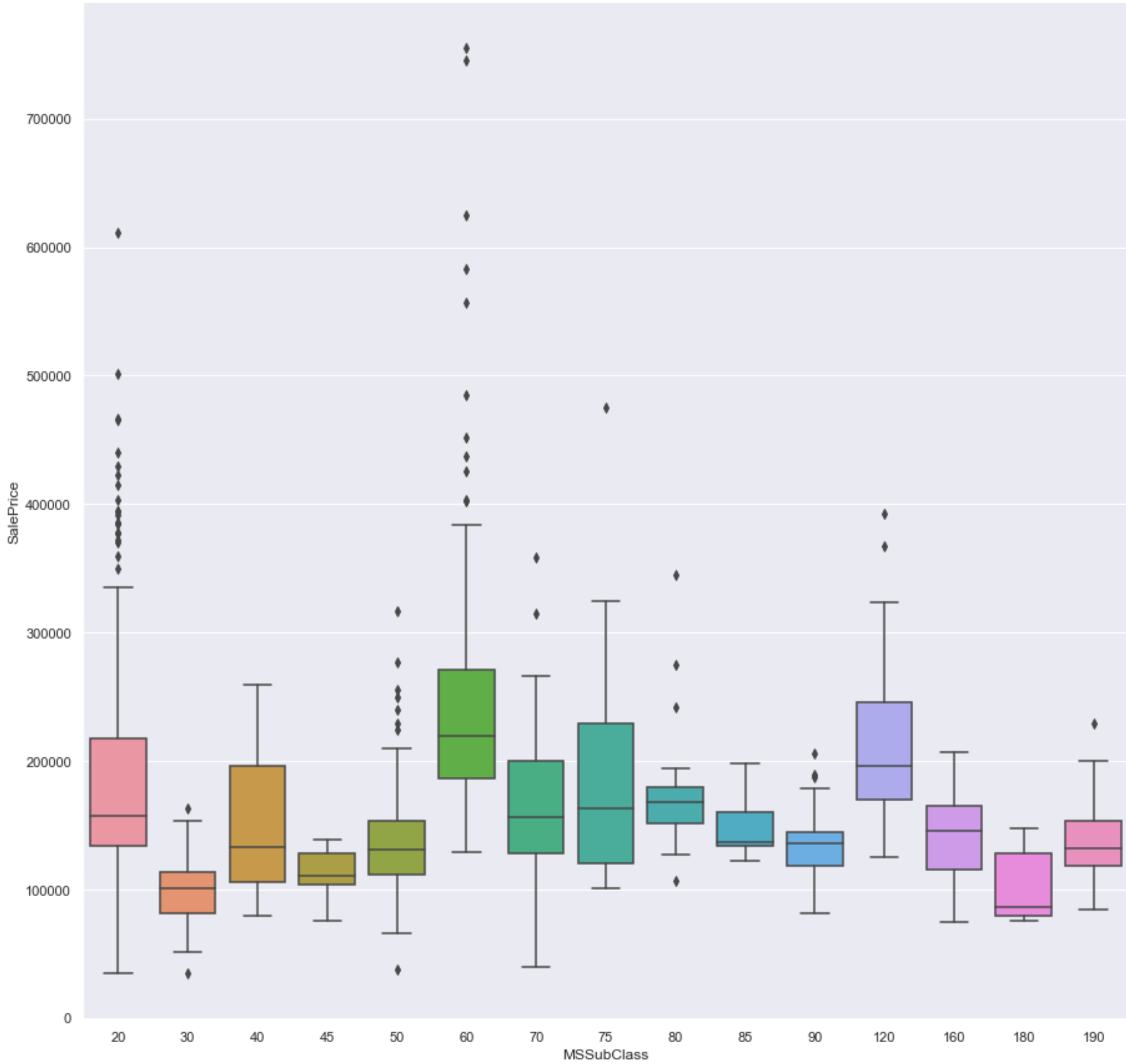

Fig.36(Ms-subclass vs Sales Price)

The dwelling type 60 (2-STORY 1946 & NEWER) in terms of price is the most expensive one.
30 (1-STORY 1945 & OLDER) type dwelling is the cheapest one
Basically Newer dwelling more expensive than older dwelling.

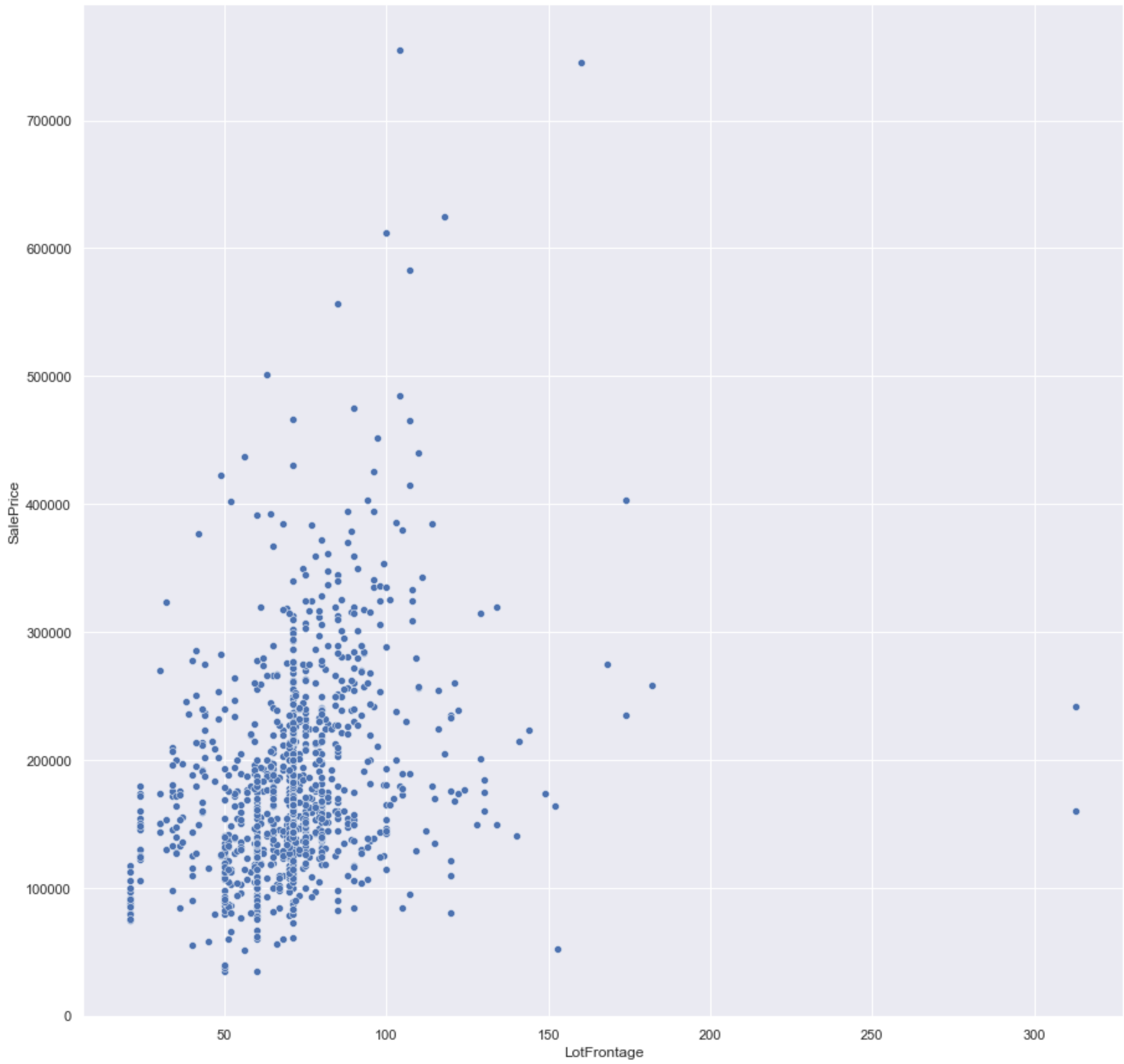

Fig.37(Lot Frontage vs Sales Price)

Most of the houses in price range 100000$, 300000$ are having Linear feet of street connected to property between (25 feet to 100 feet)

There are some very expensive houses with Lot Frontage range between(100 feet to 200 feet)

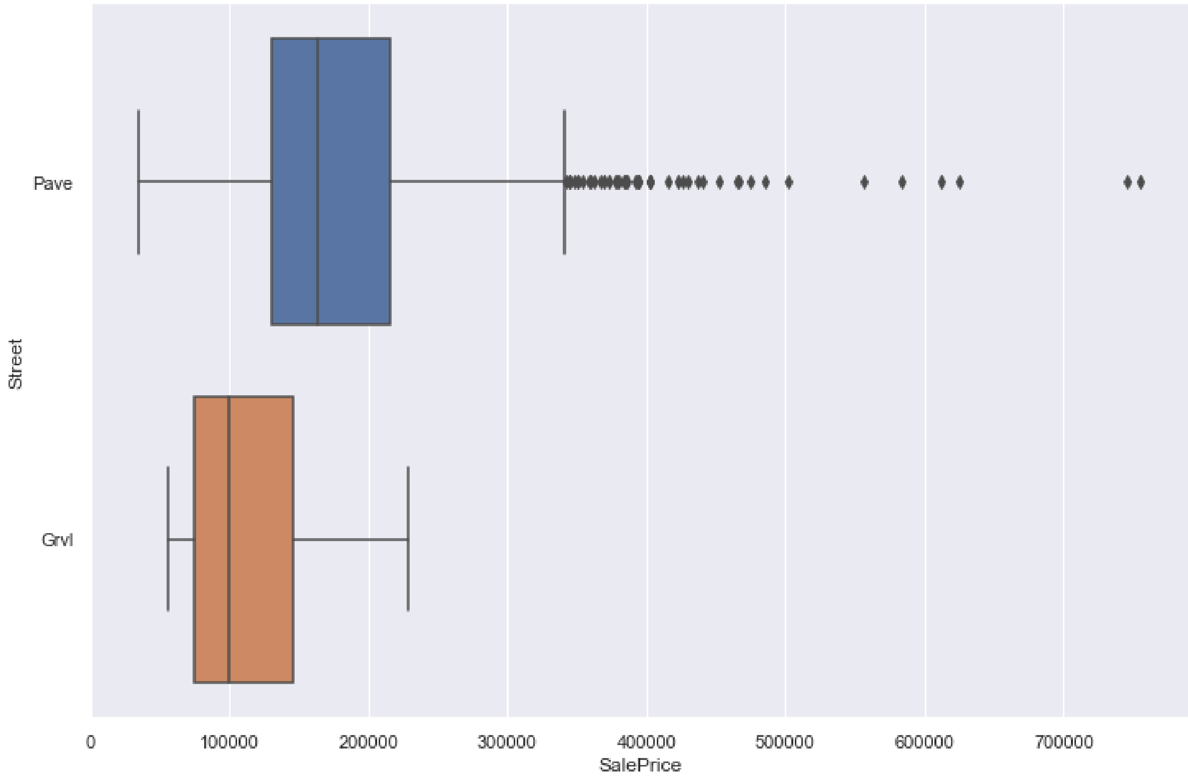

Fig.38(Street vs Sales Price)

The fig.38 is a box plot which shows the relationship between type of street and sales price, from the plot we can say that The Paved type of road access to property is more expensive then Gravel type of roads

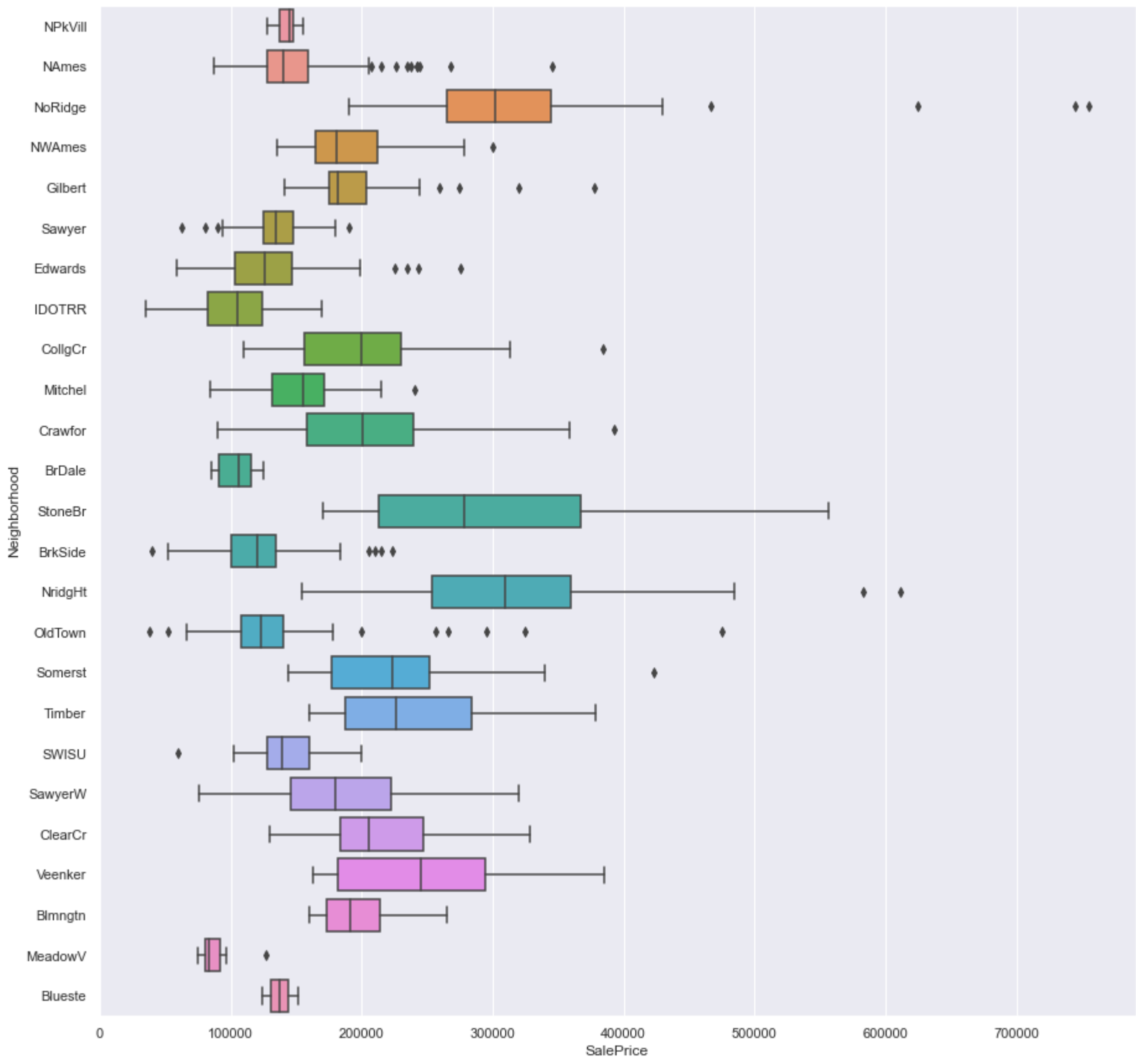

Fig.39(Neighbourhood vs Sales Price)

Neighbourhood (Physical locations within Ames city limits) NridgHt (Northridge Heights) is the most expensive location and will have a heavy impact on house price

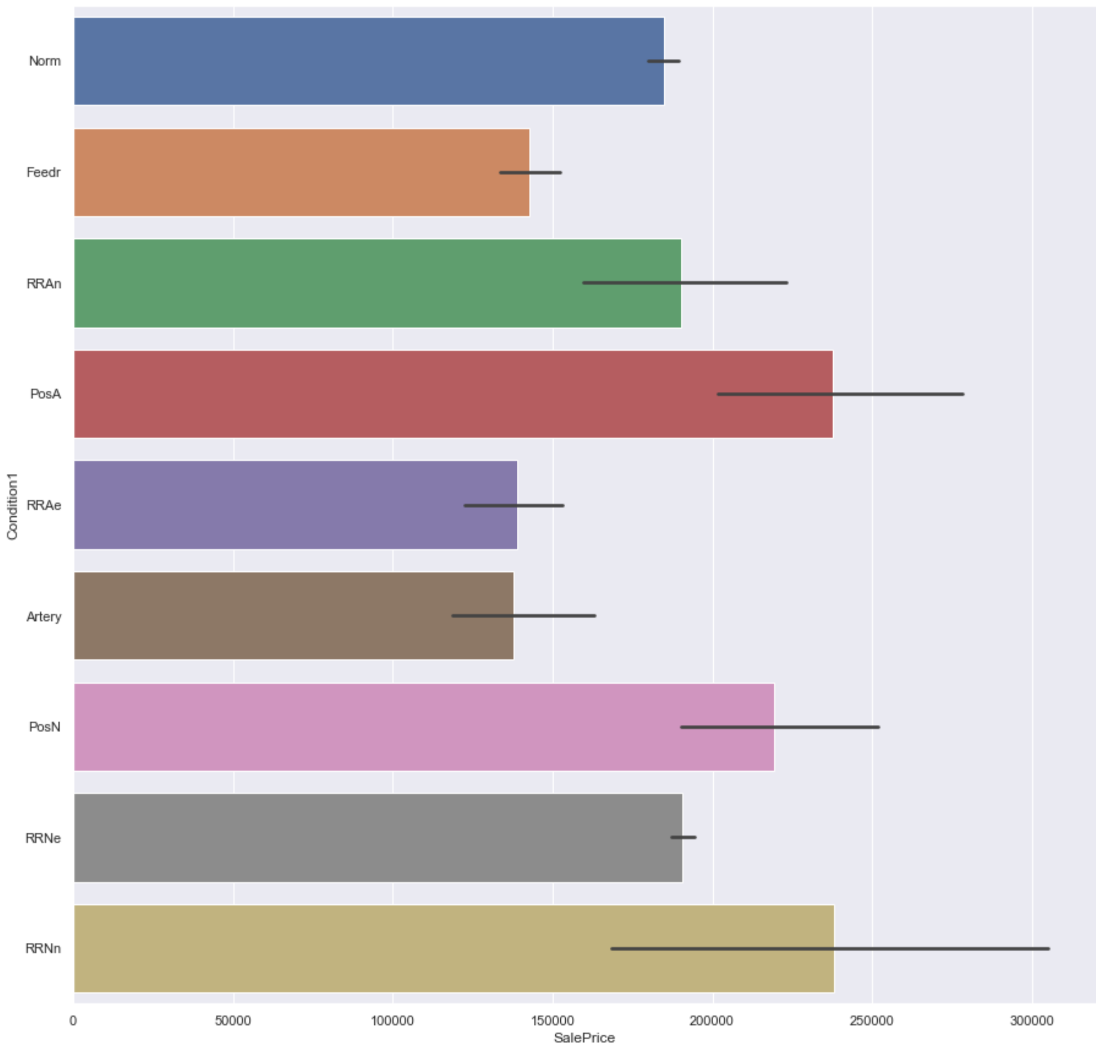

Fig.40(Condition1 vs Sales Price)

Condition1 (Proximity to various conditions) RRNn (Within 200' of North-South Railroad) and PosA (Adjacent to postive off-site feature) are the two conditions which are an expensive choice for an average customer. This conditions will impact the sale price of houses.

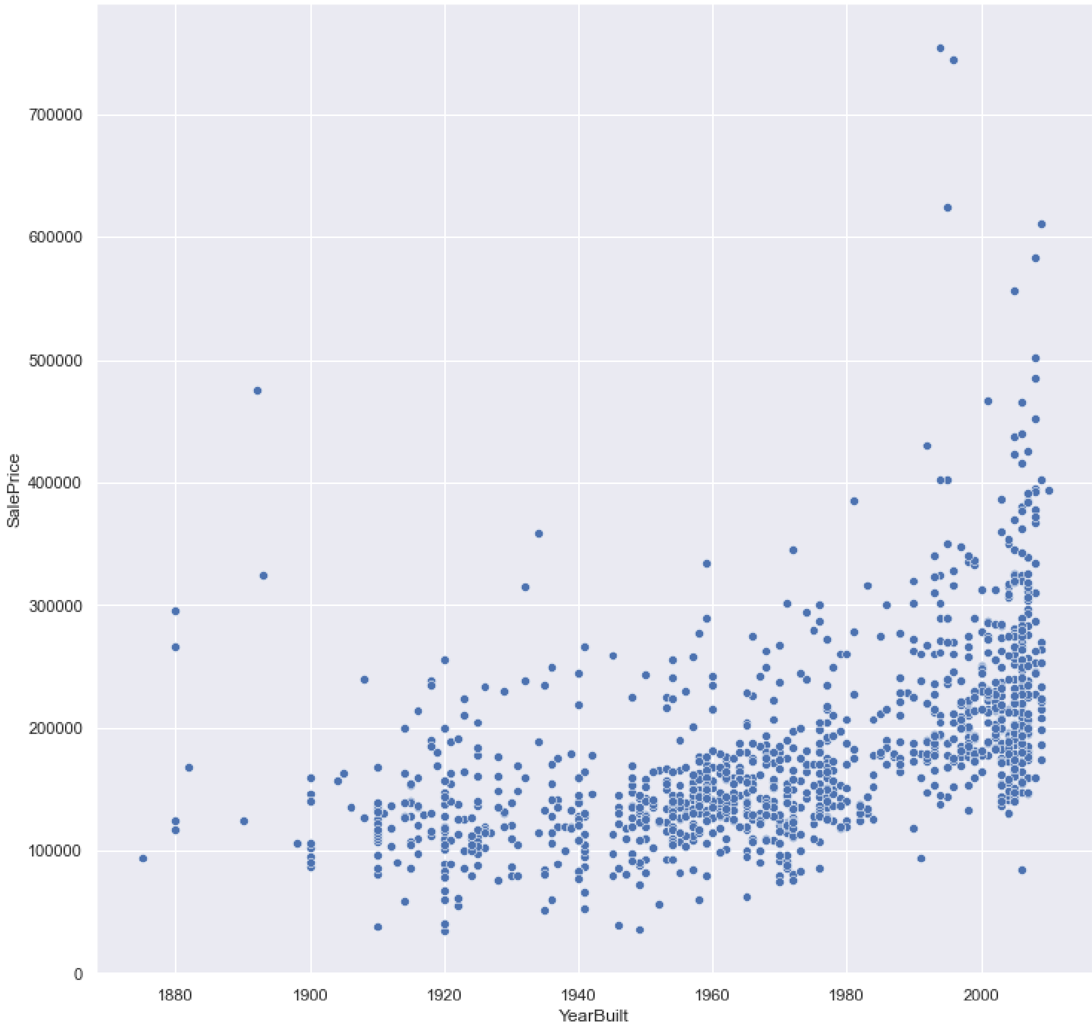

Fig.41(Year Built vs Sales Price)

The house will be more expensive if it was built latterly. More recently built houses will always be more expensive.

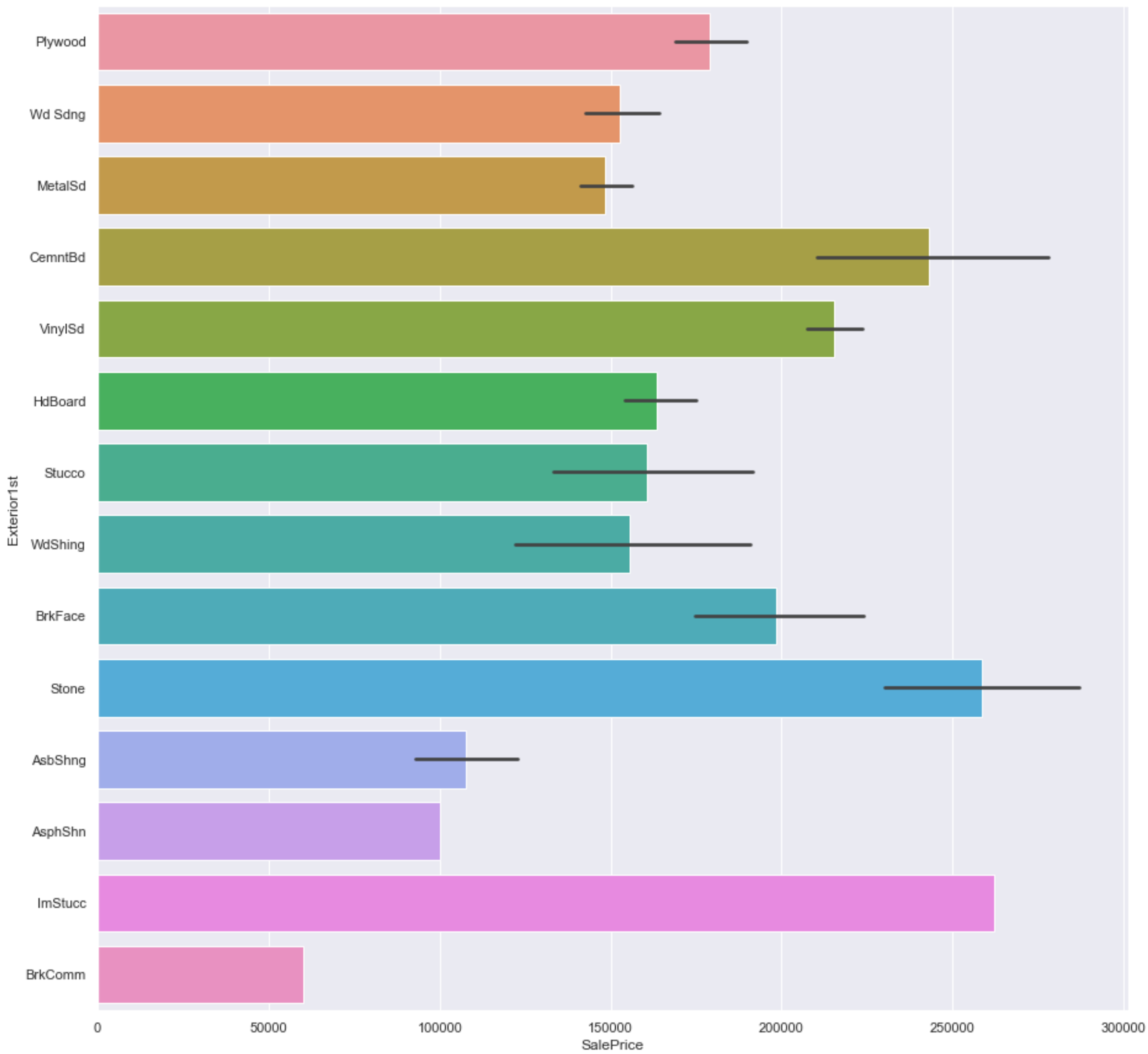

Fig.42(Exterior1st vs Sales Price)

Exterior1st is Exterior covering on house in terms of expenses houses with Exterior covering ImStucc (Imitation Stucco) are very expensive as compared to other Exterior coverings. Brick Common is the cheapest Exterior covering.

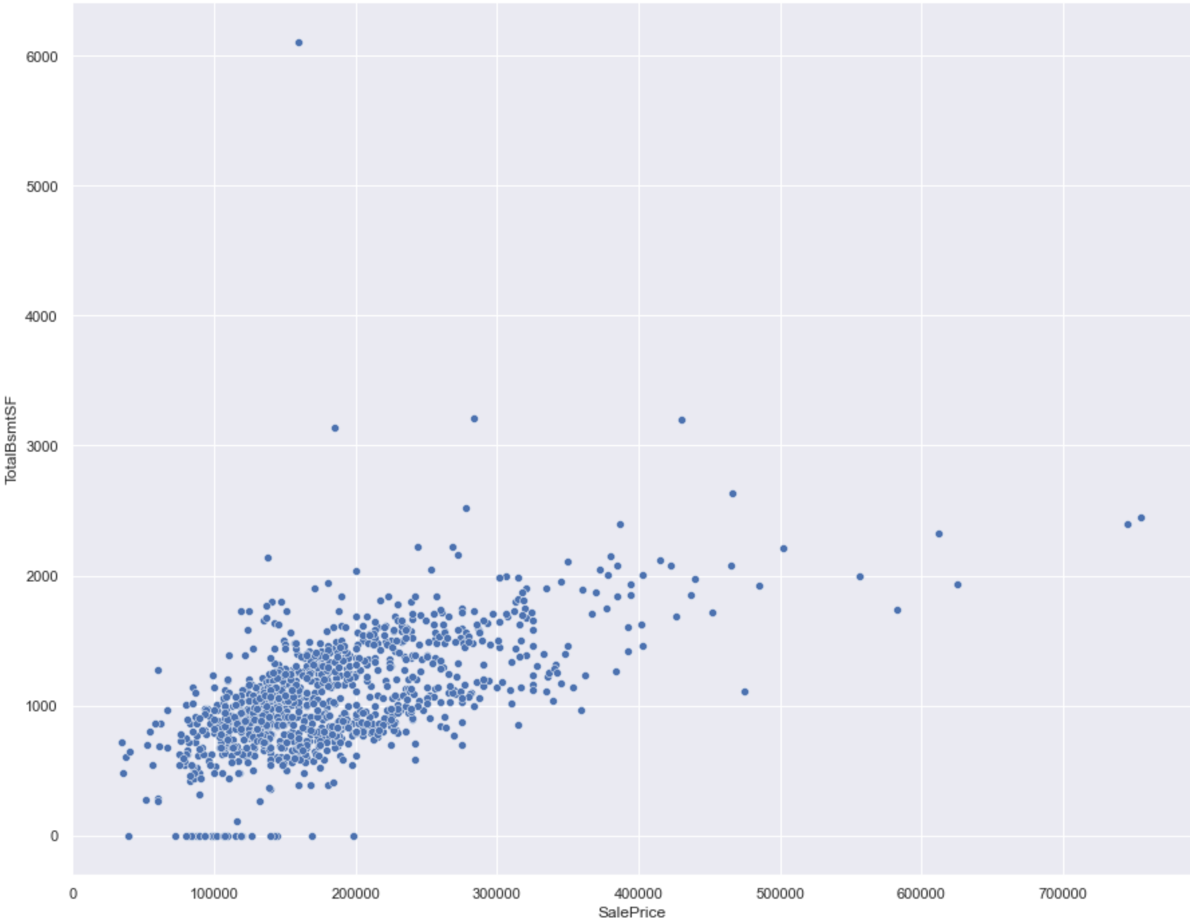

Fig.43(TotalBsmtSF vs Sales Price)

Basements with more area will evidently be more expensive.

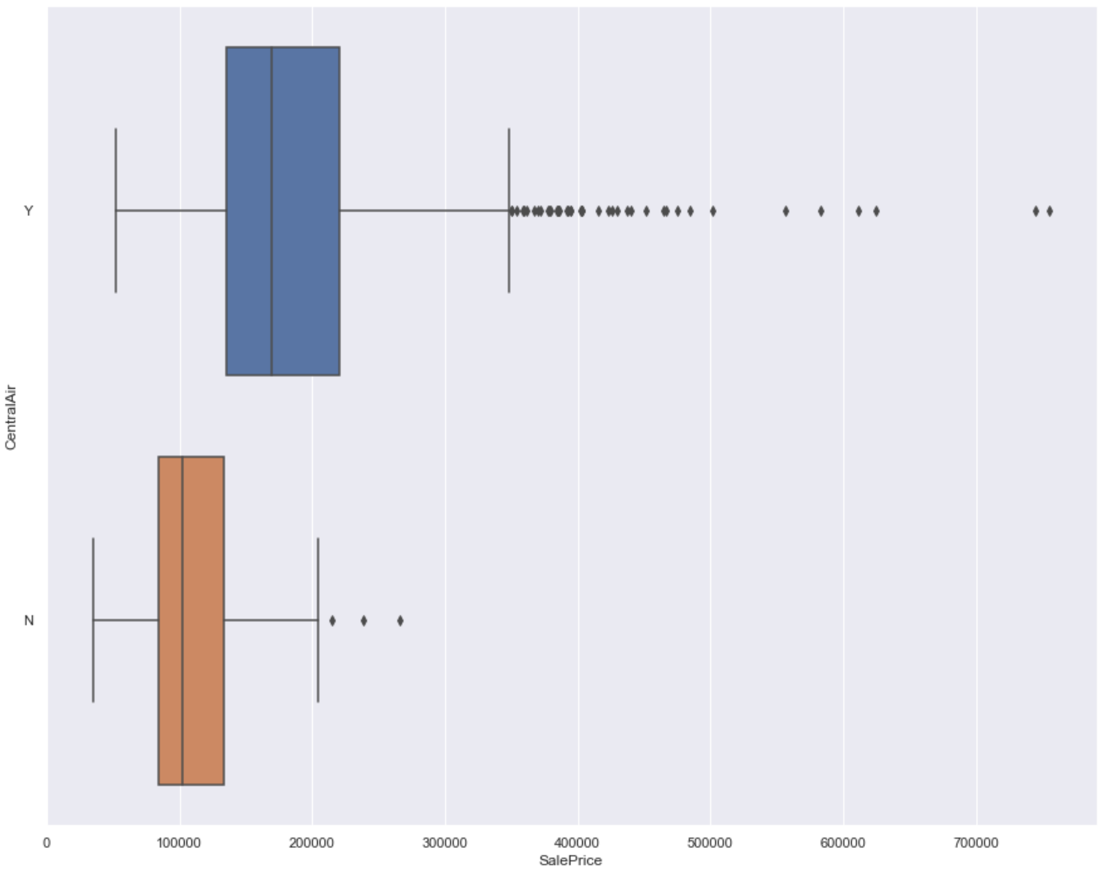

Fig.44(Central Air vs Sales Price)

Houses with Central air conditioning will be more expensive.

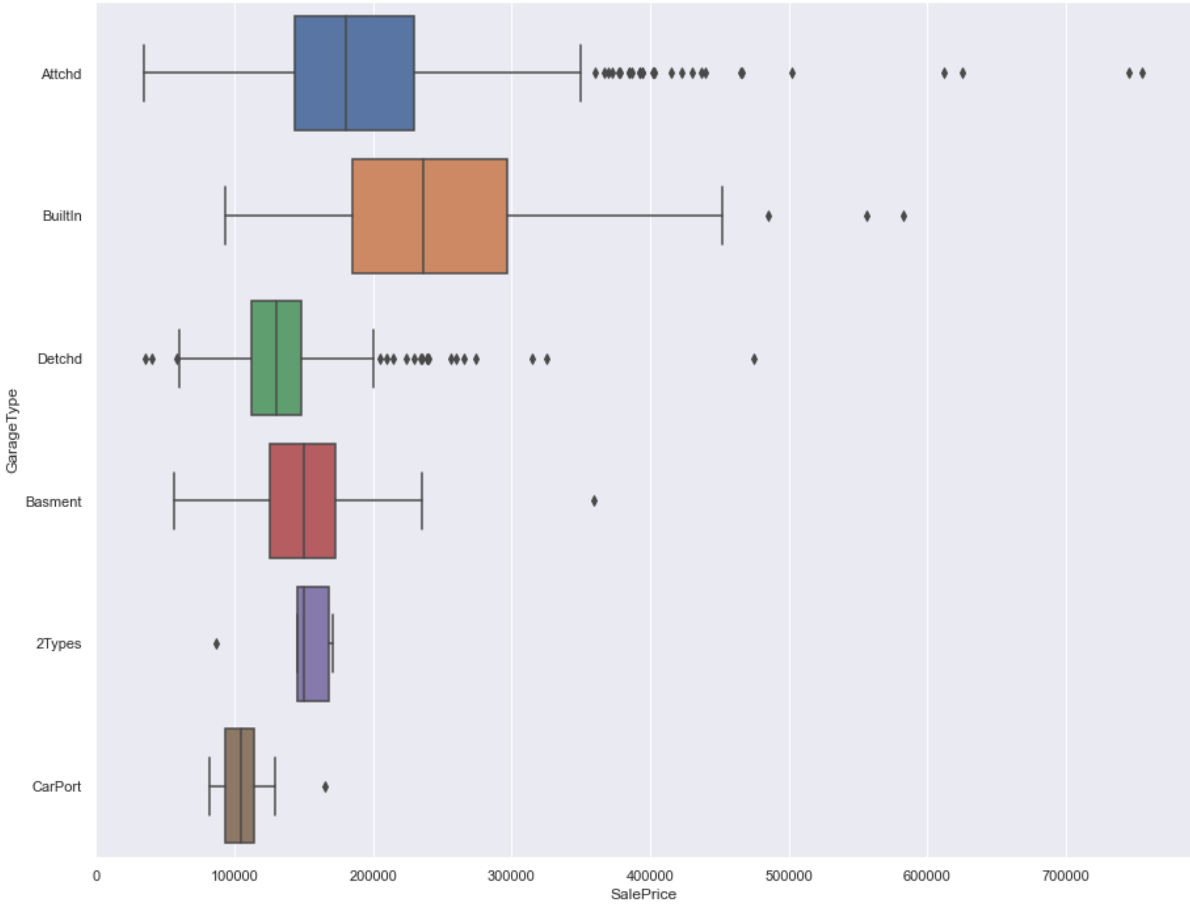

Fig.44(Garage vs Sales Price)

Houses with Built-in Built-In (Garage part of house – typically has room above garage) will be more expensive

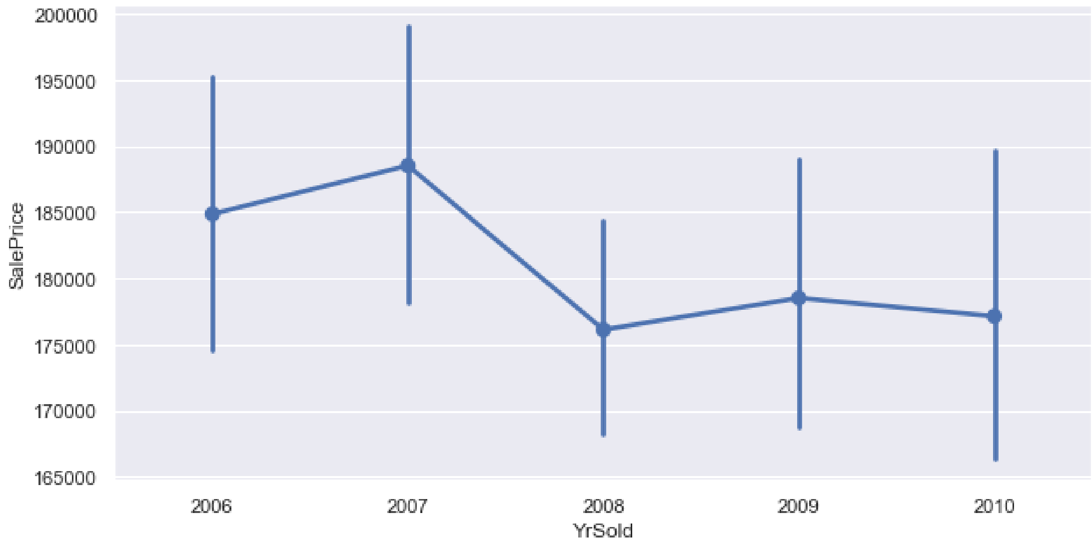

Fig.45(Yrsold vs Sales Price)

In 2007 House sale prices were very high as compared to all other years.

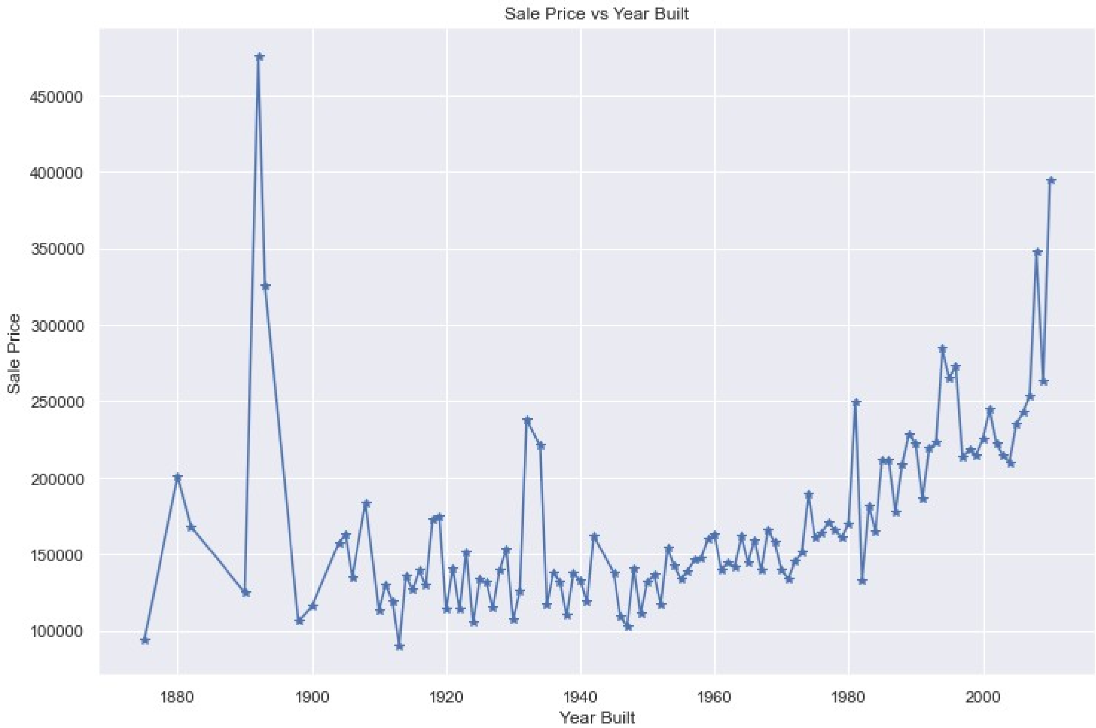

Fig.45(Ybuilt vs Sales Price)

Some of the houses around the year 1900 are very expensive.
The newer the house the more expensive it will be.

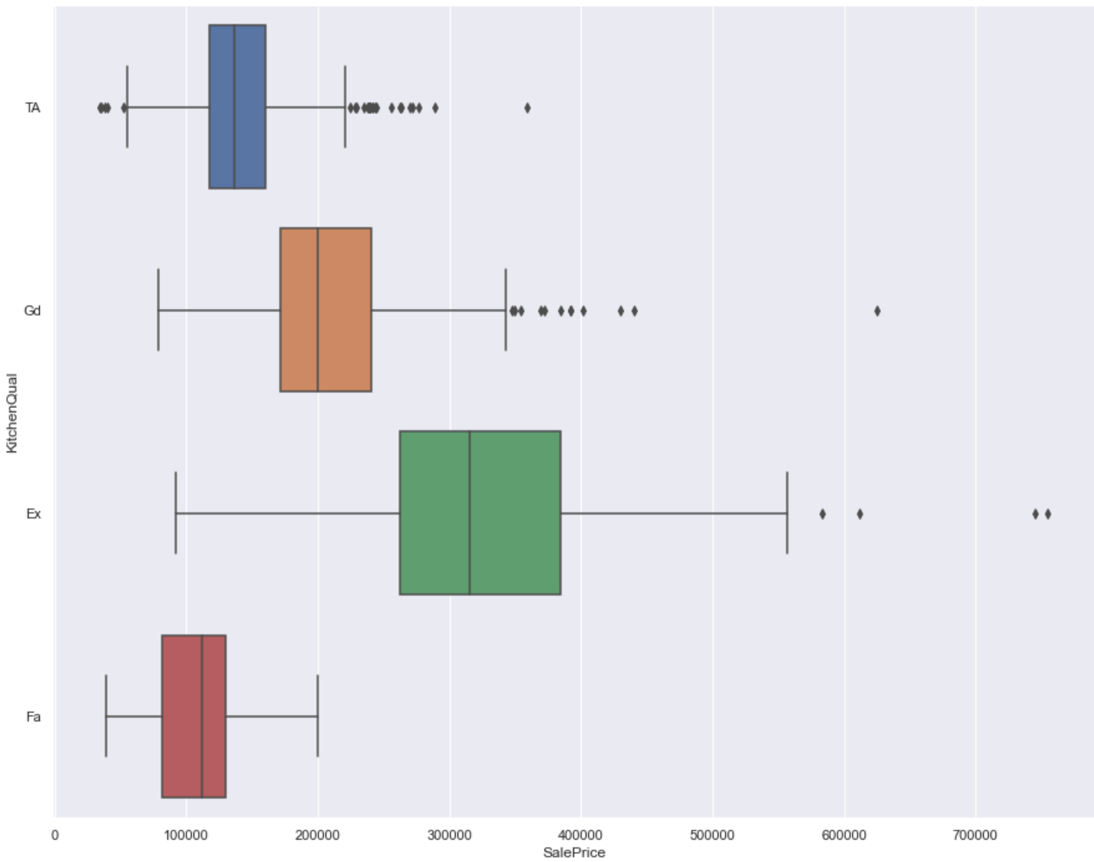

Fig.46(Kitchen Quality vs Sales Price)

Houses with Kitchen Quality (Excellent) will be more expensive.

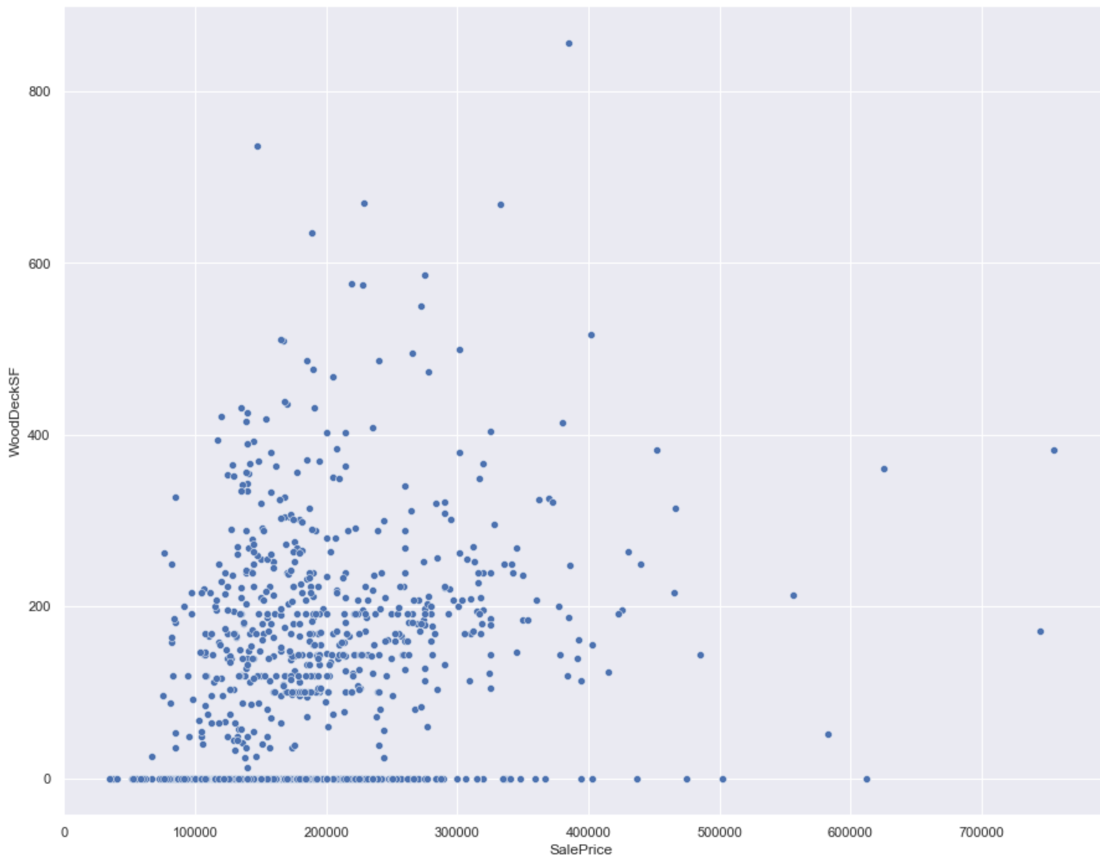

Fig.47(Wood Deck SF vs Sales Price)

We can clearly observe that, with more Wood deck area per square feet the total selling price of the house increases.

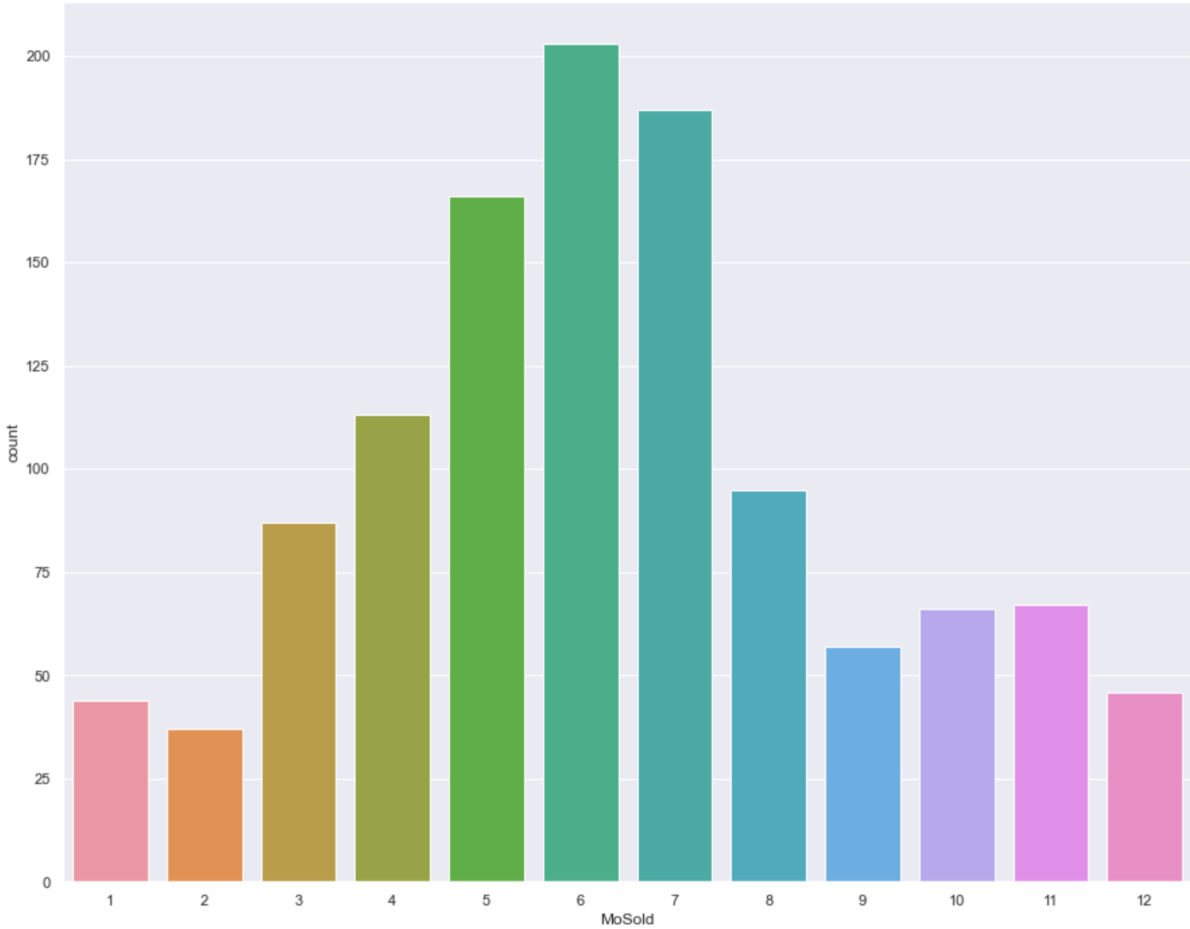

Fig.48(Month Sold)

Highest number of Houses were sold in the month of June and July.

- ## Interpretation of the Results

A total of 8 regression models were used in order to classify the target variable Label. Evaluation Matrix Like Mean Square Error, Mean Absolute Error, Root Mean Square Error, r2 Score were compared to find the optimum model.

	Algorithm	Training_Acc	R2 Score	MSE	MAE	RMSE	Cross_validation
0	Linear Regression	0.860335	0.864096	1.114169e+09	20408.876344	33379.171106	0.833876
1	Random Forest Regression	0.975890	0.904351	5.757140e+08	16380.196524	23994.040896	0.845589
2	Gredient Boosting	0.970815	0.906691	5.276473e+08	15649.906902	125.099588	0.877716
3	ADA Boost	0.863646	0.842873	1.200366e+09	24054.057217	34646.295281	0.790010
4	Bagging Regressor	0.966713	0.872546	7.207351e+08	17248.327920	26846.509088	0.822987
5	XGB Regressor	0.999948	0.880855	6.977221e+08	18108.704639	26414.429195	0.842313
6	Lasso	0.867588	0.864845	7.809582e+08	18546.194638	27945.628619	0.833968
7	Ridge	0.870072	0.851441	8.461204e+08	19654.826807	29088.148670	0.834901

Fig.49(ML model result table)

On The Basis of all the 8 machine learning algorithms and based on the cross validation scores Gradient Boosting Algorithm is giving the best results.

Hyper parametric Tuning:

Using Random search CV for Hyper parametric Tuning of ML models.

Random Forest Regression(Hyper parametric Tuning)

After applying Hyper parametric Tuning for Random forest regression we get the following results:

	Train Score	R2 Score	Adjusted_r2	Explained_variance	MAE	MSE	RMSE
0	0.910144	0.833722	0.774429	0.833817	20365.124852	9.470386e+08	30773.991513

Gradient Boosting Regression(Hyper parametric Tuning)

After applying Hyper parametric Tuning for Gradient Boosting regression we get the following results:

	Train Score	R2 Score	Adjusted_r2	Explained_variance	MAE	MSE	RMSE
0	0.99712	0.881884	0.839765	0.881897	16585.979462	6.727313e+08	25937.064365

ADA Boosting Algorithm (Hyper parametric Tuning)

After applying Hyper parametric Tuning for ADA Boosting regression we get the following results:

	Train Score	R2 Score	Adjusted_r2	Explained_variance	MAE	MSE	RMSE
0	0.910144	0.833722	0.774429	0.833817	20365.124852	9.470386e+08	30773.991513

LASSO Algorithm (Hyper parametric Tuning)

After applying Hyper parametric Tuning for Lasso regression we get the following results:

	Train Score	R2 Score	Adjusted_r2	Explained_variance	MAE	MSE	RMSE
0	0.910144	0.833722	0.774429	0.833817	20365.124852	9.470386e+08	30773.991513

Ridge Algorithm (Hyper parametric Tuning)

After applying Hyper parametric Tuning for Ridge regression we get the following results:

	Train Score	R2 Score	Adjusted_r2	Explained_variance	MAE	MSE	RMSE
0	0.910144	0.833722	0.774429	0.833817	20365.124852	9.470386e+08	30773.991513

CONCLUSION

- ## Key Findings and Conclusions of the Study

  The first step I took, was to visualize the distribution of each feature and its effect on the Sale Price (dependent variable). From the analysis, I conclude that some of the most useful features for predictions were" Overall Quality", "GrLivArea", "TotalBsmtSF", "GarageCar", "Kitchen Quality" and "Year Built". The Gradient Boosting Regression Algorithm proved to be the best model for regression based on the Cross-Validation scores. An accuracy(r2 score) of 0.88 % was achieved by hyper parametric tuning of the model.

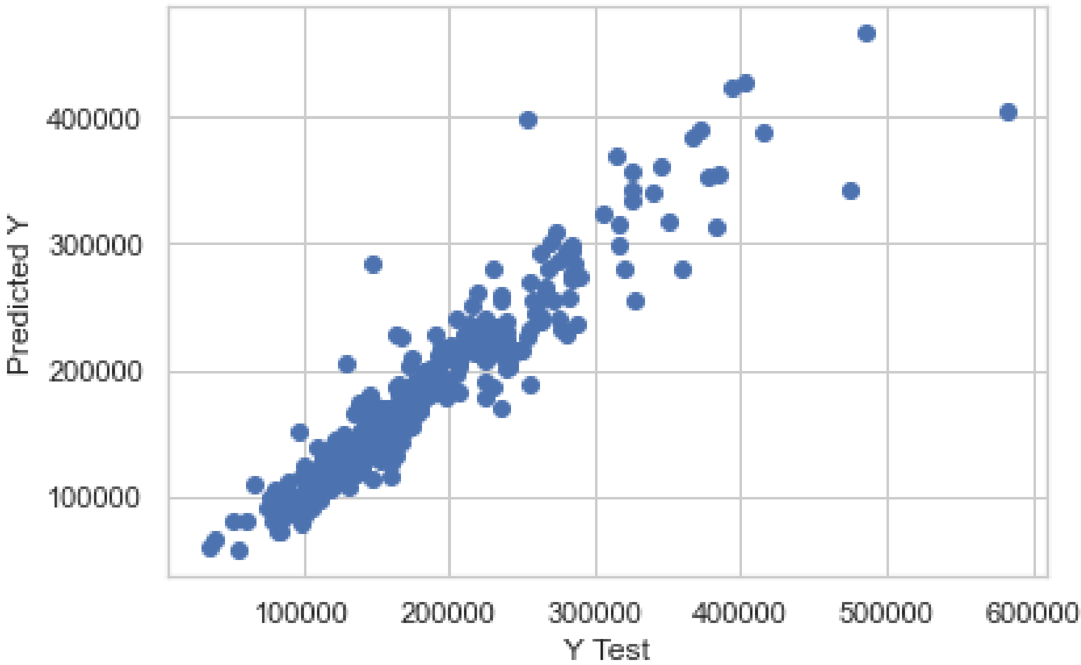

Fig.50(Actual values vs predicted values)

In Fig.50 We can observe that when we plot the predicted values with the actual values we get a graph that looks some what linear in nature

```
# Comparing predicted value vs actual value

print(np.concatenate((predictions_GB.reshape(len(predictions_GB),1), y_test.
```

```
[[189871.67326767 178000.        ]
 [124893.29185136 125000.        ]
 [139335.99692991 142500.        ]
 [143197.45773447 151500.        ]
 [252370.72308969 260000.        ]
 [122470.03929413 110000.        ]
 [355956.01629591 385000.        ]
 [141942.91514021 161500.        ]
 [128280.45163602 116900.        ]
 [209692.03717507 174000.        ]
 [168102.33630338 165000.        ]
 [169043.10319608 159000.        ]
```

Fig.51(Comparing Actual values vs predicted values)

- Learning Outcomes of the Study in respect of Data Science

New Skills Acquired: Learned a new approach to feature engineering (mutual_info_regression) from sklearn's Feature Selection Library

Algorithms Used: Linear regression, Gradient Boosting, Random Forest regression, XGB Regression, Ada Boost, Lasso Regression, Ridge Regression, Bagging regression.

Challenges Faced: The Feature Engineering ,Feature Selection and data pre-processing was quit challenging as there were a lot of non-realistic values in the dataset.

- ## Limitations of this work and Scope for Future Work

The accuracy of the model can be improved further. Was not able to reduce skewness for some features.