FLIP ROBO

Housing Project

Submitted by:

Dipesh Ramesh Limaje

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 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. You are 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.

Conceptual Background of the Domain Problem

The domain problem for housing projects is a complex one and involves a range of factors. The most fundamental issue is the lack of affordable housing in many cities and regions, especially those with high demand for housing due to population growth. This is compounded by the fact that the cost of construction often outpaces inflation, making it difficult for low-income households to find affordable housing in desirable areas. Additionally, local zoning regulations, building codes, and other restrictions can further limit the available options.

In order to address this issue, housing projects must focus on creating model that predict house sale price. The final step in the domain problem is predicting the sale price of the housing project. This is a complex task that involves analyzing a variety of factors, such as the location, the quality of the construction, the amenities offered, and the condition of the property. Additionally, the sale price can be affected by the market conditions, the availability of comparable properties, and the overall demand for housing in the area. By understanding these factors and using predictive analytics, it is possible to determine the likely sale price of a housing project.

Review of Literature

The literature review of housing projects and predict sale prices can be divided into two parts.

The first part discusses the factors that affect housing project sale prices. These factors include macroeconomic factors such as GDP growth rate, inflation rate, and population growth rate, and microeconomic factors such as location, availability of public amenities, and proximity to downtown. Other factors include environmental factors such as natural disasters, zoning regulations, and the availability of public transport. Additionally, the impact of housing projects on the local economy should also be taken into account.

The second part of the literature review focuses on the methods used to predict housing project sale prices. These methods include linear regression, multiple regression, artificial neural networks, and support vector machines. Each of these methods has its own advantages and disadvantages, and a combination of them is usually used to achieve the best results. Additionally, other forecasting techniques such as time series analysis and Monte Carlo simulations can also be used.

The literature review on housing projects and predict sale prices should also include a discussion of the challenges that exist in the field. These include the difficulty in obtaining accurate data, the lack of reliable models, and the potential for bias in the data. Additionally, the lack of standardization in the data as well.

Motivation for the Problem Undertaken

The motivation for this problem is to provide a comprehensive and accurate prediction of housing prices in order to help real estate agents, buyers, and sellers make better decisions. By accurately predicting house prices, buyers and sellers can make more informed decisions about the value of a property, and real estate agents can use the results to better inform their clients. Additionally, this information can be used to inform policy decisions and to help local governments understand the value of their local housing markets.

Mathematical/ Analytical Modeling of the Problem

The mathematical/analytical modelling of the problem for housing project and prediction of sale price can be done using multiple linear regression. Multiple linear regression is a type of linear regression that is used to model the relationship between two or more independent variables and one dependent variable. The independent variables are the factors that are used to predict the sale price, such as location, size, age, and condition of the property, while the dependent variable is the sale price of the property.

The multiple linear regression model can be formulated as follows:

Sale Price = $\beta 0 + \beta 1$*Location + $\beta 2$*Size + $\beta 3$*Age + $\beta 4$*Condition + ε

where $\beta 0$, $\beta 1$, $\beta 2$, $\beta 3$, and $\beta 4$ are the regression coefficients, and ε is the error term.

The regression coefficients can be estimated using least squares estimation and the model can be tested for its goodness of fit using the R-squared statistic. The R-squared statistic indicates the proportion of variability in the dependent variable that is explained by the independent variables. A higher R-squared value indicates a better fit of the model.

Once the model is fitted, it can be used to predict the sale price of a house.

Data Sources and their formats

The data sources and formats for the housing project and predicting sale price would include: Property data, such as square footage, lot size, Location data ,parks, public transportation, retail stores, Economic data, Historical sales data, such as prices of similar homes in the area that have sold in the past , This data would typically be in a structured format, such as CSV or PDF

Data Description

MSSubClass: 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

MSZoning: 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

LotFrontage: Linear feet of street connected to property

LotArea: 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

LotShape: General shape of property

Reg	Regular
IR1	Slightly irregular
IR2	Moderately Irregular
IR3	Irregular

LandContour: 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

<u>Overall Cond: Rates the overall condition of the house</u>

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

<u>Year Built: Original construction date</u>

<u>YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)</u>

<u>RoofStyle: Type of roof</u>

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
WdShake	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

Fa Fair

Po Poor

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

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

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

Ex Excellent (100+ inches)

Gd Good (90-99 inches)

TA Typical (80-89 inches)

Fa Fair (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

Fa Fair - 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 score average or above)

Mn Mimimum 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

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
Fa	Fair
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)

Data Pre-processing Done

Pre-processing

```
In [3]: #checking the null values
        df.isnull().sum()

Out[3]: Id                 0
        MSSubClass         0
        MSZoning           0
        LotFrontage      214
        LotArea            0
                        ...
        MoSold             0
        YrSold             0
        SaleType           0
        SaleCondition      0
        SalePrice          0
        Length: 81, dtype: int64
```

```
In [4]: # checking the shape of dataset
        df.shape

Out[4]: (1168, 81)
```

```
In [4]: # checking columns availabe in the dataset
        df.columns

Out[4]: Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
               'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
               'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
               'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
               'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
               'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
               'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
               'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
               'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
               'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
               'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
               'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
               'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
               'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
               'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
               'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
               'SaleCondition', 'SalePrice'],
              dtype='object')
```

Pre-processing I have done

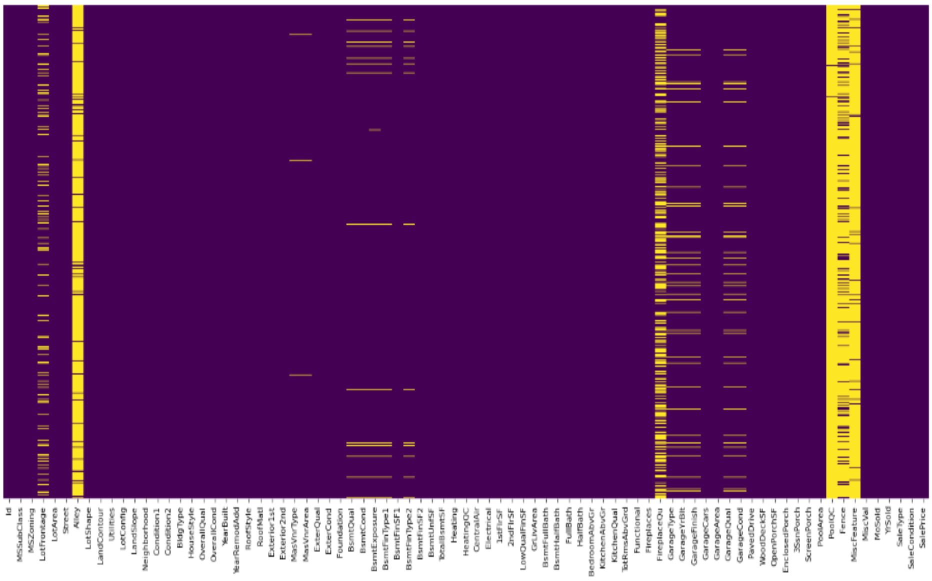
1] Check for the Null values

2] Check the Shape of the dataset

3] Check all the columns Names and Compared them with the Data Description given to us.

```
dtype='object')
```

In [5]:
```
# checking all the nulls using the heatmap to get clear view
plt.figure(figsize=(25,10))
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
plt.show()
```



Dropping columns which has more than 50% of null values

In [6]:
```
df.drop(['Alley','PoolQC'],axis=1,inplace=True)
```

In [7]:
```
df.drop(['MiscFeature'],axis=1,inplace=True)
```

In [8]:
```
df.drop(['MiscVal'],axis=1,inplace=True)
```

In [9]:
```
df.drop(['MoSold'],axis=1,inplace=True)
```

4] I cannot see the Null values by df.isnull().sum() so I Plotted Heatmap to observe the null values.

5] I dropped the unwanted Columns and The columns that have null values more than 50%

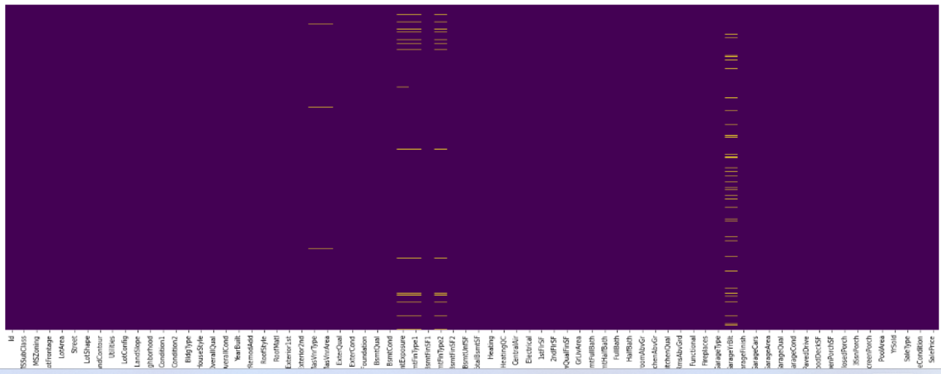
```
In [10]: # checking datatypes and remaining null values
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 76 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1168 non-null   int64
 1   MSSubClass     1168 non-null   int64
 2   MSZoning       1168 non-null   object
 3   LotFrontage    954 non-null    float64
 4   LotArea        1168 non-null   int64
 5   Street         1168 non-null   object
 6   LotShape       1168 non-null   object
 7   LandContour    1168 non-null   object
 8   Utilities      1168 non-null   object
 9   LotConfig      1168 non-null   object
 10  LandSlope      1168 non-null   object
 11  Neighborhood   1168 non-null   object
 12  Condition1     1168 non-null   object
 13  Condition2     1168 non-null   object
 14  BldgType       1168 non-null   object
 15  HouseStyle     1168 non-null   object
 16  OverallQual    1168 non-null   int64
 17  OverallCond    1168 non-null   int64
 18  YearBuilt      1168 non-null   int64
 19  YearRemodAdd   1168 non-null   int64
 20  RoofStyle      1168 non-null   object
 21  RoofMatl       1168 non-null   object
 22  Exterior1st    1168 non-null   object
 23  Exterior2nd    1168 non-null   object
 24  MasVnrType     1161 non-null   object
 25  MasVnrArea     1161 non-null   float64
 26  ExterQual      1168 non-null   object
 27  ExterCond      1168 non-null   object
 28  Foundation     1168 non-null   object
 29  BsmtQual       1138 non-null   object
 30  BsmtCond       1138 non-null   object
 31  BsmtExposure   1137 non-null   object
 32  BsmtFinType1   1138 non-null   object
 33  BsmtFinSF1     1168 non-null   int64
 34  BsmtFinType2   1137 non-null   object
 35  BsmtFinSF2     1168 non-null   int64
 36  BsmtUnfSF      1168 non-null   int64
 37  TotalBsmtSF    1168 non-null   int64
```

6] Then I have checked the Datatypes and the remaining null values in which columns the null values are present.

In [13]:
```python
## Fill Missing Values one by one by mean and mode

df['LotFrontage']=df['LotFrontage'].fillna(df['LotFrontage'].mean())

df['BsmtCond']=df['BsmtCond'].fillna(df['BsmtCond'].mode()[0])
df['BsmtQual']=df['BsmtQual'].fillna(df['BsmtQual'].mode()[0])


#df['FireplaceQu']=df['FireplaceQu'].fillna(df['FireplaceQu'].mode()[0])
df['GarageType']=df['GarageType'].fillna(df['GarageType'].mode()[0])


df['GarageFinish']=df['GarageFinish'].fillna(df['GarageFinish'].mode()[0])
df['GarageQual']=df['GarageQual'].fillna(df['GarageQual'].mode()[0])
df['GarageCond']=df['GarageCond'].fillna(df['GarageCond'].mode()[0])
```

In [14]:
```python
# again observing any null values
plt.figure(figsize=(25,10))
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
plt.show()
```



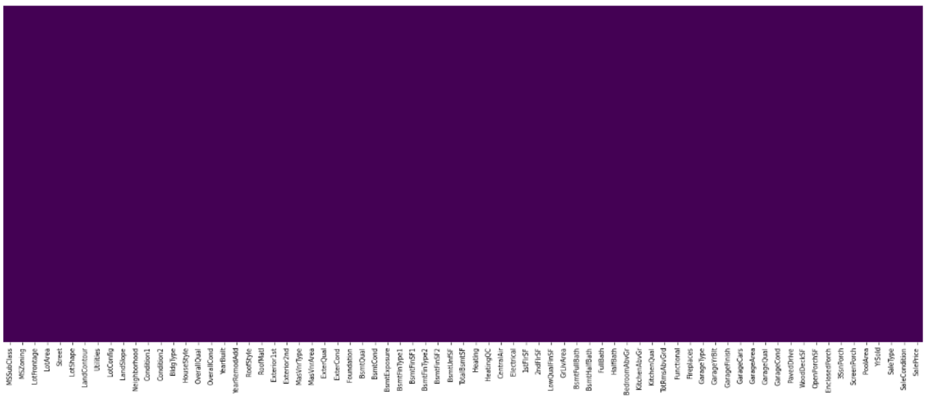
7] Then I have filled the null values with Mean and Mode method, for continuous data o have filled with the Mean Method and for Categorical Columns I have filled it with the Mode Method

8] After filling the Null values then again I check for the null values if remaining

In [21]:
```python
# the final shape of the dataset
df.shape
```

Out[21]: (1080, 73)

In [22]:
```python
# again observing any null values
plt.figure(figsize=(25,10))
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
plt.show()
```



9] then at last I have dropped all the Null values using df.dropna and again check for null values if remaining.

```
In [24]: # again observing all datatypes and null values
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1080 entries, 0 to 1167
Data columns (total 73 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   MSSubClass      1080 non-null   int64
 1   MSZoning        1080 non-null   object
 2   LotFrontage     1080 non-null   float64
 3   LotArea         1080 non-null   int64
 4   Street          1080 non-null   object
 5   LotShape        1080 non-null   object
 6   LandContour     1080 non-null   object
 7   Utilities       1080 non-null   object
 8   LotConfig       1080 non-null   object
 9   LandSlope       1080 non-null   object
 10  Neighborhood    1080 non-null   object
 11  Condition1      1080 non-null   object
 12  Condition2      1080 non-null   object
 13  BldgType        1080 non-null   object
 14  HouseStyle      1080 non-null   object
 15  OverallQual     1080 non-null   int64
 16  OverallCond     1080 non-null   int64
 17  YearBuilt       1080 non-null   int64
 18  YearRemodAdd    1080 non-null   int64
 19  RoofStyle       1080 non-null   object
 20  RoofMatl        1080 non-null   object
 21  Exterior1st     1080 non-null   object
 22  Exterior2nd     1080 non-null   object
 23  MasVnrType      1080 non-null   object
 24  MasVnrArea      1080 non-null   float64
 25  ExterQual       1080 non-null   object
 26  ExterCond       1080 non-null   object
 27  Foundation      1080 non-null   object
 28  BsmtQual        1080 non-null   object
 29  BsmtCond        1080 non-null   object
 30  BsmtExposure    1080 non-null   object
 31  BsmtFinType1    1080 non-null   object
 32  BsmtFinSF1      1080 non-null   int64
 33  BsmtFinType2    1080 non-null   object
 34  BsmtFinSF2      1080 non-null   int64
 35  BsmtUnfSF       1080 non-null   int64
 36  TotalBsmtSF     1080 non-null   int64
```

10] After treating all null values and dropping all unwanted columns finally our dataset final shape is 1080 Rows and 72 Columns.

```
In [25]: df.describe()
```

Out[25]:

F1	BsmtFinSF2	...	GarageCars	GarageArea	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	YrSold	SalePrice
00	1080.000000	...	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000
30	49.836111	...	1.883333	506.025926	100.656481	47.887037	22.212963	3.769444	16.277778	3.729630	2007.796296	187674.507407
52	168.953092	...	0.630768	187.114490	128.066825	65.228622	62.750687	29.753351	57.108231	46.680654	1.326701	78574.655649
00	0.000000	...	1.000000	160.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2006.000000	35311.000000
00	0.000000	...	1.000000	388.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2007.000000	135000.000000
00	0.000000	...	2.000000	485.000000	24.000000	28.000000	0.000000	0.000000	0.000000	0.000000	2008.000000	170000.000000
00	0.000000	...	2.000000	588.000000	180.000000	72.000000	0.000000	0.000000	0.000000	0.000000	2009.000000	222000.000000
00	1474.000000	...	4.000000	1418.000000	857.000000	547.000000	552.000000	508.000000	480.000000	738.000000	2010.000000	755000.000000

Observations

1] There are 1168 rows and and 81 Columns but after treating nulls values and dropping unwanted columns the dataset now has 1080 rows and 72 columns

2] there are no duplicates in the dataset

3] the standrd deviation and max value is high inLotFrontage and LotArea columns

4] the standrd deviation is very high and crossing mean value in MasVnrArea column and Max value is also too high as compared to 75% quantile

5] the standrd deviation is very high and crossing mean value in BsmtFinSF1 and BsmtFinSF2 column and Max value is also too high as compared to 75% quantile

6] The GarageArea column look fine to me.

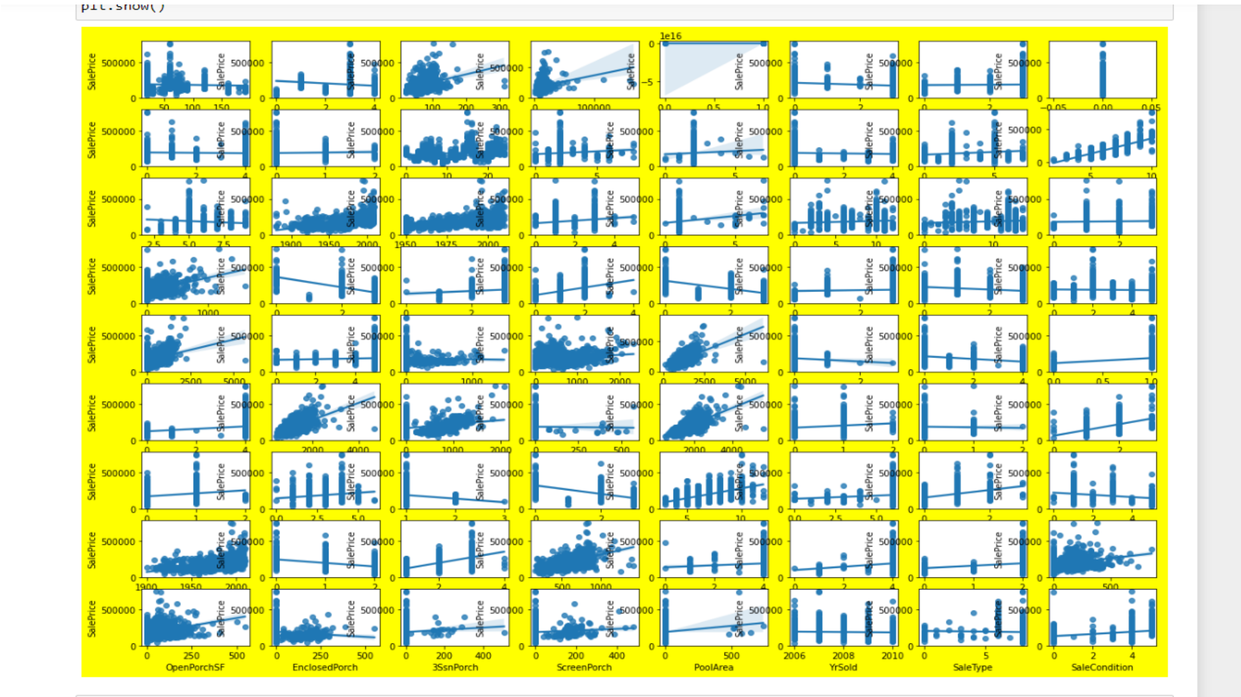
7] the standrd deviation is very high and crossing mean value in WoodDeckSF, OpenPorchSF column and Max value is also too high as compared to 75% quantile

8] I observe some problem in EnclosedPorch and ScreenPorch columns as all the quantile has zero value but max is too high.

9] the standrd deviation and max value is high SalePrice as it is our target varaible.

11] After that I have Describe the dataset to observe the numerical values and written the Observations.

Data Inputs- Logic- Output Relationships



To observe the relationship between Feature and label so I created this Regression plot to observe which features are positively co-related and which features are negatively co-related.

State the set of assumptions (if any) related to the problem under consideration

1] After Filling in all the null values by mean and mode, I dropped some nan values which I fill it is unnecessary, and Dropped some features that have missing values of more than 50%.

2] For this particular problem I have assumed that the Maximum VIF should be 10, if any of the features has a VIF which is greater than 10 we should drop that feature.

Hardware and Software Requirements and Tools Used

- **Hardware Requirements**: -Computer with minimum 8 GB RAM -High-speed internet connection -High-end graphics card -External storage device

- **Software Requirements**: -Python programming language -TensorFlow -Keras -Scikit-Learn -Pandas -Matplotlib -Seaborn

- **Tools Used**: -Jupyter Notebook -Google Colab -Tableau -Power BI

- **Predicting the sale price**: -Linear regression -Random Forest -GBoost -ADA Boost

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

Statistical Approach:

1. **Exploratory Data Analysis (EDA):** This is an important step to gain insights into the data, identify the patterns and relationships between different variables. We can look at the distribution, correlation and pattern of different variables.

2. **Feature Selection**: This is an important step to identify the important features and select the best variables that contribute to the prediction of sale prices. We can use a variety of methods such as correlation, chi-squared test, step-wise regression, etc.

3**. Model Building**: We can use different supervised learning models such as linear regression, decision tree, random forest, etc. to build a model that can accurately predict the sale prices.

Analytical Approach:

1. **Domain Knowledge**: We can use our domain knowledge to identify the important features that are most likely to influence the sale price.

2. **Data Visualization**: This is an important step to gain insights into the data, identify the patterns and relationships between different variables. We can use various visualization techniques such as bar plots, box plots, scatter plots, etc. to understand the data better.

3. **Hypothesis Testing**: We can use hypothesis testing to identify the features that are most likely to influence the sale prices. This can be done by conducting a series of hypothesis tests to test the significance of each feature.

Testing of Identified Approaches (Algorithms)

- LR (Linear Regression Model)
- GBDT (Gradient Boosting Regressor Model)
- RF (Random Forest Regressor Model)
- ADA (AdaBoost Regressor Model)

Run and Evaluate selected models

1st Model I have Created is the Logistic Regression Model

Linear Regression Model

In [100]:
```python
#lets import necessary library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

import warnings
warnings.filterwarnings('ignore')
```

Finding Best Random State

In [101]:
```python
#Best Random State
MaxAccu=0
MaxRS=0

for i in range (0,200):
    X_train,X_test,y_train,y_test=train_test_split(X_scalar,y,test_size=0.25,random_state=i)
    regression=LinearRegression()
    regression.fit(X_train,y_train)

    pred=regression.predict(X_train)
    training=regression.score(X_train,y_train)
    print ('Training Score' , training*100 , 'RandomState' ,i)

    y_pred=regression.predict(X_test)
    testing=regression.score(X_test,y_test)
```

```
Testing Score 74.12810274352803 RandomState 48
```

In [102]:
```python
print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , MaxRS)
```

```
MAXINING TESTING SCORE 86.9044746133699 ON RANDOM STATE OF 163
```

Training the model

In [103]:
```python
#splliting our data into train test split and randomstate 6
X_train,X_test,y_train,y_test=train_test_split(X_scalar,y,test_size=0.25,random_state=163)
```

In [104]:
```python
#Training the data on Linear Regression Model
regression=LinearRegression()
regression.fit(X_train,y_train)
```

Out[104]:
```
▾ LinearRegression
LinearRegression()
```

In [105]:
```python
#training score
regression.score(X_train,y_train)
```

Out[105]:
```
0.754021475973685
```

In [106]:
```python
#testing score
regression.score(X_test,y_test)
```

Out[106]:
```
0.869044746133699
```

Model Score

- Training Score = 75.4021475973685 %
- Testing Score = 86.9044746133699 %

LR (Linear Regression Model) Score are

Training score = 75.4021475973685 %

Testing Score = 86.9044746133699 %

Cross-Validation for Linear Regression Model

In [126]:
```python
#Cross Vaildation
training=regression.score(X_train,y_train)
testing=regression.score(X_test,y_test)

from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score=cross_val_score(regression,X,y,cv=j)
    cv_mean=cv_score.mean()
    print(f'At cross fold {j} the cv score is {cv_mean} and the R2 score for Training is {training} and R2 score for the Testing
    print('\n')
```

```
At cross fold 2 the cv score is 0.6932978841713491 and the R2 score for Training is 0.754021475973685 and R2 score for the Test
ing is0.869044746133699


At cross fold 3 the cv score is 0.7135903848972124 and the R2 score for Training is 0.754021475973685 and R2 score for the Test
ing is0.869044746133699


At cross fold 4 the cv score is 0.741229516881684 and the R2 score for Training is 0.754021475973685 and R2 score for the Testi
ng is0.869044746133699


At cross fold 5 the cv score is 0.7196511390369636 and the R2 score for Training is 0.754021475973685 and R2 score for the Test
ing is0.869044746133699


At cross fold 6 the cv score is 0.7358854794204976 and the R2 score for Training is 0.754021475973685 and R2 score for the Test
ing is0.869044746133699


At cross fold 7 the cv score is 0.6897383113480949 and the R2 score for Training is 0.754021475973685 and R2 score for the Test
ing is0.869044746133699


At cross fold 8 the cv score is 0.7404151830950261 and the R2 score for Training is 0.754021475973685 and R2 score for the Test
ing is0.869044746133699


At cross fold 9 the cv score is 0.6921292606617704 and the R2 score for Training is 0.754021475973685 and R2 score for the Test
ing is0.869044746133699
```

Cross Validation score

```
Cross-Validation Score at cv = 4 = 74.1229516881684 %
Training score = 75.4021475973685 %
Testing Score = 86.9044746133699 %
```

Linear Regression Model Cross-Validation Score

Cross-Validation Score at cv = 4 = 74.1229516881684 %

Training score = 75.4021475973685 %

Testing Score = 86.9044746133699 %

2nd Model I have Created is Ada Boost Regressor Model

AdaBoostRegressor Model

In [129]:
```
# IMPORT LIBRARY
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostRegressor
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

Finding the Best Random State

In [130]:
```
#Best Random State
MaxAccu=0
MaxRS=0

for i in range (0,200):
    X_train,X_test,y_train,y_test=train_test_split(X_scalar,y,test_size=0.25,random_state=i)
    ada=AdaBoostRegressor()
    ada.fit(X_train,y_train)

    pred=ada.predict(X_train)
    training=ada.score(X_train,y_train)
    print ('Training Score' , training*100 , 'RandomState' ,i)

    y_pred=ada.predict(X_test)
    testing=ada.score(X_test,y_test)
    print ('Testing Score' , testing*100 , 'RandomState' ,i)
    print('\n')


    if testing>MaxAccu:
        MaxAccu=testing
        MaxRS=i
        print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , i)
```

```
Training Score 85.01581660276176 RandomState 0
Testing Score 83.75138982248296 RandomState 0
```

```
Testing Score 73.07485390248475 RandomState 4
```

In [131]:
```
print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , MaxRS)
```

```
MAXINING TESTING SCORE 85.80302796152235 ON RANDOM STATE OF 157
```

Training the model

In [132]:
```
#splliting our data into train test split and randomstate 8
X_train,X_test,y_train,y_test=train_test_split(X_scalar,y,test_size=0.25,random_state=157)
```

In [133]:
```
# adaboost inilize
from sklearn.ensemble import AdaBoostRegressor
ada=AdaBoostRegressor()
ada.fit(X_train,y_train)
```

Out[133]:
```
AdaBoostRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [134]:
```
 # model prediction on training dataset
y_pred = ada.predict(X_train)
```

In [138]:
```
accuracy = metrics.r2_score (y_train , y_pred)
print ('R Squared Score : ' , accuracy)
```

```
R Squared Score :  0.8467411505365876
```

In [139]:
```
# model prediction on testing datadet
pred = ada.predict(X_test)
```

In [140]:
```
accuracy = metrics.r2_score(y_test,pred)
print ('R Squared Score : ' , accuracy)
```

```
R Squared Score :  0.836381985047736
```

Model Scores

```
Training Score = 84.67411505365876 %
testing Score = 83.6381985047736 %
```

Ada Boost Regressor Model Scores

Training Score = 84.67411505365876 %

testing Score = 83.6381985047736 %

Hyperparameter Tuning for Ada Boost

In [141]:
```
### HYPERPARAMETER TUNING ###
from sklearn.model_selection import RandomizedSearchCV
```

In [142]:
```
params = {'n_estimators': [45,47,53,55,60,70] ,
          'learning_rate':[0.25,0.30,0.40]}
```

In [143]:
```
rnd_srch = RandomizedSearchCV(AdaBoostRegressor() , cv=5 , param_distributions=params , n_jobs=-1)
```

In [144]:
```
rnd_srch.fit(X_train,y_train)
```

Out[144]:
```
RandomizedSearchCV(cv=5, estimator=AdaBoostRegressor(), n_jobs=-1,
                   param_distributions={'learning_rate': [0.25, 0.3, 0.4],
                                        'n_estimators': [45, 47, 53, 55, 60,
                                                         70]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [145]:
```
rnd_srch.best_params_
```

Out[145]:
```
{'n_estimators': 47, 'learning_rate': 0.3}
```

In [146]:
```
rnd_srch.best_estimator_
```

Out[146]:
```
AdaBoostRegressor(learning_rate=0.3, n_estimators=47)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [150]:
```
ada = AdaBoostRegressor(learning_rate=0.3, n_estimators=50)
ada.fit(X_train,y_train)

pred=ada.predict(X_train)
print('====Training Score====')
print(metrics.r2_score(y_train,pred))
y_pred = ada.predict(X_test)

print ('=== Testing Score ===')
print (metrics.r2_score(y_test,y_pred))
```

```
====Training Score====
0.8411012252304766
=== Testing Score ===
0.8480447033375522
```

Model Score after Hyperparameter Tuning

Training Score = 84.11012252304766 %

Testing Score = 84.80447033375522 %

Model Score after Hyperparameter Tuning

Training Score = 84.11012252304766 %

Testing Score = 84.80447033375522 %

Cross Validation for Ada Boost

In [151]:
```
#Cross Vaildation
training=ada.score(X_train,y_train)
testing=ada.score(X_test,y_test)

from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score=cross_val_score(ada,X,y,cv=j)
    cv_mean=cv_score.mean()
    print(f'At cross fold {j} the cv score is {cv_mean} and the R2 score for Training is {training} and R2 score for the Testing
    print('\n')
```

```
At cross fold 2 the cv score is 0.7454234986000483 and the R2 score for Training is 0.8411012252304766 and R2 score for the Tes
ting is0.8480447033375522


At cross fold 3 the cv score is 0.8059763594850535 and the R2 score for Training is 0.8411012252304766 and R2 score for the Tes
ting is0.8480447033375522


At cross fold 4 the cv score is 0.7704275275492436 and the R2 score for Training is 0.8411012252304766 and R2 score for the Tes
ting is0.8480447033375522


At cross fold 5 the cv score is 0.7920116392793629 and the R2 score for Training is 0.8411012252304766 and R2 score for the Tes
ting is0.8480447033375522


At cross fold 6 the cv score is 0.7822108830347632 and the R2 score for Training is 0.8411012252304766 and R2 score for the Tes
ting is0.8480447033375522


At cross fold 7 the cv score is 0.7762491905241705 and the R2 score for Training is 0.8411012252304766 and R2 score for the Tes
ting is0.8480447033375522


At cross fold 8 the cv score is 0.7758793819545073 and the R2 score for Training is 0.8411012252304766 and R2 score for the Tes
ting is0.8480447033375522


At cross fold 9 the cv score is 0.7848534244826393 and the R2 score for Training is 0.8411012252304766 and R2 score for the Tes
ting is0.8480447033375522
```

Cross Validation score for Ada Boost

Cross Validation Score at cv = 3 = 80.59763594850535 %

Training score = 84.11012252304766 %

Testing Score = 84.80447033375522 %

Cross Validation score for Ada Boost¶

Cross Validation Score at cv = 3 = 80.59763594850535 %

Training score = 84.11012252304766 %

Testing Score = 84.80447033375522 %

3rd Model I have Created is Random Forest Regressor

RandomForestRegressor Model

In [152]:
```
#import necessary library

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

Finding the Best Random State

In [153]:
```
#Best Random State
MaxAccu=0
MaxRS=0

for i in range (0,200):
    X_train,X_test,y_train,y_test=train_test_split(X_scalar,y,test_size=0.25,random_state=i)
    rf=RandomForestRegressor()
    rf.fit(X_train,y_train)

    pred=rf.predict(X_train)
    training=rf.score(X_train,y_train)
    print ('Training Score' , training*100 , 'RandomState' ,i)

    y_pred=rf.predict(X_test)
    testing=rf.score(X_test,y_test)
    print ('Testing Score' , testing*100 , 'RandomState' ,i)
    print('\n')


    if testing>MaxAccu:
        MaxAccu=testing
        MaxRS=i
        print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , i)
```

```
Training Score 96.91319986894179 RandomState 0
Testing Score 89.43397742032782 RandomState 0


MAXINING TESTING SCORE 89.43397742032782 ON RANDOM STATE OF 0
Training Score 97.36384465742964 RandomState 1
Testing Score 73.06515657350829 RandomState 1


Training Score 97.1432538917102 RandomState 2
Testing Score 82.85111165534651 RandomState 2
```

```
Testing Score 72.79114847023742 RandomState 3


Training Score 97.10531307313553 RandomState 4
Testing Score 80.67050391990307 RandomState 4
```

In [154]:
```
print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , MaxRS)
```

```
MAXINING TESTING SCORE 89.43397742032782 ON RANDOM STATE OF 0
```

Training the model

In [156]:
```
#splliting our data into train test split and randomstate 8
X_train,X_test,y_train,y_test=train_test_split(X_scalar,y,test_size=0.25,random_state=0)
```

In [157]:
```
rf=RandomForestRegressor()
rf.fit(X_train,y_train)
```

Out[157]:
```
RandomForestRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [158]:
```
 # model prediction on training dataset
y_pred = rf.predict(X_train)
```

In [159]:
```
accuracy = metrics.r2_score (y_train , y_pred)
print ('R Squared Score : ' , accuracy)
```

```
R Squared Score :  0.9723085190688893
```

In [160]:
```
# model prediction on testing datadet
pred = rf.predict(X_test)
```

In [161]:
```
accuracy = metrics.r2_score(y_test,pred)
print ('R Squared Score : ' , accuracy)
```

```
R Squared Score :  0.887964993610093
```

Model Score¶

```
Training Score = 97.23085190688893 %
Testing Score = 88.87964993610093 5 %
```

Random Forest Regressor Model Score

Training Score = 97.23085190688893 %

Testing Score = 88.879649936100935 %

```
In [163]: # define parameters
          parameters={'criterion':['mse','mae','poisson'],
                      'max_features':['auto','sqrt','log2'],
                      'min_samples_split':[1,11],
                      'max_depth':[1,15],
                      'min_samples_leaf':[1,7]}

In [164]: rf=RandomForestRegressor()
          clf=GridSearchCV(rf,parameters)
          clf.fit(X_train,y_train)

Out[164]: GridSearchCV(estimator=RandomForestRegressor(),
                       param_grid={'criterion': ['mse', 'mae', 'poisson'],
                                   'max_depth': [1, 15],
                                   'max_features': ['auto', 'sqrt', 'log2'],
                                   'min_samples_leaf': [1, 7],
                                   'min_samples_split': [1, 11]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [165]: #print best parameters
          print(clf.best_params_)

          {'criterion': 'poisson', 'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 11}

In [183]: #reassign best parameters
          rf=RandomForestRegressor(criterion= 'absolute_error', max_depth= 15, max_features= 'log2', min_samples_leaf= 1, min_samples_spli
          rf.fit(X_train,y_train)

Out[183]: RandomForestRegressor(criterion='absolute_error', max_depth=15,
                                max_features='log2', min_samples_split=11)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [184]: from sklearn.metrics import r2_score
          print ('Training R2 Score: ' ,rf.score(X_train,y_train)*100)

          Training R2 Score:  90.99536289729896

In [185]: pred_decision=rf.predict(X_test)
          rfs = r2_score(y_test,pred_decision)

In [186]: print('Testing R2 Score:' , rfs*100)

          Testing R2 Score: 85.5106099225744
```

Model Score after Hyperparameter Tuning¶

Training Score = 90.99536289729896 %
Testing Score = 85.5106099225744 %

Model Score after Hyperparameter Tuning

Training Score = 90.99536289729896 %

Testing Score = 85.5106099225744 %

Cross Vaildation for Random Forest

In [187]:
```python
#Cross Vaildation
training=rf.score(X_train,y_train)
testing=rf.score(X_test,y_test)

from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score=cross_val_score(rf,X,y,cv=j)
    cv_mean=cv_score.mean()
    print(f'At cross fold {j} the cv score is {cv_mean} and the R2 score for Training is {training} and R2 score for the Testing
    print('\n')
```

```
At cross fold 2 the cv score is 0.7953277297533681 and the R2 score for Training is 0.9099536289729896 and R2 score for the Tes
ting is0.8551060992257441


At cross fold 3 the cv score is 0.8147876958735258 and the R2 score for Training is 0.9099536289729896 and R2 score for the Tes
ting is0.8551060992257441


At cross fold 4 the cv score is 0.81071331138723 and the R2 score for Training is 0.9099536289729896 and R2 score for the Testi
ng is0.8551060992257441


At cross fold 5 the cv score is 0.8208799254184242 and the R2 score for Training is 0.9099536289729896 and R2 score for the Tes
ting is0.8551060992257441


At cross fold 6 the cv score is 0.8168526220682012 and the R2 score for Training is 0.9099536289729896 and R2 score for the Tes
ting is0.8551060992257441


At cross fold 7 the cv score is 0.8126531404929231 and the R2 score for Training is 0.9099536289729896 and R2 score for the Tes
ting is0.8551060992257441


At cross fold 8 the cv score is 0.8237671554265571 and the R2 score for Training is 0.9099536289729896 and R2 score for the Tes
ting is0.8551060992257441


At cross fold 9 the cv score is 0.816877607519066 and the R2 score for Training is 0.9099536289729896 and R2 score for the Test
ing is0.8551060992257441
```

Cross Validation score

```
Cross Vaildation Score at cv = 8 is = 82.37671554265571 %
Training score = 90.99536289729896 %
Testing Score = 85.51060992257441 %
```

Cross Validation score for Random Forest Regressor Model

Cross Vaildation Score at cv = 8 is = 82.37671554265571 %

Training score = 90.99536289729896 %

Testing Score = 85.51060992257441 %

4th Model I have Created is Gradient Boosting Regressor Model

GradientBoostingRegressor Model

In [119]:
```python
# import library

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectPercentile , chi2
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingRegressor
```

Finding the Best Random State

In [120]:
```python
#Best Random State
MaxAccu=0
MaxRS=0

for i in range (0,200):
    X_train,X_test,y_train,y_test=train_test_split(X_scalar,y,test_size=0.25,random_state=i)
    gbdt=GradientBoostingRegressor()
    gbdt.fit(X_train,y_train)

    pred=gbdt.predict(X_train)
    training=gbdt.score(X_train,y_train)
    print ('Training Score' , training*100 , 'RandomState' ,i)

    y_pred=gbdt.predict(X_test)
    testing=gbdt.score(X_test,y_test)
    print ('Testing Score' , testing*100 , 'RandomState' ,i)
    print('\n')


    if testing>MaxAccu:
        MaxAccu=testing
        MaxRS=i
        print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , i)
```

```
Testing Score 85.72961237198636 RandomState 13


Training Score 95.60965124946739 RandomState 14
Testing Score 74.85926898502332 RandomState 14
```

In [121]:
```python
print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , MaxRS)
```

```
MAXINING TESTING SCORE 90.6869855724089 ON RANDOM STATE OF 0
```

Training the model

In [122]:
```python
#splliting our data into train test split and randomstate 8
X_train,X_test,y_train,y_test=train_test_split(X_scalar,y,test_size=0.25,random_state=0)
```

In [123]:
```python
# initiate GradientBoostingClassifier
gbdt= GradientBoostingRegressor()
gbdt.fit(X_train , y_train)
```

Out[123]:
```
▾ GradientBoostingRegressor
GradientBoostingRegressor()
```

In [124]:
```python
 # model prediction on training dataset
y_pred = gbdt.predict(X_train)
```

In [129]:
```python
from sklearn.metrics import r2_score
import sklearn.metrics as metrics
accuracy = metrics.r2_score (y_train , y_pred)
print ('R Squared Score : ' , accuracy)
```

```
R Squared Score :  0.9508248740113718
```

In [130]:
```python
# model prediction on testing datadet
pred = gbdt.predict(X_test)
```

In [131]:
```python
accuracy = metrics.r2_score(y_test,pred)
print ('R Squared Score : ' , accuracy)
```

```
R Squared Score :  0.9075708209352753
```

Model Score

```
Training Score = 95.08248740113718 %
Testing Score = 90.75708209352753 %
```

Gradient Boosting Regressor Model Model Score

Training Score = 95.08248740113718 %

Testing Score = 90.75708209352753 %

```
from sklearn.model_selection import GridSearchCV
```

```
In [133]: # internally it will use decision tree as name suggest GBDT and here we are going to add one new parameter i.e learning rate

grid_params = {'max_depth' : range(1,8),
               'min_samples_split': range(2,12,1),
               'learning_rate': np.arange(0.1 , 0.9),
               'n_estimators': [90,95,100,105,110]}
```

```
In [134]: grid = GridSearchCV(GradientBoostingRegressor() , param_grid = grid_params , n_jobs = -1)
```

```
In [135]: grid.fit(X_train,y_train)
```

```
Out[135]:
GridSearchCV
estimator: GradientBoostingRegressor
GradientBoostingRegressor
```

```
In [136]: grid.best_params_
```

```
Out[136]: {'learning_rate': 0.1,
 'max_depth': 4,
 'min_samples_split': 2,
 'n_estimators': 110}
```

```
In [137]: gbdt_clf = GradientBoostingRegressor(learning_rate= 0.1,
 max_depth= 4,
 min_samples_split= 2,
 n_estimators= 90)
```

```
In [138]: gbdt_clf.fit(X_train,y_train)
```

```
Out[138]:
GradientBoostingRegressor
GradientBoostingRegressor(max_depth=4, n_estimators=90)
```

```
In [139]:  # model prediction on training dataset
y_pred = gbdt_clf.predict(X_train)
```

```
In [140]: accuracy = metrics.r2_score (y_train , y_pred)
print ('R Squared Score : ' , accuracy)

R Squared Score :  0.9714359247141823
```

```
In [141]: # model prediction on testing datadet
pred = gbdt_clf.predict(X_test)
```

```
In [142]: accuracy = metrics.r2_score(y_test,pred)
print ('R Squared Score : ' , accuracy)

R Squared Score :  0.9016720009216272
```

Model Score after Hyperparameter Tuning

Training Score = 97.14359247141823 %

Testing Score = 90.16720009216272 %

Cross Vaildation for GradientBoostingRegressor

In [238]:

```python
#Cross Vaildation
training=gbdt_clf.score(X_train,y_train)
testing=gbdt_clf.score(X_test,y_test)

from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score=cross_val_score(gbdt_clf,X,y,cv=j)
    cv_mean=cv_score.mean()
    print(f'At cross fold {j} the cv score is {cv_mean} and the R2 score for Training is {training} and R2 score for the Testing
    print('\n')
```

```
At cross fold 2 the cv score is 0.8307474564140973 and the R2 score for Training is 0.9714359247141823 and R2 score for the Tes
ting is0.9011239118063306


At cross fold 3 the cv score is 0.8746623131352403 and the R2 score for Training is 0.9714359247141823 and R2 score for the Tes
ting is0.9011239118063306


At cross fold 4 the cv score is 0.8424320089995962 and the R2 score for Training is 0.9714359247141823 and R2 score for the Tes
ting is0.9011239118063306


At cross fold 5 the cv score is 0.8564989000321532 and the R2 score for Training is 0.9714359247141823 and R2 score for the Tes
ting is0.9011239118063306


At cross fold 6 the cv score is 0.8746861723818012 and the R2 score for Training is 0.9714359247141823 and R2 score for the Tes
ting is0.9011239118063306


At cross fold 7 the cv score is 0.8461789138992112 and the R2 score for Training is 0.9714359247141823 and R2 score for the Tes
ting is0.9011239118063306


At cross fold 8 the cv score is 0.858397929837742 and the R2 score for Training is 0.9714359247141823 and R2 score for the Test
ing is0.9011239118063306


At cross fold 9 the cv score is 0.8692087893251008 and the R2 score for Training is 0.9714359247141823 and R2 score for the Tes
ting is0.9011239118063306
```

Cross Validation score for GradientBoostingRegressor

```
Cross Validation score at cv = 6 is = 87.46861723818012 %
Training score = 97.14359247141823 %
Testing Score = 90.11239118063306 %
```

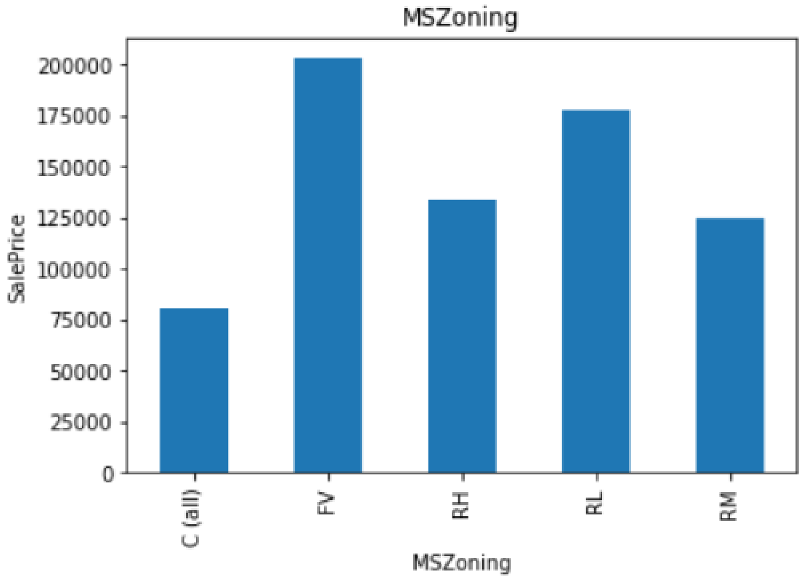
<u>Cross Validation score for Gradient Boosting Regressor</u>

Cross Validation score at cv = 6 is = 87.46861723818012 %

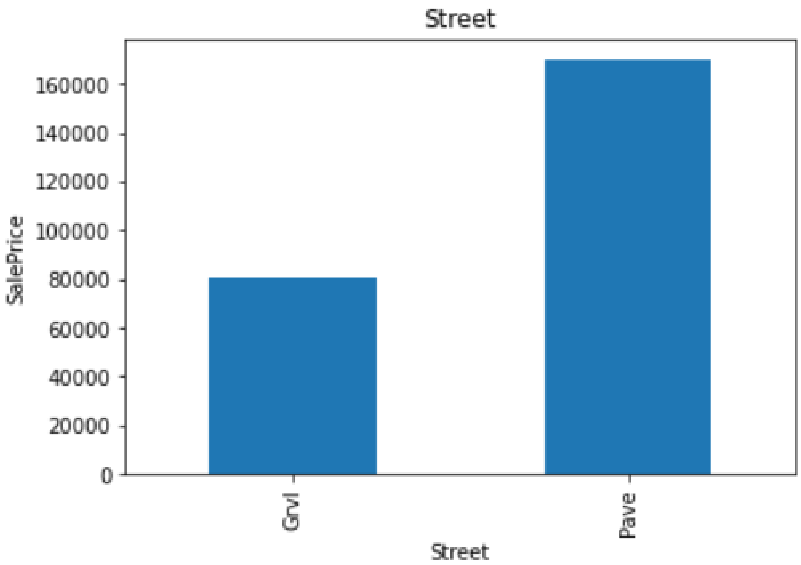
Training score = 97.14359247141823 %

Testing Score = 90.11239118063306 %

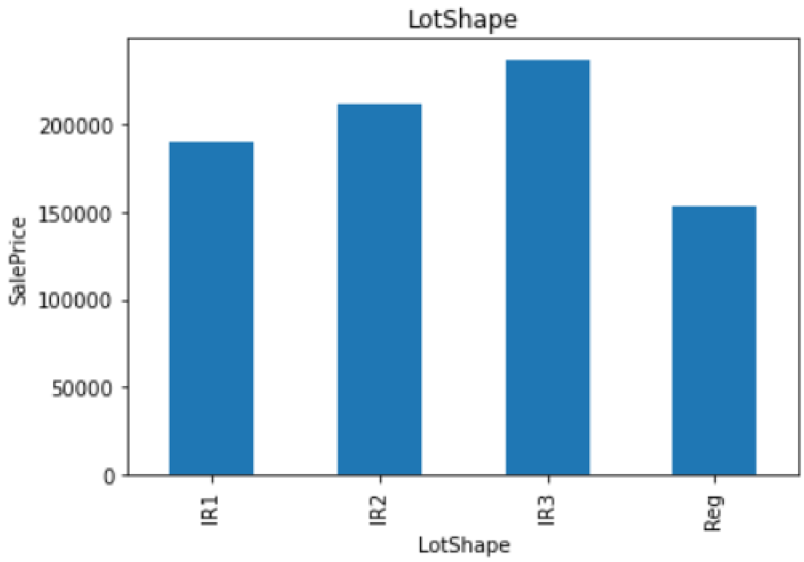
Visualizations and EDA



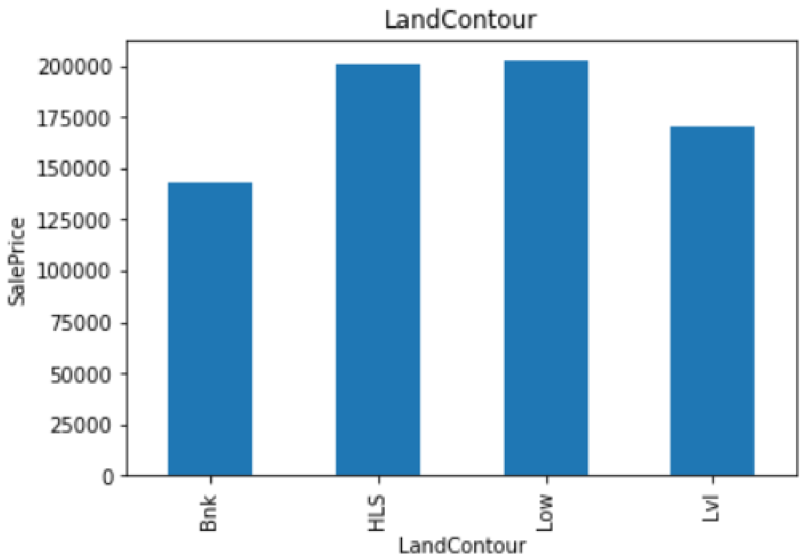
So according to the MSZoning FV (Floating Village Residential) has the highest selling price followed by RL (Residential Low Density) and C (Commercial) has a low selling price



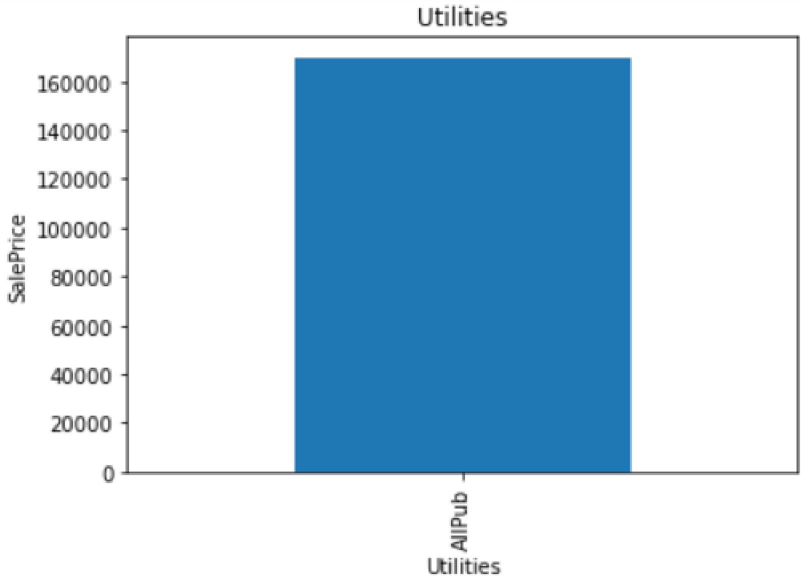
So According to the street graph, we have two types of roads Gravel and Paved so the road type pave has a high sale price as compared to gravel road type



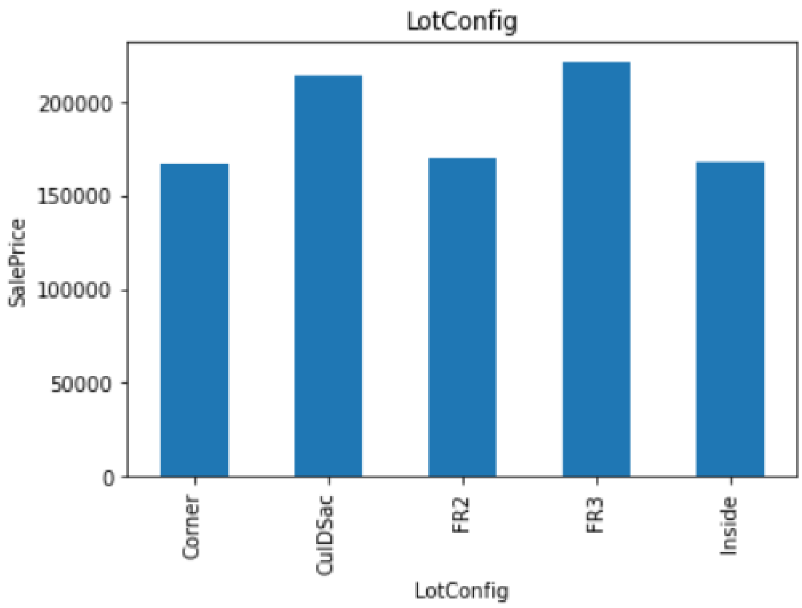
So According to the Lotshape, The IR3 (Irregular) type has a high sale price followed by IR2(Moderately Irregular) and IR1 (Slightly irregular) and the least sale price is for REG (Regular) shape lot shape.



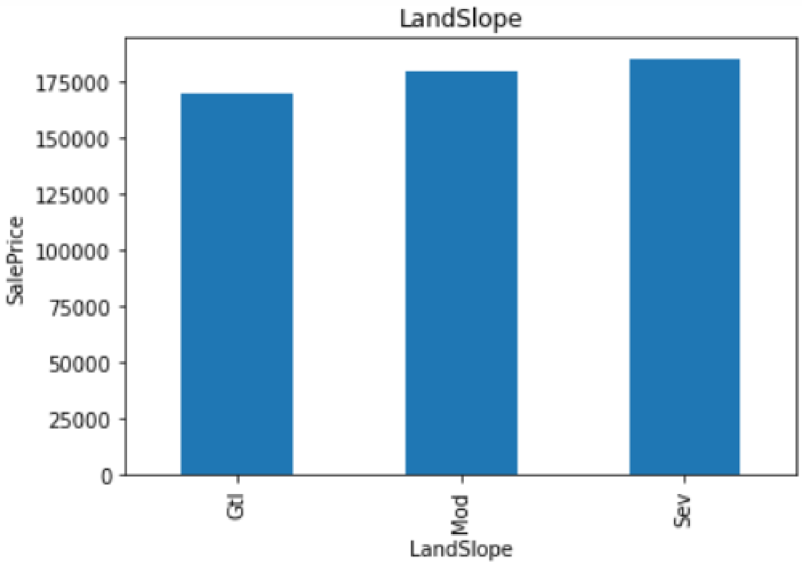
So According to the LandCountour HLS(Hillside - Significant slope from side to side) and LOW (Depression) have equal and higher sale prices as compared to other Landcontour followed by LVL (Near Flat/Level) types of the LandContour.



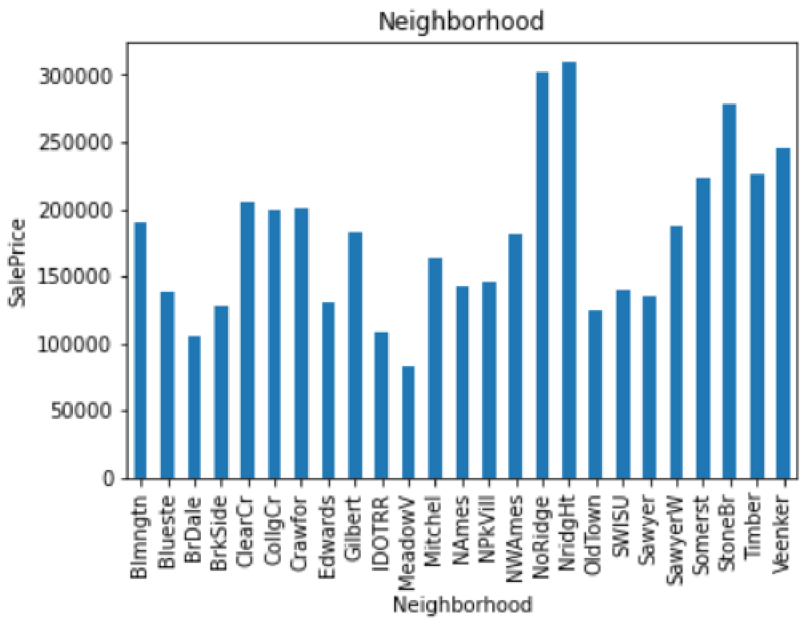
So According to the data, we have 4 types of utilities that are ALLPUB (All public Utilities (E, G, W,& S)), Nosewr (Electricity, Gas, and Water (Septic Tank)), Nosewa (Electricity and Gas Only), ELO (Electricity only) but according to our Graph we observe all the utilities in the dataset is only using 1 type of utilities that is AllPub (All public Utilities (E, G, W,& S)



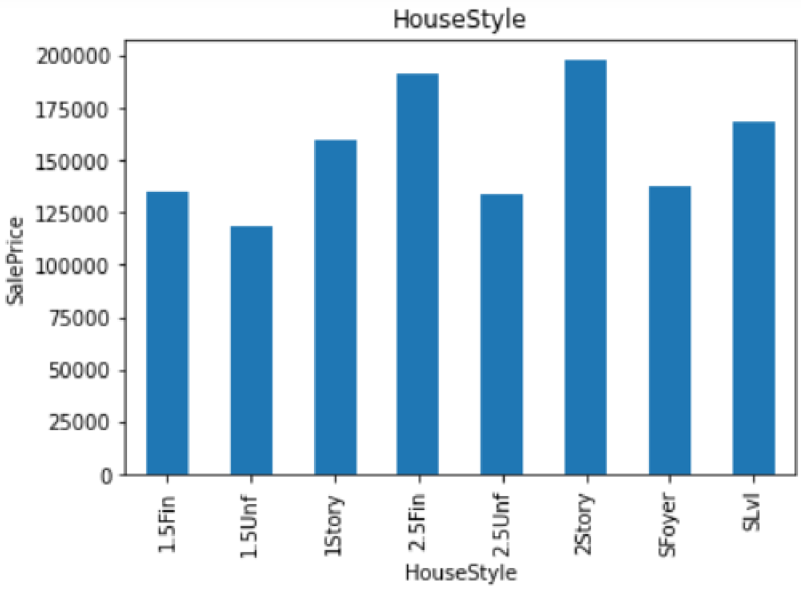
So According to the lot configuration, we have 5 types of lot configuration that is Inside (Inside lot), Corner (Corner lot), CulDSac (Cul-de-sac), FR2 (Frontage on 2 sides of the property), FR3 (Frontage on 3 sides of property) and when we compare to the sale price we observe FR3 and CulDSac has Highest Selling price.



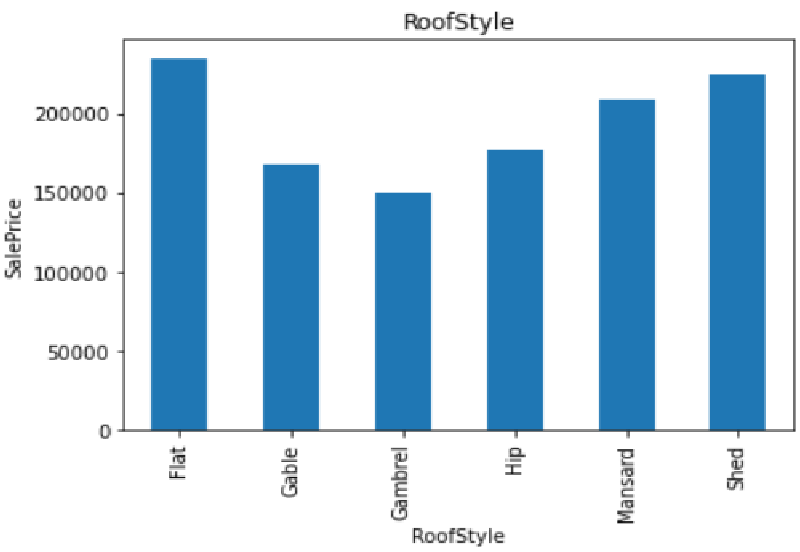
So according to the Landslope we have 3 types of landscape that is Gtl (Gentle slope), Mod (Moderate Slope), and Sev (Severe Slope), and when we compare to the sale price we observe Landslope which has the highest salingprice is Sev followed by Mod and Gtl.



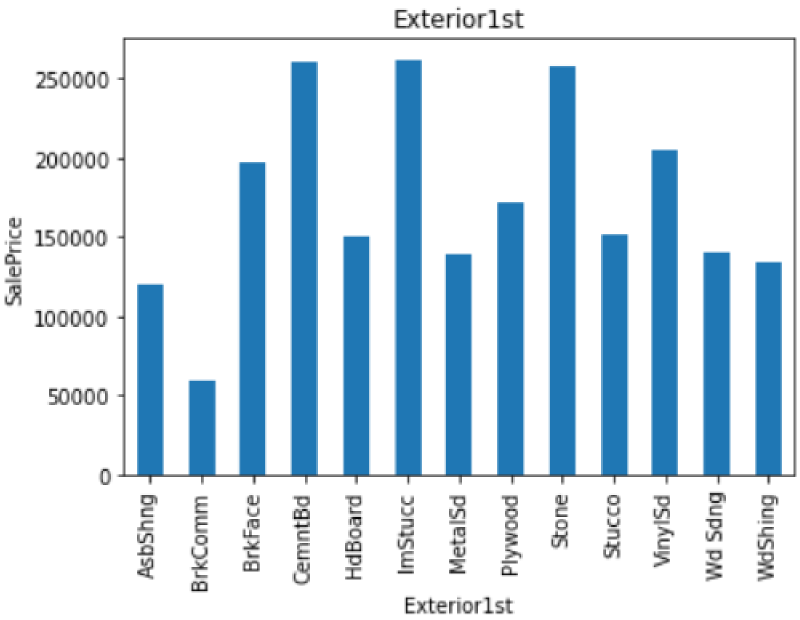
So according to the Neighbourhood, there are 25 unique types in the neighborhood and we observe NridgHt (Northridge Heights), and NWAmes (Northwest Ames) these 2 neighborhood has the highest selling price, and MeadowV (Meadow Village) has a low selling price.



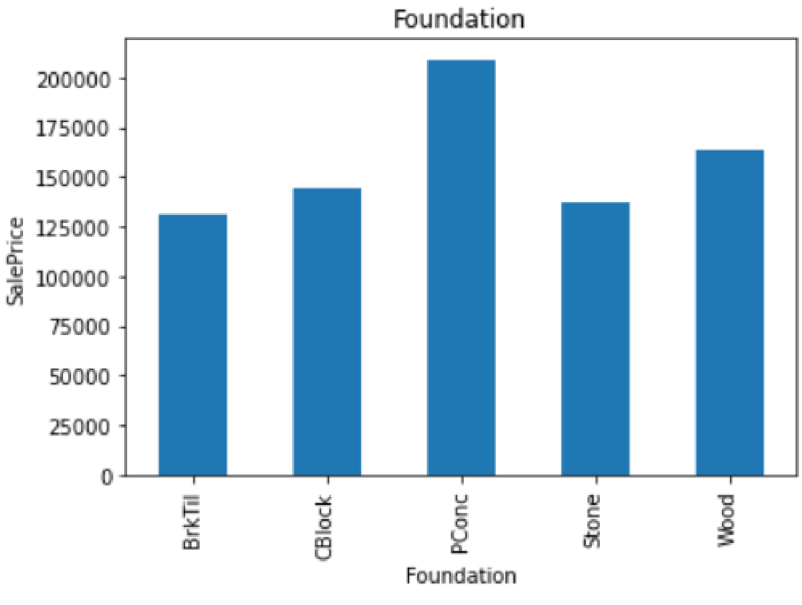
So according to the house style, we have 8 different house styles and in that 2Story and 2.5Fin (Two and one-half story: 2nd level finished) has the highest-selling Price as compared to otherHouse styles.



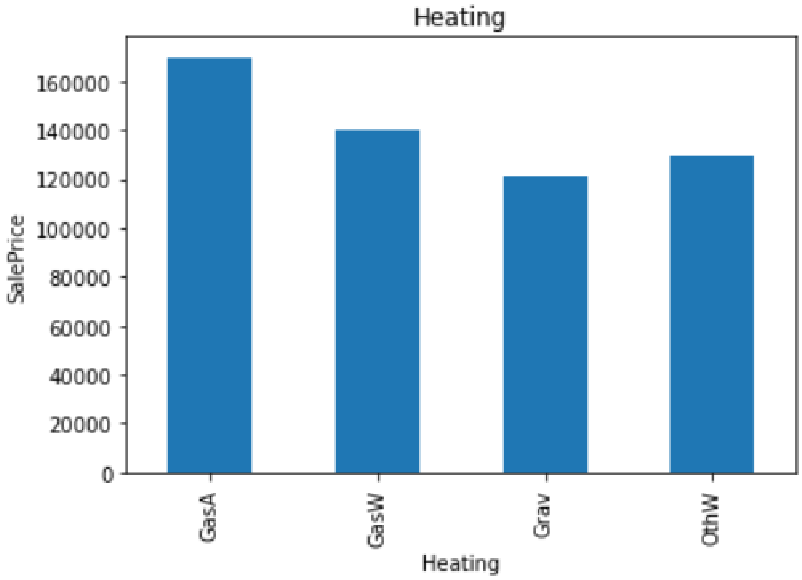
So according to the Roofstyle, we have 6 different roof style and we observe Flat and Shed has the highest selling price as compared to the other roof style.



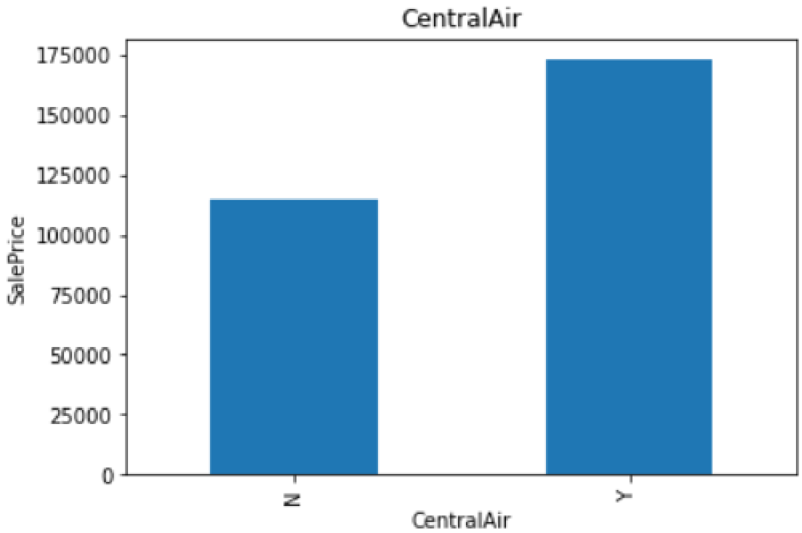
So according to the Exterior covering of the house, we have 17 different types of Exterior covering house but the most popular and highest selling are CemntBd (Cement Board), ImStucc (Imitation Stucco), and Stone (Stone).



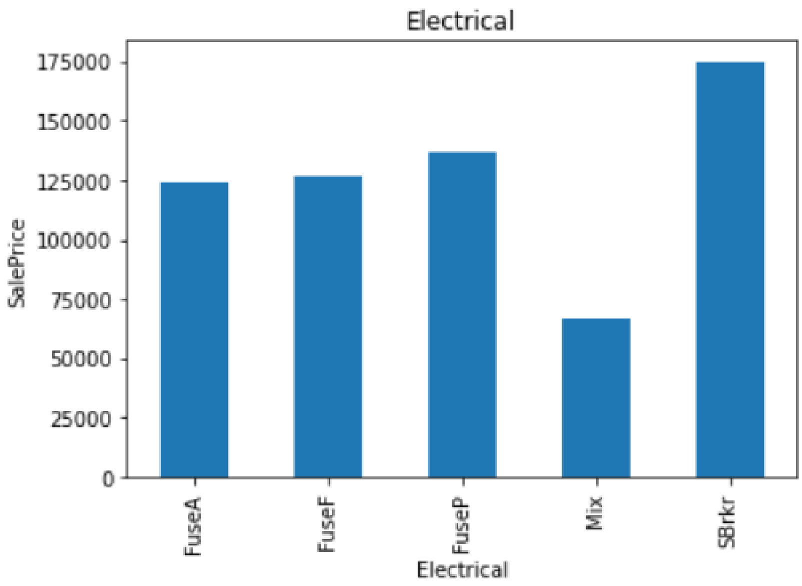
So according to the type of Foundation use we have 5 different types of foundation and the highest selling is the PConc (Poured Concrete) type of foundation.



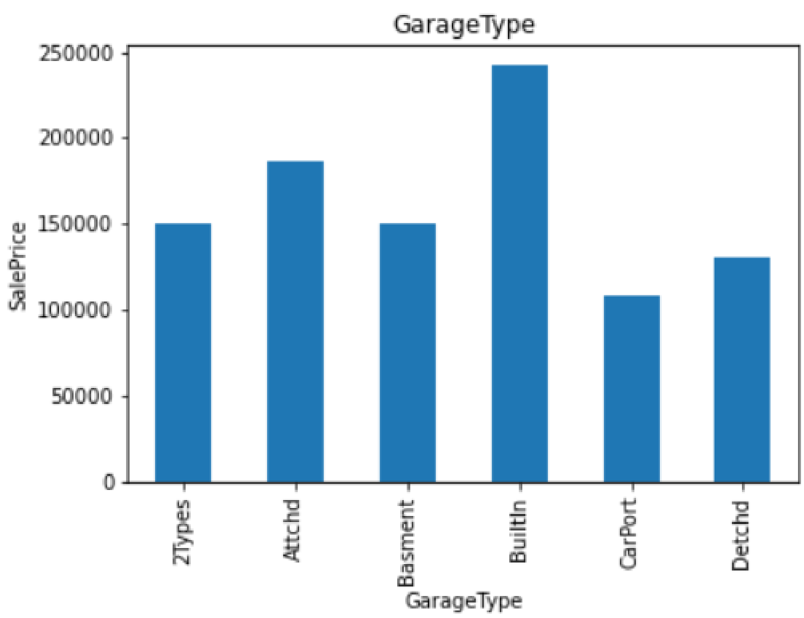
So According to the Heating, we have 5 different types of heating but the most selling type of heating is GasA (Gas forced warm air furnace).



So According to the CentrailAir which house that has Central is the highest selling as compared to the house which does not have central air.



So according to the Electrical System installed in every property the electrical system with SBrkr (Standard Circuit Breakers & Romex) has the highest selling price.



So according to the GarageType, we have 6 different types of the garage but BuiltIn Built-In (The garage part of the house - typically has room above the garage) has the highest selling price.

CONCLUSION

Key Findings and Conclusions of the Study

So from above all 4 model scores, we observe Gradient Boosting Regressor Model is best Suited model for this particular model as the training score is 97.14359247141823 % and the testing score is 90.11239118063306 % and the Cross-Validation score at cv = 6 is = 87.46861723818012 % thus saving this model and we will use this model to prediction on the test dataset.

Learning Outcomes of the Study in respect of Data Science

1) first we identify null values and applied values by using simple imputer

2)then I identified duplicates and I have dropped duplicates

3)performed EDA and wrote all observations for each graph

4)then i dropped unnecessary columns

5)then applied a label encoder to the categorical columns

6)then also plotted the Distribution plot and regression plot

7)then plotted boxplot to remove outliers

8)then treated outliers with the Z-score method

9)then scaled data and Also check for VIF

10)then find the co-relation between feature and label by the CORR method

11)then selected the top features by busing Selectkbest technique

12)then created 4 models that is Gradient Boosting Regressor, Random Forest Regressor ,linear Regressor model, Ada Boosting regressor model with hyperparameter tuning for all 4 models and also Cross-validations

13)At last I selected the best model according to their CV score and Training(R2) and testing score(R2)

Limitations of this work and Scope for Future Work

This work has mainly focused on analysing the housing data of Australia and predicting house prices using various models. The main limitation of this work is that it does not consider the changing demand for houses in different areas of Australia due to different factors like the proximity to commercial areas, schools, hospitals, etc. The scope for future work solving the housing project and predicting the sale price can include:

1. Analysing the changing demand for houses in different areas of Australia and understanding the factors that affect the demand.

2. Developing a model to predict house prices in different areas of Australia.

3. Analysing the impact of external factors like the socio-economic condition, locality, infrastructure, etc. on the house prices

. 4. Develop a recommendation system for prospective buyers to recommend the best neighbourhoods for buying a house.

5. Develop an automated system to identify potential buyers and sellers in the market.

6. Develop a model to forecast housing prices in the future.