FLIP ROBO

HOUSING PRICE PREDICTION

Submitted by:

POOJA JASWAL

ACKNOWLEDGMENT

I would like to convey my heartfelt gratitude to Ms Khushboo Garg for her tremendous support and assistance in the completion of my project. I would also like to thank for providing me with this wonderful opportunity to work on a project with the topic Housing Price Prediction. The project would not have been successful without your cooperation and inputs. Your useful advice and suggestions were really helpful to me during the project's completion. In this aspect, I am eternally grateful to you.

Other reference sources are mentioned below:-

Google

Medium.com

Analytics Vidhya

Stack Overflow

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. 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.

This is a real estate problem where a US based housing company named Surprise Housing has decided to invest in Australian Market. Their agenda is to buy houses in Australia at prices below their actual value in the market and sell them at high prices to gain profit. To do this this company uses data analytics to decide in which property they must invest. Company has collected the data of previously sold houses in Australia and with the help of this data they want to know to the value of prospective properties to decide whether it will suitable to invest in the properties or not.

Conceptual Background of the Domain Problem

The value of property also depends on the proximity of the property, its size its neighbourhood and audience for which the property is subjected to be sold. For example if audience is mainly concerned of commercial purpose. Then the property which is located in densely populated area will be sold very fast and at high prices compared to the one located at remote place. Similarly if audience is concerned only on living place then property with less dense area having large area with all services will be sold at higher prices.

The company is looking at prospective properties to buy houses to enter the market. We 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.

Review of Literature

House is one of human life's most essential needs, along with other fundamental needs such as food, water, and much more. Demand for houses grew rapidly over the years as people's living standards improved. While there are people who make their house as an investment and property, yet most people around the world are buying a house as their shelter or as their livelihood.

We 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.

Motivation for the Problem Undertaken

Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company. To understand real world problems where Machine Learning and Data Analysis can be applied to help organizations in various domains to make better decisions with the help of which they can gain profit or can be escaped from any loss which otherwise could be possible without the study of data .One of such domain is Real Estate.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

In Housing Price Prediction project we have performed various mathematical and statistical analysis such as we checked description or statistical summary of the data using describe, checked correlation and also visualized it using heatmap then we have used Z-Score to plot outliers and remove them.

```
In [24]: train_df.describe()
```

Out[24]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000
mean	56.767979	70.988470	10484.749144	6.104452	5.595890	1970.930651	1984.758562	102.310078	444.726027	46.647260	569.721747
std	41.940650	22.437056	8957.442311	1.390153	1.124343	30.145255	20.785185	182.047152	462.664785	163.520016	449.375525
min	20.000000	21.000000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	0.000000	0.000000
25%	20.000000	60.000000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	0.000000	216.000000
50%	50.000000	70.988470	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	0.000000	474.000000
75%	70.000000	79.250000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	0.000000	816.000000
max	190.000000	313.000000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.000000	2336.000000

- In the columns Id, MSSubclass, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfsF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath, HalfBath, TotRmsAbvGrd, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, Miscval, salePrice mean is considerably greater than median so the columns are positively skewed.
- In the columns FullBath, BedroomAbvGr, Fireplaces, Garagecars, GarageArea, YrSold Median is greater than mean so the columns are negatively skewed.
- In the columns Id, MSSubClass, LotFrontage, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtHalfBath, BedroomAbvGr, ToRmsAbvGrd, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, MiscVal, SalePrice there is considerable difference between the 75 percentile and maximum so outliers are present.

Data Sources and their formats

The variable features of this problem statement are as :

MSSubClass: Identifies the type of dwelling involved in the sale
MSZoning: Identifies the general zoning classification of the sale
LotFrontage: Linear feet of street connected to property
LotArea: Lot size in square feet
Street: Type of road access to property
Alley: Type of alley access to property
LotShape: General shape of property
LandContour: Flatness of the property
Utilities: Type of utilities available
LotConfig: Lot configuration
LandSlope: Slope of property
Neighborhood: Physical locations within Ames city limits
Condition1: Proximity to various conditions
Condition2: Proximity to various conditions (if more than one is present)
BldgType: Type of dwelling
HouseStyle: Style of dwelling
OverallQual: Rates the overall material and finish of the house
OverallCond: Rates the overall condition of the house
YearBuilt: Original construction date
YearRemodAdd: Remodel date (same as construction date if no remodelling or additions)
RoofStyle: Type of roof
RoofMatl: Roof material
Exterior1st: Exterior covering on house
Exterior2nd: Exterior covering on house (if more than one material)
MasVnrType: Masonry veneer type
MasVnrArea: Masonry veneer area in square feet
ExterQual: Evaluates the quality of the material on the exterior
ExterCond: Evaluates the present condition of the material on the exterior
Foundation: Type of foundation
BsmtQual: Evaluates the height of the basement
BsmtCond: Evaluates the general condition of the basement
BsmtExposure: Refers to walkout or garden level walls
BsmtFinType1: Rating of basement finished area
BsmtFinSF1: Type 1 finished square feet
BsmtFinType2: Rating of basement finished area (if multiple types)
BsmtFinSF2: Type 2 finished square feet
BsmtUnfSF: Unfinished square feet of basement area
TotalBsmtSF: Total square feet of basement area
Heating: Type of heating
HeatingQC: Heating quality and condition
CentralAir: Central air conditioning
Electrical: Electrical system
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
TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
Functional: Home functionality (Assume typical unless deductions are warranted)
Fireplaces: Number of fireplaces
FireplaceQu: Fireplace quality
GarageType: Garage location
GarageYrBlt: Year garage was built
GarageFinish: Interior finish of the garage
GarageCars: Size of garage in car capacity
GarageArea: Size of garage in square feet
GarageQual: Garage quality
GarageCond: Garage condition
PavedDrive: Paved driveway
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
Fence: Fence quality
MiscFeature: Miscellaneous feature not covered in other categories
MiscVal: $Value of miscellaneous feature
MoSold: Month Sold (MM)
YrSold: Year Sold (YYYY)
SaleType: Type of sale
SaleCondition: Condition of sale

```
In [18]: train_df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1168 entries, 0 to 1167
Data columns (total 79 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MSSubClass     1168 non-null   int64
 1   MSZoning       1168 non-null   object
 2   LotFrontage    954 non-null    float64
 3   LotArea        1168 non-null   int64
 4   Street         1168 non-null   object
 5   Alley          77 non-null     object
 6   LotShape       1168 non-null   object
 7   LandContour    1168 non-null   object
 8   LotConfig      1168 non-null   object
 9   LandSlope      1168 non-null   object
 10  Neighborhood   1168 non-null   object
 11  Condition1     1168 non-null   object
 12  Condition2     1168 non-null   object
 13  BldgType       1168 non-null   object
 14  HouseStyle     1168 non-null   object
 15  OverallQual    1168 non-null   int64
 16  OverallCond    1168 non-null   int64
 17  YearBuilt      1168 non-null   int64
 18  YearRemodAdd   1168 non-null   int64
 19  RoofStyle      1168 non-null   object
 20  RoofMatl       1168 non-null   object
 21  Exterior1st    1168 non-null   object
 22  Exterior2nd    1168 non-null   object
 23  MasVnrType     1161 non-null   object
 24  MasVnrArea     1161 non-null   float64
 25  ExterQual      1168 non-null   object
 26  ExterCond      1168 non-null   object
 27  Foundation     1168 non-null   object
 28  BsmtQual       1138 non-null   object
 29  BsmtCond       1138 non-null   object
 30  BsmtExposure   1137 non-null   object
 31  BsmtFinType1   1138 non-null   object
```

```
 32  BsmtFinSF1     1168 non-null   int64
 33  BsmtFinType2   1137 non-null   object
 34  BsmtFinSF2     1168 non-null   int64
 35  BsmtUnfSF      1168 non-null   int64
 36  TotalBsmtSF    1168 non-null   int64
 37  Heating        1168 non-null   object
 38  HeatingQC      1168 non-null   object
 39  CentralAir     1168 non-null   object
 40  Electrical     1168 non-null   object
 41  1stFlrSF       1168 non-null   int64
 42  2ndFlrSF       1168 non-null   int64
 43  LowQualFinSF   1168 non-null   int64
 44  GrLivArea      1168 non-null   int64
 45  BsmtFullBath   1168 non-null   int64
 46  BsmtHalfBath   1168 non-null   int64
 47  FullBath       1168 non-null   int64
 48  HalfBath       1168 non-null   int64
 49  BedroomAbvGr   1168 non-null   int64
 50  KitchenAbvGr   1168 non-null   int64
 51  KitchenQual    1168 non-null   object
 52  TotRmsAbvGrd   1168 non-null   int64
 53  Functional     1168 non-null   object
 54  Fireplaces     1168 non-null   int64
 55  FireplaceQu    617 non-null    object
 56  GarageType     1104 non-null   object
 57  GarageYrBlt    1104 non-null   float64
 58  GarageFinish   1104 non-null   object
 59  GarageCars     1168 non-null   int64
 60  GarageArea     1168 non-null   int64
 61  GarageQual     1104 non-null   object
 62  GarageCond     1104 non-null   object
 63  PavedDrive     1168 non-null   object
 64  WoodDeckSF     1168 non-null   int64
 65  OpenPorchSF    1168 non-null   int64
 66  EnclosedPorch  1168 non-null   int64
 67  3SsnPorch      1168 non-null   int64
 68  ScreenPorch    1168 non-null   int64
 69  PoolArea       1168 non-null   int64
 70  PoolQC         7 non-null      object
 71  Fence          237 non-null    object
 72  MiscFeature    44 non-null     object
 73  MiscVal        1168 non-null   int64
 74  MoSold         1168 non-null   int64
 75  YrSold         1168 non-null   int64
 76  SaleType       1168 non-null   object
 77  SaleCondition  1168 non-null   object
 78  SalePrice      1168 non-null   int64
dtypes: float64(3), int64(34), object(42)
memory usage: 730.0+ KB
```

In [19]:
```
train_df.dtypes
```

Out[19]:
```
MSSubClass        int64
MSZoning         object
LotFrontage     float64
LotArea           int64
Street           object
Alley            object
LotShape         object
LandContour      object
LotConfig        object
LandSlope        object
Neighborhood     object
Condition1       object
Condition2       object
BldgType         object
HouseStyle       object
OverallQual       int64
OverallCond       int64
YearBuilt         int64
YearRemodAdd      int64
RoofStyle        object
RoofMatl         object
Exterior1st      object
Exterior2nd      object
MasVnrType       object
MasVnrArea      float64
ExterQual        object
ExterCond        object
Foundation       object
BsmtQual         object
BsmtCond         object
BsmtExposure     object
BsmtFinType1     object
BsmtFinSF1        int64
BsmtFinType2     object
BsmtFinSF2        int64
BsmtUnfSF         int64
TotalBsmtSF       int64
Heating          object
```

```
2ndFlrSF          int64
LowQualFinSF      int64
GrLivArea         int64
BsmtFullBath      int64
BsmtHalfBath      int64
FullBath          int64
HalfBath          int64
BedroomAbvGr      int64
KitchenAbvGr      int64
KitchenQual      object
TotRmsAbvGrd      int64
Functional       object
Fireplaces        int64
FireplaceQu      object
GarageType       object
GarageYrBlt     float64
GarageFinish     object
GarageCars        int64
GarageArea        int64
GarageQual       object
GarageCond       object
PavedDrive       object
WoodDeckSF        int64
OpenPorchSF       int64
EnclosedPorch     int64
3SsnPorch         int64
ScreenPorch       int64
PoolArea          int64
PoolQC           object
Fence            object
MiscFeature      object
MiscVal           int64
MoSold            int64
YrSold            int64
SaleType         object
SaleCondition    object
SalePrice         int64
dtype: object
```

Now we have to loaded the require libraries after that we have to loaded the data into our jupyter notebook.

```
In [1]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        import warnings
        warnings.filterwarnings('ignore')
```

```
In [93]: from sklearn.model_selection import train_test_split
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.metrics import r2_score
         from sklearn.metrics import mean_absolute_error
         from sklearn.metrics import mean_squared_error
```

```
In [1]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        import warnings
        warnings.filterwarnings('ignore')
```

```
In [2]: path = "/Users/aerryjoop/Desktop/train.csv"
```

```
In [3]: train_df=pd.read_csv(path)
```

```
In [4]: pd.set_option('display.max_rows', 1500)
        pd.set_option('display.max_columns', 1500)
        pd.set_option('display.width', 1000)
        pd.set_option('display.max_rows',None)
```

```
In [7]: train_df.head().T
```

Out[7]:

	0	1	2	3	4
Id	127	889	793	110	422
MSSubClass	120	20	60	20	20
MSZoning	RL	RL	RL	RL	RL
LotFrontage	NaN	95.0	92.0	105.0	NaN
LotArea	4928	15865	9920	11751	16635
Street	Pave	Pave	Pave	Pave	Pave
Alley	NaN	NaN	NaN	NaN	NaN
LotShape	IR1	IR1	IR1	IR1	IR1
LandContour	Lvl	Lvl	Lvl	Lvl	Lvl
Utilities	AllPub	AllPub	AllPub	AllPub	AllPub
LotConfig	Inside	Inside	CulDSac	Inside	FR2

```
In [8]: train_df.tail().T
```

Out[8]:

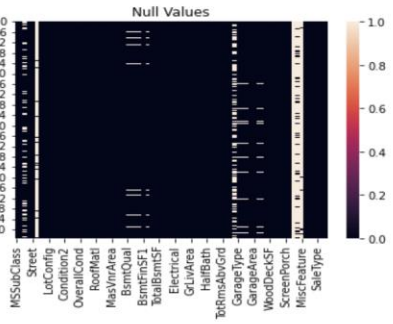
	1163	1164	1165	1166	1167
Id	289	554	196	31	617
MSSubClass	20	20	160	70	60
MSZoning	RL	RL	RL	C (all)	RL
LotFrontage	NaN	67.0	24.0	50.0	NaN
LotArea	9819	8777	2280	8500	7861
Street	Pave	Pave	Pave	Pave	Pave
Alley	NaN	NaN	NaN	Pave	NaN
LotShape	IR1	Reg	Reg	Reg	IR1
LandContour	Lvl	Lvl	Lvl	Lvl	Lvl
Utilities	AllPub	AllPub	AllPub	AllPub	AllPub

Here we have to use the Feature Engineering for clean the data. Some unused columns have been deleted and even some columns have been bifurcated which was used in the prediction. We first done data cleaning. We first looked percentage of values missing in columns then we imputed missing values.

In [15]:
```
train_df.isnull().sum()
```
Out[15]:
```
MSSubClass          0
MSZoning            0
LotFrontage       214
LotArea             0
Street              0
Alley            1091
LotShape            0
LandContour         0
LotConfig           0
LandSlope           0
Neighborhood        0
Condition1          0
Condition2          0
BldgType            0
HouseStyle          0
OverallQual         0
OverallCond         0
YearBuilt           0
YearRemodAdd        0
RoofStyle           0
RoofMatl            0
Exterior1st         0
Exterior2nd         0
MasVnrType          7
MasVnrArea          7
ExterQual           0
ExterCond           0
Foundation          0
BsmtQual           30
BsmtCond           30
BsmtExposure       31
BsmtFinType1       30
BsmtFinSF1          0
BsmtFinType2       31
BsmtFinSF2          0
BsmtUnfSF           0
TotalBsmtSF         0
Heating             0
HeatingQC           0
CentralAir          0
Electrical          0
1stFlrSF            0
2ndFlrSF            0
LowQualFinSF        0
GrLivArea           0
BsmtFullBath        0
BsmtHalfBath        0
FullBath            0
HalfBath            0
```

In [16]:
```
train_df.isnull().sum().sum()
```
Out[16]:
```
5558
```

In [17]:
```
sns.heatmap(train_df.isnull())
plt.title("Null Values")
plt.show
```
Out[17]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

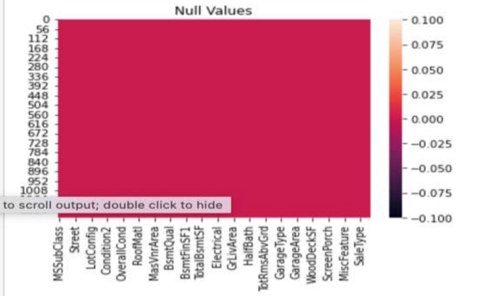


In [20]:
```
train_df['LotFrontage'] = train_df['LotFrontage'].fillna(train_df['LotFrontage'].mean())
train_df['Alley'] = train_df['Alley'].fillna(train_df['Alley'].mode()[0])
train_df['MasVnrType'] = train_df['MasVnrType'].fillna(train_df['MasVnrType'].mode()[0])
train_df['MasVnrArea'] = train_df['MasVnrArea'].fillna(train_df['MasVnrArea'].mean())
train_df['BsmtQual'] = train_df['BsmtQual'].fillna(train_df['BsmtQual'].mode()[0])
train_df['BsmtCond'] = train_df['BsmtCond'].fillna(train_df['BsmtCond'].mode()[0])
train_df['BsmtExposure'] = train_df['BsmtExposure'].fillna(train_df['BsmtExposure'].mode()[0])
train_df['BsmtFinType1'] = train_df['BsmtFinType1'].fillna(train_df['BsmtFinType1'].mode()[0])
train_df['BsmtFinType2'] = train_df['BsmtFinType2'].fillna(train_df['BsmtFinType2'].mode()[0])
train_df['FireplaceQu'] = train_df['FireplaceQu'].fillna(train_df['FireplaceQu'].mode()[0])
train_df['GarageType'] = train_df['GarageType'].fillna(train_df['GarageType'].mode()[0])
train_df['GarageYrBlt'] = train_df['GarageYrBlt'].fillna(train_df['GarageYrBlt'].mean())
train_df['GarageFinish'] = train_df['GarageFinish'].fillna(train_df['GarageFinish'].mode()[0])
train_df['GarageQual'] = train_df['GarageQual'].fillna(train_df['GarageQual'].mode()[0])
train_df['GarageCond'] = train_df['GarageCond'].fillna(train_df['GarageCond'].mode()[0])
train_df['PoolQC'] = train_df['PoolQC'].fillna(train_df['PoolQC'].mode()[0])
train_df['Fence'] = train_df['Fence'].fillna(train_df['Fence'].mode()[0])
train_df['MiscFeature'] = train_df['MiscFeature'].fillna(train_df['MiscFeature'].mode()[0])
```

In [21]:
```
train_df.isnull().sum().sum()
```
Out[21]:
```
0
```

In [22]:
```
sns.heatmap(train_df.isnull())
plt.title("Null Values")
plt.show
```

Out[22]: <function matplotlib.pyplot.show(close=None, block=None)>



In [24]:
```
train_df.describe()
```

Out[24]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000
mean	56.767979	70.988470	10484.749144	6.104452	5.595890	1970.930651	1984.758562	102.310078	444.726027	46.647260	569.721747
std	41.940650	22.437056	8957.442311	1.390153	1.124343	30.145255	20.785185	182.047152	462.664785	163.520016	449.375525
min	20.000000	21.000000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	0.000000	0.000000
25%	20.000000	60.000000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	0.000000	216.000000
50%	50.000000	70.988470	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	0.000000	474.000000
75%	70.000000	79.250000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	0.000000	816.000000
max	190.000000	313.000000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.000000	2336.000000

In [85]:
```
plt.figure(figsize=(30,25))
sns.heatmap(correlation,linewidths=.1,vmin=-1, vmax=1, fmt='.1g',linecolor="black", annot = True, annot_kws={'size':10},cmap="cool")
plt.title('Correlation matrix using heat map',fontsize=30)
plt.show()
```

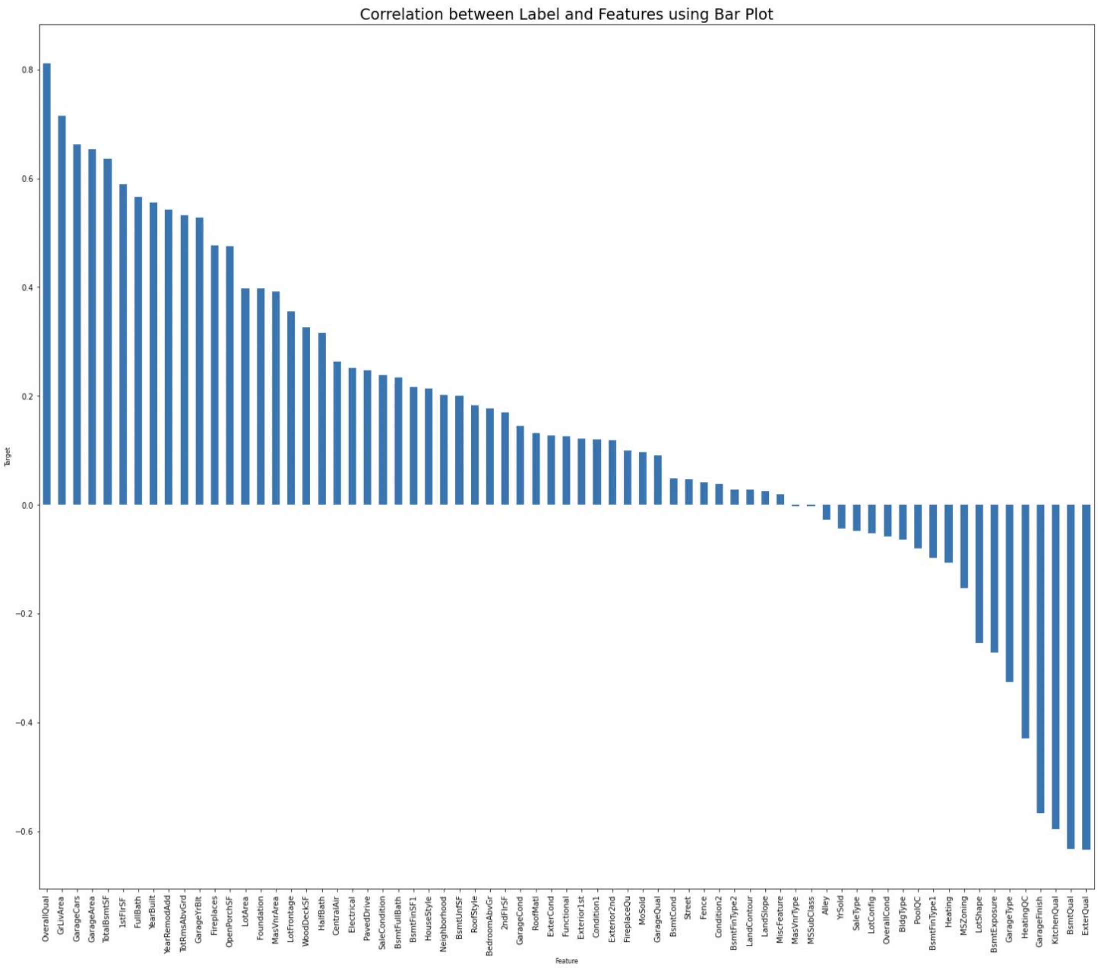
In this heatmap of the correlation we have observed the following:-

- No correlation has been observed between the column Id and other columns so we will be dropping this column.
- We observe multicollinearity in between columns so we will be using Principal Component Analysis.
- SalePrice is negatively correlated with OverallCond, KitchenAbvGr, Encloseporch, YrSold.
- SalePrice is highly positively correlated with the columns OverallQual, YearBuilt, YearRemodAdd, TotalBsmtSF, 1stFlrSF, GrLivArea, FullBath, TotRmsAbvGrd, GarageCars, GarageArea.

Data Inputs- Logic- Output Relationships

In [87]:

```
plt.figure(figsize=(25,20))
train_df.corr()['SalePrice'].sort_values(ascending=False).drop('SalePrice').plot(kind='bar')
plt.xlabel('Feature',fontsize=8)
plt.ylabel('Target',fontsize=8)
plt.title('Correlation between Label and Features using Bar Plot',fontsize=20)
plt.show()
```



Here we check the correlation between all our feature variables with target variable label

1. The column OverallQual is most positively correlated with SalePrice.

2. The column KitchenAbvGrd is most negatively correlated with SalePrice.

Set of assumptions related to the problem under consideration

Here we have observed that only one single unique value was present in Utilities column so we assumed that we will be dropping these columns. We have also observed multicollinearity in between columns so we assumed that we will be using Principal Component Analysis.

By looking into the target variable label we assumed that it was a Regression type of problem.

Hardware and Software Requirements and Tools Used

Hardware

HP ENVI X360AQ105X

Software

Jupyter Notebook (Anaconda 3) – Python 3.7.6 Microsoft package 2013

Library

The tools, libraries and packages we used for accomplishing this project are pandas, numpy, matplotlib, seaborn, scipy stats, sklearn.decomposition pca, sklearn standardscaler, GridSearchCV, joblib.

From sklearn.preprocessing import StandardScaler

As these columns are different in scale, they are standardized to have common scale while building machine learning model. This is useful when you want to compare data that correspond to different units.

from sklearn.preprocessing import Label Encoder

Label Encoder and One Hot Encoder. These two encoders are parts of the SciKit Learn library in Python, and they are used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

from sklearn.model_selection import train_test_split,cross_val_score

Train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.

Through pandas library we loaded our csv file ‘Data file’ into dataframe and performed data manipulation and analysis. With the help of numpy we worked with arrays. With the help of matplotlib and seaborn we did plot various graphs and figures and done data visualization.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

We have observed skewness in data so we tried to remove the skewness through treating outliers with technique. We first converted all our categorical variables to numeric variables with the help of dummy variables to checkout and dropped the columns which we felt were unnecessary. The data was improper scaled so we scaled the feature variables on a single scale using sklearn's StandardScaler package.

Testing of Identified Approaches (Algorithms)

The algorithms we used for the training and testing are as follows:-

- Random Forest Regressor
- SVR
- Linear Regressor
- SGD Regressor
- KNeighbors Regressor
- Gradient Boosting Regressor

Run and Evaluate Selected Models

In [101…

```
rfr = RandomForestRegressor()
rfr.fit(x_train,y_train)
rfrpred=rfr.predict(x_test)
print('R2_Score:',r2_score(y_test,rfrpred))
print('MAE:',mean_absolute_error(y_test, rfrpred))
print('MSE:',mean_squared_error(y_test, rfrpred))
print("RMSE:",np.sqrt(mean_squared_error(y_test, rfrpred)))
print("Cross_Validaton_Score",cross_val_score(rfr,x,y,cv=5).mean())
```

```
R2_Score: 0.9163313160526597
MAE: 15006.15555555556
MSE: 415877560.9065705
RMSE: 20393.076298257958
Cross_Validaton_Score 0.8775743191625309
```

In [102…

```
from sklearn.svm import SVR
svr = SVR()
svr.fit(x_train,y_train)
svrpred=svr.predict(x_test)
print('R2_Score:',r2_score(y_test,svrpred))
print('MAE:',mean_absolute_error(y_test, svrpred))
print('MSE:',mean_squared_error(y_test, svrpred))
print("RMSE:",np.sqrt(mean_squared_error(y_test, svrpred)))
print("Cross_Validaton_Score",cross_val_score(svr,x,y,cv=5).mean())
```

```
R2_Score: -0.055043446709241106
MAE: 53644.365441885915
MSE: 5244123303.577413
RMSE: 72416.31931807507
Cross_Validaton_Score -0.06453280296115219
```

In [103…

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train,y_train)
lrpred=lr.predict(x_test)
print('R2_score:',r2_score(y_test,lrpred))
print('MAE:',mean_absolute_error(y_test, lrpred))
print('MSE:',mean_squared_error(y_test, lrpred))
print("RMSE:",np.sqrt(mean_squared_error(y_test, lrpred)))
print("Cross_Validaton_Score",cross_val_score(lr,x,y,cv=5).mean())
```

```
R2_score: 0.9055889437624671
MAE: 16874.651833305837
MSE: 469272826.32280904
RMSE: 21662.70588644939
Cross_Validaton_Score 0.8520731190269025
```

In [104…

```
from sklearn.linear_model import SGDRegressor
sgd = SGDRegressor()
sgd.fit(x_train,y_train)
sgdpred=sgd.predict(x_test)
print('R2_Score:',r2_score(y_test,sgdpred))
print('MAE:',mean_absolute_error(y_test, sgdpred))
print('MSE:',mean_squared_error(y_test, sgdpred))
print("RMSE:",np.sqrt(mean_squared_error(y_test, sgdpred)))
print("Cross_Validaton_Score",cross_val_score(sgd,x,y,cv=5).mean())
```

```
R2_Score: 0.9073961119317414
MAE: 16845.415148848773
MSE: 460290245.80488443
RMSE: 21454.375912733616
Cross_Validaton_Score 0.8518245688960219
```

In [105…

```
from sklearn.neighbors import KNeighborsRegressor
knr = KNeighborsRegressor()
knr.fit(x_train,y_train)
knrpred=knr.predict(x_test)
print('R2_Score:',r2_score(y_test,knrpred))
print('MAE:',mean_absolute_error(y_test, knrpred))
print('MSE:',mean_squared_error(y_test, knrpred))
print("RMSE:",np.sqrt(mean_squared_error(y_test, knrpred)))
print("Cross_Validaton_Score",cross_val_score(knr,x,y,cv=5).mean())
```

```
R2_Score: 0.8271192598168915
MAE: 21402.447863247864
MSE: 859308610.6185757
RMSE: 29313.96613593213
Cross_Validaton_Score 0.7820142349615625
```

In [106…

```
from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor()
gbr.fit(x_train,y_train)
gbrpred=gbr.predict(x_test)
print('R2_Score:',r2_score(y_test,gbrpred))
print('MAE:',mean_absolute_error(y_test, gbrpred))
print('MSE:',mean_squared_error(y_test, gbrpred))
print("RMSE:",np.sqrt(mean_squared_error(y_test, gbrpred)))
print("Cross_Validaton_Score",cross_val_score(gbr,x,y,cv=5).mean())
```

```
R2_Score: 0.9155816162726059
MAE: 14751.17165201443
MSE: 419603964.87555355
RMSE: 20484.236985437205
Cross_Validaton_Score 0.8878142934333052
```

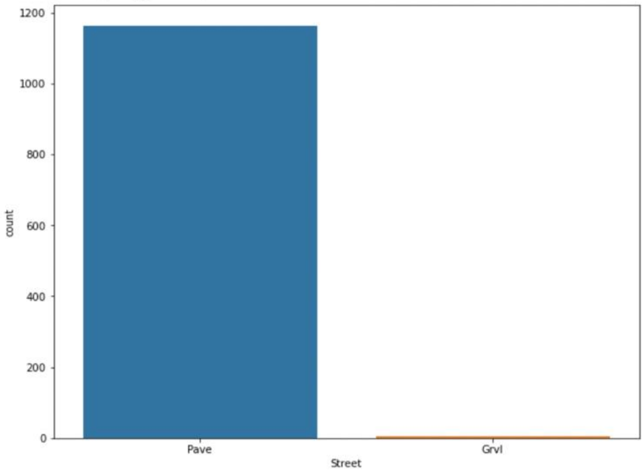
Key Metrics for success in solving problem under consideration

We used the metric Root Mean Squared Error by selecting the Ridge Regressor model which was giving us best(minimum) RMSE score.

Visualizations

```
In [33]: plt.figure(figsize=(10,8))
         ax = sns.countplot(x="Street",data=train_df)
         print(train_df["Street"].value_counts())
```

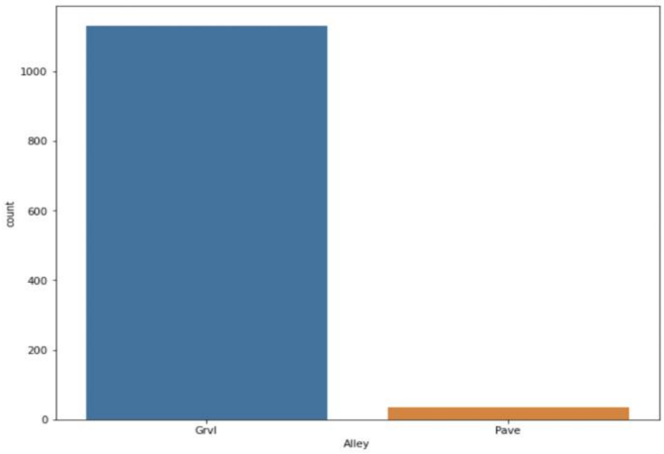
```
Pave    1164
Grvl       4
Name: Street, dtype: int64
```



Here we observed maximum number of Pave Street

```
In [34]: plt.figure(figsize=(10,8))
         ax = sns.countplot(x="Alley",data=train_df)
         print(train_df["Alley"].value_counts())
```

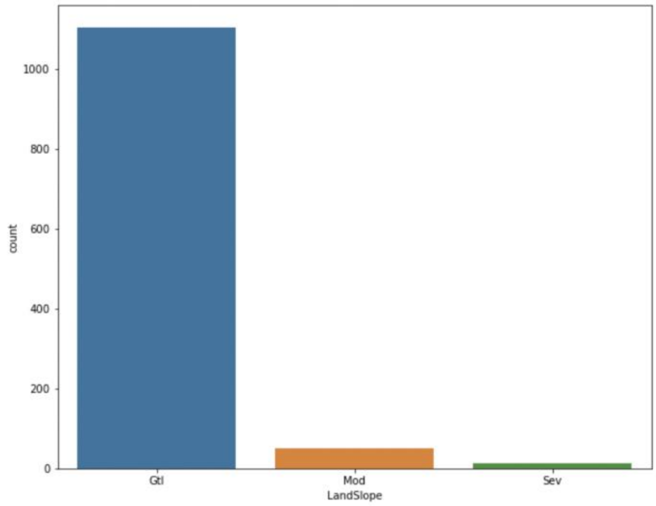
```
Grvl    1132
Pave      36
Name: Alley, dtype: int64
```



Maximum number of Grvl Alley

In [35]:
```
plt.figure(figsize=(10,8))
ax = sns.countplot(x="LandSlope",data=train_df)
print(train_df["LandSlope"].value_counts())
```

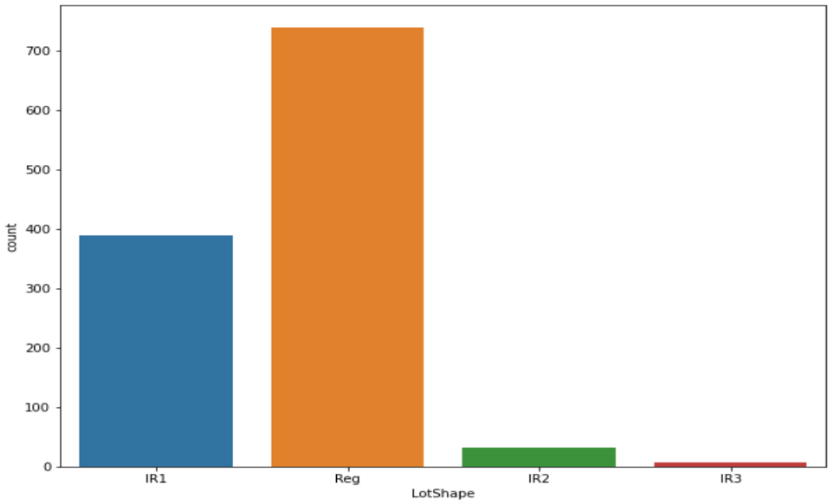
```
Gtl    1105
Mod      51
Sev      12
Name: LandSlope, dtype: int64
```



Here we have observed maximum number of Gtl LansdSlope

In [36]:
```
plt.figure(figsize=(10,8))
ax = sns.countplot(x="LotShape",data=train_df)
print(train_df["LotShape"].value_counts())
```

```
Reg    740
IR1    390
IR2     32
IR3      6
Name: LotShape, dtype: int64
```

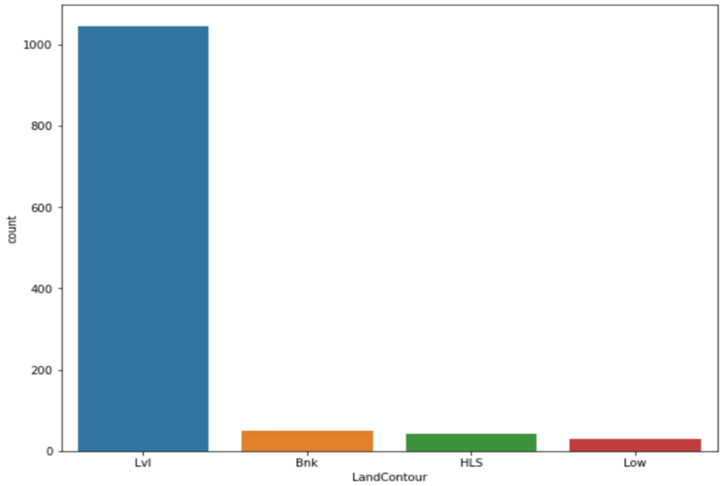


Maximum number of Reg LotShape

In [37]:

```
plt.figure(figsize=(10,8))
ax = sns.countplot(x="LandContour",data=train_df)
print(train_df["LandContour"].value_counts())
```

```
Lvl    1046
Bnk      50
HLS      42
Low      30
Name: LandContour, dtype: int64
```

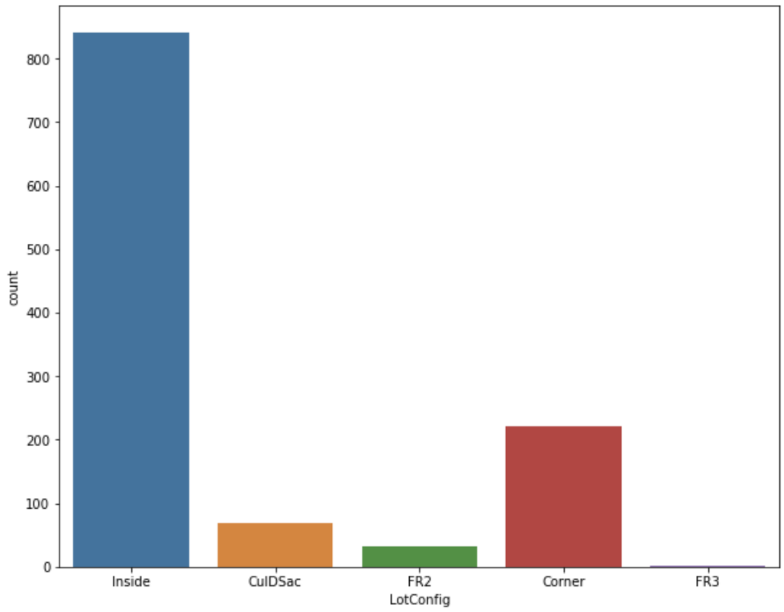


Here we observed maximum number of Lvl LandContour

In [38]:

```
plt.figure(figsize=(10,8))
ax = sns.countplot(x="LotConfig",data=train_df)
print(train_df["LotConfig"].value_counts())
```

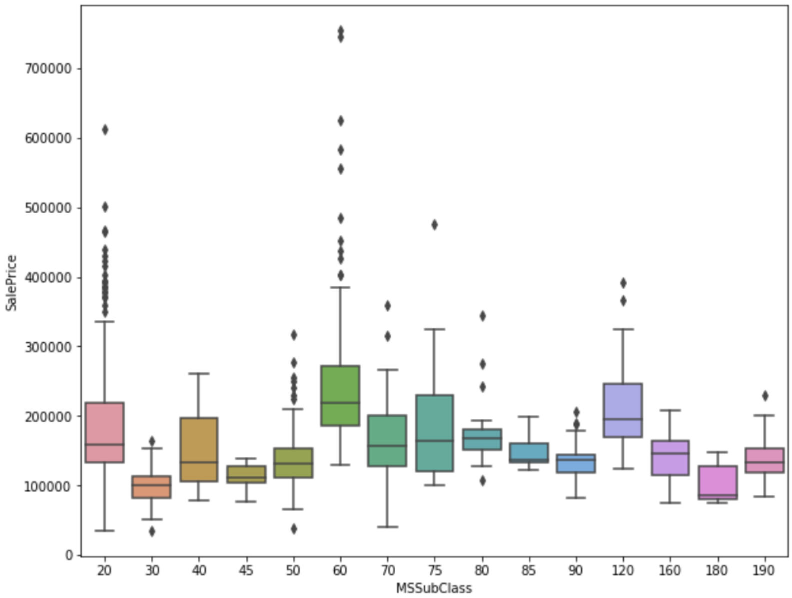
```
Inside     842
Corner     222
CulDSac     69
FR2         33
FR3          2
Name: LotConfig, dtype: int64
```



Maximum numbers is showed in Inside Lotconfig

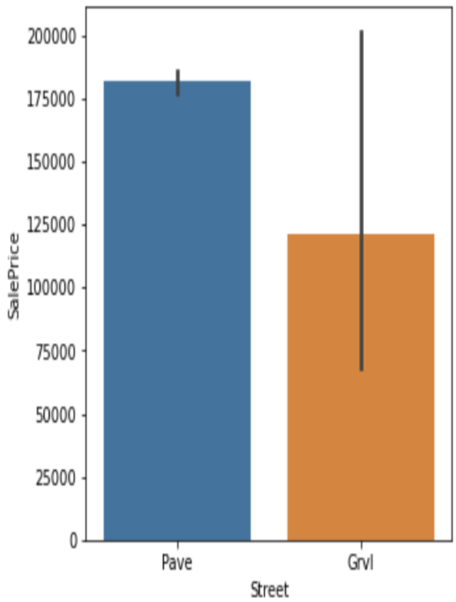
In [39]:

```
plt.figure(figsize=(10,8))
sns.boxplot(x='MSSubClass',y='SalePrice',data=train_df.sort_values('SalePrice',ascending=False))
plt.show()
```



In [40]:

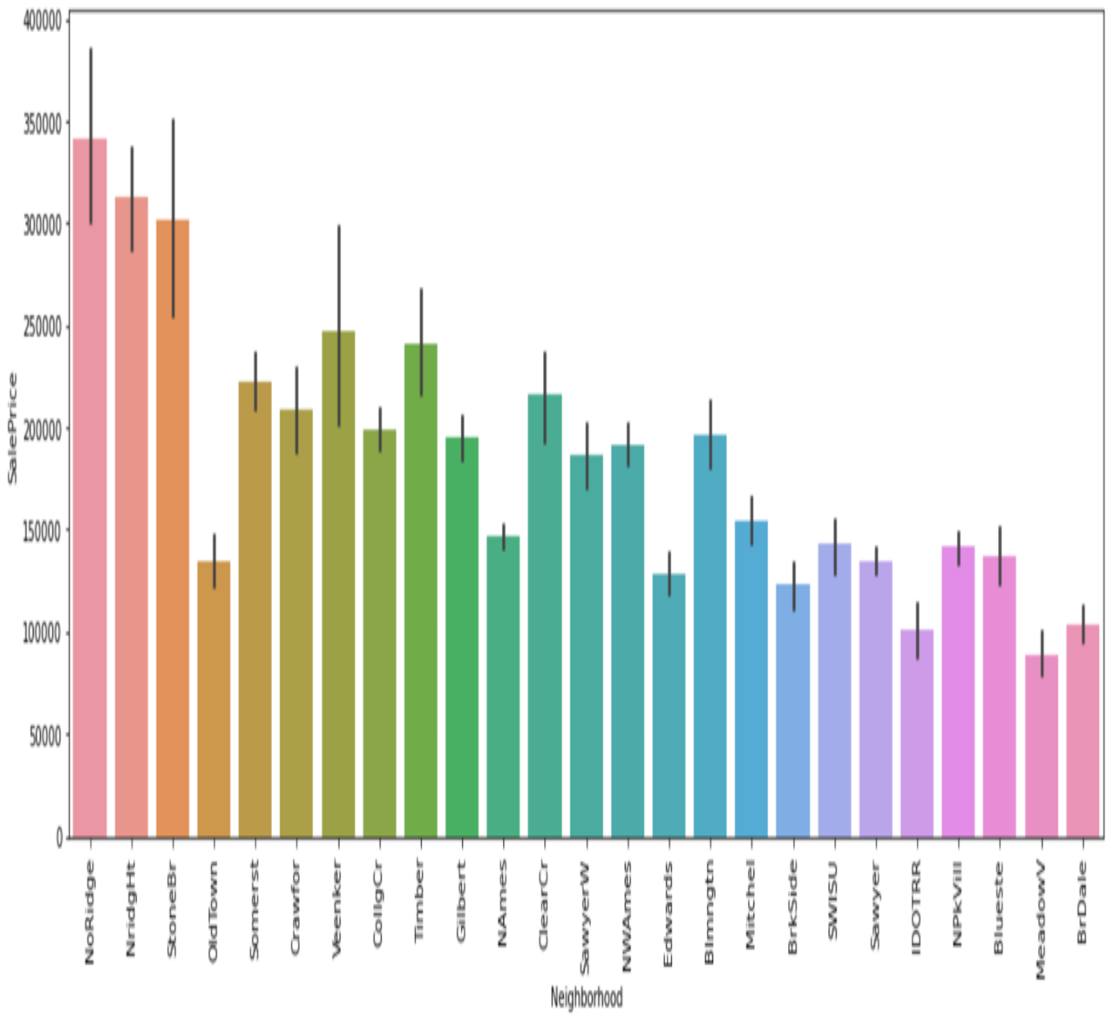
```
plt.figure(figsize=[5,5])
sns.barplot(x='Street', y='SalePrice', data = train_df.sort_values('SalePrice', ascending=False))
plt.show()
```



Here we have observed maximum number of Pave Street

In [46]:

```
plt.figure(figsize=[18,6])
sns.barplot(x='Neighborhood', y='SalePrice', data=train_df.sort_values('SalePrice', ascending=False))
plt.xticks(rotation=90)
plt.show()
```



In this bar plot we observed difference between Saleprice and Neighborhood columns

Interpretation of the Results

From the preprocessing we interpreted that data was improper scaled. From the visualization we interpreted that the target variable SalePrice was highly positively correlated with the columns GrLivArea, YearBuilt, OverallQual, GarageCars, GarageArea.

.

In [107…
```
parameters={'loss' : ['squared_error', 'absolute_error', 'huber', 'quantile'],
            'max_features' : ['auto', 'sqrt', 'log2'],
           'learning_rate':[0.1,.2,.4,.5],
           'criterion' : ['friedman_mse', 'squared_error', 'mse'],
}
```

In [108…
```
from sklearn.model_selection import GridSearchCV
```

In [109…
```
gscv=GridSearchCV(GradientBoostingRegressor(),parameters,cv=5)
gscv.fit(x_train,y_train)
```

Out[109…
```
GridSearchCV(cv=5, estimator=GradientBoostingRegressor(),
             param_grid={'criterion': ['friedman_mse', 'squared_error', 'mse'],
                         'learning_rate': [0.1, 0.2, 0.4, 0.5],
                         'loss': ['squared_error', 'absolute_error', 'huber',
                                  'quantile'],
                         'max_features': ['auto', 'sqrt', 'log2']})
```

In [110…
```
gscv.best_params_
```

Out[110…
```
{'criterion': 'mse',
 'learning_rate': 0.1,
 'loss': 'absolute_error',
 'max_features': 'auto'}
```

In [111…
```
gscv.best_estimator_
```

Out[111…
```
GradientBoostingRegressor(criterion='mse', loss='absolute_error',
                          max_features='auto')
```

In [112…
```
gscvpred1=gscv.best_estimator_.predict(x_test)
gscv.best_estimator_.score(x_train,y_train)
```

Out[112…
```
0.9420947302913055
```

In [113…
```
final_model=GradientBoostingRegressor(max_depth=4)
```

In [114…
```
final_model.fit(x_test,y_test)
pred=final_model.predict(x_test)
final_model.fit(x_train,y_train)
pred1=final_model.predict(x_train)
```

In [115…
```
print("Test Accuracy=",final_model.score(x_test,y_test))
print("Train Accuracy=",final_model.score(x_train,y_train))
```

```
Test Accuracy= 0.9238113053495993
Train Accuracy= 0.9865543876443633
```

CONCLUSION

Key Findings and Conclusions of the Study

Housing Price Prediction here we have tried to show how the house prices vary and what are the factors related to the changing of house prices. The best RMSE score was achieved using the best parameters of Ridge Regressor through GridSearchCV though Lasso Regressor model performed well too.

Learning Outcomes of the Study in respect of Data Science

With the help of different powerful tools of visualization we were able to analyse and interpret different hidden insights about the data. This project has demonstrated the importance of sampling effectively, modelling and predicting data. Through data cleaning we were able to remove unnecessary columns and outliers from our dataset due to which our model would have suffered from overfitting or underfitting.

The few challenges while working on this project where:-

- Improper scaling
- Too many features
- Missing values
- Skewed data due to outliers

There were lot of missing values present in different columns which we imputed on the basis of our understanding. The data was improper scaled so we scaled it to a single scale using sklearns's package StandardScaler.

Limitations of this work and Scope for Future Work

The very nature of this project allows for multiple algorithms to be integrated together as modules and their results can be combined to increase the accuracy of the final result. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project. While we couldn't reach out goal of minimum RMSE in house price prediction without letting the model to overfit, we did end up creating a system that can with enough time and data get very close to that goal. As with any project there is room for improvement here.