



HOUSING PRICE PREDICTION

Submitted by:
Priyanka Saikia

ACKNOWLEDGMENT

I would like to express my deep sense of gratitude to my SME (Subject Matter Expert) **Mr. Shubham Yadav** as well as **Flip Robo Technologies** who gave me the golden opportunity to do this data analysis project on **Housing: Price Prediction**, which also helped me in doing lots of research and I came to know about so many new things.

I am very much thankful to **Dr. Deepika, Trainer (DataTrained)**, for her valuable guidance, keen interest, and encouragement at various stages of my training period which eventually helped me a lot in doing this project.

I have utilized a few external resources that helped me to complete the project which are listed down:

- Real Estate Value Prediction Using Linear Regression - Nehal N Ghosalkar, Sudhir N Dhage
- Affordable Housing in the United States - By Gordon Davis, Jaime Bordenave, Roger Williams, Richard A. Hanson, and Richard Shields - June 12, 2006
- Housing Affordability Literature Review and Affordable Housing Program Audit – Elena Sliogeris, Louise Crabtree, Peter Phibbs and Kate Johnston
- <https://github.com/>
- <https://www.kaggle.com/>
- <https://towardsdatascience.com/>
- <https://www.analyticsvidhya.com/>

INTRODUCTION

Business Problem Framing:

Houses are one of the necessary needs of every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors to the world's economy. It is a very large market and various companies are 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, improve their marketing strategies, and focus on changing trends in house sales and purchases. Predictive modeling, Market mix modeling, 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. It is required to build a model using Machine Learning to predict the actual value of the prospective properties and decide whether to invest in them or not. So we are building a model that helps to determine which variables are important to predict the price of variables & also how do these variables describe the price of the house.

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

Conceptual Background of the Domain Problem:

As we are working on the Housing project dataset we can easily understand that the data belongs to the Housing and Real Estate which will eventually involve several Financial, Costing, Area & Neighbourhood, Statistical, and Technical terms in the dataset. This data belongs to A US-based housing company named Surprise Housing as they have 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. So here we need to understand several words related to the Core Domain.

The domain-related concepts which are useful for a better understanding of the project are. Some relevant details of individual columns are,

1. 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

2. 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

3. LotFrontage: Linear feet of street connected to property

4. LotArea: Lot size in square feet

5. Street: Type of road access to property

Grvl	Gravel
Pave	Paved

6. Alley: Type of alley access to property

Grvl	Gravel
Pave	Paved
NA	No alley access

7. LotShape: General shape of property

Reg	Regular
IR1	Slightly irregular
IR2	Moderately Irregular
IR3	Irregular

8. 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

9. 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

10. 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

11. LandSlope: Slope of property

Gtl	Gentle slope
Mod	Moderate Slope
Sev	Severe Slope

12. 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

13. 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 positive off-site feature
RRNe	Within 200' of East-West Railroad
RR Ae	Adjacent to East-West Railroad

14. 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 positive off-site feature
RRNe	Within 200' of East-West Railroad
RR Ae	Adjacent to East-West Railroad

15. 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

16. HouseStyle: 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

17. 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

18. OverallCond: Rates the overall condition of the house

10	Very Excellent
9	Excellent
8	Very Good
7	Good
6	Above Average
5	Average
4	Below Average
3	Fair
2	Poor
1	Very Poor

19. YearBuilt: Original construction date

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

21. RoofStyle: Type of roof

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

22. 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

23. 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

24. 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

25. MasVnrType: Masonry veneer type

BrkCmn	Brick Common
BrkFace	Brick Face
CBlock	Cinder Block
None	None
Stone	Stone

26. MasVnrArea: Masonry veneer area in square feet

27. ExterQual: Evaluates the quality of the material on the exterior

Ex	Excellent
Gd	Good
TA	Average/Typical
Fa	Fair
Po	Poor

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

Ex	Excellent
Gd	Good
TA	Average/Typical
Fa	Fair
Po	Poor

29. Foundation: Type of foundation

BrkTil	Brick & Tile
CBlock	Cinder Block
PConc	Poured Contrete
Slab	Slab
Stone	Stone
Wood	Wood

30. 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

31. 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

32. 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

33. 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

34. BsmtFinSF1: Type 1 finished square feet

35. 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	Unfinished
NA	No Basement

36. BsmtFinSF2: Type 2 finished square feet

37. BsmtUnfSF: Unfinished square feet of basement area

38. TotalBsmtSF: Total square feet of basement area

39. 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

40. HeatingQC: Heating quality and condition

Ex	Excellent
Gd	Good
TA	Average/Typical
Fa	Fair
Po	Poor

41. CentralAir: Central air conditioning

N	No
Y	Yes

42. 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

43. 1stFlrSF: First Floor square feet

44. 2ndFlrSF: Second floor square feet

45. LowQualFinSF: Low quality finished square feet (all floors)

46. GrLivArea: Above grade (ground) living area square feet

47. BsmtFullBath: Basement full bathrooms

48. BsmtHalfBath: Basement half bathrooms

49. FullBath: Full bathrooms above grade

50. HalfBath: Half baths above grade

51. Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

52. Kitchen: Kitchens above grade

53. KitchenQual: Kitchen quality

Ex	Excellent
Gd	Good
TA	Typical/Average
Fa	Fair
Po	Poor

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

55. 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

56. Fireplaces: Number of fireplaces

57. 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

58. 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

59. GarageYrBlt: Year garage was built

60. GarageFinish: Interior finish of the garage

Fin	Finished
RFn	Rough Finished
Unf	Unfinished
NA	No Garage

61. GarageCars: Size of garage in car capacity

62. GarageArea: Size of garage in square feet

63. GarageQual: Garage quality

Ex	Excellent
Gd	Good
TA	Typical/Average
Fa	Fair
Po	Poor
NA	No Garage

64. GarageCond: Garage condition

Ex	Excellent
Gd	Good
TA	Typical/Average
Fa	Fair
Po	Poor
NA	No Garage

65. PavedDrive: Paved driveway

Y	Paved
P	Partial Pavement
N	Dirt/Gravel

66. WoodDeckSF: Wood deck area in square feet

67. OpenPorchSF: Open porch area in square feet

68. EnclosedPorch: Enclosed porch area in square feet

69. 3SsnPorch: Three season porch area in square feet

70. ScreenPorch: Screen porch area in square feet

71. PoolArea: Pool area in square feet

72. PoolQC: Pool quality

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

73. Fence: Fence quality

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

74. 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

75. MiscVal: \$Value of miscellaneous feature

76. MoSold: Month Sold (MM)

77. YrSold: Year Sold (YYYY)

78. 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

79. 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
Partial
with New Homes)

Sale between family members
Home was not completed when last assessed (associated

Review of Literature:

In this section, we are discussing the reported work carried out in various fields of Housing projects & Real Estate sectors.

Nehal N Ghosalkar et. Al., describes that the real estate market is a standout amongst the most focused regarding pricing and keeps fluctuating. It is one of the prime fields to apply the ideas of machine learning on how to enhance and foresee the costs with high accuracy. Three factors influence the price of a house which include physical conditions, concepts, and location. The current framework includes estimating the price of houses without any expectations of market prices and cost increments. The objective of the paper is the prediction of residential prices for the customers considering their financial plans and needs. By breaking down past market patterns and value ranges, and coming advancements future costs will be anticipated. This examination means to predict house prices in the city with Linear Regression. It will help clients to put resources into a bequest without moving toward a broker.

George Earl et. Al. describe that markets are the central institutions of economies, allowing people to buy and sell goods and services in a manner that potentially makes everyone better off. However, markets can only be formed under certain conditions, and when these conditions are absent, markets may struggle to exist or the conditions may lead to market failures. This is the basic underlying principle of missing or incomplete markets; that is, failure to produce some goods and services despite being needed or wanted. It is well known that microeconomic equilibrium occurs when the demand for goods is equal to the supply. A missing market, therefore, is a sign that the market is out of equilibrium; a situation where markets do not exist or where the equilibrium price is not related to either marginal social benefits or marginal social costs. For decades, the market has been failing to meet the housing needs of its lowest-income residents, and the situation is getting steadily worse. Many people on low incomes cannot afford to buy their own homes, and housing rents have also become increasingly unaffordable in recent years. Therefore, by definition, sustainable housing means that everyone should have the opportunity to live in a decent home at a price they can afford, in a place in which they want to live and work.

Elena Sliogeris et. Al. describe that the incidence of the problem has spread from very low-income through low-income into moderate-income households. There is now a consistent call for housing schemes to retain 'key workers' and 'the working poor' in established areas to ensure access to employment, education, public transport, and other facilities and amenities. Land com has a strategic position within this landscape and there exists a range of current and potential mechanisms land com might utilize to create and maintain a pool of affordable houses. Numerous interrelated factors have driven the loss of affordability, including an increased willingness and capacity to pay for housing due to increased incomes and more accessible lines of credit. Concurrent increases in a population decrease in household size, and increases in house size have further compounded the problem. The role of supply-side

impediments to housing development that contributes to a loss of affordability is strongly contested. The planning processes may have a role to play in addressing affordability concerns. However, the house prices are largely countered by recent market conditions in city areas that have continued to increase in price while house prices in outer areas have stagnated or decreased in value. The landscape of affordability is influenced to a large extent by access to jobs, public transport, and other social amenities. This highlights the need for future housing provision to address employment, transport, and other infrastructure as well as the volume of housing supply.

Motivation for the Problem Undertaken

The prime intention of performing this project is to build a Machine Learning model to predict house prices with the help of other supporting feature attributes. The sample data is provided by our client database. To improve the selection of customers, the client wants some predictions that could help them in further investment and improvement in the selection of customers. House Price Index (HPI) is commonly used to estimate the changes in housing prices. Since housing price is strongly correlated to other factors such as location, area, etc, it requires other information apart from HPI to predict individual housing prices.

There has been a considerably large number of papers adopting traditional machine learning approaches to predict housing prices accurately, but they rarely concern themselves with the performance of individual models and neglect the less popular yet complex models.

As a result, to explore various impacts of features on prediction methods, this paper will apply both traditional and advanced machine learning approaches to investigate the difference among several advanced models. This analysis will also comprehensively validate multiple techniques in model implementation on regression and provide an optimistic result for housing price prediction.

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

Analytical Problem Framing

- **Mathematical/ Analytical Modeling of the Problem:**

The provided dataset is in CSV format. We will begin with loading the dataset and reading the dataset from the CSV file using the `read_csv()` function from the Pandas Python package. Next, we will perform Non-Graphical Exploratory Data Analysis (EDA) such as checking the data types and missing values using `pandas.info()` function. Then, we will get statistical information about the numeric columns in

our dataset using `pandas.DataFrame.describe()` method. We want to know the mean, the standard deviation, the minimum, the maximum, and the 50th percentile (the median) for each numeric column in the dataset. After that, we will move on to perform graphical EDA to get more insights from our dataset and how the feature attribute affects the target attribute.

Next, we will perform data pre-processing to treat the missing values, dropping columns not helpful for our model prediction, removing outliers if there is, removing skewness, etc. Further, we will build a full pipeline for our model prediction.

Since we have to predict house prices in this dataset therefore we will be building and training a few regression-based models to predict the price of the houses.

- **Data Sources and their formats:**

The dataset is being provided by Flip Robo and is in the format of CSV (Comma Separated Values). Let's load it and start the analysis.

Importing Libraries

```
: # linear algebra
import numpy as np
from numpy import mean

import warnings
warnings.simplefilter("ignore")
warnings.filterwarnings('ignore')

# data processing
import pandas as pd
from pandas_profiling import ProfileReport

# data visualization
import seaborn as sns
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style
import matplotlib.gridspec as gridspec
from matplotlib.gridspec import GridSpec
from scipy.stats import norm
import plotly.express as px
import klib
import scikitplot as skplt
```

Loading the dataset

```
df_train=pd.read_csv("train.csv")
```

```
#checking the first 5 entries of the dataset
df_train.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2
0	127	120	RL	NaN	4928	Pave	NaN	IR1		Lvl	AllPub	Inside	Gtl	NPkVill	Norm
1	889	20	RL	95.0	15865	Pave	NaN	IR1		Lvl	AllPub	Inside	Mod	NAmes	Norm
2	793	60	RL	92.0	9920	Pave	NaN	IR1		Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm
3	110	20	RL	105.0	11751	Pave	NaN	IR1		Lvl	AllPub	Inside	Gtl	NWAmes	Norm
4	422	20	RL	NaN	16635	Pave	NaN	IR1		Lvl	AllPub	FR2	Gtl	NWAmes	Norm

The dimension of data is 1168 rows and 81 columns.

```
#checking the dimension of the dataset
df_train.shape

(1168, 81)
```

Two data sets are given. One is training data and one is testing data.

1) Train file will be used for training the model, i.e., the model will learn from this file. It contains all the independent variables and the target variable. Size of training set: 1168 records.

2) Test file contains all the independent variables, but not the target variable. We will apply the model to predict the target variable for the test data. Size of testing set: 292 records.

The data type information:

```
#checking the data type information
df_train.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 81 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   Alley                  77 non-null     object  
 7   LotShape                1168 non-null   object  
 8   LandContour            1168 non-null   object  
 9   Utilities               1168 non-null   object  
10  LotConfig              1168 non-null   object  
11  LandSlope              1168 non-null   object  
12  Neighborhood            1168 non-null   object  
13  Condition1             1168 non-null   object  
14  Condition2             1168 non-null   object  
15  BldgType               1168 non-null   object  
16  HouseStyle             1168 non-null   object  
17  OverallQual            1168 non-null   int64  
18  OverallCond            1168 non-null   int64  
19  YearBuilt              1168 non-null   int64  
20  YearRemodAdd           1168 non-null   int64  
21  ...                   ...              ...      ...
```

Observations made from the data type information-

There are 38 numerical columns and 43 categorical columns

Some columns should be an object such as mssubclass, overallqual, etc. but are in numerical format.

Checking missing/null values present in the dataset

```
: #checking for null values
df_train.isna().sum()
```

```
: Id                0
  MSSubClass        0
  MSZoning          0
  LotFrontage      214
  LotArea          0
  Street           0
  Alley           1091
  LotShape         0
  LandContour      0
  Utilities        0
  LotConfig        0
  LandSlope        0
  Neighborhood     0
  Condition1       0
  Condition2       0
  BldgType         0
  HouseStyle       0
  OverallQual      0
  OverallCond      0
  YearBuilt        0
  YearRemodAdd     0
```

There are many missing values that we have to handle before model training. I am treating some columns that have numerous missing values by replacing them with None for categorical columns, and numerical columns, I am replacing them with 0 only when if they have a lot of missing values.

And for the columns having fewer missing values, I am replacing them with mode for categorical columns, and for numerical I am replacing them with mean.

Statistical summary of the data:

```
df_train.describe().T.style.background_gradient(cmap='flag')
```

	count	mean	std	min	25%	50%	75%	max
Id	1168.000000	724.136130	416.159877	1.000000	360.500000	714.500000	1079.500000	1460.000000
MSSubClass	1168.000000	56.767979	41.940650	20.000000	20.000000	50.000000	70.000000	190.000000
LotFrontage	954.000000	70.988470	24.828750	21.000000	60.000000	70.000000	80.000000	313.000000
LotArea	1168.000000	10484.749144	8957.442311	1300.000000	7621.500000	9522.500000	11515.500000	164660.000000
OverallQual	1168.000000	6.104452	1.390153	1.000000	5.000000	6.000000	7.000000	10.000000
OverallCond	1168.000000	5.595890	1.124343	1.000000	5.000000	5.000000	6.000000	9.000000
YearBuilt	1168.000000	1970.930651	30.145255	1875.000000	1954.000000	1972.000000	2000.000000	2010.000000
YearRemodAdd	1168.000000	1984.758562	20.785185	1950.000000	1966.000000	1993.000000	2004.000000	2010.000000
MasVnrArea	1161.000000	102.310078	182.595606	0.000000	0.000000	0.000000	160.000000	1600.000000
BsmtFinSF1	1168.000000	444.726027	462.664785	0.000000	0.000000	385.500000	714.500000	5644.000000
BsmtFinSF2	1168.000000	46.647260	163.520016	0.000000	0.000000	0.000000	0.000000	1474.000000
BsmtUnfSF	1168.000000	569.721747	449.375525	0.000000	216.000000	474.000000	816.000000	2336.000000

From this summary, we can see all the numeric value details and also see the spread of the data like the target "SalePrice" has a min value of 34900 and a max value of 755000. Also, the count is not the same for all the columns indicating that there are missing values present in the dataset.

- **Data Pre-processing Done:**

Data pre-processing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for building and training Machine Learning models. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. Therefore, it is extremely important that we pre-process our data before feeding it into our model as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn. The steps that were performed are as follows-

- **Filling /Treating Missing values:** We have defined a function to treat the missing values present. I am treating some columns that have numerous missing values by replacing them with None for categorical columns, and numerical columns, I am replacing them with 0 only when if they have a lot of missing values. And for the columns having fewer missing values, I am replacing them with mode for categorical columns, and for numerical I am replacing them with mean.

```
class Preprocess_Missingvalues:
    def __init__(self):
        pass

    def fit(self,X,y=None):
        return self

    def transform(self,X):
        X['LotFrontage']=X.groupby('Neighborhood')['LotFrontage'].transform(lambda x:x.fillna(x.mean()))
        X['Electrical']=X['Electrical'].fillna(X['Electrical'].mode()[0])
        for col in ['FireplaceQu','GarageType','GarageFinish','BsmtFinType2','PoolQC','Fence','MiscFeature','GarageQual','GarageCond',
                    'BsmtFinType1','BsmtFinType2','MasVnrType','Alley']:
            X[col]=X[col].fillna('None')
        for col in ['GarageYrBlt','MasVnrArea']:
            X[col]=X[col].fillna(0)

        X=X.drop(['Utilities','Id'],axis=1)
        return X
```

- **Feature Engineering:** we have defined a class to create more features so our machine learning models performs well. Feature engineering is one of the most important method in ML.

```
class Feature_Engineering:
    def __init__(self):
        pass
    def fit(self,X,y=None):
        return self
    def transform(self,X):
        X['SqFtPerRoom']=X['GrLivArea']/X['TotRmsAbvGrd']+(X['TotRmsAbvGrd']+
                                                            X['FullBath']+
                                                            X['HalfBath']+
                                                            X['KitchenAbvGr'])

        X['Total_Home_Quality']=X['OverallQual']+X['OverallCond']
        X['Total_Bathrooms']=(X['FullBath']+(0.5*X['HalfBath'])+
                              X['BsmtFullBath']+(0.5*X['BsmtHalfBath']))
        X['HighQualSF']=X['1stFlrSF']+X['2ndFlrSF']
        return X
```

- **Feature Transformation:** We are using LabelEncode to encode the **categorical features**

```
class Encoding:
    def __init__(self):
        pass
    def fit(self,X,y=None):
        return self
    def transform(self,X):
        le=LabelEncoder()
        cols= ('MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour',
               'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
               'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st',
               'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation',
               'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
               'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual',
               'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual',
               'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature',
               'SaleType', 'SaleCondition')
        for c in cols:
            X[c]=le.fit_transform(X[c])
        return X
```

- **Handling Multi-collinearity:** With the help of heatmap & correlation bar graph we were able to understand the feature vs target relativity and insights on multi-collinearity amongst the feature columns.

```
#defining a function to check multicollinearity
def correlation(dataset,threshold):
    col_corr=set()
    corr_matrix=dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i,j]) > threshold:
                colname=corr_matrix.columns[i]
                print(corr_matrix.iloc[i,j],corr_matrix.columns[i],corr_matrix.columns[j])
                col_corr.add(colname)
    return col_corr
```

- **Removing Outliers:** with the help of percentile method as we can replace the outliers with percentile value and don't lose any data.

```
class remove_outliers:
    def __init__(self):
        pass
    def fit(self,X,y=None):
        return self
    def transform(self,X):
        for col in X.columns:
            percentile=X[col].quantile([0.01,0.98]).values
            X[col][X[col]<=percentile[0]]=percentile[0]
            X[col][X[col]>=percentile[1]]=percentile[1]
        return X
```

- **Removing Skewness** of continuous data (with threshold value -1 to +1) using power_transform function from sklearn.preprocessing.

```
class skewness_remove:
    def __init__(self,skew=0.5):
        self.skew=skew

    def fit(self,X,y=None):
        return self

    def transform(self,X):
        x=X.copy()
        X_num=X.select_dtypes(exclude='object')
        skewness=X_num.apply(lambda x:x.skew())
        skewness_col=skewness[abs(skewness)>=self.skew].index
        X[skewness_col]=power_transform(X[skewness_col])
        X=pd.DataFrame(X,columns=x.columns)
        X = pd.get_dummies(X)
        return X
```

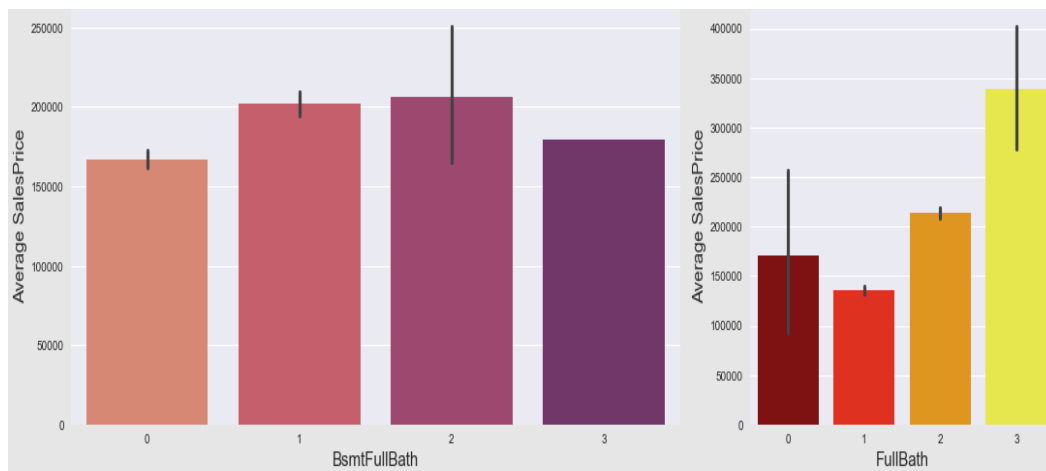
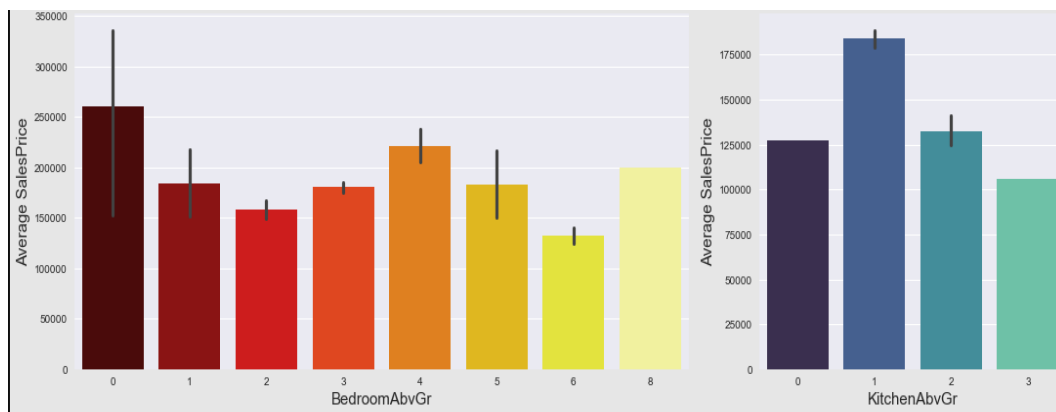
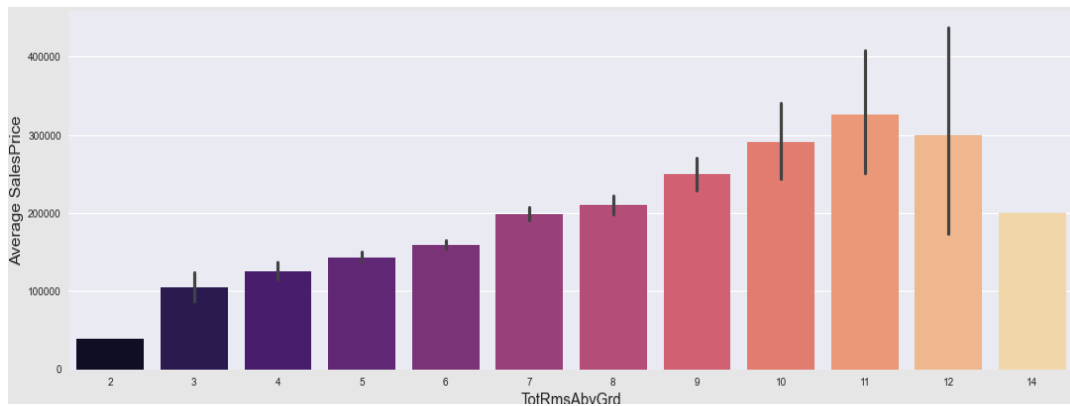
- **Feature Scaling:** we are using StandardScaler to scale all high values to same scale

```
class Feature_Scaling:
    def __init__(self):
        pass
    def fit(self,X,y=None):
        return self
    def transform(self,X):
        x=X.copy()
        scale=StandardScaler()
        X=scale.fit_transform(X)
        X=pd.DataFrame(X,columns=x.columns)
        return X
```

- **Data Inputs- Logic- Output Relationships:**

The given dataset has 78 input variables after pre-processing of the data that needs to be provided to the logic to get the output i.e. SalePrice. The below EDA describes the relationship behind the data input, its format, the logic in between, and the output. For EDA we have used different types of plots such as count plot, barplot, strip plot, boxplot, heatmap, etc.

BarPlots:

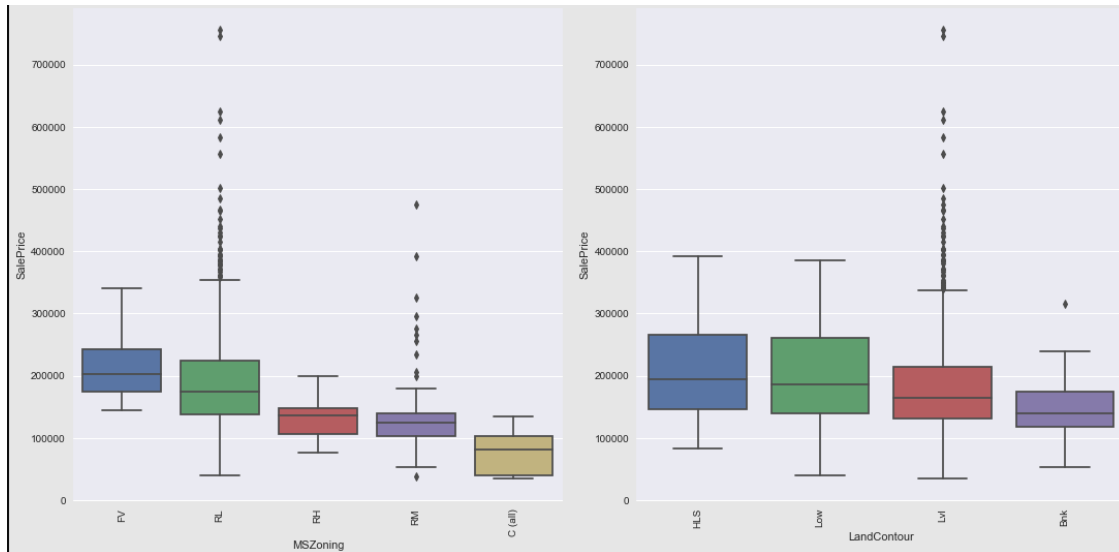


Observations:

1. From TotRmsAbvGrd vs SalePrice plot, we see that as total rooms Above Ground are increasing, the average price is also increasing until the 11th room and the price starts to decrease.
2. From the Bedroom Above Ground Vs Saleprice plot, we can see that for the 0, 4, 8 Bedroom number, the price is high and the price is very less for 6 and 2 bedroom numbers.
3. From the Kitchen above Ground Vs saleprice plot, we can see that as the no. of the kitchen is increasing, the price is decreasing indicating mostly people take one kitchen house only.

4. In Basement full bathroom & half bathrooms vs saleprice plot, we see that as the bathroom size increases, the price is also increasing.

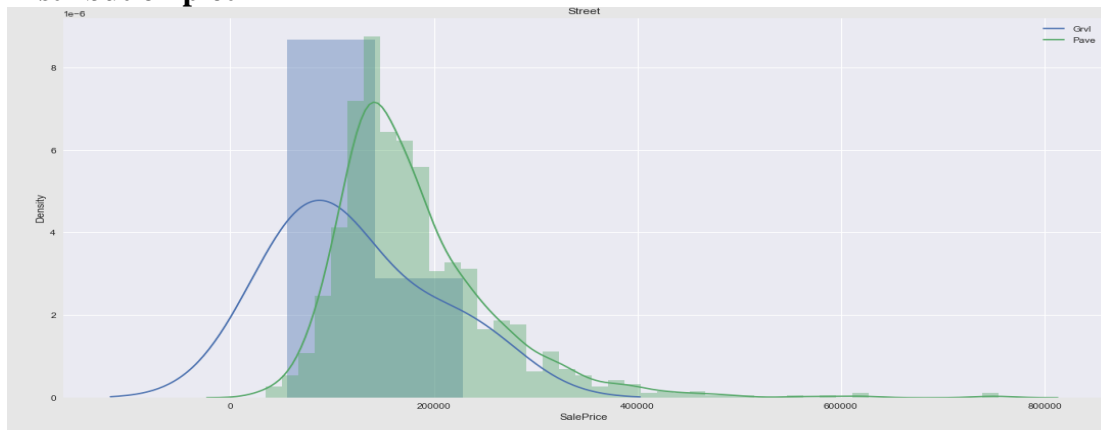
Boxplots:



Observations:

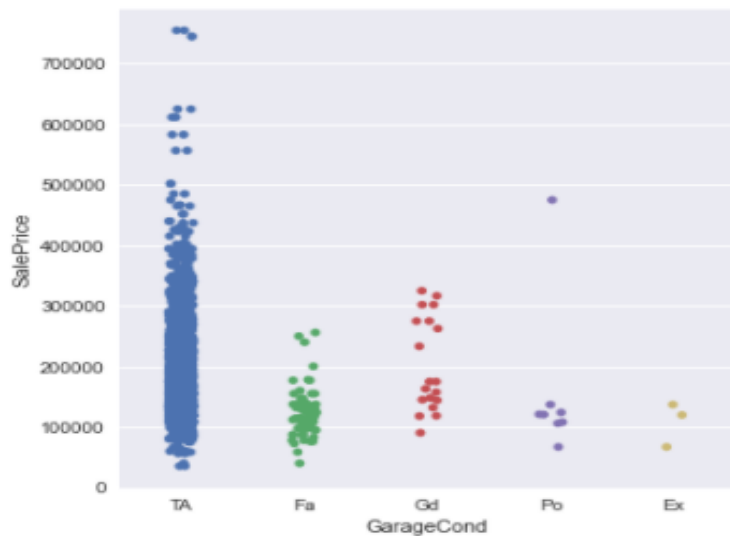
1. From MSZoning vs Saleprice plot, we see that the Floating Village Residential zone (FV) is having a higher sale price of around 2 lakhs While Commercial Zones are having the lowest sale prices and we see a variation in sale prices in different zones which makes it an important feature for our prediction.
2. From LandCountour vs saleprice plot, we see that if the house has a significant slope on both sides (HLS) or has depression (Low), sale prices are more. For nearly flat/level and banked slopes, the Sale price is less.

Distribution plot:



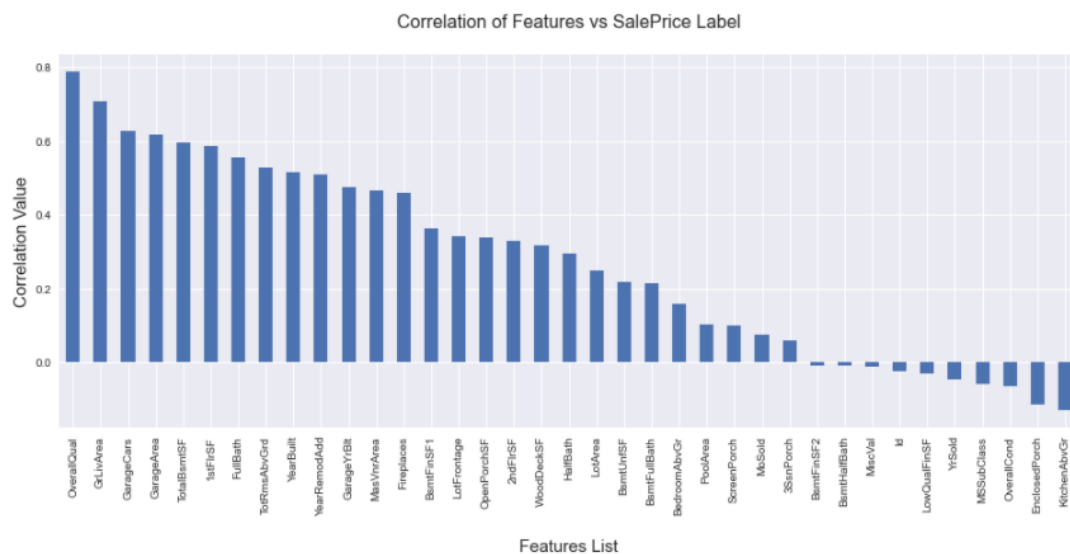
Observation: We see that For flat stones or bricks access streets, the Price is higher and the majority of houses with gravel streets have a price around 2 lakhs but as there is very little data for gravel street type this feature is not that important for prediction.

Stripplot:



Observation: From Garagecond vs saleprice plot, we see that houses with typical/average condition garages account for higher sale prices than garages with excellent or poor condition.

Checking Correlation of features with target attribute:



Remarks: It gives us an insight on positive and negative correlated column details.

These are some of the Graphical EDA I did to check how the features affect the target attribute. For detailed EDA, please go through my Jupyter notebook file in my GitHub.

Hardware and Software Requirements and Tools Used:

➤ Hardware Used:

RAM: 8 GB

CPU: AMD A8 Quad Core 2.2 Ghz

GPU: AMD Redon R5 Graphics

➤ Software Tools used:

Programming language: Python 3.0

Distribution: Anaconda Navigator

Browser-based language shell: Jupyter Notebook

➤ Libraries/Packages Used:

- Pandas
- Numpy
- Matplotlib
- Seaborn
- Scipy.stats
- Sklearn

Model/s Development and Evaluation

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

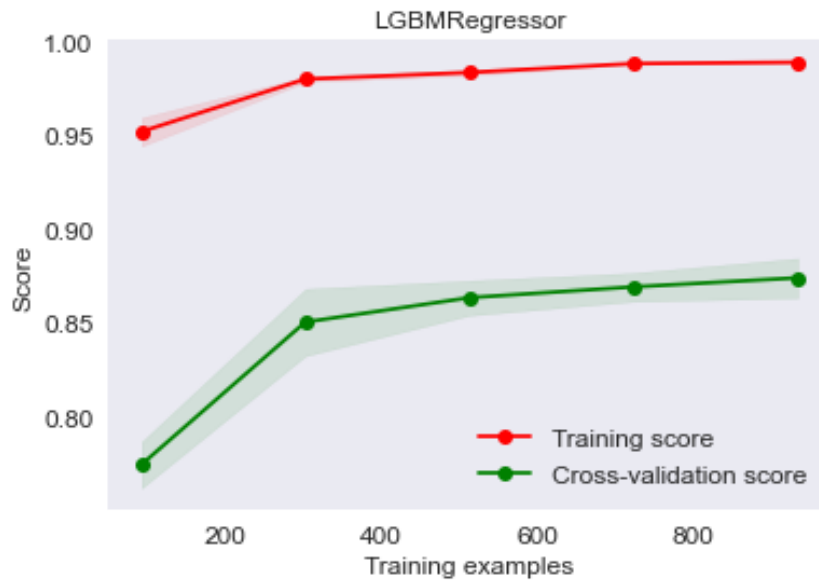
Both statistical and analytical approaches have been used to solve the problem which mainly includes the pre-processing of the data and EDA to check the correlation of independent and dependent features. Also, before building the model, I made sure that the input data is cleaned and scaled before it was fed into the machine learning models.

For this project, we need to predict the target attribute “SalePrice” o which is continuous output so this is a Regression problem. We have used various Regression algorithms and tested them for the prediction.

By doing various evaluations I have selected LightGBM Regressor as the best suitable algorithm for our final model as it is giving a good r2-score and good performance metrics. Other regression algorithms are also showing good accuracy but some are over-fitting and some are under-fitting the results which may be because of fewer amounts of data.

The Learning curve for LGBMRegressor-

MODEL PERFORMANCE CURVE



To get good performance as well as accuracy and to check my model from over-fitting and under-fitting I have made use of the K-Fold cross-validation and then hyperparameter tuned the final model.

Once I was able to get my desired final model I ensured to save that model before I loaded the testing data to analyze it and finally obtain the predicted sale price values.

- **Testing of Identified Approaches (Algorithms):**

The algorithms used are as follows:

- ExtratreesRegressor
- RandomForest
- DecisionTree
- LightGBM
- LinearRegression
- Lasso

- **Run and Evaluate selected models:**

A total of 6 algorithms has been used on the dataset for training testing purpose. To perform training and testing operation(s) following functions has been defined for which codes are as follows:

Splitting data for model building using train test split:

```
: X_train,X_test,y_train,y_test=train_test_split(X_new,np.log(y),test_size=0.3,random_state=42)
```

Training Multiple Models:

```
: models={
    "ExtraTreesRegressor":ExtraTreesRegressor(),
    "RandomForestRegressor":RandomForestRegressor(),
    "LinearRegression":LinearRegression(),
    "DecisionTreeRegressor":DecisionTreeRegressor(),
    "Lasso":Lasso(),
    "LGBMRegressor":LGBMRegressor()
}
```

Finding the best model:

```
CVS=[]
R2=[]
MSE=[]
MAE=[]
RMSE=[]
NAME=[]
kf=KFold(n_splits=5,shuffle=True)
for name, model in models.items():
    print("*****",name,"*****")
    NAME.append(name)
    model.fit(X_train,y_train)
    y_pred=model.predict(X_test)
    mse=mean_squared_error(y_test,y_pred)
    MSE.append(mse)
    print("MEAN SQUARED ERROR",mse)
    mae=mean_absolute_error(y_test,y_pred)
    MAE.append(mae)
    print('\n')
    print("MEAN ABSOLUTE ERROR",mae)
    cvs=cross_val_score(model,X_new,np.log(y),scoring='r2',cv=kf).mean()
    CVS.append(cvs)
    print('\n')
    print("CVS_SCORE",cvs)
    r2=r2_score(y_test,y_pred)
    R2.append(r2)
    print('\n')
    print("R2_SCORE",r2)
    rmse=np.sqrt(mse)
    RMSE.append(rmse)
    print('\n')
    print("RMSE",rmse)
    print('\n')
    print('MODEL PERFORMANCE CURVE')
    skplt.estimators.plot_learning_curve(model,X_new,np.log(y),cv=kf,scoring='r2',title=name,text_fontsize='large')
    plt.show()
```

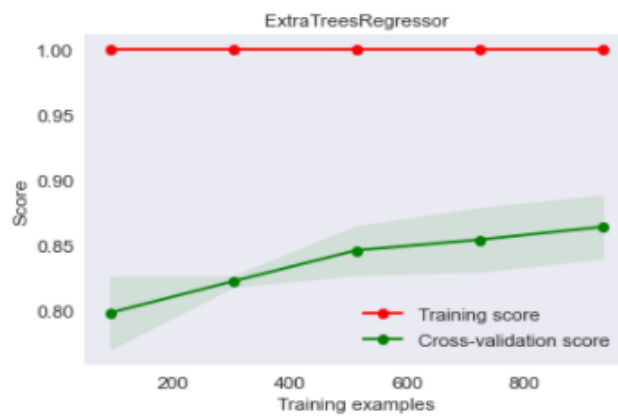
Evaluation metrics of the models:

models_result

	NAME	Cross_Val_Score	R2_score	Mean_squared_error	Mean_Absolute_Error	RMSE
0	ExtraTreesRegressor	0.864158	0.846082	0.025243	0.106210	0.158881
1	RandomForestRegressor	0.856580	0.843025	0.025745	0.108678	0.160451
2	LinearRegression	0.874925	0.870077	0.021308	0.094128	0.145972
3	DecisionTreeRegressor	0.723774	0.690034	0.050836	0.159130	0.225468
4	Lasso	-0.008094	-0.001338	0.164224	0.319727	0.405246
5	LGBMRegressor	0.874619	0.867441	0.021740	0.096217	0.147446

Learning Curve of all Models:

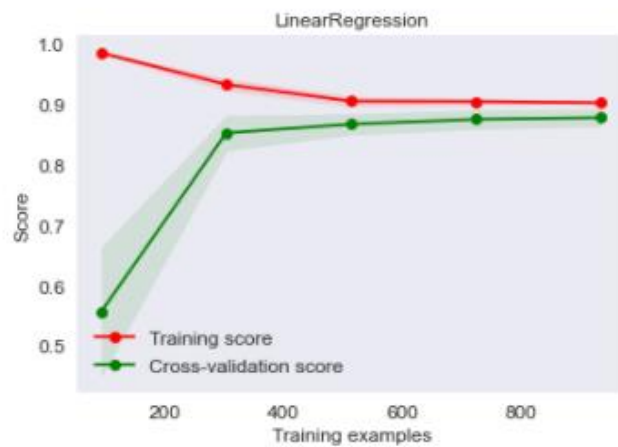
MODEL PERFORMANCE CURVE



MODEL PERFORMANCE CURVE



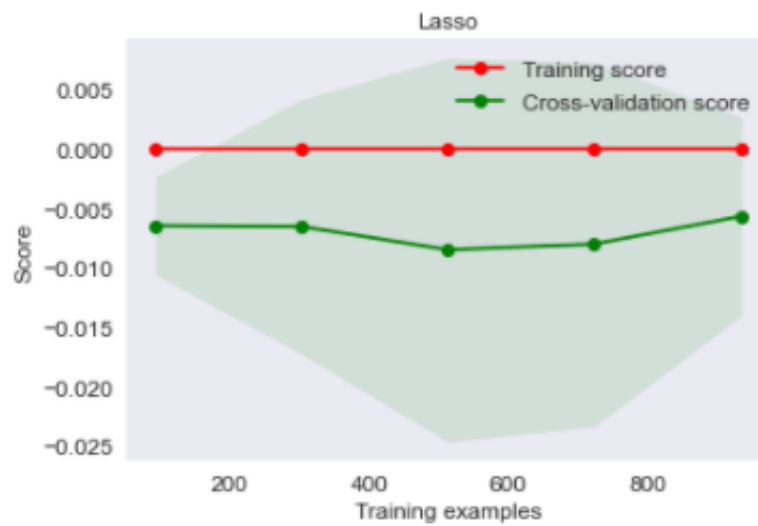
MODEL PERFORMANCE CURVE



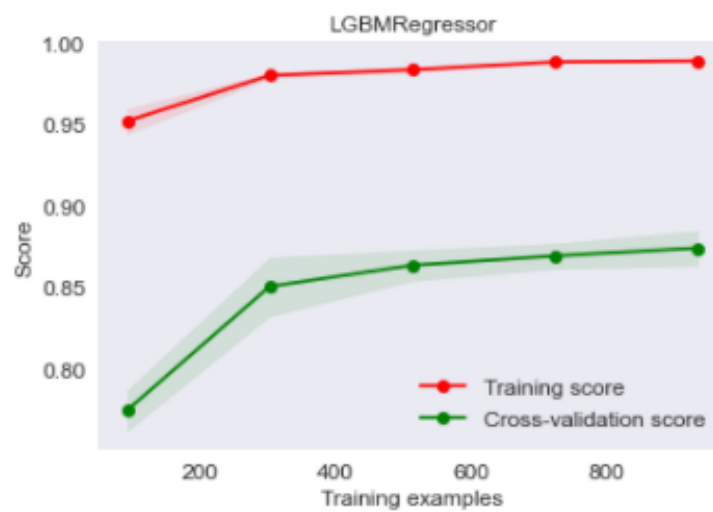
MODEL PERFORMANCE CURVE



MODEL PERFORMANCE CURVE



MODEL PERFORMANCE CURVE



Remarks: LinearRegression model is showing a good result but when we see its Learning curve we observe that it is underfitting so We are going to proceed with LGBMRegressor as our best model because it is performing well and also have a good Learning curve.

Key Metrics for success in solving problem under consideration:

To find out the best performing model following metrics are used:

1. R2 Score: It is used to check the model performance score between 0.0 to 1.0
2. Mean Squared Error: The mean squared error of a model concerning a test set is the mean of the squared prediction errors over all instances in the test set.
3. Root Mean Squared Error: It is the root of mean squared error.
4. Mean Absolute Error: It is the average of absolute errors for a group of predictions and observations as a measurement of the magnitude of errors for the entire group.

Hyperparameter Tuning of the best model :

Listing down the chosen parameters above after selecting LGBMRegressor as our best model:

```
param={
    'boosting_type': ['gbdt','dart'],
    'n_estimators':[100,200,500,600],
    'importance_type': ['mse','mae'],
    'max_depth': [-1,1,2,3,4,5]
}
```

Using the Grid Search CV method for hyperparameter tuning of the best model:

```
: Grid=GridSearchCV(estimator=LGBM,scoring='r2',param_grid=param,n_jobs=-1,cv=kf)
```

Remarks: Using the Grid Search CV method for hyperparameter tuning of the best model.

```
: Grid.fit(X_new,np.log(y))  
: GridSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=True),  
               estimator=LGBMRegressor(), n_jobs=-1,  
               param_grid={'boosting_type': ['gbdt', 'dart'],  
                           'importance_type': ['mse', 'mae'],  
                           'max_depth': [-1, 1, 2, 3, 4, 5],  
                           'n_estimators': [100, 200, 500, 600]},  
               scoring='r2')
```

Remarks: We have trained the Grid Search CV with the list of parameters we think it should check for best possible outcomes.

```
: Grid.best_params_  
: {'boosting_type': 'gbdt',  
   'importance_type': 'mse',  
   'max_depth': 2,  
   'n_estimators': 600}
```

Remarks: In the above cell, the Grid Search CV has provided the best parameters list out of all the combinations it used to train the model.

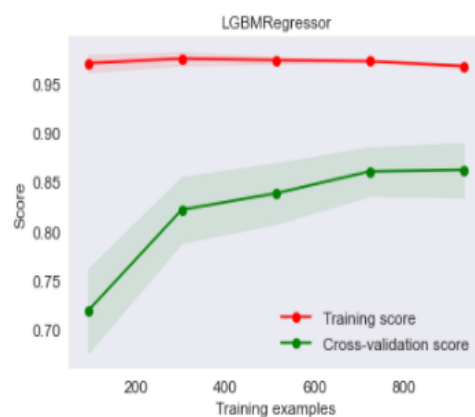
```
: Grid.best_score_  
: 0.8843869306000196
```

After successfully incorporating the HyperParameter Tuning on the final Model, we have received the accuracy score of 88.43%.

Final Model Learning Curve:

```
print('MODEL PERFORMANCE CURVE')  
skplot.estimators.plot_learning_curve(lgb,X_new,y,cv=kf,scoring='r2',title='LGBMRegressor',text_fontsize='large')  
plt.show()
```

MODEL PERFORMANCE CURVE



Final Model metrics:

```
print("Mean Squared Error",mean_squared_error(y_test,Y_Pred))
print("Mean Absolute Error",mean_absolute_error(y_test,Y_Pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,Y_Pred)))
print("R2 Score",r2_score(y_test,Y_Pred))
```

```
Mean Squared Error 0.0203794875635031
Mean Absolute Error 0.09366757992141704
RMSE 0.14275674262010568
R2 Score 0.8757385668737673
```

- **Interpretation of the Results:**

Visualizations: It helped to understand the correlation between independent and dependent features. Also, helped with feature importance and checking for multicollinearity issues. Detected outliers/skewness with the help of boxplot and distribution plot. I got to know the count of a particular category for each feature by using a count plot and most importantly with predicted target value distribution.

Pre-processing: Basically, before building the model the dataset should be cleaned and scaled. As we have mentioned above in the pre-processing steps where all the important features are present in the dataset and ready for model building.

Model Creation: After performing the train test split, we have x_train, x_test, y_train & y_test, which are required to build Machine learning models. I have built multiple regression models to get the best R2 score, MSE, RMSE & MAE out of all the models.

CONCLUSION

- **Key Findings and Conclusions of the Study:**

- The LightGBMRegressor model is working with a well R2 score of 86.74.
- The cross_val_score of the model is 87.46 which again shows that the cross_val_score is better as compared to other models in the table.
- All these points prove that the LightGBMRegressor model is working best and can be considered as the finalized model.

- **Learning Outcomes of the Study in respect of Data Science**

1. Price Prediction modeling – This allows predicting the prices of houses & how they are varying in nature considering the different factors affecting the prices in real-time scenarios.

2. Prediction of Sale Price – This helps to predict the future revenues based on inputs from the past and different types of factors related to real estate & property-related cases. This is best done using predictive data analytics to calculate the future values of houses. This helps in segregating houses, identifying the ones with high future value, and investing more resources in them.

3. **Deployment of ML models** – The Machine learning models can also predict the houses depending upon the needs of the buyers and recommend them, so customers can make final decisions as per the needs.

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

Current model is limited to housing prediction data but this can further be improved for other sectors of property price prediction by training the model accordingly. The overall score can also be improved further by training the model with more specific data.