HOUSING: PRICE PREDICTION

Submitted By
Amruta Shah

Problem Statement:

Houses are one of the necessary needs of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company.

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

The company is looking at prospective properties to buy houses to enter the market. 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. For this company wants to know:

• Which variables are important to predict the price of variable?

• How do these variables describe the price of the house?

Business Goal:

You are required to build model 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.

Technical Requirements:

• Data contains 1460 entries each having 81 variables.
• Data contains Null values. You need to treat them using the domain knowledge and your own understanding.
• Extensive EDA has to be performed to gain relationships of important variable and price.
• Data contains numerical as well as categorical variable. You need to handle them accordingly.
• You have to build Machine Learning models, apply regularization and determine the optimal values of Hyper Parameters. If any.
• You need to find important features which affect the price positively or negatively.
• Two datasets are being provided to you (test.csv, train.csv). You will train on train.csv dataset and predict on
test.csv file.
The "Data file.csv" and "Data description.txt" are enclosed with this file.

Dataset 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

LandSlope: 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

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

OverallQual: Rates the overall material and finish of the house

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

OverallCond: Rates the overall condition of the house

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

YearBuilt: Original construction date

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

RoofStyle: Type of roof

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

RoofMatl: Roof material

ClyTile Clay or Tile
CompShg Standard (Composite) Shingle
Membran Membrane
Metal Metal
Roll Roll
Tar&Grv Gravel & Tar
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
GdGood
TAAverage/Typical
Fa Fair
Po Poor

CentralAir: Central air conditioning
N No
Y Yes

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

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

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

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality
Ex Excellent
GdGood
TATypical/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
GdGood - Masonry Fireplace in main level
TAAverage - 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
GdGood
TATypical/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)

Methodology:

The steps followed in this work, right from the dataset preparation to obtaining results are represented in Fig.

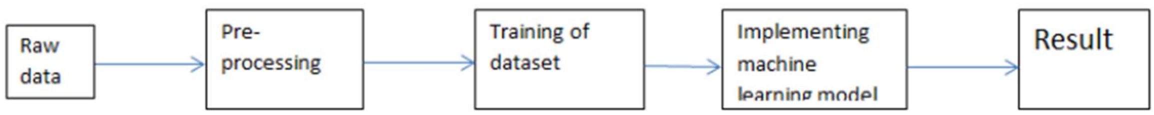


Fig1: Steps followed for obtaining results

Data Analysis:

In this project, we have a dataset which has the details of the prospective properties to buy houses to enter the market.

The given dataset contains 1460 rows × 81 columns. The column names like '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',etc.

EDA (Exploratory Data Analysis):

We should first perform an EDA as it will connect us with the dataset at an emotional level and yes, of course, will help in building good hypothesis function. EDA is a very crucial step. It gives us a glimpse of what our data set is all about, its uniqueness, its anomalies and finally it summarizes the main characteristics of the dataset for us. In order to perform EDA, we will require the following python packages.

Import libraries:

Let's use collected data set to solve the problem. For that we need to import some necessary python libraries.

```
import pandas as pd        # for data manipulation
import numpy as np         # for mathematical calculations
import seaborn as sns      # for data visualization

import matplotlib.pyplot as plt  #for graphical analysis
%matplotlib inline

from scipy.stats import zscore # to remove outliers

from sklearn.preprocessing import StandardScaler  # for normalize the model
from sklearn.preprocessing import LabelEncoder  # to convert object into int


from sklearn.model_selection import train_test_split  # for train and test model

import warnings                          # to ignore any warnings
warnings.filterwarnings("ignore")

from sklearn import metrics  # for model evaluation
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, accuracy_score, confusion_matrix, classificat
```

Once we have imported the packages successfully, we will move on to importing our dataset.

Load Dataset:

This collected data is in the .csv form so we need to use Pandas. read method to read the data.

```
data=pd.read_csv('train.csv')  # read the data
data
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

1168 rows × 81 columns

Dimensions of Dataset:

The dataset has been successfully imported. Let's have a look at the dataset. head () gives us a glimpse of the dataset. It can be considered similar to select * from database table limit 5 in SQL. Let's go ahead and explore a little bit more about the different fields in the dataset. info () gives uses all the relevant information on the dataset. If your dataset has more numerical variables, consider using describe () too to summarize data along mean, median, standard variance, variance, unique values, frequency etc. isna (). sum () gives us sum of null values are present in the dataset. See below screenshot.

```
(1168, 81)
------------------------------------------------------------
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      214
LotArea            0
                ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 81 columns):
```

```
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  RoofStyle      1168 non-null   object 
 22  RoofMatl       1168 non-null   object 
 23  Exterior1st    1168 non-null   object 
 24  Exterior2nd    1168 non-null   object 
 25  MasVnrType     1161 non-null   object 
 26  MasVnrArea     1161 non-null   float64
 27  ExterQual      1168 non-null   object 
```

The number of null values present is as above.

Dataset has object as well as numeric types. Object type in pandas is similar to strings. Now let’s try to classify these columns as Categorical, Ordinal or Numerical/Continuous.
In this our dataset has we have all **Ordinal Variables & Numerical or Continuous Variables only.**

Categorical Variables:
Categorical variables are those data fields that can be divided into definite groups. In this case, no categorical variables are present.

Ordinal Variables:
Ordinal variables are the ones that can be divided into groups, but these groups have some kind of order. Like, high, medium, low. Dependents field can be considered ordinal since the data can be clearly divided into 4 categories: 0, 1, 2, 3 and there is a definite ordering also. In this case we have different type of categories which is mentioned in data description etc.

Numerical or Continuous Variables:
Numerical variables are those that can take up any value within a given range. In this case MSSubClass, LotFrontage, LotArea, BsmtUnfSF, TotalBsmtSF, SalePrice, etc. & many.

Statistical Summary:

In the raw data, there can be various types of underlying patterns which also gives an in-depth knowledge about subject of interest and provides insights about the problem. But caution should be observed with respect to data as it may contain null values, or redundant values, or various types of ambiguity, which also demands for pre-processing of data. Dataset should therefore be explored as much as possible. Various factors important by statistical means like mean, standard deviation, median, count of values and maximum value etc. are shown in Fig below.

```
# Lets understand data at high level check the stastics of dataset
data.describe(include='all')
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	PoolArea	PoolQC	Fence	MiscI
count	1168.000000	1168	954.00000	1168.000000	1168	77	1168	1168	1168	1168	...	1168.000000	7	237	
unique	NaN	5	NaN	NaN	2	2	4	4	1	5	...	NaN	3	4	
top	NaN	RL	NaN	NaN	Pave	Grvl	Reg	Lvl	AllPub	Inside	...	NaN	Gd	MnPrv	
freq	NaN	928	NaN	NaN	1164	41	740	1046	1168	842	...	NaN	3	129	
mean	56.767979	NaN	70.98847	10484.749144	NaN	NaN	NaN	NaN	NaN	NaN	...	3.448630	NaN	NaN	
std	41.940650	NaN	24.82875	8957.442311	NaN	NaN	NaN	NaN	NaN	NaN	...	44.896939	NaN	NaN	
min	20.000000	NaN	21.00000	1300.000000	NaN	NaN	NaN	NaN	NaN	NaN	...	0.000000	NaN	NaN	
25%	20.000000	NaN	60.00000	7621.500000	NaN	NaN	NaN	NaN	NaN	NaN	...	0.000000	NaN	NaN	
50%	50.000000	NaN	70.00000	9522.500000	NaN	NaN	NaN	NaN	NaN	NaN	...	0.000000	NaN	NaN	
75%	70.000000	NaN	80.00000	11515.500000	NaN	NaN	NaN	NaN	NaN	NaN	...	0.000000	NaN	NaN	
max	190.000000	NaN	313.00000	164660.000000	NaN	NaN	NaN	NaN	NaN	NaN	...	738.000000	NaN	NaN	

11 rows × 80 columns

Observations: 1) null values are present 2)we have categorical data type(object type) 3)outliers are present in the dataset 4) In some columns there is too much difference between mean and std. deviation.

Data Visualization:

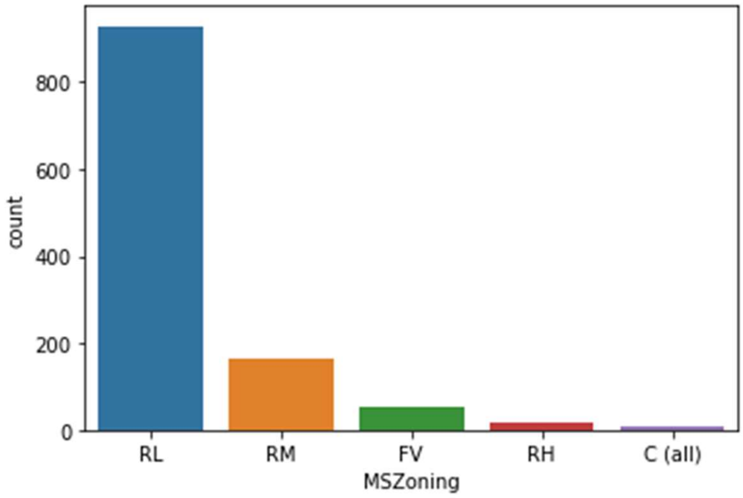
We now have a basic idea about the data. We need to extend that with some visualizations. We are going to look at three types of plots:

1.Univariate plots to better understand each variable.
2.Bivariate plots to find relationship between two variables,
3.Multivariate plots to better understand the relationships between variables.

Univariate Plots:

We start with some univariate plots, that is, plots of each individual variable. Given that the input variables are numeric, we can create box or count plots of each. Now we are all set to perform Univariate Analysis. Univariate analysis involves analysis of one variable at a time. Let's say "MSSubClass" then we will analyse "MSZoning" & others field in the dataset. The analysis is usually summarized in the form of count. For visualization, we have many options such as frequency tables, bar graphs, pie charts, histograms etc. We will stick to count charts.

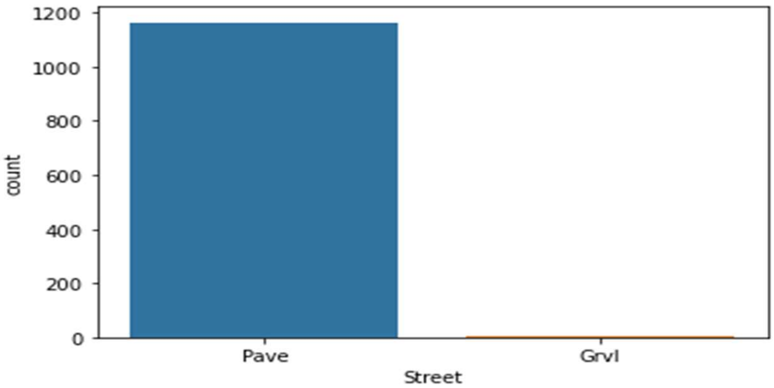
```
#plot each class frequency
sns.countplot(x='MSZoning',data=data)
plt.show()
print(data['MSZoning'].value_counts())
```




```
RL         928
RM         163
FV          52
RH          16
C (all)      9
Name: MSZoning, dtype: int64
```

Residential Low Density is the most important general zoning classification of the sale than the others.

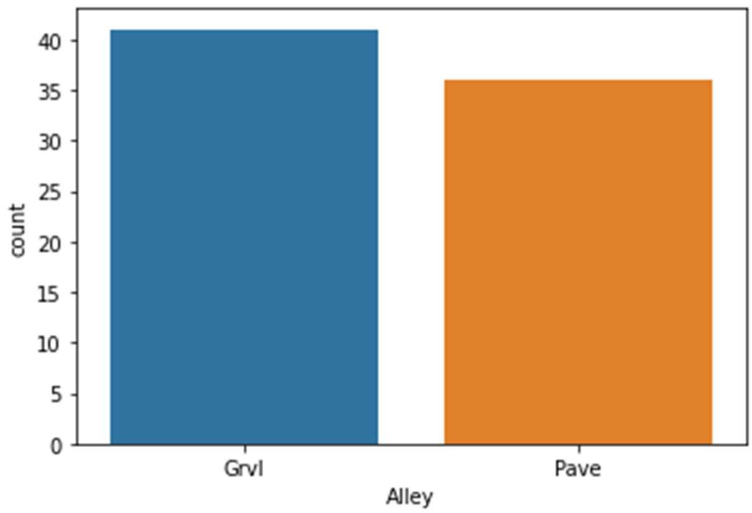
```
#plot each class frequency
sns.countplot(x='Street',data=data)
plt.show()
print(data['Street'].value_counts())
```




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

We can see the more Pave type of road properties are there to access the property.

```
#plot each class frequency
sns.countplot(x='Alley',data=data)
plt.show()
print(data['Alley'].value_counts())
```




```
Grvl    41
Pave    36
Name: Alley, dtype: int64
```

Both the type of properties has alley access to property.

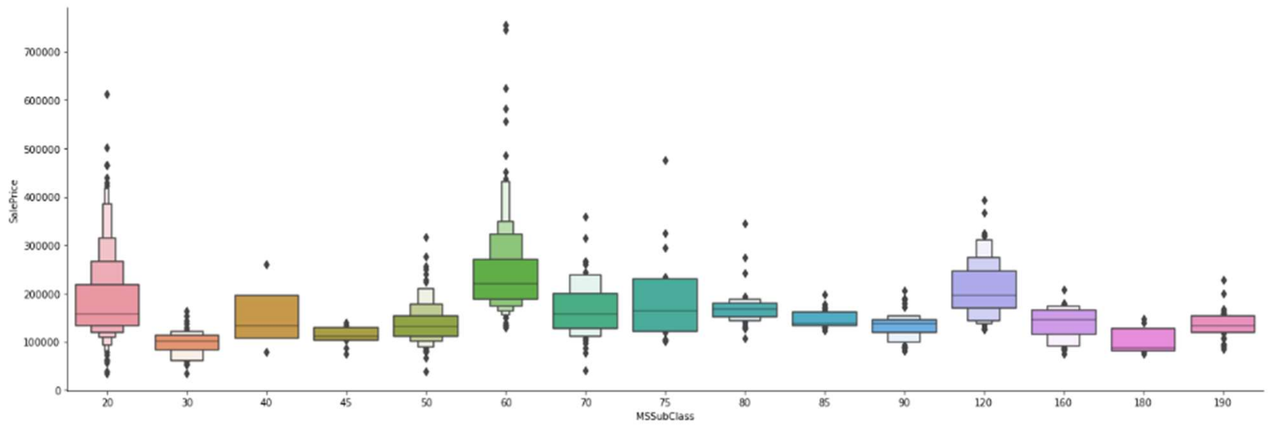
Similarly, I have plotted for the other independent variables and concluded in the notebook.

Bivariate Plot:

Now let's find some relationship between two variables, particularly between the target variable and a predictor variable from the dataset. Formally, this is known as bivariate analysis. Hear we have target variable is SalesPrice, so let's check this relationship between two variables and other independent variables. For visualization, we will be using seaborn. countplot (). It can be considered similar to the histogram for categorical variables

```
#Bivariant graph
plt.figure(figsize =(10, 6))
sns.catplot(x ='MSSubClass', y ='SalePrice', data = data, kind = "boxen", height = 6, aspect = 3)
plt.show()
```

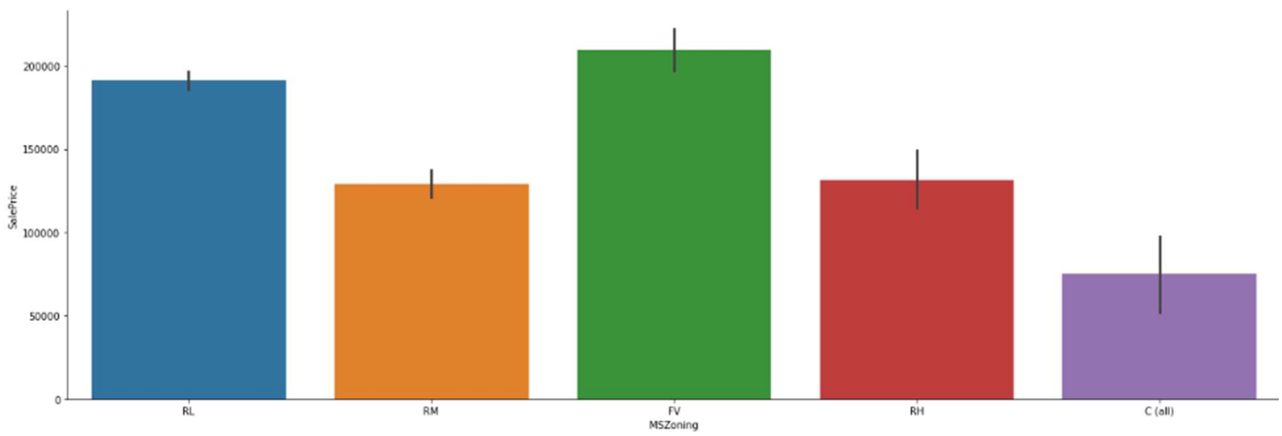
<Figure size 720x432 with 0 Axes>



the maximum 2-STORY 1946 & NEWER the type of dwelling involved in the sale

```
#Bivariant graph
plt.figure(figsize =(10, 6))
sns.catplot(x ='MSZoning', y ='SalePrice', data = data, kind = "bar", height = 6, aspect = 3)
plt.show()
```

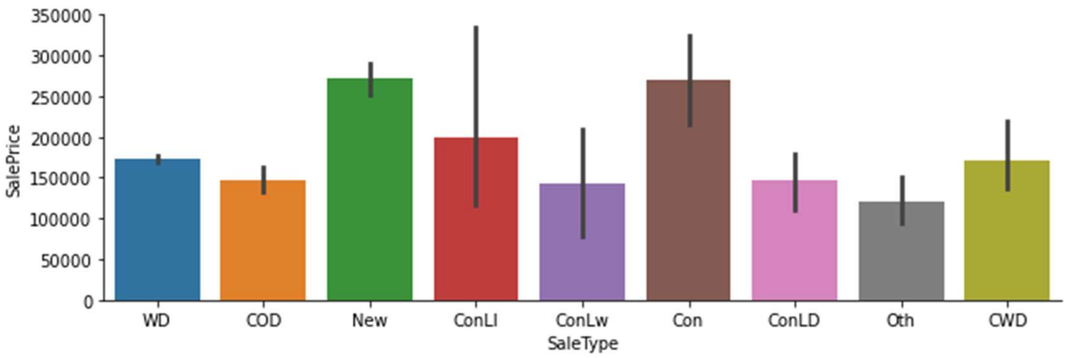
<Figure size 720x432 with 0 Axes>



The maximum sale price is for the Floating Village Residential and Residential Low Density

```
#Bivariant graph
plt.figure(figsize =(10, 6))
sns.catplot(x ='SaleType', y ='SalePrice', data = data, kind = "bar", height = 3, aspect = 3)
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



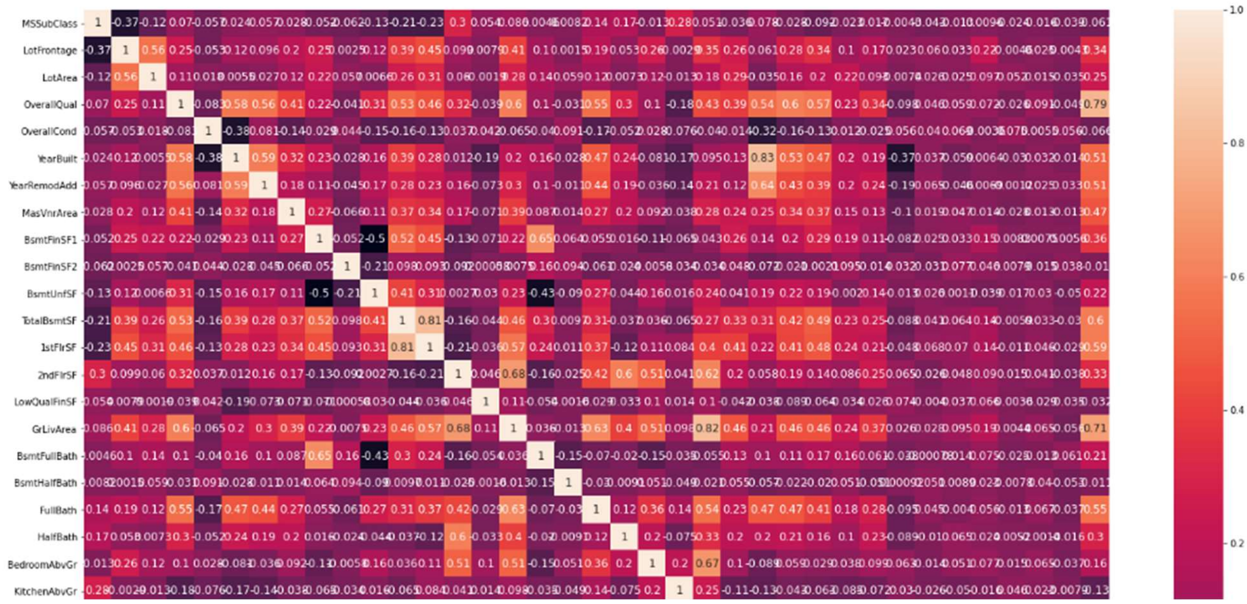
The maximum type of sale property is Home just constructed and sold & Contract 15% Down payment regular terms.

Similarly, done for the other variable with target variable in notebook.

Multivariate Plot:

Let's move on to analysing more than two variables now. We call it "Multivariate analysis". Now we can look at the interactions between the variables. First, let's look at heatmap plot of all pairs of attributes. This can be helpful to spot structured relationships between input variables and target variable. Let's visualize the data in this correlation matrix using a heat map.

```
#check multicolinearity
plt.figure(figsize=(25,20))
sns.heatmap(data.corr(),annot=True,annot_kws={'size':12})
plt.show()
```



Data pre-processing:

Pre-processing of this dataset includes doing analysis on the independent variables like checking for null values in each column and then replacing or filling them with supported appropriate data types like mean, mode method or Imputer methods, so that analysis and model fitting is not hindered from its way to accuracy.

Shown above are some of the representations obtained by using Pandas tools which tells about variable count for numerical columns and model values for categorical columns. Maximum and minimum values in numerical columns, along with their percentile values for median, plays an important factor in deciding which value to be chosen at priority for further exploration tasks and analysis.

Data types of different columns are used further in label processing and Label encoding scheme during model building.

There are many stages involved in data pre-processing.

Data cleaning attempts to impute missing values, removing outliers from the dataset.

Data integration integrates data from a multitude of sources into a single data warehouse.

Data transformation such as normalization, may be applied. For example, normalization may improve the accuracy and efficiency of mining algorithms involving distance measurement.

Data reduction can reduce the data size by dropping out redundant features. Feature selection and feature extraction techniques can be used.

In this dataset there are some objectives are mentioned as NA so first lets change to particular value as per mentioned in data description. Refer below fig.

Alley: Type of alley access to property

```
Grvl    Gravel
Pave    Paved
NA      No alley access
```

```
26]: # As per the information in data description file the NA in the column Alley is No Alley Access so lets replace all NA with No al
data['Alley']=data['Alley'].replace("NA","No alley access")
data[['BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2']]=data[['BsmtQual','BsmtCond','BsmtExposure','BsmtFinTyp
data['FireplaceQu']=data['FireplaceQu'].replace("NA","No Fireplace")
data['PoolQC']=data['PoolQC'].replace("NA","No Pool")
data['Fence']=data['Fence'].replace("NA","No Fence")
data[['GarageCond','GarageQual','GarageFinish','GarageType']]=data[['GarageCond','GarageQual','GarageFinish','GarageType']].repla
```

As in our dataset have null values & objective data type so let's use Imputer techniques & Encoding techniques to convert data into numerical form as shown below.

Label Encoder refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning. Label encoding in python can be imported from the Sklearn library. Sklearn provides a very efficient tool for encoding. Label encoders encode labels with a value between 0 and n_classes-1.

```
: # Lets frist covert categorical data(type & column) into int
label = LabelEncoder()
for i in cat_col:
    df=label.fit_transform(data[i])
    pd.Series(df)
    data[i]=df
```

Iterative imputer is a hidden gem of the sklearn library in python.

The iterative imputer library provides us with tools to tackle the problem mentioned above. Instead of just replacing values with mean/median, we can have a regressor (Linear/Decision Tree/Random Forest/KNN) to impute missing values. By using the iterative imputer we can intelligently impute the missing values, avoid bias, maintain the relationship between variables, and can get better results.

```
# Creating null value list which we want to impute by using iterative inputer
df=data[['LotFrontage','MasVnrArea','GarageYrBlt','MSZoning','Street','Alley','LotShape','LandContour','Utilities','LotConfig','L
 'ExterCond','Foundation','BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2','Heating','HeatingQC','CentralAir','
```

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
imputer = IterativeImputer()

for i in df:
    df1=imputer.fit_transform(data[[i]])
    pd.Series([df1])
    data[i]=df1
```

Also, as I say data pre-processing is a predominant step in machine learning to yield highly accurate and insightful results. Greater the quality of data, the greater is the reliability of the produced results.

Incomplete, noisy, and inconsistent data are the inherent nature of real-world datasets. Data pre-processing helps in increasing the quality of data by filling in missing incomplete data, smoothing noise, and resolving inconsistencies.

Incomplete data can occur due to many reasons. Appropriate data may not be persisted due to a misunderstanding, or because of instrument defects and malfunctions.

Noisy data can occur for a number of reasons (having incorrect feature values). The instruments used for the data collection might be faulty. Data entry may contain human or instrument errors. Data transmission errors might occur as well.

```
data.skew(axis=0)
```

```
MSSubClass        1.422019
MSZoning         -1.796785
LotFrontage       2.710383
LotArea          10.659285
Street          -17.021969
                   ...
MoSold            0.220979
YrSold            0.115765
SaleType         -3.660513
SaleCondition    -2.671829
SalePrice         1.953878
Length: 80, dtype: float64
```

From above filling we can see the skewed data is present as the weight of values is more than +-0.5 values. So, to remove this skewness we are using Power Transformation.

```
# Using power transformation to remove skewed data
from sklearn.preprocessing import PowerTransformer
pt=PowerTransformer()
data[df1]=pt.fit_transform(data[df1].values)
```

Outliers are data points that are distant from other similar points. They may be due to variability in the measurement or may indicate experimental errors. If possible, outliers should be excluded from the data set. However, detecting that anomalous instance might be very difficult, and is not always possible.

Methods we used to remove outliers from our dataset is:

Z-score — Call scipy.stats.zscore() with the given data-frame as its argument to get a numpy array containing the z-score of each value in a dataframe. Call numpy.abs() with the previous result to convert each element in the dataframe to its absolute value. Use the syntax (array < 3).all(axis=1) with array as the previous result to create a boolean array.

```
# from above graph we see there is outliers in featurs Let's remove outliers from above columns by using Zscore
z_score=zscore(data[df1])
abs_z_score=np.abs(z_score)
filtering_entry=(abs_z_score<3).all(axis=1)
data=data[filtering_entry]
```

Correlations among variables:

Heatmap was plotted for variables with correlation coefficient. The Variance Inflation Factor (VIF) measures the severity of multicollinearity in regression analysis. It is a statistical concept that indicates the increase in the variance of a regression coefficient as a result of collinearity. Variance inflation factor (VIF) is used to detect the severity of multicollinearity in the ordinary least square (OLS) regression analysis. Multicollinearity inflates the variance and type II error. It makes the coefficient of a variable consistent but unreliable. VIF measures the number of inflated variances caused by multicollinearity. Checking correlation between dependent and independent variables.

```
          vif        featurs
0    9.456248     MSSubClass
1    1.725692       MSZoning
2    2.268559    LotFrontage
3    3.527018        LotArea
4    0.000000         Street
..        ...            ...
74        NaN        MiscVal
75   1.142510         MoSold
76   1.202151         YrSold
77   1.665228       SaleType
78   1.829983  SaleCondition

[79 rows x 2 columns]
```

```
# Lets make thumb rule we can drop the columns if VIF values are more than the 10 pvalue
index=np.where(vif['vif']>10)
vif.loc[index]
```

	vif	featurs
18	16.825635	YearBuilt
34	28.202149	BsmtFinType2
35	29.100187	BsmtFinSF2
37	10.831835	TotalBsmtSF
42	20.155439	1stFlrSF
43	26.117573	2ndFlrSF
45	27.504631	GrLivArea
58	10.631115	GarageYrBlt

As we see the pvalues of above predictors is more than 10 so lets drop the same & reduce the colinarity.

Building machine learning models:

For building machine learning models there are several models present inside the Sklearn module.

Sklearn provides two types of models i.e., regression and classification. Our datasets contain continuous type target variable i.e**., Sales Price.** So,this kind of problem, we use regression models. But before fitting our dataset to its model first we have to separate the predictor variable and the target variable, then scaled the independent variables or predictors by using StandardScaler method and then we pass this variable to the train_test_split method to create a random test and train subset.

What is train_test_split, it is a function in sklearn model selection for splitting data arrays into two subsets for training data and 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.

However, you can also specify a random state for the operation. It gives us four outputs x_train, x_test, y_train and y_test. The x_train and x_test contains the training and testing predictor variables while y_train and y_test contains the training and testing target variable. After performing train_test_split we have to choose the models to pass the training variable.

We can build as many models as we want to compare the accuracy given by these models and to select the best model among them.

I have selected 5 models for the problem statement that is prediction of Sales Price:

1) **LinearRegression from sklearn.linear_model:** Linear regression models are most preferably used with the least-squares approach, where the implementation might require other ways by minimising the deviations and the cost functions, for instance. The general

linear models include a response variable that is a vector in nature and not directly scalar. The conditional linearity is still presumed positive over the modelling process. They vary over a large scale, but they are better described as the skewed distribution, which is related to the log-normal distribution.

```
# Model no.1
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)

print_score(lr,x_train,x_test,y_train,y_test,train=True)
print_score(lr,x_train,x_test,y_train,y_test,train=False)
model_accuracy(lr)
```

```
Train Report: 0.9080276081529983
Test Report: 0.9056093871247141
RMSE: 0.2514498866075719
MAE: 0.19281181814548082
MSE: 0.06322704547496076
Accuracy: 85.98 %
Standard Deviation: 3.99 %
```

2) **Lasso regression from sklearn.model:** Lasso regression is a method we can use to fit a regression model when multicollinearity is present in the data.

In a nutshell, least squares regression tries to find coefficient estimates that minimize the sum of squared residuals (RSS):

RSS = Σ(yi – ŷi)2
where:
Σ: A greek symbol that means sum
yi: The actual response value for the ith observation
ŷi: The predicted response value based on the multiple linear regression model

Conversely, lasso regression seeks to minimize the following:

RSS + λΣ|βj|
where j ranges from 1 to p predictor variables and λ ≥ 0.

This second term in the equation is known as a shrinkage penalty. In lasso regression, we select a value for λ that produces the lowest possible test MSE (mean squared error).

```
# Model no.2
from sklearn.linear_model import LinearRegression, Lasso, LassoCV

lcv=LassoCV(alphas=None,max_iter=10000,normalize=True)
lcv.fit(x_train,y_train)
alpha=lcv.alpha_
print(alpha)
Lasso_reg=Lasso(alpha).fit(x_train,y_train)

print_score(Lasso_reg,x_train,x_test,y_train,y_test,train=True)
print_score(Lasso_reg,x_train,x_test,y_train,y_test,train=False)
model_accuracy(Lasso_reg)
```

```
0.0004869863116328419
Train Report: 0.9079791547504575
Test Report: 0.9060285419169292
RMSE: 0.2508909661323906
MAE: 0.19217707449440888
MSE: 0.06294627688684437
Accuracy: 86.16 %
Standard Deviation: 4.05 %
```

3) **DecisionTreeRegressor from sklearn.tree:** Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. The two main entities of a tree are decision nodes, where the data is split and leaves, where we get the outcome.

```
#Model no.5
from sklearn.tree import DecisionTreeRegressor

dt=DecisionTreeRegressor()

dt.fit(x_train,y_train)

print_score(dt,x_train,x_test,y_train,y_test,train=True)
print_score(dt,x_train,x_test,y_train,y_test,train=False)
model_accuracy(dt)
```

```
Train Report: 1.0
Test Report: 0.7027666658783476
RMSE: 0.44620640259849315
MAE: 0.319052518153771
MSE: 0.19910015371988854
Accuracy: 58.24 %
Standard Deviation: 8.17 %
```

4) **RandomForestRegressor from sklearn.ensemble:** As we know that a forest is made up of trees and more trees means more robust forest. Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

```
: # Model no.3
from sklearn.ensemble import RandomForestRegressor

rand_regressor= RandomForestRegressor()
rand_regressor.fit(x_train,y_train)

print_score(rand_regressor,x_train,x_test,y_train,y_test,train=True)
print_score(rand_regressor,x_train,x_test,y_train,y_test,train=False)
model_accuracy(rand_regressor)
```

```
Train Report: 0.9763639515432244
Test Report: 0.8688353356347611
RMSE: 0.29641178538100743
MAE: 0.2243813261332307
MSE: 0.0878599465127564
Accuracy: 81.50 %
Standard Deviation: 4.56 %
```

5) **XGBRegressor from XGBoost:** XGBoost is short for "eXtreme Gradient Boosting." The "eXtreme" refers to speed enhancements such as parallel computing and cache awareness that makes XGBoost approximately 10 times faster than traditional Gradient Boosting. In addition, XGBoost includes a unique split-finding algorithm to optimise trees, along with built-in regularisation that reduces over-fitting. Generally speaking, XGBoost is a faster, more accurate version of Gradient Boosting.

```
: # Model no.3
from xgboost import XGBRegressor

xgb=XGBRegressor()
xgb.fit(x_train,y_train)

print_score(xgb,x_train,x_test,y_train,y_test,train=True)
print_score(xgb,x_train,x_test,y_train,y_test,train=False)
model_accuracy(xgb)
```

```
Train Report: 0.9999988777699308
Test Report: 0.866975159506532
RMSE: 0.29850624031898
MAE: 0.22965790186721105
MSE: 0.08910597550937263
Accuracy: 80.38 %
Standard Deviation: 4.12 %
```

6) **HistGradientBoostingRegressor:** The number of tree that are built at each iteration. For regressors, this is always 1. train_score_ndarray, shape (n_iter_+1,) The scores at each iteration on the training data. The first entry is the score of the ensemble before the first iteration. Scores are computed according to the scoring parameter.

```
#Model no.4
from sklearn.ensemble import HistGradientBoostingRegressor

gbdt=HistGradientBoostingRegressor()

gbdt.fit(x_train,y_train)

print_score(gbdt,x_train,x_test,y_train,y_test,train=True)
print_score(gbdt,x_train,x_test,y_train,y_test,train=False)
model_accuracy(gbdt)
```

```
Train Report: 0.986284834190471
Test Report: 0.8909983490646709
RMSE: 0.2702113448903229
MAE: 0.21280564525111753
MSE: 0.07301417090743703
Accuracy: 83.38 %
Standard Deviation: 3.40 %
```

As we have both the train and test dataset, we have used train dataset to build the model and based on this model we are going to predict the sales price for this data so, we are doing all the EDA steps as per train data and predicting the sales price.

We got **our** best model looking at accuracy, is **LassoRegression** with Kfold cross validation method with the accuracy score of 88.43 %. With RMSE: 0.21, MAE: 0.16, & MSE: 0.04 % error.

Understand what does MSE, MAE & RMSE do:

Mean squared error	$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}}$
Mean absolute error	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e$

MSE: Mean squared error

In machine learning, the mean squared error (MSE) is used to evaluate the performance of a regression model. In regression models, the RMSE is used as a metric to measure model performance and the MSE score is used to evaluate the performance. The MSE score is used to evaluate the performance of a machine learning model while working on regression problems. When the distance is higher it represents a high error rate and when the distance is lower than you are near to the best fit line.

MAE: Mean Absolute Error
Mean Absolute Error metric is to subtract our predicted value and actual value at each time point to obtain the absolute value, and then average it out.

RMSE: Root Mean Squared Error

Root Mean Squared Error (RMSE) is a common metric for assessing the performance of regression machine learning models. It is often used to provide a metric that is related to the unit being measured.

Squared error, also known as L2 loss, is a row-level error calculation where the difference between the prediction and the actual is squared. RMSE is the aggregated mean and subsequent square root of these errors, which helps us understand the model performance over the whole dataset.

A benefit of using RMSE is that the metric it produces is on the same scale as the unit being predicted. For example, calculating RMSE for a house price prediction model would give the error in terms of house price, which can help end users easily understand model performance.

Cross validation:

Models are trained with a 5-fold cross validation. A technique that takes the entirety of your training data, randomly splits it into train and validation data sets over 5 iterations.

You end up with 5 different training and validation data sets to build and test your models. It's a good way to counter overfitting.

More generally, cross validation of this kind is known as k-fold cross validation.

Hyper parameter tuning:

Hyper parameter optimisation in machine learning intends to find the hyper parameters of a given machine learning algorithm that deliver the best performance as measured on a validation set. Hyper parameters, in contrast to model parameters, are set by the machine learning engineer before training.

The number of trees in a random forest is a hyper parameter while the weights in a neural network are model parameters learned during training. I like to think of hyper parameters as the model settings to be tuned so that the model can optimally solve the machine learning problem.

We will use RandomizedSearchCV for the hyper parameter tuning.

RandomizedSearchCV :

Randomized search on hyper parameters. RandomizedSearchCV implements a "fit" and a "score" method. It also implements "score_samples", "predict", "predict_proba", "decision_function", "transform" and "inverse_transform" if they are implemented in the estimator used.

But in this our problem after using the hyper tuning parameter there is no change in accuracy and error so no need to do the same.

Feature importance using the LASSO:

When creating a model not all of the features in our training data are of equal importance. If we have sufficient computational resources at our disposal then we could indeed include all of the available features in our model, but this has (at least) two drawbacks; this can lead to overfitting, and also reduces the interpretability of our model. It is much more informative to create our model on a subset of the most influential features, in other words create a sparse model.

With LASSO (Least Absolute Shrinkage and Selection Operator) regression we now have the following penalty term (ESL Eq. 3.53):

$PRSS=RSS+\lambda\sum_{j=1}^{p}|\beta_j|$

note now the use of the ℓ_1 norm, which is sometimes known as the Manhattan distance. This change has the effect of actually forcing some of the coefficients to become zero, in other words this is actually removing features from the model, and is effectively performing feature selection.

We can fit a LinearRegression model on the regression dataset and retrieve the coeff_ property that contains the coefficients found for each input variable.

These coefficients can provide the basis for a crude feature importance score. This assumes that the input variables have the same scale or have been scaled prior to fitting a model.

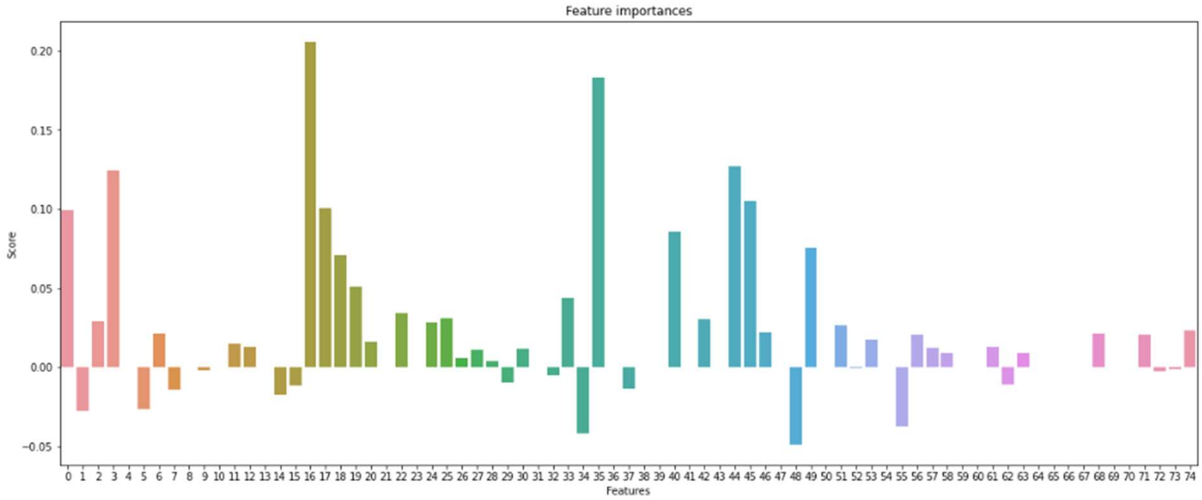
All of these algorithms find a set of coefficients to use in the weighted sum in order to make a prediction. These coefficients can be used directly as a crude type of feature importance score.

Let’s take a closer look at using coefficients as feature importance for classification and regression. We will fit a model on the dataset to find the coefficients, then summarize the importance scores for each input feature and finally create a bar chart to get an idea of the relative importance of the features.

```
Feature: 0, Score: 0.09935
Feature: 1, Score: -0.02756
Feature: 2, Score: 0.02920
Feature: 3, Score: 0.12447
Feature: 4, Score: 0.00000
Feature: 5, Score: -0.02606
Feature: 6, Score: 0.02161
Feature: 7, Score: -0.01401
Feature: 8, Score: 0.00000
Feature: 9, Score: -0.00185
Feature: 10, Score: 0.00000
Feature: 11, Score: 0.01509
Feature: 12, Score: 0.01277
Feature: 13, Score: 0.00000
Feature: 14, Score: -0.01697
Feature: 15, Score: -0.01129
Feature: 16, Score: 0.20574
Feature: 17, Score: 0.10060
Feature: 18, Score: 0.07114
Feature: 19, Score: 0.05107
Feature: 20, Score: 0.01635
Feature: 21, Score: 0.00000
Feature: 22, Score: 0.03453
Feature: 23, Score: 0.00000
Feature: 24, Score: 0.02853
Feature: 25, Score: 0.03079
Feature: 26, Score: 0.00615
Feature: 27, Score: 0.01130
Feature: 28, Score: 0.00392
Feature: 29, Score: -0.00957
Feature: 30, Score: 0.01186
Feature: 31, Score: -0.00000
Feature: 32, Score: -0.00491
Feature: 33, Score: 0.04395
```

Form this score we can find the important features to predict the price.

```
# plot feature importance
plt.figure(figsize=(20,8))
sns.barplot([x for x in range(len(importance))], importance)
plt.xlabel("Features")
plt.ylabel("Score")
plt.title("Feature importances")
plt.show()
```



Now, lets predict the sales price by using the saved Lasso Regression Model on test data.

```
data1["predictions"] = data1.apply(lambda s: Lasso_reg.predict(s.values[None])[0], axis=1)
```

```
data1["predictions"]
```

```
0       3.206583
1       1.407278
2       1.752104
3       1.295778
4       2.161446
          ...
287     2.223310
288    -0.029469
289     0.615764
290     0.316112
291    -1.758343
Name: predictions, Length: 292, dtype: float64
```

Remarks:

In this project we build the regression model that can predict sales price of house. The challenge behind predicting models is EDA & feature selection.

We have gone through how to implement the entire machine learning pipeline, and we have an intuitive understanding of machine learning algorithms. The larger the dataset gets, the more complex each of the mentioned steps gets. Therefore, using this as a base will help while you build your knowledge of machine learning pipelines.

This Paper has presented a supervised housing sales price learning model which used machine learning algorithms to predict the price. We used different machine learning algorithm to check the accuracy of price prediction.

References:

- Regression Model
- Hyper-parameter Tuning
- Cross Validation
- Getting started with XGBoost
- scikit

In a nutshell….

Exploring and knowing your datasets is a very essential step. It not only helps in finding anomalies, uniqueness and pattern in the dataset but also helps us in building better hypothesis functions. If you wish to see the entire code, here Link is the to my Jupiter notebook