HOUSE PRICE PREDICTION

JANUARY 22

FLIP ROBO TECHNOLOGIES
Authored by: I N VENKATESWARA RAO

FLIP ROBO

HOUSE PRICE PREDICTION

ACKNOWLEDGMENT

I would like to express my sincere thanks of gratitude to my mentors from Data Trained academy and Flip Robo company for giving me the opportunity to work on this project.

ABSTRACT

Business Problem Framing:
Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company.
A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file.

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

Conceptual Background of the Domain Problem

The project will require knowledge and practice in building Graphs /plots and analyzing them to get the relationship between dataset, Knowledge of Different Learning Models to build and predict the required output. Basic Data science concepts to increase the quality of the dataset and Python Knowledge (Coding Language) which will be used to solve the complete Micro Credit Defaulter project. Understanding of calculating F2 score, accuracy, skewness and basic mathematics/statistical approaches will help to build an accurate model for this project.

Analytical Problem Framing

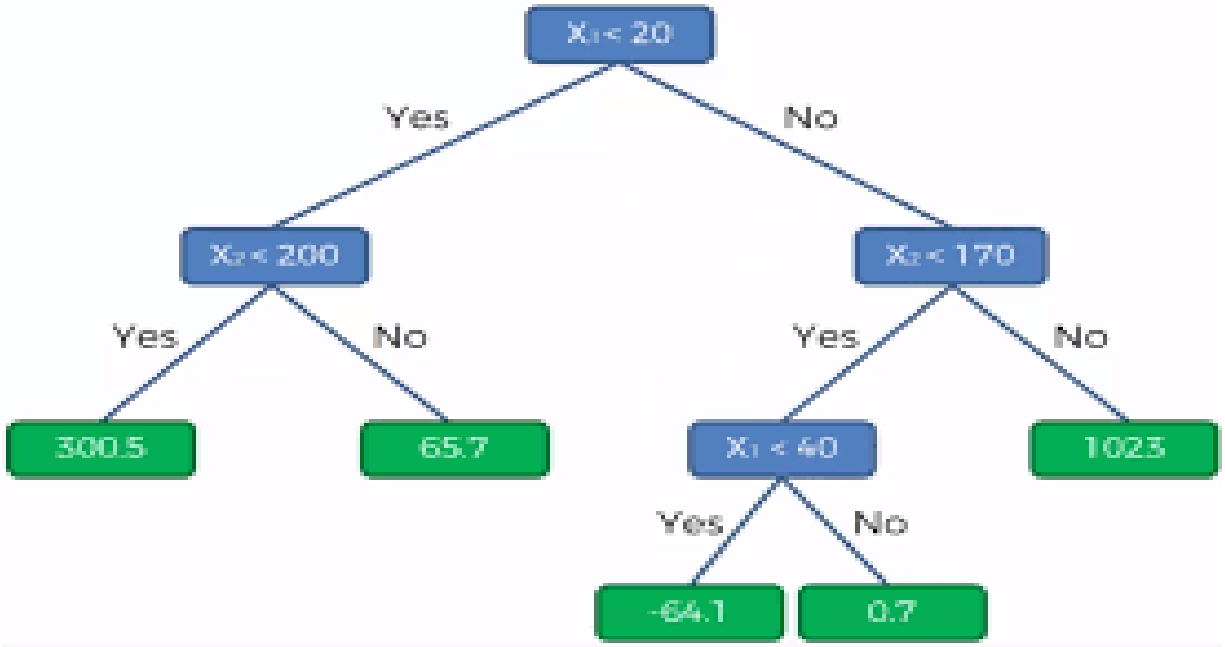
Mathematical/ Analytical Modelling of the Problem:

Regression Models-

Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (s) (predictor). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables. For example, relationship between rash driving and number of road accidents by a driver is best studied through regression.

Decision Tree –

It is a decision-making tool that uses a flowchart-like tree structure or is a Model of decisions and all of their possible results, including outcomes, input costs and utility. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables. The branches/edges represent the result of the node and the nodes have either: Conditions [Decision Nodes] Result [End Nodes] The branches/edges represent the truth/falsity of the statement and takes makes a decision based on that in the example below which shows a decision tree that evaluates the smallest of three numbers:



Random Forest –

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, 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.

Naive Bayes –

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Linear Regression –

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x). When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, literature from statistics often refers to the method as multiple linear regression.

SVM –

Supervised Machine Learning Algorithm used for classification and/or regression. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line.

We used different Plots/ graphs to perform EDA on the dataset:

Box Plot: It is a type of chart that depicts a group of numerical data through their quartiles. It is a simple way to visualize the shape of our data. It makes comparing characteristics of data between categories very easy.

Count Plot: IT is kind of like a histogram or a bar graph for some categorical area. It simply shows the number of occurrences of an item based on a certain type of category

Heat Map: It contains values representing various shades of the same color for each value to be plotted. Usually the darker shades of the chart represent higher values than the lighter shade. For a very different value a completely different color can also be used.

Scatter Plot: A scatter plot is a diagram where each value in the data set is represented by a dot. The Matplotlib module has a method for drawing scatter plots

Data Sources and their formats

Below are the fields present in our dataset with the information what these fields describe

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
CornerCorner 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
GilbertGilbert
IDOTRR Iowa DOT and Rail Road
MeadowV Meadow Village
Mitchel Mitchell
NamesNorth 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
TwnhsITownhouse Inside Unit
HouseStyle: Style of dwelling
1Story One story
1.5Fin One and one-half story: 2nd level finished
1.5UnfOne 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
Gd Good
TA Average/Typical
Fa Fair
Po Poor
CentralAir: Central air conditioning
N No
Y Yes
Electrical: Electrical system
SBrkr Standard Circuit Breakers & Romex
FuseA Fuse Box over 60 AMP and all Romex wiring
(Average)
FuseF 60 AMP Fuse Box and mostly Romex wiring (Fair)
FuseP 60 AMP Fuse Box and mostly knob & tube wiring
(poor)
Mix Mixed

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet
LowQualFinSF: Low quality finished square feet (all floors)
GrLivArea: Above grade (ground) living area square feet
BsmtFullBath: Basement full bathrooms
BsmtHalfBath: Basement half bathrooms
FullBath: Full bathrooms above grade
HalfBath: Half baths above grade
Bedroom: Bedrooms above grade (does NOT include basement
bedrooms)
Kitchen: Kitchens above grade
KitchenQual: Kitchen quality
Ex Excellent

Gd Good
TA Typical/Average
Fa Fair
Po Poor

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
Functional: Home functionality (Assume typical unless deductions are warranted)
Typ Typical Functionality
Min1 Minor Deductions 1
Min2 Minor Deductions 2
Mod Moderate Deductions
Maj1 Major Deductions 1

Maj2 Major Deductions 2
Sev Severely Damaged
Sal Salvage only

Fireplaces: Number of fireplaces
FireplaceQu: Fireplace quality
Ex Excellent - Exceptional Masonry Fireplace
Gd Good - Masonry Fireplace in main level
TA Average - Prefabricated Fireplace in main living area or Masonry Fireplace in basement
Fa Fair - Prefabricated Fireplace in basement
Po Poor - Ben Franklin Stove
NA No Fireplace

GarageType: Garage location
2TypesMore than one type of garage
Attchd Attached to home
Basment Basement Garage

BuiltIn Built-In (Garage part of house - typically has room above garage)
CarPort Car Port
Detchd Detached from home
NA No Garage

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage
Fin Finished
RFn Rough Finished
Unf Unfinished
NA No Garage
GarageCars: Size of garage in car capacity
GarageArea: Size of garage in square feet
GarageQual: Garage quality
Ex Excellent
Gd Good
TA Typical/Average
Fa Fair
Po Poor
NA No Garage
GarageCond: Garage condition
Ex Excellent
Gd Good
TA Typical/Average
Fa Fair
Po Poor
NA No Garage
PavedDrive: Paved driveway
Y Paved
P Partial Pavement
N Dirt/Gravel

WoodDeckSF: Wood deck area in square feet
OpenPorchSF: Open porch area in square feet
EnclosedPorch: Enclosed porch area in square feet
3SsnPorch: Three season porch area in square feet
ScreenPorch: Screen porch area in square feet
PoolArea: Pool area in square feet
PoolQC: Pool quality
Ex Excellent
Gd Good
TA Average/Typical
Fa Fair
NA No Pool

Fence: Fence quality
GdPrv Good Privacy
MnPrv Minimum Privacy
GdWo Good Wood
MnWw Minimum Wood/Wire
NA No Fence
MiscFeature: Miscellaneous feature not covered in other categories
Elev Elevator
Gar2 2nd Garage (if not described in garage section)
Othr Other
Shed Shed (over 100 SF)

TenC Tennis Court
NA None
MiscVal: $Value of miscellaneous feature
MoSold: Month Sold (MM)
YrSold: Year Sold (YYYY)
SaleType: Type of sale
WD Warranty Deed - Conventional

CWD Warranty Deed - Cash
VWD Warranty Deed - VA Loan
New Home just constructed and sold
COD Court Officer Deed/Estate
Con Contract 15% Down payment regular terms
ConLw Contract Low Down payment and low interest
ConLI Contract Low Interest
ConLD Contract Low Down
Oth Other
SaleCondition: Condition of sale
Normal Normal Sale
Abnorml Abnormal Sale - trade, foreclosure, short sale
AdjLand Adjoining Land Purchase
Alloca Allocation - two linked properties with separate
deeds, typically condo with a garage unit
Family Sale between family members
Partial Home was not completed when last assessed
(associated with New Homes)

Data types of the fields:
Below is the information of all the attributes with their respective
datatypes:
Column name datatype
Id int64
MSSubClass int64
MSZoning object
LotFrontage float64
LotArea int64
Street object
Alley object
LotShape object
LandContour object
Utilities object

LotConfig object
LandSlope object
Neighborhood object
Condition1 object
Condition2 object
BldgType object
HouseStyle object
OverallQual int64
OverallCond int64
YearBuilt int64
YearRemodAdd int64
RoofStyle object

RoofMatl object
Exterior1st object
Exterior2nd object
MasVnrType object
MasVnrArea float64
ExterQual object
ExterCond object
Foundation object
BsmtQual object
BsmtCond object
BsmtExposure object
BsmtFinType1 object
BsmtFinSF1 int64
BsmtFinType2 object
BsmtFinSF2 int64
BsmtUnfSF int64
TotalBsmtSF int64
Heating object
HeatingQC object
CentralAir object

Electrical object
1stFlrSF int64
2ndFlrSF int64
LowQualFinSF int64
GrLivArea int64

BsmtFullBath int64
BsmtHalfBath int64
FullBath int64
HalfBath int64
BedroomAbvGr int64
KitchenAbvGr int64
KitchenQual object
TotRmsAbvGrd int64
Functional object
Fireplaces int64
FireplaceQu object
GarageType object
GarageYrBlt float64
GarageFinish object
GarageCars int64
GarageArea int64
GarageQual object
GarageCond object
PavedDrive object
WoodDeckSF int64
OpenPorchSF int64
EnclosedPorch int64
3SsnPorch int64
ScreenPorch int64
PoolArea int64

PoolQC object

Fence object
MiscFeature object
MiscVal int64
MoSold int64
YrSold int64
SaleType object
SaleCondition object
SalePrice int64

Data Pre-processing Done

1. First we checked the data set dimensions:

2. Then we checked whether there is any missing data and replaced those with appropriate data

3. We checked the outliers using the Box Plot and replaced the outliers with more appropriate values. Removal of outliers can also be done but taking the Data Loss percentage into consideration It is better to replace the outlier

Hardware and Software Requirements and Tools Used

1) Software: Jupyter Notebook - To code and build the project in python

2) Libraries:
   a) numpy - To perform basic math operations
   b) pandas - To perform basic File operations
   c) Matplotlib - To plot Different Graphs/ Plots
   d) Seaborn - Advance library to enhance the quality of graphs/plots
   e) warnings - To ignore the unwanted warnings raised while interpreting code
   f) sklearn - To build the Prediction models
   g) imblearn - To balance our dataset distribution

Univariate Analysis:

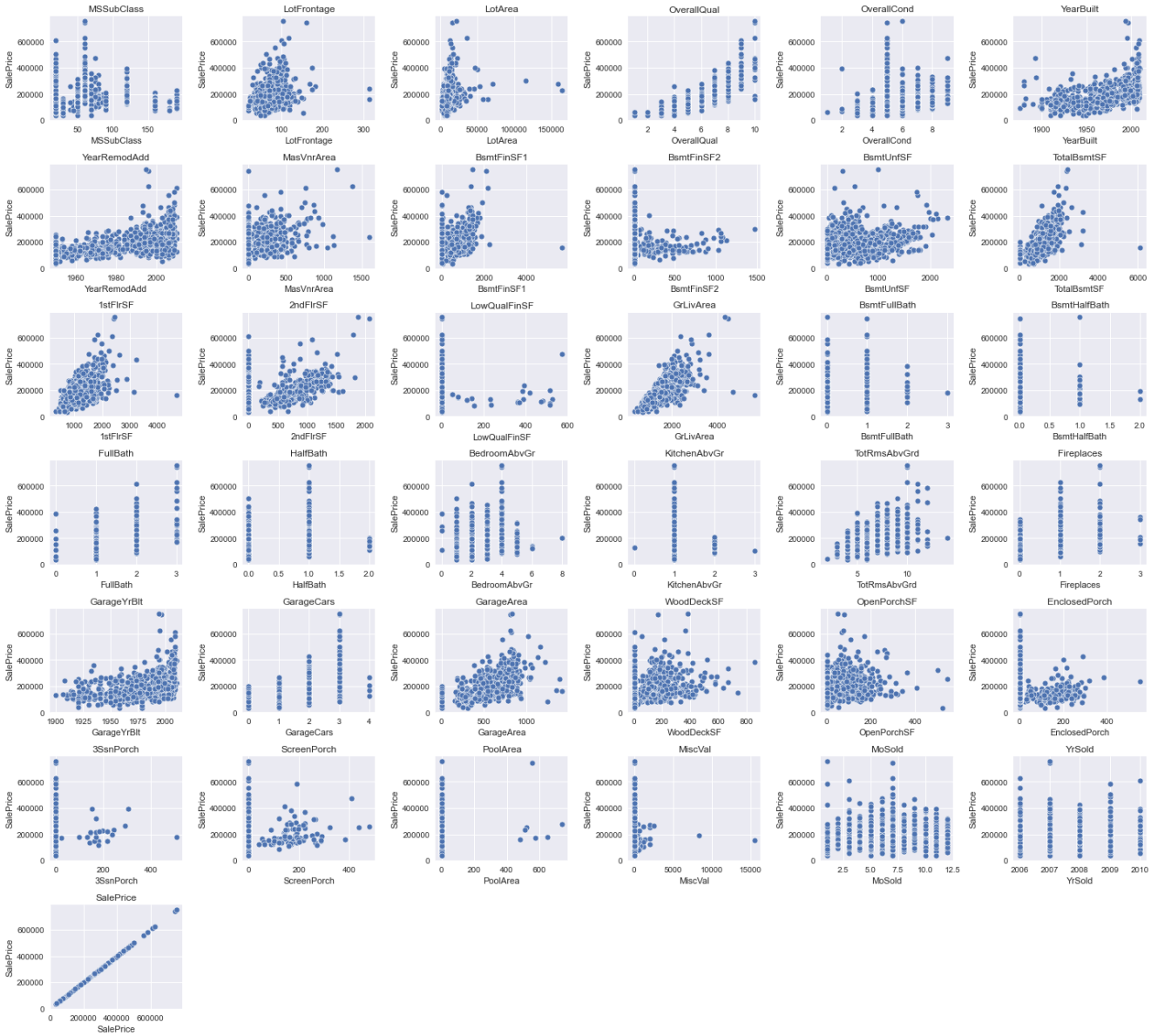
Uni means one, so in other words, the data has only one variable. Univariate data requires analysing each variable separately. It doesn't deal with causes or relationships (unlike regression) and its major purpose is to describe; It takes data, summarizes that data and finds patterns in the data.

Analysing Numerical columns:

Here in all the numerical columns we can see through scatter plot that many numerical features are skewed on either the left side or right side. Also, there are so many outliers in the features.

Bivariate Analysis:

Bivariate analysis is finding some kind of empirical relationship between two variables. Specifically, the dependent vs independent Variables.

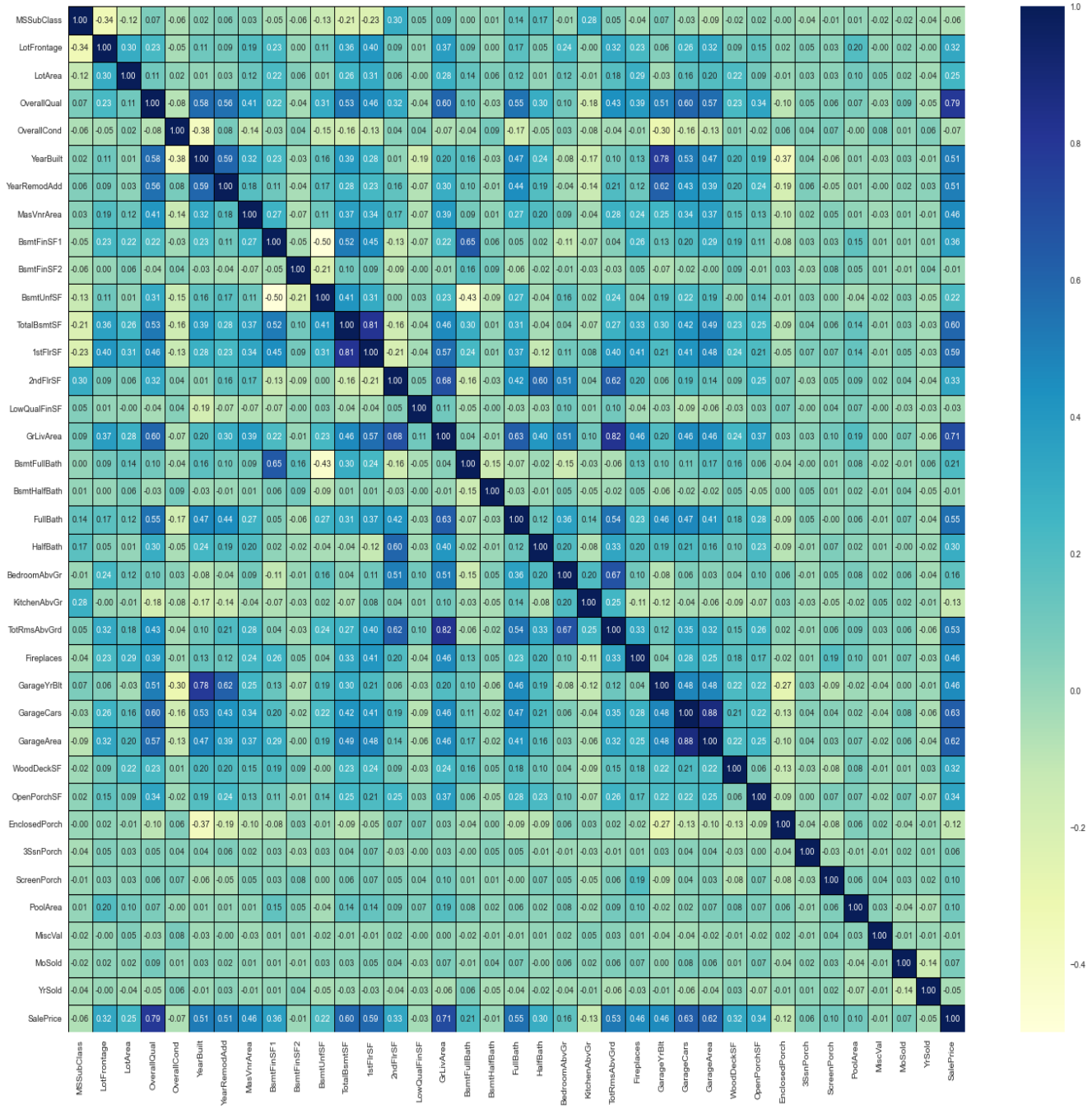


From the above graphs the following observations were made:

- MSSubClass -> not normally distributed
- LotFrontage -> normally distributed
- LotArea -> Normally distributed
- OverallQual-> Not normally distributed
- Overall cond-> Not normally distributed
- Year Built -> Not normally distributed
- Year remod add->not normally distributed
- BsmtFinSF1 ->not normally distributed
- GaragerBlt ->not normally distributed
- Garage Area -> not normally distributed
- Sale prices of houses were less in the past. But with time price increased.
- With LotArea there is increase in Price, but there may be outliers
- With OverallQual, OverallCond, YearBuilt, YearReMOdAdd,MassVnrArea,GrLivArea there is increase in Price, but there may be outliers
- With TotRmsAbvGrd, GarageArea, FullBath,2ndFlrSF,1stFlrSF, TotalBsmtSF,BsmtUnfSF, BsmtFinSF1,MasVnrArea increase, prices increasing
- The more is the Ground living area, the more is the sale price.
- The more is the garage area, the more is the sale price

Checking for Correlation with Output Features:

- After this, we found the most important features relative to the target by building a correlation matrix. A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. The correlation coefficient has values between -1 to 1.
- Correlations are very useful in many applications, especially when conducting regression analysis.



Observation:

- We can see that there isn't much correlation among the input features . Thus there is no Multicollinearity among the features. This is good.
- Few Features have greater than 0.5 Pearson Correlation with output feature. A value closer to 0 implies weaker correlation (exact 0 implying no correlation) A value closer to 1 implies stronger positive correlation A value closer to -1 implies stronger negative correlation.

Separating Independent and Dependent (target) features from Train Data:

- Then we separated the data into two categories input and output/Target variable.

```
X = df_train.drop('SalePrice', axis=1)
y = df_train['SalePrice'].values
```

Check for Outliers:

An outlier is a data point that is noticeably different from the rest. They represent errors in measurement, bad data collection, or simply show variables not considered when collecting the data. A value that "lies outside" (is much smaller or larger than) most of the other values in a set of data.

Box Plot

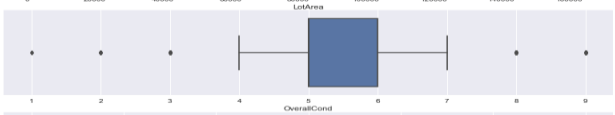
This is the visual representation of the depicting groups of numerical data through their quartiles. Boxplot is also used for detecting the outlier in the data set.

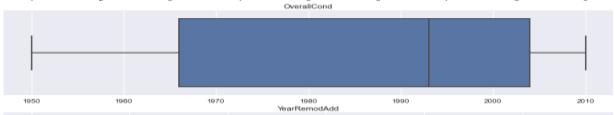
I used a box plot in this dataset because It captures the summary of the data efficiently with a simple box and whiskers and allows me to compare easily across groups.

Boxplots for the variables before removing the outliers is shown Below. After that we used the IQR method for removing the outliers.


25 50 75 100 125 150 175
MSSubClass

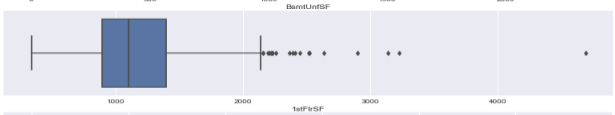

0 20000 40000 60000 80000 100000 120000 140000 160000
LotArea


1 2 3 4 5 6 7 8 9
OverallCond

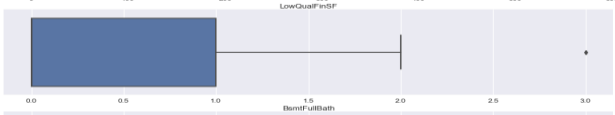

1950 1960 1970 1980 1990 2000 2010
YearRemodAdd

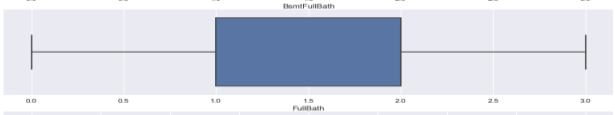

0 1000 2000 3000 4000 5000
BsmtFinSF1


0 500 1000 1500 2000
BsmtUnfSF


1000 2000 3000 4000
1stFlrSF

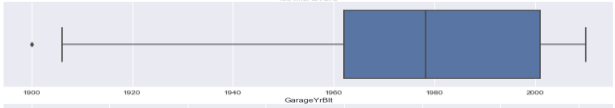

0 100 200 300 400 500 600
LowQualFinSF

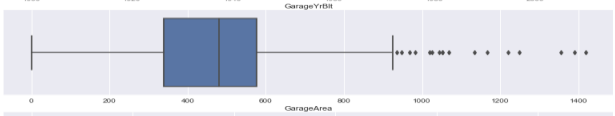

0.0 0.5 1.0 1.5 2.0 2.5 3.0
BsmtFullBath


0.0 0.5 1.0 1.5 2.0 2.5 3.0
FullBath


0 1 2 3 4 5 6 7 8
BedroomAbvGr

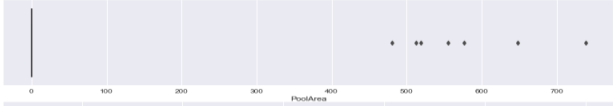

2 4 6 8 10 12 14
TotRmsAbvGrd

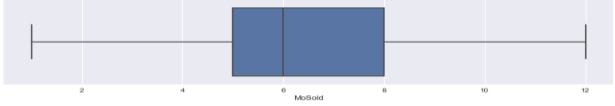

1900 1920 1940 1960 1980 2000
GarageYrBlt

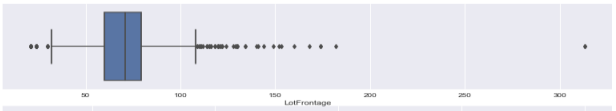

0 200 400 600 800 1000 1200 1400
GarageArea

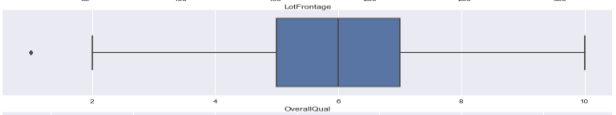

0 100 200 300 400 500
OpenPorchSF

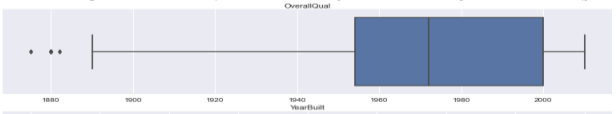

0 100 200 300 400 500
3SsnPorch


0 100 200 300 400 500 600 700
PoolArea


2 4 6 8 10 12
MoSold


50 100 150 200 250 300
LotFrontage


2 4 6 8 10
OverallQual

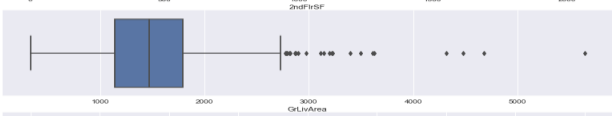

1880 1900 1920 1940 1960 1980 2000
YearBuilt


0 200 400 600 800 1000 1200 1400 1600
MasVnrArea

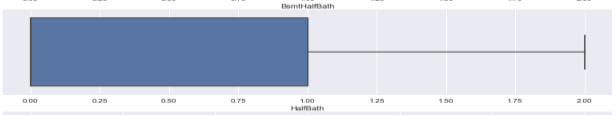

0 200 400 600 800 1000 1200 1400
BsmtFinSF2


0 1000 2000 3000 4000 5000 6000
TotalBsmtSF


0 500 1000 1500 2000
2ndFlrSF


1000 2000 3000 4000 5000
GrLivArea

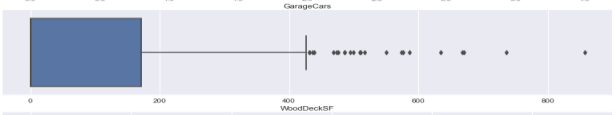

0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
BsmtHalfBath

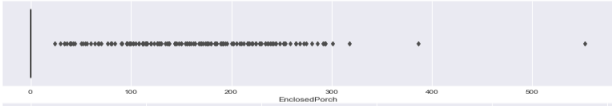

0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
HalfBath

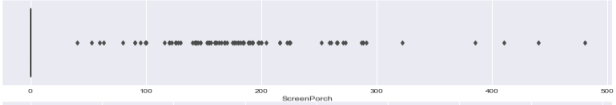

0.0 0.5 1.0 1.5 2.0 2.5 3.0
KitchenAbvGr


0.0 0.5 1.0 1.5 2.0 2.5 3.0
Fireplaces

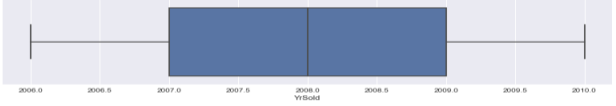

0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
GarageCars


0 200 400 600 800
WoodDeckSF

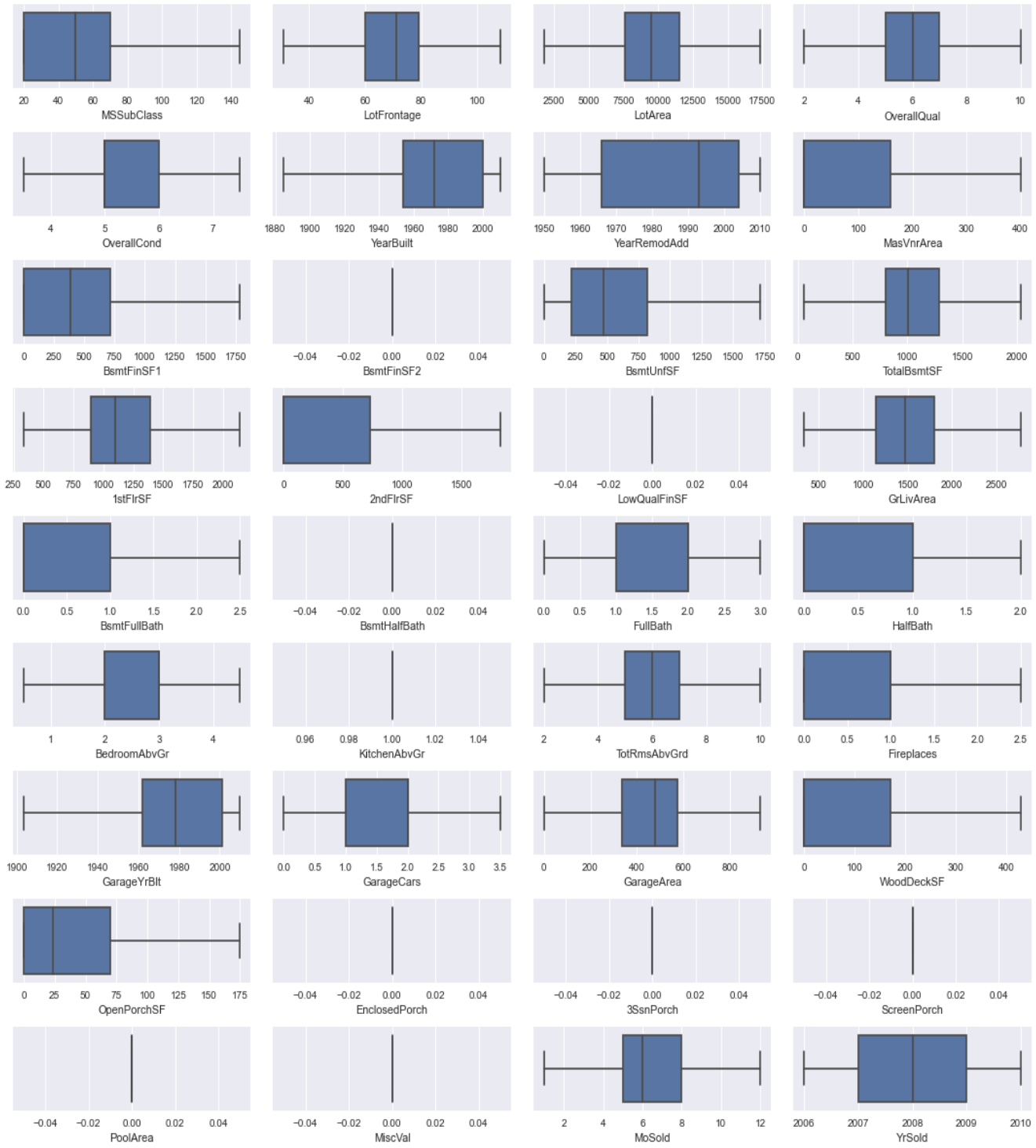

0 100 200 300 400 500
EnclosedPorch


0 100 200 300 400 500
ScreenPorch


0 2000 4000 6000 8000 10000 12000 14000 16000
MiscVal


2006.0 2006.5 2007.0 2007.5 2008.0 2008.5 2009.0 2009.5 2010.0
YrSold

BOXPLOTS AFTER REMOVING THE OUTLIERS



LABEL ENCODING:

After removing the outliers, we used the encoding method to turn the objective/categorical columns into integers/Numerical

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
#For train data
df_train=df_train.apply(LabelEncoder().fit_transform)
print('For train data')
print(df_train.head())
print('\n\n')
#For test data
df_test_le=df_test.apply(LabelEncoder().fit_transform)
print('For test data')
print(df_test_le.head())
```

Features Scaling / Standard Scaler:

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units.

```
X = df_train.drop('SalePrice', axis=1)
y = df_train['SalePrice'].values

# Performing Standard scaler
#For train data
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
#For test data
price_test = sc.fit_transform(df_test_le)
```

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods) We used different approaches from checking the dataset quality to building the model.

First, We will find the best random state on which I will get the maximum score.

```
# Finding Best Random State
maxScore = 0
maxRS = 0

for i in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=i)
    lr = LinearRegression()
    lr.fit(x_train,y_train)
    pred_train = lr.predict(x_train)
    pred_test = lr.predict(x_test)
    acc=r2_score(y_test,pred_test)
    if acc>maxScore:
        maxScore=acc
        maxRS=i
print('Best score is',maxScore,'on Random State',maxRS)
```

```
Best score is 0.9240949406322587 on Random State 84
```

Testing of Identified Approaches (Algorithms)

1) Linear Regression
2) Ridge
3) Lasso
4) Decision Tree
5) KNeighborsRegressor

```
model =
[LinearRegression(),Lasso(alpha=1.0),Ridge(alpha=1.0),DecisionTreeRegressor(cr
iterion='squared_error'),
         KNeighborsRegressor()]
for i in model:
    X_train1,X_test1,y_train1,y_test1 = train_test_split(X,y, test_size = 0.3,
random_state =maxRS)
    i.fit(X_train1,y_train1)
    pred = i.predict(X_test1)
    print('Train Score of', i , 'is:' , i.score(X_train1,y_train1))
    print("r2_score", r2_score(y_test1, pred))
    print("mean_squred_error", mean_squared_error(y_test1, pred))
    print("RMSE", np.sqrt(mean_squared_error(y_test1, pred)),"\n")
```

```
Train Score of LinearRegression() is: 0.8949552137686356
r2_score 0.9240949406322587
mean_squred_error 1818.3788124788762
RMSE 42.642453171444956

Train Score of Lasso() is: 0.8917401227582257
r2_score 0.9261040022874991
mean_squred_error 1770.249805304877
RMSE 42.074336659118906

Train Score of Ridge() is: 0.8949527937715197
r2_score 0.9242053679462534
mean_squred_error 1815.73341980335
RMSE 42.611423583393105

Train Score of DecisionTreeRegressor() is: 1.0
r2_score 0.7139434297385598
mean_squred_error 6852.760683760684
RMSE 82.78140300671814

Train Score of KNeighborsRegressor() is: 0.8761395853867664
r2_score 0.8369131698871278
mean_squred_error 3906.9021082621084
RMSE 62.505216648389506
```

From above test Lasso is having highest score with 1.0 but its having the MSE and RMSE very high compared to others. So we take Lasso with difference in test train scores less and also the low MSE and RMSE

Then we used ensemble technique to boost the performance of the model

RandomForestRegressor¶

```
from sklearn.ensemble import RandomForestRegressor
rf=RandomForestRegressor(n_estimators=100,random_state=maxRS,criterion='squared_error', min_samples_split=2, min_samples_leaf=1)
#RandomForestClassifier(100)---Default
rf.fit(X_train1,y_train1)
predrf=rf.predict(X_test1)
print('Train Score of', rf , 'is:' , rf.score(X_train1,y_train1))
print("r2_score", r2_score(y_test1, predrf))
print("mean_squred_error", mean_squared_error(y_test1, predrf))
print("RMSE", np.sqrt(mean_squared_error(y_test1, predrf)))
Train Score of RandomForestRegressor(random_state=84) is: 0.9813767323691924
r2_score 0.8930452761631832 mean_squred_error 2562.203433333333 RMSE
50.61821246679236
```

AdaBoostRegressor¶

```
from sklearn.ensemble import AdaBoostRegressor
ABr=AdaBoostRegressor(
base_estimator=Lasso(),n_estimators=50,learning_rate=1.0,loss='linear',random_
state=maxRS,)
#RandomForestClassifier(50)---Default
ABr.fit(X_train1,y_train1)
predAbr=ABr.predict(X_test1)
print('Train Score of', ABr , 'is:' , ABr.score(X_train1,y_train1))
print("r2_score", r2_score(y_test1, predAbr))
print("mean_squred_error", mean_squared_error(y_test1, predAbr))
print("RMSE", np.sqrt(mean_squared_error(y_test1, predAbr)))
```

```
Train Score of AdaBoostRegressor(base_estimator=Lasso(), random_state=84) is:
0.8622186362608182 r2_score 0.8681156446260295 mean_squred_error
3159.4167701999245 RMSE 56.20868945456676
```

GradientBoostingRegressor¶

```
from sklearn.ensemble import GradientBoostingRegressor
Gradient_Boost=GradientBoostingRegressor(n_estimators=100,loss='squared_error'
,learning_rate=0.1,criterion='friedman_mse', min_samples_split=2,
min_samples_leaf=1)
#GradientBoostingRegressor(100)---Default
Gradient_Boost.fit(X_train1,y_train1)
predgb=Gradient_Boost.predict(X_test1)
print('Train Score of', Gradient_Boost , 'is:' ,
Gradient_Boost.score(X_train1,y_train1))
print("r2_score", r2_score(y_test1, predgb))
print("mean_squred_error", mean_squared_error(y_test1, predgb))
print("RMSE", np.sqrt(mean_squared_error(y_test1, predgb)),"\n")
```

```
Train Score of GradientBoostingRegressor() is: 0.9679385176546844 r2_score
0.9175740653553743 mean_squred_error 1974.592660950527 RMSE 44.436388927888
```

From the above all three, Gradient boost regressor is having the low MSE, RMSE and high Train Score, But the difference between the test and train scores are high So we are selecting the AdaBoost as the difference between the test train scores are low

Hyper Parameter Tuning:

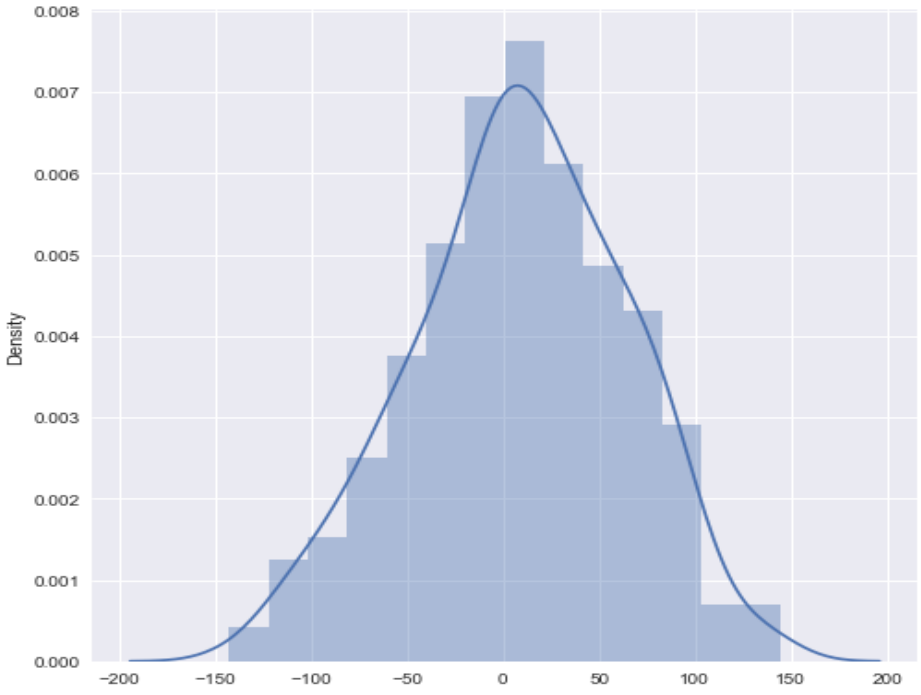
Hyperparameter tuning (or hyperparameter optimization) is the process of determining the right combination of hyperparameters that maximizes the model performance. It works by running multiple trials in a single training process.

We are using Randomsearchcv method for hyperparameter tuning to find best parameters for AdaBoostRegressor.

Hyper Parameter Tuning¶

```
Ada_Boost = AdaBoostRegressor()
Para ={'n_estimators' : [50, 100, 150, 200],
        'learning_rate' : [0.001, 0.01, 0.1, 1],
        'loss' : ["linear", "square", "exponential"],
        'random_state' : [21, 42, 104, 111]
        }
Rand_search = RandomizedSearchCV(Ada_Boost,Para,cv = 5,scoring = "r2",n_jobs
=-1,verbose = 2)
Rand_search.fit(X_train1,y_train1)
print(Rand_search.best_params_)
Fitting 5 folds for each of 10 candidates, totalling 50 fits {'random_state':
42, 'n_estimators': 100, 'loss': 'exponential', 'learning_rate': 1}
```

The predicted y value is having a normalized curve which is good.



Cross Validation:

Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

Cross Validation¶

```
best_Ada_Boost = AdaBoostRegressor(n_estimators= 100, loss= 'exponential',
learning_rate =1, random_state=104)
for i in range(2,11):
    cross_score = cross_val_score(best_Ada_Boost,X,y,cv = i,n_jobs = -1)
    print(i,"mean",cross_score.mean() ,"and STD" , cross_score.std())
```

```
2 mean 0.8443214554419438 and STD 0.009555629109518848 3 mean
0.8492086515080733 and STD 0.006408722736634356 4 mean 0.8476405781779779 and
STD 0.010834588125154383 5 mean 0.8494251839238531 and STD
0.011282459225452415 6 mean 0.8479310962999165 and STD 0.015742375342086924 7
mean 0.845274131430127 and STD 0.021166154396682322 8 mean 0.8469293653677799
and STD 0.024957283297494973 9 mean 0.8476671999403956 and STD
0.027442190645623893 10 mean 0.8496127177699639 and STD 0.02682516694571256
```

Applying cross validation score 10

```
# Selecting Cv score as 10
# Cross validate of GradientBoostingRegressor using cv=10
from sklearn.model_selection import cross_val_score
score=cross_val_score(best_Ada_Boost,X,y,cv=10,scoring='r2')
print('Score:', score)
print('Mean Score:', score.mean())
print('Standard Deviation:', score.std())
```

```
Score: [0.8410553 0.87217638 0.87497133 0.81981079 0.88227262 0.79766244
0.86755785 0.87187117 0.83019316 0.83855615] Mean Score: 0.8496127177699639
Standard Deviation: 0.02682516694571256
```

Plotting y_test1 vs predictions:

- Simply plotting our predictions vs the true values.
- Ideally, it should be a straight line.

Saving the Model:

We are saving the model by using python's pickle library. It will be used further for the predictio n. Also, we have loaded the prediction file to predict the target of the test data.

Saving Model¶

```
import pickle
# Saving the AdaBoostRegressor
best_Ada_Boost.fit(X,y)
pred = best_Ada_Boost.predict(price_test)
# Saving model
filename = "House_Saleprice_Prediction.pkl"
with open(filename,"wb") as f:
    pickle.dump(best_Ada_Boost,f)
```

Loading the Model

```
loaded_model=pickle.load(open('House_Saleprice_Prediction.pkl','rb'))
```

```
Test_pred=loaded_model.predict(price_test)
Y_tst=pd.DataFrame(data=Test_pred)

Y_tst
```

Here we have predicted the target of test dataset and checked the shape of test dataset and target of test dataset to join them.

CONCLUSION:

So, as we saw that we have done a complete EDA process, getting data insights, feature engineering, and data visualization as well so after all these steps one can go for the prediction using machine learning model-making steps.

We have training and test file separately available with us. we have both numerical and categorical data types features in both datasets and the dependent variable of train data i.e. the price is the numerical data type. So, I applied the regression method for prediction.

Once data has been cleaned for both test and train datasets, Label encoding is applied to them to convert them into Numerical ones. I trained the model on five different algorithms but for most of the models, train and test data was having a high variance, and the model was overfitting.

Only Ada Boost regressor worked well out of all the models, as there was less difference between train score and test score and RMSE was also low hence I used it as the final model and have done further processing.

After applying hyperparameter tuning I got an accuracy(r2_score) of 86% from the Ada Boost Regressor model which is a good score. Then I applied that score to the test dataset to get the target variable which is price.