FLIP ROBO

Surprise Housing Price Prediction Project Use Case Report

Submitted by:

Sri Bharath Vasa

ACKNOWLEDGMENT

Thanks to Flip Robo and Datatrained for guiding and giving us projects in helping in learning the data science course. Thanks to my SME(Shubham Yadav) for guiding us in the projects.

Also, I have utilized a few external resources that helped me to complete the project. I ensured that I learn from the samples and modify things according to my project requirement. All the external resources that were used in creating this project are listed below:

1) https://www.google.com/
2) https://www.youtube.com/
3) https://scikit-learn.org/stable/user_guide.html
4) https://github.com/
5) https://www.kaggle.com/
6) https://medium.com/
7) https://towardsdatascience.com/
8) https://www.analyticsvidhya.com/

INTRODUCTION

- Business Problem Framing

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.

We are required to 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.

- Conceptual Background of the Domain Problem

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:

1. Which variables are important to predict the price of a variable?
2. How do these variables describe the price of the house?

- Review of Literature

Based on the sample data provided to us from our client database where we have understood that the company is looking at prospective properties to buy houses to enter the market. The data set explains it is a regression problem as we need 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. Also, we have other independent features that would help to decide which all variables are important to predict the price of the variable and how do these variables describe the price of the house.

- Motivation for the Problem Undertaken

Our main objective of doing this project is to build a model to predict the house prices with the help of other supporting features. We are going to predict by using Machine Learning algorithms.
The sample data is provided to us from our client database. In order to improve the selection of customers, the client wants some predictions that could help them in further investment and improvement in selection of customers.
House Price Index is commonly used to estimate the changes in housing price. Since housing price is strongly correlated to other factors such as location, area, population, it requires other information apart from HPI to predict individual housing price.
There has been a considerably large number of papers adopting traditional machine learning approaches to predict housing prices accurately, but they rarely concern themselves with the performance of individual models and neglect the less popular yet complex models.
As a result, to explore various impacts of features on prediction methods, this paper will apply both traditional and advanced machine learning approaches to investigate the difference among several advanced models. This paper will also comprehensively validate multiple techniques in model implementation on regression and provide an optimistic result for housing price prediction.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

  We are building a model in Machine Learning to predict the actual value of the prospective properties and decide whether to invest in them or not. So, this model will help us to determine which variables are important to predict the price of variables & also how do these variables describe the price of the house. This will help to determine the price of houses with the available independent variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns.

  Regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome variable') and one or more independent variables (often called 'predictors', 'covariates', or 'features'). The most common form of regression analysis is linear regression, in which one finds the line (or a more complex linear combination) that most closely fits the data according to a specific mathematical criterion. For specific mathematical reasons this allows the researcher to estimate the

  conditional expectation of the dependent variable when the independent variables take on a given set of values.

  Regression analysis is also a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (predictor). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables.

- ## Data Sources and their formats

  Data set provided by Flip Robo was in the format of CSV (Comma Separated Values). The dimension of data is 1168 rows and 81 columns. There are 2 data sets that are given. One is training data and one is testing data.

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

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

- Data Pre-processing Done

Data pre-processing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre- process our data before feeding it into our model. Therefore, it is the first and crucial step while creating a machine learning model. I have used some following pre-processing steps:

a. Loading the training dataset as a dataframe
b. Used pandas to set display I ensuring we do not see any truncated information
c. Checked the number of rows and columns present in our training dataset
d. Checked for missing data and the number of rows with null values
e. Verified the percentage of missing data in each column and decided to discard the ones that have more than 50% of null values
f. Dropped all the unwanted columns and duplicate data present in our dataframe
g. Separated categorical column names and numeric column names in separate list variables for ease in visualization
h. Checked the unique values information in each column to get a gist for categorical data
i. Performed imputation to fill missing data using mean on numeric data and mode for categorical data columns
j. Used Pandas Profiling during the visualization phase along with pie plot, count plot, scatter plot and the others
k. With the help of ordinal encoding technique converted all object datatype columns to numeric datatype
l. Thoroughly checked for outliers and skewness information
m. With the help of heatmap, correlation bar graph was able to understand the Feature vs Label relativity and insights on multicollinearity amongst the feature columns

n. Separate feature and label data to ensure feature scaling is performed avoiding any kind of biasness
o. Checked for the best random state to be used on our Regression Machine Learning model pertaining to the feature importance details
p. Finally created a regression model function along with evaluation metrics to pass through various model formats

- Data Inputs- Logic- Output Relationships

When we loaded the training dataset, we had to go through various data pre-processing steps to understand what was given to us and what we were expected to predict for the project. When it comes to logical part the domain expertise of understanding how real estate works and how we are supposed to cater to the customers came in handy to train the model with the modified input data. In Data Science community there is a saying “Garbage In Garbage Out” therefore we had to be very cautious and spent almost 80% of our project building time in understanding each and every aspect of the data how they were related to each other as well as our target label.

With the objective of predicting hosing sale prices accurately we had to make sure that a model was built that understood the customer priorities trending in the market imposing those norms when a relevant price tag was generated. I tried my best to retain as much data possible that was collected but I feel discarding columns that had lots of missing data was good. I did not want to impute data and then cause a biasness in the machine learning model from values that did not come from real people.

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

The assumption part for me was relying strictly on the data provided to me and taking into consideration that the separate training and testing datasets were obtained from real people surveyed for their preferences and how reasonable a price for a house with various features inclining to them were.

- Hardware and Software Requirements and Tools Used

Hardware Used:

i. RAM: 12 GB
ii. CPU: 11th Gen Intel(R) Core TM) i5-1135G7 @ 2.40GHz
iii. GPU: intel iRISXe Graphics card

Software Used:

i. Programming language: Python
ii. Distribution: Anaconda Navigator
iii. Browser based language shell: Jupyter Notebook

Libraries/Packages Used:

Pandas, NumPy, matplotlib, seaborn, scikit-learn and pandas_profiling

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

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

For this project we need to predict the sale price of houses, means our target column is continuous so this is a regression problem. I have used various regression algorithms and tested for the prediction. By doing various evaluations I have selected Extra Trees Regressor as best suitable algorithm for our final model as it is giving good r2-score and least difference in r2-score and CV-score among all the algorithms used. Other regression algorithms are also giving me good accuracy but some are over-fitting and some are with under-fitting the results which may be because of less amount of data.

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

Once we are able to get our desired final model, we can ensure to save that model before loading the testing data and start performing the data pre-processing as the training dataset and obtaining the predicted sale price values out of the Regression Machine Learning Model.

- Testing of Identified Approaches (Algorithms)

The algorithms used on training and test data are as follows:

A. Linear Regression Model
B. Ridge Regularization Regression Model
C. Lasso Regularization Regression Model
D. Support Vector Regression Model
E. Decision Tree Regression Model
F. Random Forest Regression Model
G. K Nearest Neighbours Regression Model
H. Gradient Boosting Regression Model
I. Ada Boost Regression Model
J. Extra Trees Regression Model

- Running and Evaluating Selected Models
  Used the above said 10 models and tested for the best random state number among 1-1000 and defined a function to evaluate and train the model.

Finding the best random state for building Regression Models

```
maxAccu=0
maxRS=0

for i in range(1, 1000):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=i)
    lr=LinearRegression()
    lr.fit(X_train, Y_train)
    pred = lr.predict(X_test)
    r2 = r2_score(Y_test, pred)

    if r2>maxAccu:
        maxAccu=r2
        maxRS=i

print("Best R2 score is", maxAccu,"on Random State", maxRS)
```

```
Best R2 score is 0.8856355344351948 on Random State 340
```

First, we start by importing the needed libraries and load the train dataset and look for the first and last data in the dataset.

```python
import warnings
warnings.simplefilter("ignore")
warnings.filterwarnings("ignore")
import joblib

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import pandas_profiling
from sklearn import metrics
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor

from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

```python
train_df = pd.read_csv("train.csv")
```

I am importing the train dataset comma separated values file and storing it into our dataframe for further usage.

```python
train_df # checking the first 5 and last 5 rows
```

	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

Here we are taking a look at the first 5 and last 5 rows of our dataset. It shows that we have a total of 1168 rows and 81 columns present in our dataframe. In the above cell we can see the training dataset which includes the target label "SalePrice" column and the remaining feature columns that determine or help in predicting the sales. Since sales is a continous value it makes this to be a Regression problem!

EDA (Exploratory Data Analysis)

```
pd.set_option('display.max_columns', None) # show all columns in a dataframe
pd.set_option('display.max_rows', None) # show all rows in a dataframe
```

Ensuring that in future observations we do not have any truncated information being displayed in our Jupter Notebook.

```
print("We have {} Rows and {} Columns in our dataframe".format(train_df.shape[0], train_df.shape[1]))
train_df.head()
```

```
We have 1168 Rows and 81 Columns in our dataframe
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Con
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm	Nor
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm	Nor
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm	Nor
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	Nor
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norm	Nor

Then we are checking for the null values present in the train data.

```
train_df.isna().sum() # checking for missing values
```

```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage     214
LotArea           0
Street            0
Alley          1091
LotShape          0
LandContour       0
Utilities         0
LotConfig         0
LandSlope         0
Neighborhood      0
Condition1        0
Condition2        0
BldgType          0
HouseStyle        0
OverallQual       0
OverallCond       0
YearBuilt         0
YearRemodAdd      0
RoofStyle         0
RoofMatl          0
Exterior1st       0
Exterior2nd       0
MasVnrType        7
MasVnrArea        7
ExterQual         0
ExterCond         0
Foundation        0
BsmtQual         30
BsmtCond         30
BsmtExposure     31
BsmtFinType1     30
BsmtFinSF1        0
BsmtFinType2     31
BsmtFinSF2        0
BsmtUnfSF         0
TotalBsmtSF       0
Heating           0
HeatingQC         0
CentralAir        0
```

```
Electrical          0
1stFlrSF            0
2ndFlrSF            0
LowQualFinSF        0
GrLivArea           0
BsmtFullBath        0
BsmtHalfBath        0
FullBath            0
HalfBath            0
BedroomAbvGr        0
KitchenAbvGr        0
KitchenQual         0
TotRmsAbvGrd        0
Functional          0
Fireplaces          0
FireplaceQu       551
GarageType         64
GarageYrBlt        64
GarageFinish       64
GarageCars          0
GarageArea          0
GarageQual         64
GarageCond         64
PavedDrive          0
WoodDeckSF          0
OpenPorchSF         0
EnclosedPorch       0
3SsnPorch           0
ScreenPorch         0
PoolArea            0
PoolQC           1161
Fence             931
MiscFeature      1124
MiscVal             0
MoSold              0
YrSold              0
SaleType            0
SaleCondition       0
SalePrice           0
dtype: int64
```

Using the isna() and sum functions together on our dataframe we can take a look at missing data information. It looks like we do have missing values present in few of our columns. However, we will check the percentage of missing information before we began treating them.

Then we are checking the percentage of missing data and dropping the non-required columns and checking for duplicity of data in the rows and columns.

So the column names and the percentage of missing data can be seen below: LotFrontage : 18.322 % Alley : 93.408 % MasVnrType : 0.599 % MasVnrArea : 0.599 % BsmtQual : 2.568 % BsmtCond : 2.568 % BsmtExposure : 2.654 % BsmtFinType1 : 2.568 % BsmtFinType2 : 2.654 % FireplaceQu : 47.175 % GarageType : 5.479 % GarageYrBlt : 5.479 % GarageFinish : 5.479 % GarageQual : 5.479 % GarageCond : 5.479 % PoolQC : 99.401 % Fence : 79.709 % MiscFeature : 96.233 %

Now I have decided to drop columns that have most of their values or almost of their values filled with a "null". The columns that I am going to lose are as follows: Alley : 93.408 % FireplaceQu : 47.175 % PoolQC : 99.401 % Fence : 79.709 % MiscFeature : 96.233 %

```
# data preprocessing 1
train_df.drop(["Alley", "FireplaceQu", "PoolQC", "Fence", "MiscFeature"], axis=1, inplace=True)
```

I have successfully got rid of all the columns that had most of the values filled with null because treating them would mean manually entering data that was not originally collected properly and that would only make the model biased towards the few information we could get hold of.

```
print("We had {} Rows and {} Columns before dropping duplicates.".format(train_df.shape[0], train_df.shape[1]))
train_df.drop_duplicates(inplace=True)
print("We have {} Rows and {} Columns after dropping duplicates.".format(train_df.shape[0], train_df.shape[1]))
```

```
We had 1168 Rows and 76 Columns before dropping duplicates.
We have 1168 Rows and 76 Columns after dropping duplicates.
```

With the drop_duplicates option I was trying to get rid of all the duplicate values present in our dataset. However, we can see that there are no duplicate data present in our dataset. I tried doing the same thing for dropping null values but we were losing lots of data.

Further we are checking we are checking the datatype of each column and uniqueness of values in the dataset.

```
train_df.info()
```

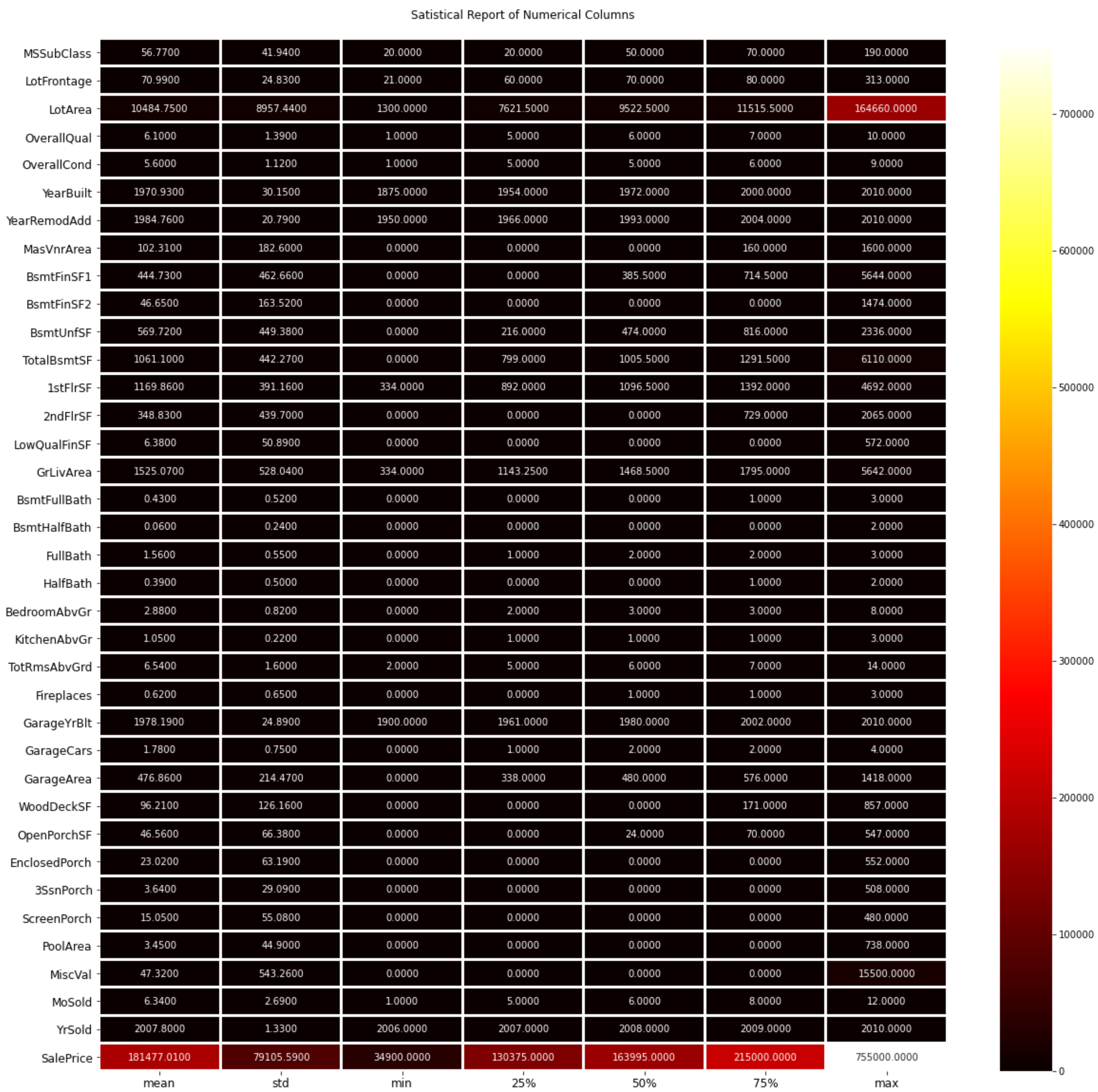
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1168 entries, 0 to 1167
Data columns (total 76 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1168 non-null   int64
 1   MSSubClass     1168 non-null   int64
 2   MSZoning       1168 non-null   object
 3   LotFrontage    954 non-null    float64
 4   LotArea        1168 non-null   int64
 5   Street         1168 non-null   object
 6   LotShape       1168 non-null   object
 7   LandContour    1168 non-null   object
 8   Utilities      1168 non-null   object
 9   LotConfig      1168 non-null   object
 10  LandSlope      1168 non-null   object
 11  Neighborhood   1168 non-null   object
 12  Condition1     1168 non-null   object
 13  Condition2     1168 non-null   object
 14  BldgType       1168 non-null   object
 15  HouseStyle     1168 non-null   object
 16  OverallQual    1168 non-null   int64
 17  OverallCond    1168 non-null   int64
 18  YearBuilt      1168 non-null   int64
 19  YearRemodAdd   1168 non-null   int64
 20  RoofStyle      1168 non-null   object
 21  RoofMatl       1168 non-null   object
 22  Exterior1st    1168 non-null   object
 23  Exterior2nd    1168 non-null   object
 24  MasVnrType     1161 non-null   object
 25  MasVnrArea     1161 non-null   float64
 26  ExterQual      1168 non-null   object
 27  ExterCond      1168 non-null   object
 28  Foundation     1168 non-null   object
 29  BsmtQual       1138 non-null   object
 30  BsmtCond       1138 non-null   object
 31  BsmtExposure   1137 non-null   object
 32  BsmtFinType1   1138 non-null   object
 33  BsmtFinSF1     1168 non-null   int64
 34  BsmtFinType2   1137 non-null   object
 35  BsmtFinSF2     1168 non-null   int64
```

In the above cell we see that there are 3 columns with float datatype, 35 columns with integer datatype and 38 columns with object datatype.

```
train_df.nunique().to_frame("Unique Values")
```

	Unique Values
Id	1168
MSSubClass	15
MSZoning	5
LotFrontage	106
LotArea	892
Street	2
LotShape	4
LandContour	4
Utilities	1
LotConfig	5
LandSlope	3
Neighborhood	25
Condition1	9
Condition2	8
BldgType	5
HouseStyle	8
OverallQual	10
OverallCond	9
YearBuilt	110
YearRemodAdd	61
RoofStyle	6
RoofMatl	8
Exterior1st	14

```
# visualizing the statistical description of numeric datatype columns

plt.figure(figsize = (20,20))
sns.heatmap(round(train_df.describe()[1:].transpose(),2), linewidth = 2, annot= True, fmt = ".4f", cmap="hot")
plt.title("Satistical Report of Numerical Columns\n")
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
plt.show()
```



Satistical Report of Numerical Columns

	mean	std	min	25%	50%	75%	max
MSSubClass	56.7700	41.9400	20.0000	20.0000	50.0000	70.0000	190.0000
LotFrontage	70.9900	24.8300	21.0000	60.0000	70.0000	80.0000	313.0000
LotArea	10484.7500	8957.4400	1300.0000	7621.5000	9522.5000	11515.5000	164660.0000
OverallQual	6.1000	1.3900	1.0000	5.0000	6.0000	7.0000	10.0000
OverallCond	5.6000	1.1200	1.0000	5.0000	5.0000	6.0000	9.0000
YearBuilt	1970.9300	30.1500	1875.0000	1954.0000	1972.0000	2000.0000	2010.0000
YearRemodAdd	1984.7600	20.7900	1950.0000	1966.0000	1993.0000	2004.0000	2010.0000
MasVnrArea	102.3100	182.6000	0.0000	0.0000	0.0000	160.0000	1600.0000
BsmtFinSF1	444.7300	462.6600	0.0000	0.0000	385.5000	714.5000	5644.0000
BsmtFinSF2	46.6500	163.5200	0.0000	0.0000	0.0000	0.0000	1474.0000
BsmtUnfSF	569.7200	449.3800	0.0000	216.0000	474.0000	816.0000	2336.0000
TotalBsmtSF	1061.1000	442.2700	0.0000	799.0000	1005.5000	1291.5000	6110.0000
1stFlrSF	1169.8600	391.1600	334.0000	892.0000	1096.5000	1392.0000	4692.0000
2ndFlrSF	348.8300	439.7000	0.0000	0.0000	0.0000	729.0000	2065.0000
LowQualFinSF	6.3800	50.8900	0.0000	0.0000	0.0000	0.0000	572.0000
GrLivArea	1525.0700	528.0400	334.0000	1143.2500	1468.5000	1795.0000	5642.0000
BsmtFullBath	0.4300	0.5200	0.0000	0.0000	0.0000	1.0000	3.0000
BsmtHalfBath	0.0600	0.2400	0.0000	0.0000	0.0000	0.0000	2.0000
FullBath	1.5600	0.5500	0.0000	1.0000	2.0000	2.0000	3.0000
HalfBath	0.3900	0.5000	0.0000	0.0000	0.0000	1.0000	2.0000
BedroomAbvGr	2.8800	0.8200	0.0000	2.0000	3.0000	3.0000	8.0000
KitchenAbvGr	1.0500	0.2200	0.0000	1.0000	1.0000	1.0000	3.0000
TotRmsAbvGrd	6.5400	1.6000	2.0000	5.0000	6.0000	7.0000	14.0000
Fireplaces	0.6200	0.6500	0.0000	0.0000	1.0000	1.0000	3.0000
GarageYrBlt	1978.1900	24.8900	1900.0000	1961.0000	1980.0000	2002.0000	2010.0000
GarageCars	1.7800	0.7500	0.0000	1.0000	2.0000	2.0000	4.0000
GarageArea	476.8600	214.4700	0.0000	338.0000	480.0000	576.0000	1418.0000
WoodDeckSF	96.2100	126.1600	0.0000	0.0000	0.0000	171.0000	857.0000
OpenPorchSF	46.5600	66.3800	0.0000	0.0000	24.0000	70.0000	547.0000
EnclosedPorch	23.0200	63.1900	0.0000	0.0000	0.0000	0.0000	552.0000
3SsnPorch	3.6400	29.0900	0.0000	0.0000	0.0000	0.0000	508.0000
ScreenPorch	15.0500	55.0800	0.0000	0.0000	0.0000	0.0000	480.0000
PoolArea	3.4500	44.9000	0.0000	0.0000	0.0000	0.0000	738.0000
MiscVal	47.3200	543.2600	0.0000	0.0000	0.0000	0.0000	15500.0000
MoSold	6.3400	2.6900	1.0000	5.0000	6.0000	8.0000	12.0000
YrSold	2007.8000	1.3300	2006.0000	2007.0000	2008.0000	2009.0000	2010.0000
SalePrice	181477.0100	79105.5900	34900.0000	130375.0000	163995.0000	215000.0000	755000.0000

Using the above visualization on the describe method we are able to observe that our target label "SalePrice" has values that are higher than the other feature column details.

Checking the correlation values for all the numeric datatype columns.

```
train_df.corr()
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUn
MSSubClass	1.000000	-0.365220	-0.124151	0.070462	-0.056978	0.023988	0.056618	0.027868	-0.052236	-0.062403	-0.13417
LotFrontage	-0.365220	1.000000	0.557257	0.247809	-0.053345	0.118554	0.096050	0.202225	0.247780	0.002514	0.12394
LotArea	-0.124151	0.557257	1.000000	0.107188	0.017513	0.005506	0.027228	0.121448	0.221851	0.056656	0.00660
OverallQual	0.070462	0.247809	0.107188	1.000000	-0.083167	0.575800	0.555945	0.409163	0.219643	-0.040893	0.30867
OverallCond	-0.056978	-0.053345	0.017513	-0.083167	1.000000	-0.377731	0.080669	-0.137882	-0.028810	0.044336	-0.14638
YearBuilt	0.023988	0.118554	0.005506	0.575800	-0.377731	1.000000	0.592829	0.323006	0.227933	-0.027682	0.15555
YearRemodAdd	0.056618	0.096050	0.027228	0.555945	0.080669	0.592829	1.000000	0.181869	0.114430	-0.044694	0.17473
MasVnrArea	0.027868	0.202225	0.121448	0.409163	-0.137882	0.323006	0.181869	1.000000	0.267066	-0.065723	0.10985
BsmtFinSF1	-0.052236	0.247780	0.221851	0.219643	-0.028810	0.227933	0.114430	0.267066	1.000000	-0.052145	-0.49986
BsmtFinSF2	-0.062403	0.002514	0.056656	-0.040893	0.044336	-0.027682	-0.044694	-0.065723	-0.052145	1.000000	-0.21358
BsmtUnfSF	-0.134170	0.123943	0.006600	0.308676	-0.146384	0.155559	0.174732	0.109850	-0.499861	-0.213580	1.00000
TotalBsmtSF	-0.214042	0.386261	0.259733	0.528285	-0.162481	0.386265	0.280720	0.366833	0.518940	0.098167	0.41418
1stFlrSF	-0.227927	0.448186	0.312843	0.458758	-0.134420	0.279450	0.233384	0.339938	0.445876	0.093442	0.30743
2ndFlrSF	0.300366	0.099250	0.059803	0.316624	0.036668	0.011834	0.155102	0.173358	-0.127656	-0.092049	0.00273
LowQualFinSF	0.053737	0.007885	-0.001915	-0.039295	0.041877	-0.189044	-0.072526	-0.070518	-0.070932	-0.000577	0.03008
GrLivArea	0.086448	0.410414	0.281360	0.599700	-0.065006	0.198644	0.295048	0.387891	0.217160	-0.007484	0.23292
BsmtFullBath	0.004556	0.104255	0.142387	0.101732	-0.039680	0.164983	0.104643	0.086720	0.645126	0.163518	-0.43174
BsmtHalfBath	0.008207	0.001528	0.059282	-0.030702	0.091016	-0.028161	-0.011375	0.014198	0.063895	0.093692	-0.09037
FullBath	0.140807	0.189321	0.123197	0.548824	-0.171931	0.471264	0.444446	0.268545	0.054511	-0.060773	0.27219
HalfBath	0.168423	0.053168	0.007271	0.296134	-0.052125	0.243227	0.194943	0.200926	0.015767	-0.023734	-0.04402
BedroomAbvGr	-0.013283	0.264010	0.117351	0.099639	0.028393	-0.080639	-0.035847	0.091717	-0.114888	-0.005788	0.15605
KitchenAbvGr	0.283506	-0.002890	-0.013075	-0.178220	-0.076047	-0.167869	-0.139943	-0.038281	-0.065450	-0.034411	0.01553
TotRmsAbvGrd	0.051179	0.351969	0.184546	0.432579	-0.039952	0.095476	0.206923	0.279391	0.043499	-0.033702	0.23704
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

First, we checked for the presence of missing values and got it treated and using the below function we can see that there are no more null values present in the dataset.

Below, using the mean and mode method we can see that no null values present in the dataset.

Filling the missing values using mean and mode options

```
# data preprocessing 3
mode = ["MasVnrType", "BsmtQual", "BsmtCond", "BsmtExposure", "BsmtFinType1", "BsmtFinType2", "GarageType",
        "GarageFinish", "GarageQual", "GarageCond"]
mean = ["LotFrontage", "MasVnrArea", "GarageYrBlt"]

for i in mode:
    train_df[i] = train_df[i].fillna(train_df[i].mode()[0])

for j in mean:
    train_df[j] = train_df[j].fillna(train_df[j].mean())

print("Missing values count after filling the data")
print(train_df.isna().sum())
```

```
Missing values count after filling the data
MSSubClass       0
MSZoning         0
LotFrontage      0
LotArea          0
Street           0
LotShape         0
LandContour      0
LotConfig        0
LandSlope        0
Neighborhood     0
Condition1       0
Condition2       0
BldgType         0
HouseStyle       0
OverallQual      0
OverallCond      0
YearBuilt        0
YearRemodAdd     0
RoofStyle        0
RoofMatl         0
Exterior1st      0
Exterior2nd      0
MasVnrType       0
MasVnrArea       0
ExterQual        0
ExterCond        0
Foundation       0
```

Visualization

```
train_df.nunique().sort_values()
```

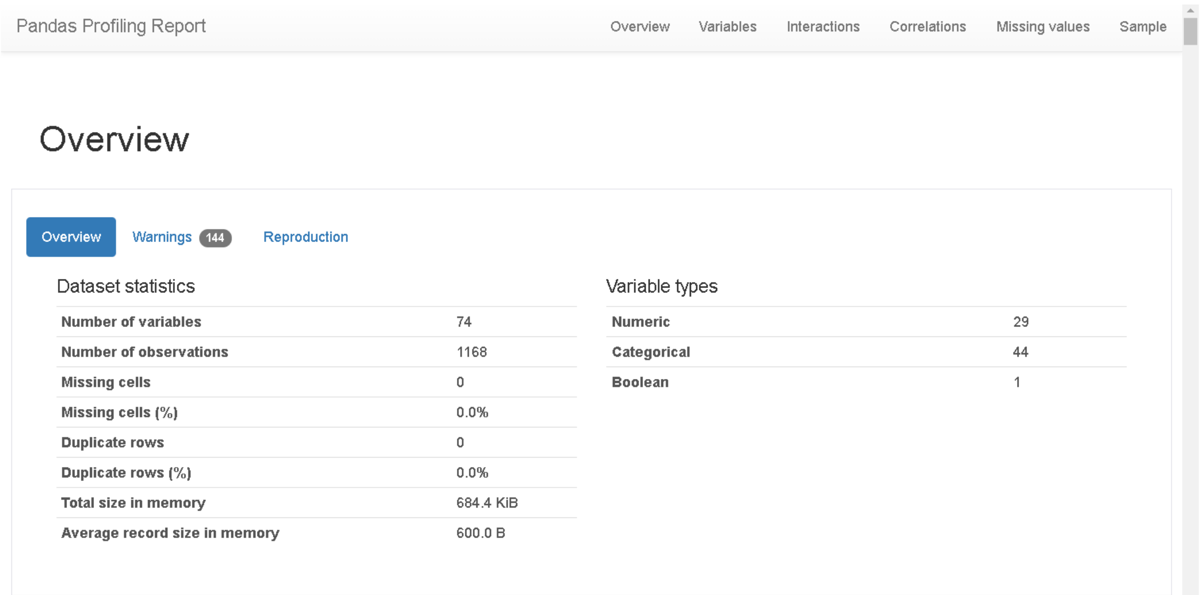
```
CentralAir        2
Street            2
GarageFinish      3
HalfBath          3
LandSlope         3
BsmtHalfBath      3
PavedDrive        3
BsmtExposure      4
BsmtCond          4
BsmtQual          4
MasVnrType        4
Fireplaces        4
KitchenQual       4
ExterQual         4
FullBath          4
KitchenAbvGr      4
BsmtFullBath      4
LotShape          4
LandContour       4
Electrical        5
MSZoning          5
YrSold            5
ExterCond         5
LotConfig         5
BldgType          5
GarageCond        5
HeatingQC         5
GarageQual        5
GarageCars        5
GarageType        6
SaleCondition     6
BsmtFinType2      6
Heating           6
Foundation        6
RoofStyle         6
BsmtFinType1      6
Functional        7
RoofMatl          8
PoolArea          8
HouseStyle        8
Condition2        8
```

```
Condition2          8
BedroomAbvGr        8
OverallCond         9
Condition1          9
SaleType            9
OverallQual        10
TotRmsAbvGrd       12
MoSold             12
Exterior1st        14
MSSubClass         15
Exterior2nd        15
3SsnPorch          18
MiscVal            20
LowQualFinSF       21
Neighborhood       25
YearRemodAdd       61
ScreenPorch        65
GarageYrBlt        98
EnclosedPorch     106
LotFrontage       107
YearBuilt         110
BsmtFinSF2        122
OpenPorchSF       176
WoodDeckSF        244
MasVnrArea        284
2ndFlrSF          351
GarageArea        392
BsmtFinSF1        551
SalePrice         581
TotalBsmtSF       636
1stFlrSF          669
BsmtUnfSF         681
GrLivArea         746
LotArea           892
dtype: int64
```

I have sorted the unique values column name list to see the one's with least unique values and the one's with the most in them.

Then we are profiling the pandas and getting the output.

```
pandas_profiling.ProfileReport(train_df)
```



Pandas Profiling Report — Overview | Variables | Interactions | Correlations | Missing values | Sample

Overview

Overview | Warnings 144 | Reproduction

Dataset statistics

Number of variables	74
Number of observations	1168
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	684.4 KiB
Average record size in memory	600.0 B

Variable types

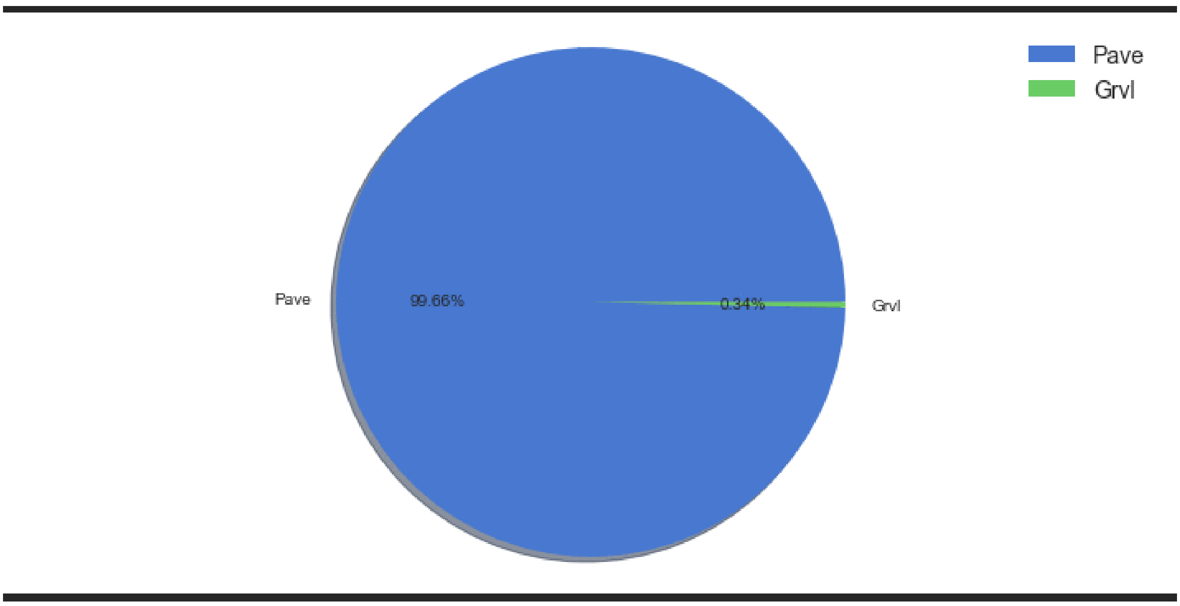
Numeric	29
Categorical	44
Boolean	1

Then created Pie plots', Count plots and Scatter plots to get further insight about the various variables featuring in our training dataset.

Code:

```
plt.style.use('seaborn-muted')
def generate_pie(x):
    plt.style.use('seaborn-white')
    plt.figure(figsize=(10,5))
    plt.pie(x.value_counts(), labels=x.value_counts().index, shadow=True, autopct='%1.2f%%')
    plt.legend(prop={'size':14})
    plt.axis('equal')
    plt.tight_layout()
    return plt.show()

for i in train_df[single]:
    print(f"Single digit category column name:", i)
    generate_pie(train_df[i])
```

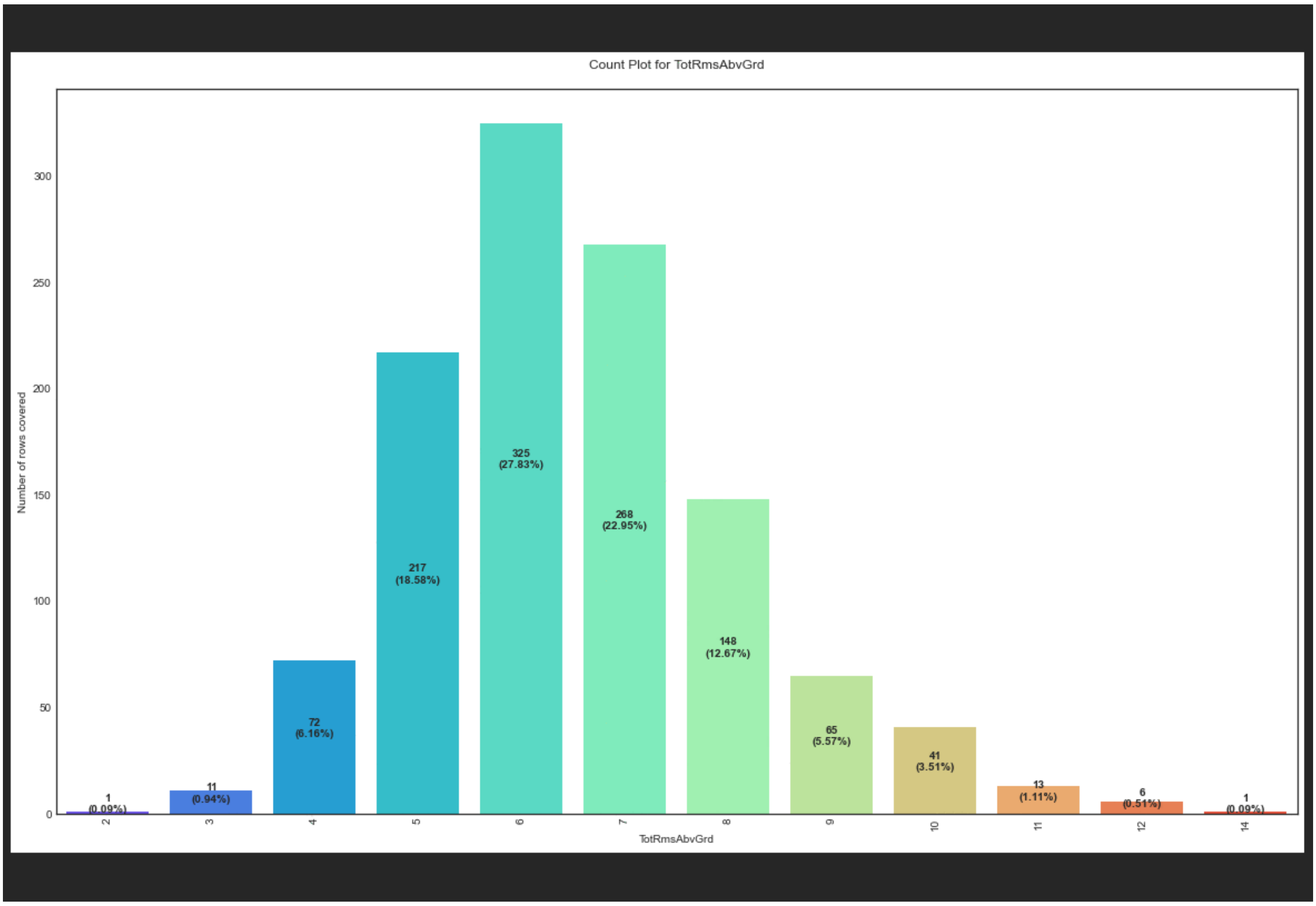
Output:



Code:

```
for col in train_df[double]:
    plt.figure(figsize=(20,12))
    col_name = col
    values = train_df[col_name].value_counts()
    index = 0
    ax = sns.countplot(train_df[col_name], palette="rainbow")

    for i in ax.patches:
        h = i.get_height() # getting the count of each value
        t = len(train_df[col_name]) # getting the total number of records using length
        s = f"{h}\n({round(h*100/t,2)}%)" # making the string for displaying in count bar
        plt.text(index, h/2, s, ha="center", fontweight="bold")
        index += 1

    plt.title(f"Count Plot for {col_name}\n")
    plt.xlabel(col_name)
    plt.ylabel(f"Number of rows covered")
    plt.xticks(rotation=90)
    plt.show()
```

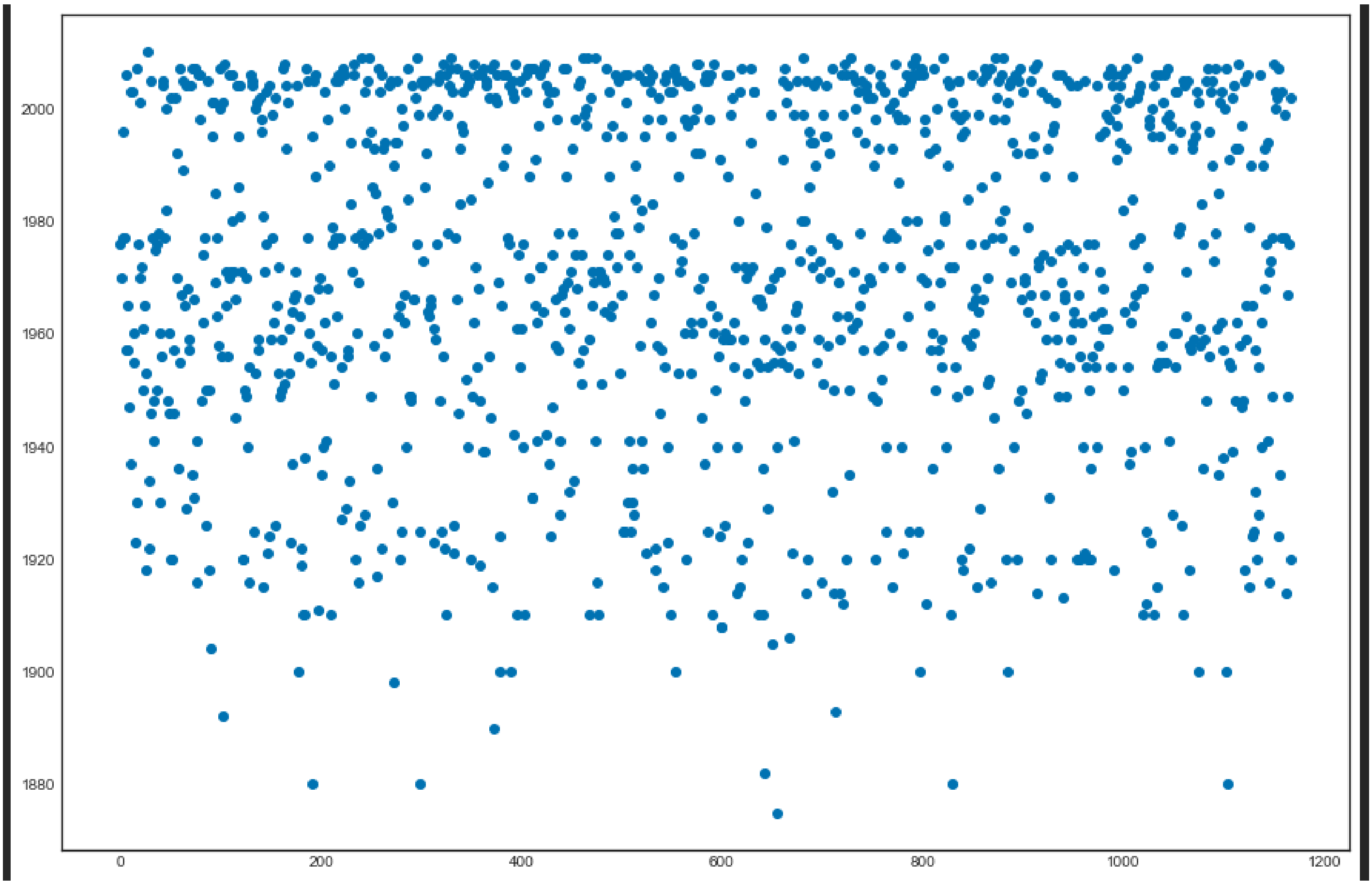
Output:



Code:

```
plt.style.use('seaborn-colorblind')
for j in train_df[triple]:
    plt.figure(figsize=(15,10))
    print(f"Scatter plot for {j} column with respect to the rows covered ->")
    plt.scatter(train_df.index, train_df[j])
    plt.show()
```

Output:



- Interpretation of the Results

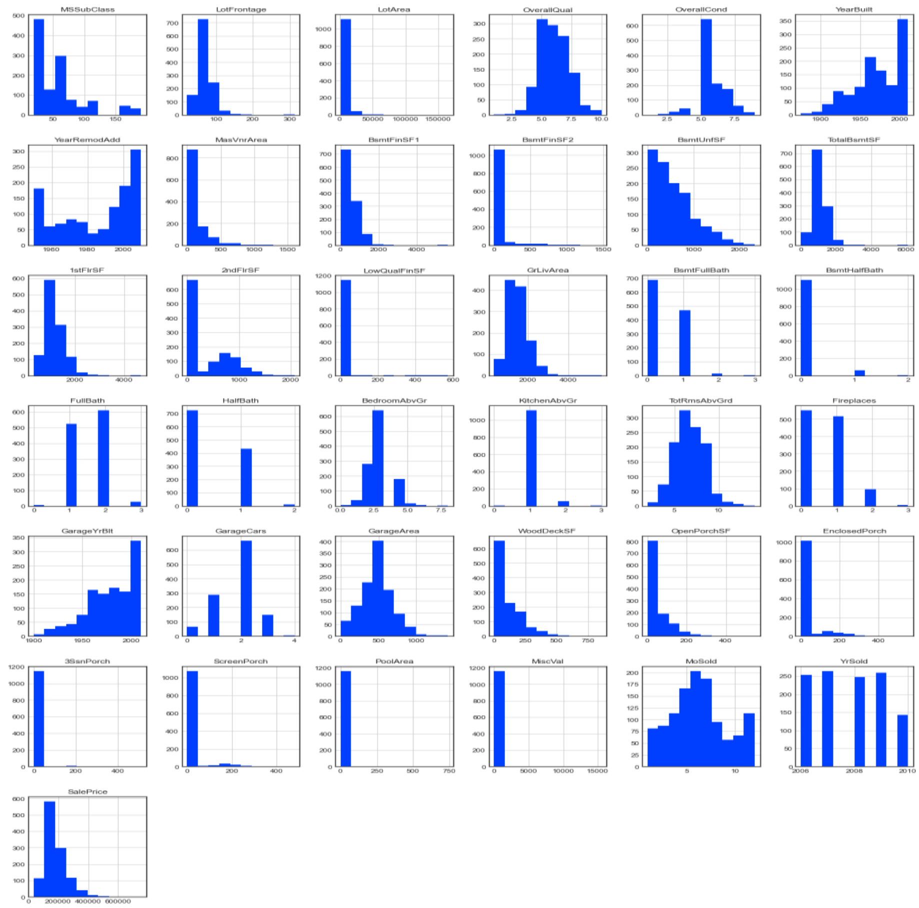
Visualizations: It helped me to understand the correlation between independent and dependent features. Also, helped me with feature importance and to check for multi collinearity issues.

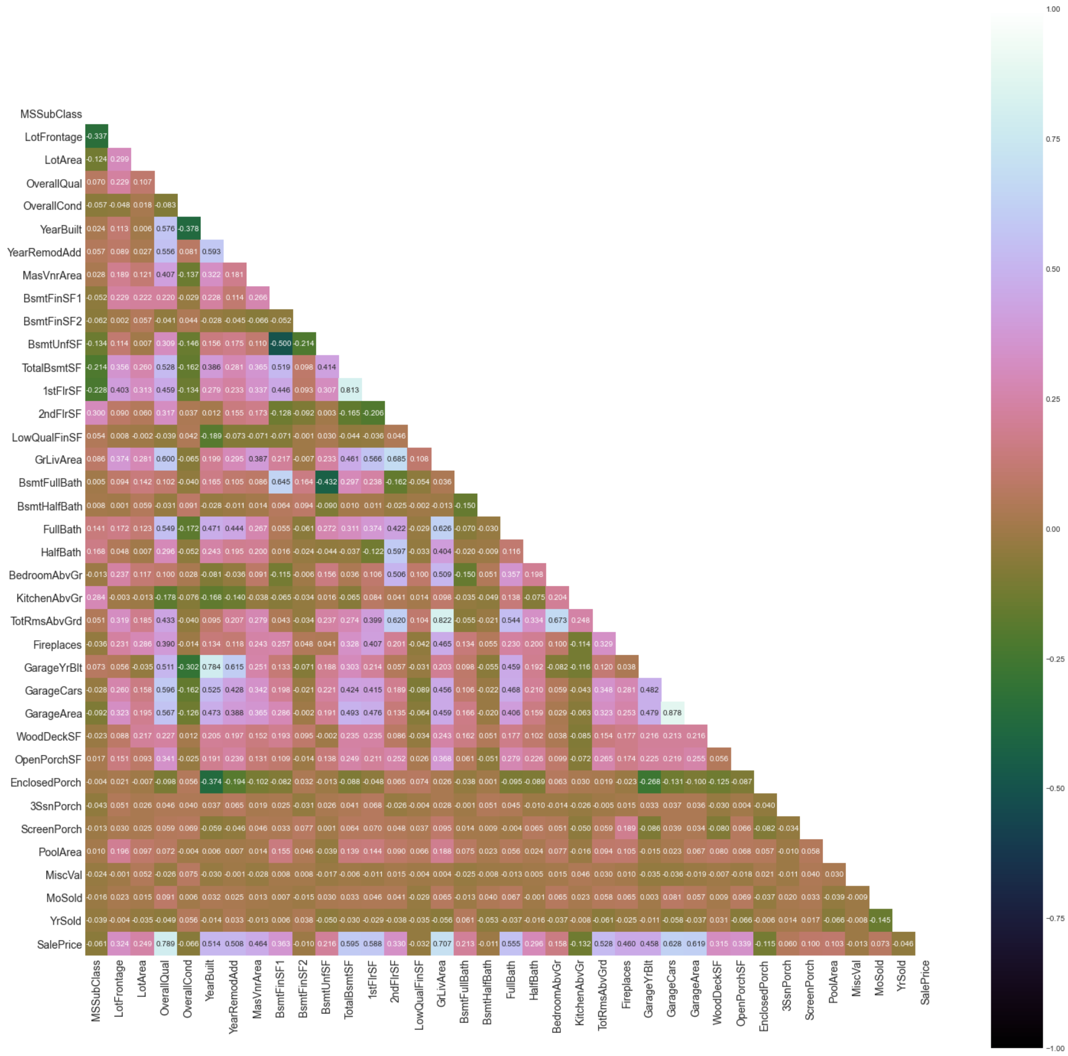
Detected outliers/skewness with the help of boxplot and distribution plot. I got to know the count of a particular category for each feature by using count plot and most importantly with predicted target value distribution as well as scatter plot helped me to select the best model.

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

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

Then we got Histogram, Heatmap for further insight on the price variance.


MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageYrBlt
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
SalePrice


MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageYrBlt
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
SalePrice
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00

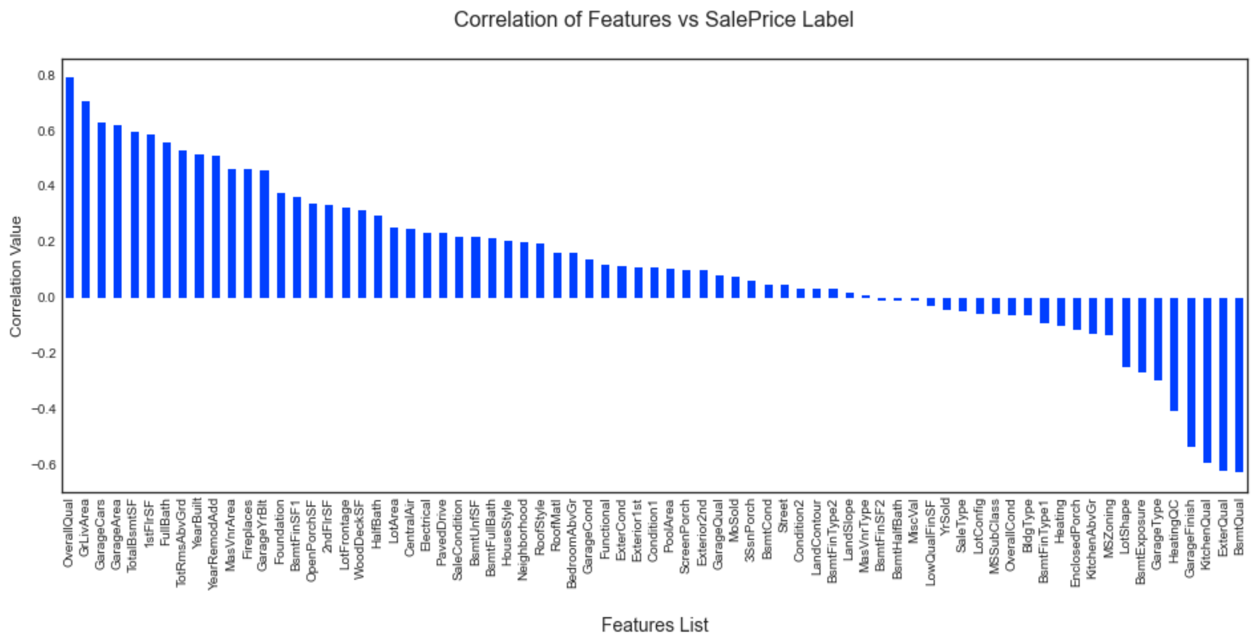
Encoding the categorical object datatype columns

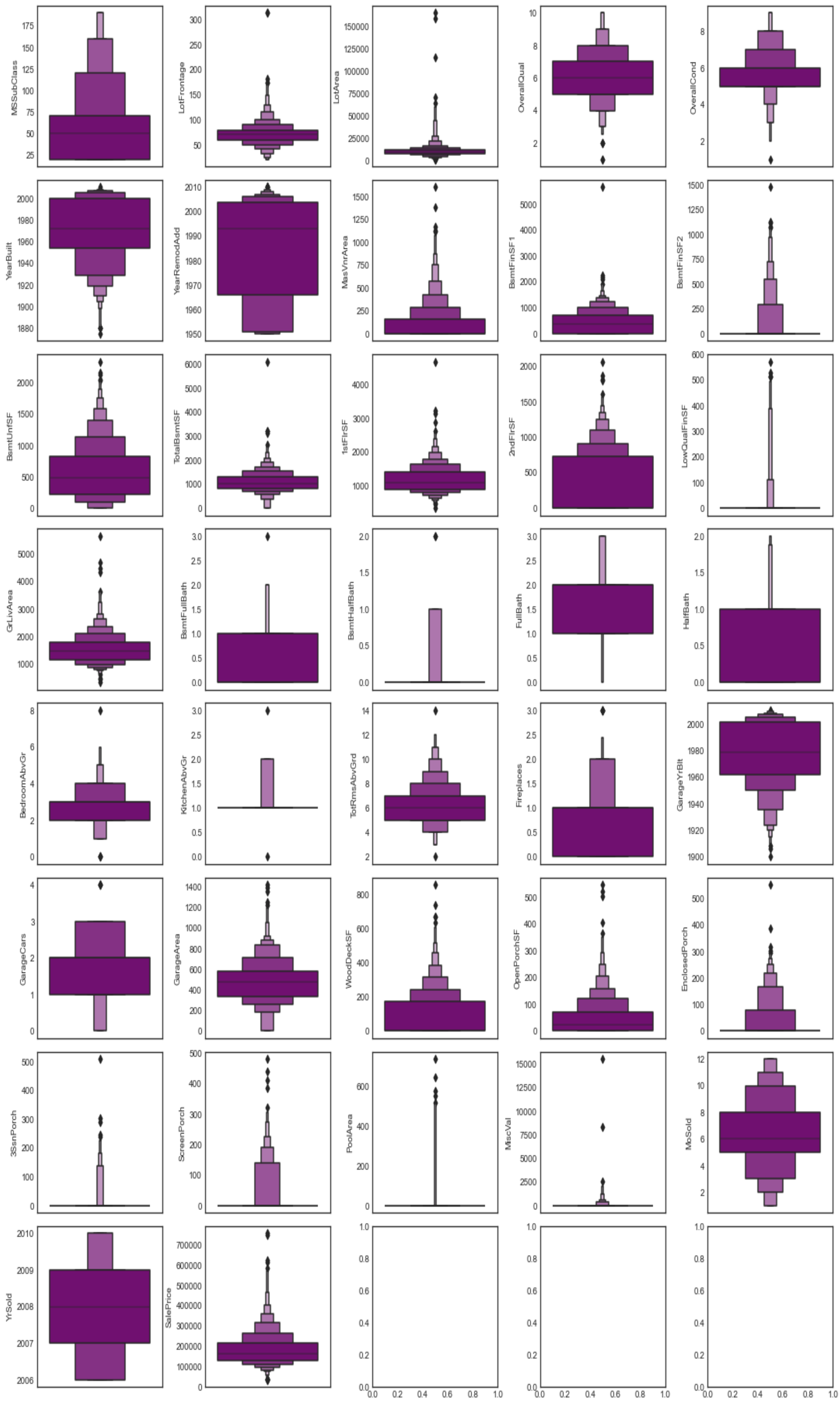
```python
# Ordinal Encoder

oe = OrdinalEncoder()
def ordinal_encode(df, column):
    df[column] = oe.fit_transform(df[column])
    return df

df=ordinal_encode(train_df, object_datatype)
df.head()
```

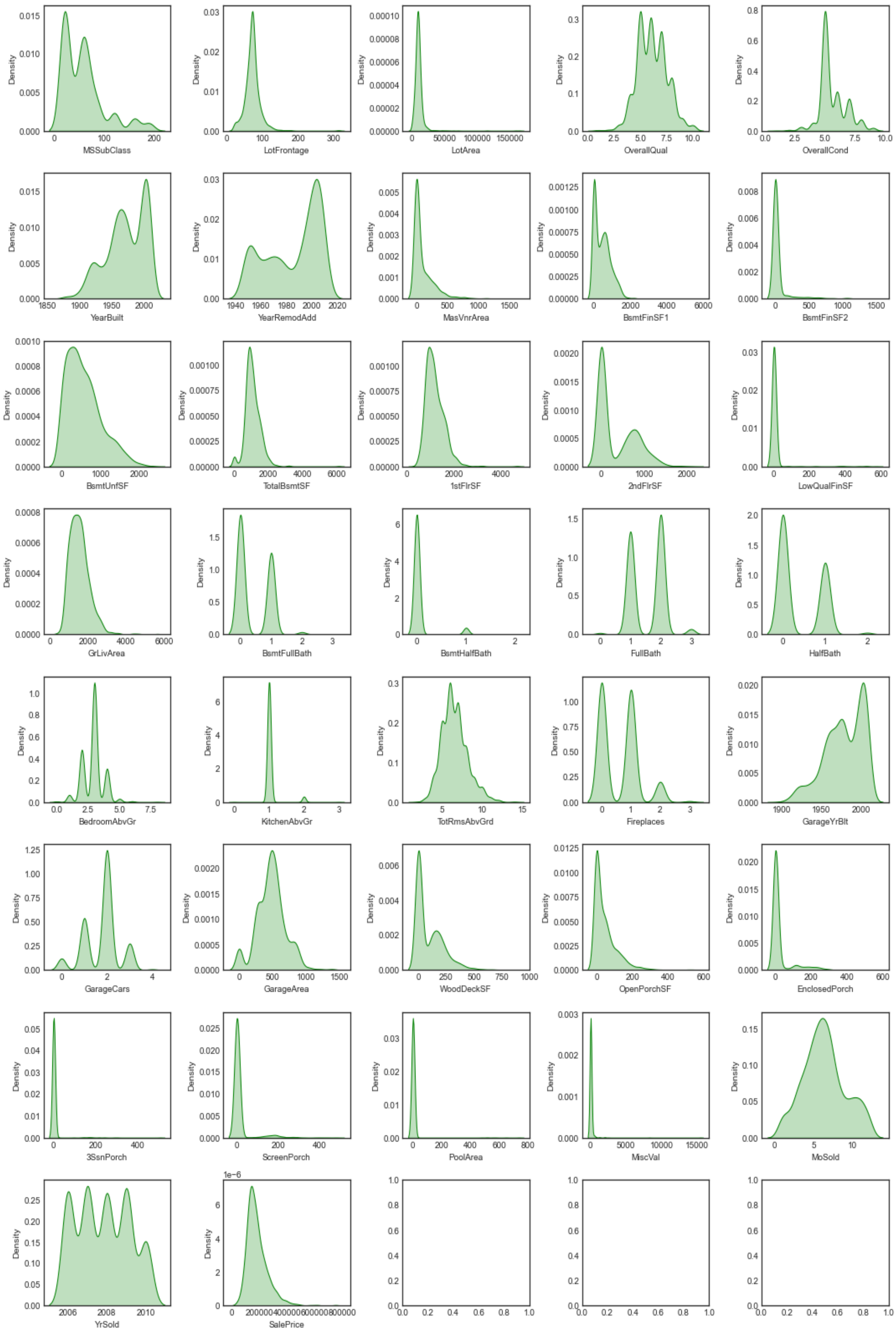
	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	H
0	120	3.0	70.98847	4928	1.0	0.0	3.0	4.0	0.0	13.0	2.0	2.0	4.0	2
1	20	3.0	95.00000	15865	1.0	0.0	3.0	4.0	1.0	12.0	2.0	2.0	0.0	2
2	60	3.0	92.00000	9920	1.0	0.0	3.0	1.0	0.0	15.0	2.0	2.0	0.0	5
3	20	3.0	105.00000	11751	1.0	0.0	3.0	4.0	0.0	14.0	2.0	2.0	0.0	2
4	20	3.0	70.98847	16635	1.0	0.0	3.0	2.0	0.0	14.0	2.0	2.0	0.0	2

The heatmap gives us the correlation between positive and negative correlation of the dataset and the feature columns to convert the object datatype columns to numeric format.

Further Bar Chart, Boxen Plot and Distribution plot are formed for our further price increase of housing.




MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageYrBlt
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
SalePrice



Post plotting of all the plots and X Y variable splitting we do

Feature importance dataframe

```
rf=RandomForestRegressor()
rf.fit(X_train, Y_train)
importances = pd.DataFrame({'Features':X.columns, 'Importance':np.round(rf.feature_importances_,3)})
importances = importances.sort_values('Importance', ascending=False).set_index('Features')
importances
```

	Importance
Features	
OverallQual	0.575
GrLivArea	0.091
1stFlrSF	0.035
MasVnrArea	0.031
GarageArea	0.026
TotalBsmtSF	0.025
BsmtQual	0.022
BsmtFinSF1	0.019
2ndFlrSF	0.017
FullBath	0.014
LotArea	0.013
GarageCars	0.012
LotFrontage	0.009
Neighborhood	0.009
YearRemodAdd	0.008
YearBuilt	0.008
OverallCond	0.005
WoodDeckSF	0.005
OpenPorchSF	0.005

Regression Model Function:

Machine Learning Model for Regression with Evaluation Metrics

```
# Regression Model Function

def reg(model, X, Y):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=340)

    # Training the model
    model.fit(X_train, Y_train)

    # Predicting Y_test
    pred = model.predict(X_test)

    # RMSE - a lower RMSE score is better than a higher one
    rmse = mean_squared_error(Y_test, pred, squared=False)
    print("RMSE Score is:", rmse)

    # R2 score
    r2 = r2_score(Y_test, pred, multioutput='variance_weighted')*100
    print("R2 Score is:", r2)

    # Cross Validation Score
    cv_score = (cross_val_score(model, X, Y, cv=5).mean())*100
    print("Cross Validation Score:", cv_score)

    # Result of r2 score minus cv score
    result = r2 - cv_score
    print("R2 Score - Cross Validation Score is", result)
```

Linear Regression being tested:

```
# Linear Regression Model

model=LinearRegression()
reg(model, X, Y)
```

```
RMSE Score is: 24876.373485691707
R2 Score is: 88.56355344351948
Cross Validation Score: 74.14529018813273
R2 Score - Cross Validation Score is 14.418263255386748
```

Ridge Regularization:

```
# Ridge Regularization

model=Ridge(alpha=1e-2, normalize=True)
reg(model, X, Y)
```

```
RMSE Score is: 24815.18998074428
R2 Score is: 88.6197402024921
Cross Validation Score: 74.45483255058483
R2 Score - Cross Validation Score is 14.16490765190727
```

Lasso Regularization:

```
# Lasso Regularization

model=Lasso(alpha=1e-2, normalize=True, max_iter=1e5)
reg(model, X, Y)
```

```
RMSE Score is: 24917.18385422086
R2 Score is: 88.52599905988447
Cross Validation Score: 74.1554161073105
R2 Score - Cross Validation Score is 14.370582952573969
```

Support Vector Regressor:

```
# Support Vector Regression

model=SVR(C=1.0, epsilon=0.2, kernel='poly', gamma='auto')
reg(model, X, Y)
```

```
RMSE Score is: 76592.05128076131
R2 Score is: -8.413750687388166
Cross Validation Score: -6.214424099645246
R2 Score - Cross Validation Score is -2.1993265877429202
```

Decision Tree Regressor:

```
# Decision Tree Regressor

model=DecisionTreeRegressor(criterion="poisson", random_state=111)
reg(model, X, Y)
```

```
RMSE Score is: 57727.62379648374
R2 Score is: 38.41366921116711
Cross Validation Score: 41.26696984258857
R2 Score - Cross Validation Score is -2.8533006314214617
```

Random Forest Regressor:

```
# Random Forest Regressor

model=RandomForestRegressor(max_depth=2, max_features="sqrt")
reg(model, X, Y)
```

```
RMSE Score is: 40625.4396140173
R2 Score is: 69.49906765983303
Cross Validation Score: 64.61456200338246
R2 Score - Cross Validation Score is 4.884505656450571
```

K Nearest Neighbour Regressor:

```
# K Neighbors Regressor

KNeighborsRegressor(n_neighbors=2, algorithm='kd_tree')
reg(model, X, Y)
```

```
RMSE Score is: 40466.730494501026
R2 Score is: 69.73691471173798
Cross Validation Score: 64.42251920085333
R2 Score - Cross Validation Score is 5.314395510884651
```

Gradient Boosting Regressor:

```
# Gradient Boosting Regressor

model=GradientBoostingRegressor(loss='quantile', n_estimators=200, max_depth=5)
reg(model, X, Y)
```

```
RMSE Score is: 34539.463803694656
R2 Score is: 77.95306863017093
Cross Validation Score: 78.2983938466606
R2 Score - Cross Validation Score is -0.34532521648966963
```

Ada Boost Regressor:

```
# Ada Boost Regressor

model=AdaBoostRegressor(n_estimators=300, learning_rate=1.05, random_state=42)
reg(model, X, Y)
```

```
RMSE Score is: 31820.346272586143
R2 Score is: 81.28771728128767
Cross Validation Score: 79.16566313678824
R2 Score - Cross Validation Score is 2.1220541444994296
```

Extra Tree Regressor:

```
# Extra Trees Regressor

model=ExtraTreesRegressor(n_estimators=200, max_features='sqrt', n_jobs=6)
reg(model, X, Y)
```

```
RMSE Score is: 23816.88408105236
R2 Score is: 89.51696939850329
Cross Validation Score: 84.8703100074016
R2 Score - Cross Validation Score is 4.646659391101693
```

Post testing on the regression models we are hyper tuning the parameter.

Code:

Hyper parameter tuning

```
# Choosing Extra Trees Regressor

fmod_param = {'n_estimators' : [100, 200, 300],
              'criterion' : ['squared_error', 'mse', 'absolute_error', 'mae'],
              'n_jobs' : [-2, -1, 1],
              'random_state' : [42, 111, 340]
             }
```

```
GSCV = GridSearchCV(ExtraTreesRegressor(), fmod_param, cv=5)
```

I am using the Grid Search CV method for hyper parameter tuning my best model.

```
Final_Model = ExtraTreesRegressor(criterion='mse', n_estimators=100, n_jobs=-2, random_state=42)
Model_Training = Final_Model.fit(X_train, Y_train)
fmod_pred = Final_Model.predict(X_test)
fmod_r2 = r2_score(Y_test, fmod_pred, multioutput='variance_weighted')*100
print("R2 score for the Best Model is:", fmod_r2)
```

```
R2 score for the Best Model is: 83.64443386563624
```

After getting the best R2 score of the model we are saving the model.

Saving the best model

```
filename = "SurpriseHousingSalePrice.pkl"
joblib.dump(Final_Model, filename)
```

```
['SurpriseHousingSalePrice.pkl']
```

Finally, I am saving my best regression model using the joblib library.

Then, we start testing the test dataset available to us.

```
test_df = pd.read_csv("test.csv")
test_df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Co
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	Corner	Gtl	StoneBr	Norm	No
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	StoneBr	Norm	No
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	No
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	Inside	Gtl	Crawfor	Norm	No
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	Somerst	Feedr	No

Then, we are checking the number of rows and columns and using isna() function to find null values present in the dataset or not.

```
print("We have {} Rows and {} Columns in our dataframe".format(test_df.shape[0], test_df.shape[1]))
```

```
We have 292 Rows and 80 Columns in our dataframe
```

Using the shape option we are checking the total number of rows and columns present in our testing dataset.

```
test_df.isnull().sum()
```

```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage      45
LotArea           0
Street            0
Alley           278
LotShape          0
LandContour       0
Utilities         0
LotConfig         0
LandSlope         0
Neighborhood      0
Condition1        0
Condition2        0
BldgType          0
HouseStyle        0
OverallQual       0
OverallCond       0
YearBuilt         0
YearRemodAdd      0
RoofStyle         0
RoofMatl          0
Exterior1st       0
Exterior2nd       0
MasVnrType        1
```

After checking for the missing data percentage just like the train dataset we are doing for test data set too and treating the missing values and looking if null values are further present or not as below.

```python
# data preprocessing 1
test_df.drop(["Alley", "FireplaceQu", "PoolQC", "Fence", "MiscFeature"], axis=1, inplace=True)

# data preprocessing 2
test_df.drop(["Id", "Utilities"], axis=1, inplace=True)

# data preprocessing 3
for i in mode:
    test_df[i] = test_df[i].fillna(test_df[i].mode()[0])

for j in mean:
    test_df[j] = test_df[j].fillna(test_df[j].mean())

print("Missing values count after filling the data")
print(test_df.isna().sum())
```

```
Missing values count after filling the data
MSSubClass       0
MSZoning         0
LotFrontage      0
LotArea          0
Street           0
LotShape         0
LandContour      0
LotConfig        0
LandSlope        0
Neighborhood     0
Condition1       0
Condition2       0
BldgType         0
HouseStyle       0
```

Further Predicting the sale price

```python
final_test_data = pd.concat([test_df, predicted_output], axis=1)
final_test_data.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	H
0	20.0	2.0	86.000000	14157.0	1.0	0.0	1.0	0.0	0.0	21.0	2.0	0.0	0.0	2
1	120.0	2.0	66.425101	5814.0	1.0	0.0	3.0	1.0	0.0	21.0	2.0	0.0	4.0	2
2	20.0	2.0	66.425101	11838.0	1.0	3.0	3.0	4.0	0.0	4.0	2.0	0.0	0.0	2
3	70.0	2.0	75.000000	12000.0	1.0	3.0	0.0	4.0	0.0	5.0	2.0	0.0	0.0	5
4	60.0	2.0	86.000000	14598.0	1.0	0.0	3.0	1.0	0.0	20.0	1.0	0.0	0.0	5

Here I am concatenating the test dataset and predicted Sale Price dataframe so that they can resemble the training dataset.

Conclusion:

-> After getting an insight of this dataset, we were able to understand that the Housing prices are done on basis of different features.

-> First, we loaded the train dataset and did the EDA process and other pre-processing techniques like outlier and skewness check, handling the null values present, filling the missing data with mean and mode, visualizing the distribution of data, etc.

-> Then we did the model training, building the model and finding out the best model on the basis of different metrices scores we got like R2 score, Cross Validation score, Mean Absolute Error, Mean Squared Error, Root Mean Squared Error, etc.

-> We got Extra Trees Regressor as the best algorithm among all as it gave more r2_score and cross_val_score. Then for finding out the best parameter and improving the scores, we performed Hyperparameter Tuning.

-> As the scores were not increased, we also tried using Ensemble Techniques like RandomForestRegressor, AdaBoostRegressor and GradientBoostingRegressor algorithms for boosting up our scores. Finally, we concluded that Extra Trees Regressor remained the best performing algorithm, although there were more errors in it and it had less RMSE compared to other algorithms. It gave an r2_score of 89.51 and cross_val_score of 84.87 which is the highest scores among all.

-> We saved the model in a pickle with a filename in order to use whenever we require. -> We predicted the values obtained and saved it separately in a csv file.

-> Then we used the test dataset and performed all the pre-processing pipeline methods to it. -> After treating missing values, we loaded the saved model that we obtained and did the predictions over the test data and then saving the predictions separately in a csv file.

-> From this project, I learnt how to handle train and test data separately and how to predict the values from them. This will be useful while we are working in a real-time case study as we can get any new data from the client we work on and we can proceed our analysis by loading the best model we obtained and start working on the analysis of the new data we have.

-> The final result will be the predictions we get from the new data and saving it separately.

-> Overall, we can say that this dataset is good for predicting the Housing prices using regression analysis and Extra Trees Regressor is the best working algorithm model we obtained.

-> We can improve the data by adding more features that are positively correlated with the target variable, having less outliers, normally distributed values, etc.

-> Also, we can work upon many factors to originally improve the quality of our features before providing it as an input for our machine learning models

Learning Outcomes of the Study in respect of Data Science

The above study helps one to understand the business of real estate. How the price is changing across the properties. With the Study we can tell how multiple real estate amenities like swimming pool, garage, pavement and lawn size of Lot Area, and type of Building raise decides the cost. With the help of the above analysis, one can sketch the needs of a property buyer and according to need we can project the price of the property.

Future Work

- ✓ The used pre-processing methods do help in the prediction accuracy. However, experimenting with different combinations of pre-processing methods to achieve better prediction accuracy.
- ✓ Make use of the available features and if they could be combined as binning features has shown that the data got improved.
- ✓ Training the datasets with different regression methods such as Elastic net regression that combines both L1 and L2 norms. In order to expand the comparison and check the performance.
- ✓ The correlation has shown the association in the local data. Thus, attempting to enhance the local data is required to make rich with features that vary and can provide a strong correlation relationship.
- ✓ The factors that have been studied in this study has a weak correlation with the sale price. Hence, by adding more factors to the local dataset that affect the house price, such as GDP, average income, and the population. In order to increase the number of factors that have an impact on house prices.
- ✓ The results of this study have shown that ANN is prone to overfitting. However, ANN still a strong algorithm that has a lot of options that could, with the right methods, provide a better prediction accuracy. ANN has a lot of possibilities that could lead to a different output. For instance, experimenting with the model when using combinations of layers and neurons over several iterations in order to find what fits the algorithm.
- ✓ ANN model was designed using feed-forward architecture. The model could make use of applying the proper back-propagation method to reduce the weight between neurons and give a better training performance resulting in better prediction accuracy.