Project Report on

Surprise Housing - Housing Price Predication & Analysis Project

NAME OF THE PROJECT

Surprise Housing - Housing Price Predication & Analysis Project

Submitted by:

Mr. Vikas Kumar Mishra

FLIPROBO SME:

Ms. Khushboo Garg

Project Report on

Surprise Housing - Housing Price Predication & Analysis Project

ACKNOWLEDGMENT

I would like to express my special gratitude to the "Flip Robo" team, who has allowed me to deal with a beautiful dataset and helped me improve my analysis skills. And I want to express my huge gratitude to Ms. Khushboo Garg (SME Flip Robo).

Thanks to "Data trained" who are the reason behind my Internship at FlipRobo Technologies.

SOURCE USED IN THIS PROJECT:

1. Learn Library Documentation
2. Help from YouTube Channels, Blogs from Educational Websites
3. Notes on Machine Learning (GitHub)
4. SCIKIT Learn Library Documentation

Project Report on

Surprise Housing - Housing Price Predication & Analysis Project

INTRODUCTION

Business Problem Framing

Real Estate Property is not only the basic need of a man but today it also represents the riches and prestige of a person. Investment in real estate generally seems to be profitable because their property values do not decline rapidly. The market demand for housing is always increasing every year due to increase in population and migrating to other cities for their financial purpose. Changes in the real estate price can affect various household investors, bankers, policy makers and many. Investment in Housing seems to be an attractive choice for the investments.

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.

Project Report on

Surprise Housing - Housing Price Predication & Analysis Project

In general, purchasing and investing in any real estate project will involve various transactions between different parties. Thus, it could be a vital decision for both households and enterprises. How to construct a realistic model to precisely predict the price of real estate has been a challenging topic with great potential for further research.

There are many factors that have an impact on house prices, such as the number of bedrooms and bathrooms. House price depends upon its location as well. A house with great accessibility to highways, schools, malls, employment opportunities, would have a greater price as compared to a house with no such accessibility.

Regression is a supervised learning algorithm in machine learning which is used for prediction by learning and forming a relationship between present statistical data and target value i.e., Sale Price in this case. Different factors are taken into consideration while predicting the worth of the house like location, neighbourhood and various amenities

like garage space etc. if learning is applied to above parameters with target values for a certain geographical region as different areas differ in price like land price, housing style, material used, availability of public

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

Project Report on

Surprise Housing - Housing Price Predication & Analysis Project

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.
This company wants to know:

- Which variables are important to predict the price of the variable?
- How do these variables describe the price of the house?

It is 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.

Analytical Problem Framing

Mathematical / Analytical Modelling of the Problem

Our objective is to predict House price which can be resolve by use of regression-based algorithm. In this project we are going to use different types of algorithms which uses their own mathematical equation on background. This project comes with two separate data set for training & testing model. Initially data cleaning & pre-processing perform over data. Feature engineering is performed to remove unnecessary feature & for dimensionality

reduction. In model building, the Final model is selected based on evaluation benchmarks among different models with different algorithms.
Further Hyperparameter tuning was performed to build the more accurate model out of the best model.

Data Sources and their formats

The data set provided by Flip Robo was in the format of CSV (Comma Separated Values). 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. The dimension of the data is 1168 rows and 81 columns.
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. The dimension of the data is 292 rows and 80 columns.

First Import Libraries

Importing Required Libraries

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

Project Report on

Surprise Housing - Housing Price Predication & Analysis Project

```
from sklearn.metrics import accuracy_score, mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import ExtraTreesRegressor
```

Retriving Dataset House_pred_train csv File

```
data = pd.read_csv('House_pred_train.csv')
data
```

```
print('No. of Rows :',data.shape[0])
print('No. of Columns :', data.shape[1])

# See truncated columns

pd.set_option('display.max_columns',None)
data.head()
```

```
No. of Rows : 1168
No. of Columns : 81
```

The dataset contains 81 columns with 1168 rows.

❖ The data types of different features are as shown below:

Sort columns by their datatypes

```
data.columns.to_series().groupby(data.dtypes).groups
```

```
{int64: ['Id', 'MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'BsmtFinSF1', 'BsmtFinSF2',
'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'H
alfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchS
F', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SalePrice'], float64: ['LotFrontag
e', 'MasVnrArea', 'GarageYrBlt'], object: ['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Ext
erior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtF
inType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageType', 'Garage
Finish', 'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'SaleCondition']}
```

Surprise Housing - Housing Price Predication & Analysis Project

Data Pre-processing

Before pre-processing data, the integrity of data is checked for missing values, and possible duplicates are present or not.

CHEK DATA INTEGRITY AND ANY WHITESPACE

```
: data.duplicated().sum()
: 0
```

```
: data.isin(['NA','N/A', '-', ' ', '?']).sum().sum()
: 0
```

CHEK MISSING VALUE

```
: data.isnull().sum().sum()
: 5558
```

❖ Some features contain missing values as shown below:

```
pd.set_option('display.max_rows',None)
missing_values = data.isnull().sum().sort_values(ascending=False)
per_miss_val = (missing_values/len(data))*100
print(pd.concat([missing_values, per_miss_val], axis = 1, keys =['Missing Values', 'Percentage(%) of Missing data']))
```

	Missing Values	Percentage(%) of Missing data
PoolQC	1161	99.400685
MiscFeature	1124	96.232877
Alley	1091	93.407534
Fence	931	79.708904
FireplaceQu	551	47.174658
LotFrontage	214	18.321918
GarageYrBlt	64	5.479452
GarageFinish	64	5.479452
GarageType	64	5.479452
GarageQual	64	5.479452
GarageCond	64	5.479452
BsmtExposure	31	2.654110
BsmtFinType2	31	2.654110
BsmtQual	30	2.568493
BsmtCond	30	2.568493
BsmtFinType1	30	2.568493
MasVnrType	7	0.599315
MasVnrArea	7	0.599315

Surprise Housing - Housing Price Predication & Analysis Project

We have removed features that contain a high amount of missing values e.g., the Top 5 features with missing values in the above list. The rest of the features are handled based on mean, median, or mode imputation depending on outliers & distribution of features.

These Above features contain high amount of Missing Data, There is no way to imputate these data. So, we are going to drop these features.

```
data.drop(['PoolQC','MiscFeature','Alley','Fence','FireplaceQu'],axis =1, inplace = True)
```

```
data.head()
```

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

```
data.shape[1]
```

76

```
print('No. of Rows :',data.shape[0])
print('No. of Columns :', data.shape[1])

# See truncated columns

pd.set_option('display.max_columns',None)
data.head()
```

```
No. of Rows : 1168
No. of Columns : 76
```

Removing Unused Column from Training and Testing Dataset

```
data.drop(['Id','Utilities'],axis=1,inplace=True)
dtest.drop(['Id','Utilities'],axis=1,inplace=True)
```

```
data.drop(['BsmtFinSF1','BsmtFinSF2','BsmtUnfSF'],axis=1,inplace=True)
dtest.drop(['BsmtFinSF1','BsmtFinSF2','BsmtUnfSF'],axis=1,inplace=True)
```

```
data.drop(['1stFlrSF','2ndFlrSF','LowQualFinSF'],axis=1,inplace=True)
dtest.drop(['1stFlrSF','2ndFlrSF','LowQualFinSF'],axis=1,inplace=True)
```

```
data.drop(['EnclosedPorch','3SsnPorch','ScreenPorch','PoolArea','MiscVal'],axis=1,inplace=True)
dtest.drop(['EnclosedPorch','3SsnPorch','ScreenPorch','PoolArea','MiscVal'],axis=1,inplace=True)
```

Surprise Housing - Housing Price Predication & Analysis Project

Label Encoding of Categorical features:

The categorical Variable in the training & testing dataset is converted into numerical data using a label encoder from the scikit library.

Encoding Categorical Features

Encoding Training data

```
# Using Label encoder

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in Categorical_Data:
    data[i] = le.fit_transform(data[i])
data.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	Hou
0	120	3	70.0	4928	1	0	3	4	0	13	2	2	4	
1	20	3	95.0	15865	1	0	3	4	1	12	2	2	0	
2	60	3	92.0	9920	1	0	3	1	0	15	2	2	0	
3	20	3	105.0	11751	1	0	3	4	0	14	2	2	0	
4	20	3	70.0	16635	1	0	3	2	0	14	2	2	0	

Encoding Testing data

```
# Using Label encoder

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in Categorical_Data:
    dtest[i] = le.fit_transform(dtest[i])
dtest.head()
```

➢ Standard Scaling:

Surprise Housing - Housing Price Predication & Analysis Project

Use Scaling Techniques

STANDARD SCALING FOR TRAINING DATASET

```
from sklearn.preprocessing import StandardScaler

st = StandardScaler()
x = st.fit_transform(x)
x
```

```
array([[ 1.72132516, -0.30715915, -0.03387126, ..., -0.64697732,
         0.33133632, -0.2689097 ],
       [ 0.17558668, -0.30715915,  1.12599853, ..., -0.64697732,
         0.33133632, -0.2689097 ],
       [ 0.17558668, -0.30715915, -0.66652751, ..., -1.3830511 ,
        -2.36453649,  2.80352665],
       ...,
       [-0.85490564, -0.30715915, -0.03387126, ...,  1.56124401,
         0.33133632, -0.2689097 ],
       [ 2.75181748, -0.30715915, -2.45905356, ...,  0.82517023,
         0.33133632, -0.2689097 ],
       [ 0.17558668, -0.30715915, -0.03387126, ..., -1.3830511 ,
         0.33133632, -0.2689097 ]])
```

STANDARD SCALING FOR TESTING DATASET

```
ss = StandardScaler()
xt = ss.fit_transform(dtest)
xt
```

```
array([[-0.85605433, -0.28700579,  0.99228791, ..., -0.65090813,
         0.23029007,  0.14865423],
       [ 1.43198105, -0.28700579, -0.06042999, ...,  0.86355541,
        -6.49418003, -3.32390858],
       [-0.85605433, -0.28700579, -0.06042999, ...,  0.86355541,
         0.23029007,  0.14865423],
       ...,
       [-0.85605433, -0.28700579, -0.06042999, ...,  1.62078718,
         0.23029007,  0.14865423],
       [-0.16964372,  1.80813647, -0.81237135, ..., -1.4081399 ,
         0.23029007,  0.14865423],
       [ 2.3471952 ,  1.80813647, -2.26612464, ..., -1.4081399 ,
         0.23029007,  0.14865423]])
```

Project Report on

Surprise Housing - Housing Price Predication & Analysis Project

Hardware & Software Requirements Tool Used

Hardware Used:

Processor — AMD Ryzen 5
RAM - 8 GB
ROM – 512 GB SSD
4GB Nvidia GEFORCE GTX Graphics card

Software utilized:

Anaconda – Jupyter Notebook

Models Development & Evaluation

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES:

- Our objective is to predict house prices and analyze features impacting Sale prices. This problem can be solved using regression-based machine learning algorithms like linear regression. For that purpose, the first task is to convert a categorical variable into numerical features. Once data encoding is done the data is scaled using a standard scalar.
- The final model is built over this scaled data. For building the ML model before implementing the regression algorithm, data is split into training & test data using train_test_split from the model_selection module of the sklearn library.
- Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is,

to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. After that model is trained with various regression algorithms and 5-fold cross-validation is performed. Further Hyperparameter tuning was performed to build a more accurate model out of the best model.

Testing of Identified Approaches (Algorithms)

The different regression algorithms used in this project to build the ML model are as below:

- Linear Regression
- Random Forest Regressor
- Decision Tree Regressor
- Ridge Regression
- Support Vector Regression (SVR)
- KNN Regression (KNeighbors Regression)

RUN AND EVALUATE SELECTED MODELS

Finding Best Random State:

Finding Best RandomState

```
lr=LinearRegression()
for i in range(20, 500):
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=i)
    lr.fit(x_train, y_train)
    pred_train = lr.predict(x_train)
    pred_test = lr.predict(x_test)
    if round(r2_score(y_train, pred_train)*100, 1)==round(r2_score(y_test, pred_test)*100, 1):
        print('At random state ', i, 'The model perform very well')
        print('At random state ', i)
        print('r2 score traning :', round(r2_score(y_train, pred_train)*100, 1))
        print('r2 score testing :', round(r2_score(y_test, pred_test)*100, 1), '\n\n')
```

```
At random state  62 The model perform very well
At random state  62
r2 score traning : 90.2
r2 score testing : 90.2


At random state  176 The model perform very well
At random state  176
r2 score traning : 90.1
r2 score testing : 90.1
```

Linear Regression:

Linear Regression

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state= 62, test_
lin_reg= LinearRegression()
lin_reg.fit(x_train, y_train)
y_pred = lin_reg.predict(x_test)
print('\033[1m'+ 'Error :'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(y_test, y_pred))
print('Mean squared error :', mean_squared_error(y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(y_test, y_pred)))
print('\033[1m'+' R2 Score :'+'\033[0m')
print(round(r2_score(y_test, y_pred)*100, 1))
```

```
Error :
Mean absolute error : 13650.573227096547
Mean squared error : 321353241.3811743
Root Mean squared error : 17926.328162263857
 R2 Score :
90.2
```

Chek Cross validation Score

```
from sklearn.model_selection import cross_val_score

score = cross_val_score(lin_reg, x, y, cv=5)
print('Score :', score)
print('\033[1m'+'Cross Validation Score :',lin_reg,":"+'\033[0m\n')
print("Mean CV Score :", score.mean())
print("Standard Deviation :",score.std())
print('Difference in R2 & CV Score:', (r2_score(y_test,y_pred)*100)-(score.mean()*100))
```

```
Score : [0.87779946 0.91167841 0.82534751 0.91456004 0.8454766 ]
Cross Validation Score : LinearRegression() :

Mean CV Score : 0.8749724044601928
Standard Deviation : 0.03536986378955725
Difference in R2 & CV Score: 2.714452019839044
```

Random Forest Regressor

Random Forest Regressor

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state= 62, test_size=0.33)
rfr = RandomForestRegressor()
rfr.fit(x_train, y_train)
rfr_prd = rfr.predict(x_test)
print("RFR Score : ", rfr.score(x_train, y_train))
print("RFR r2 Score : ", r2_score(y_test, rfr_prd))
print("RFR Mean Squared Error : ", mean_squared_error(y_test, rfr_prd))
print("RFR Root Mean Squared Error : ", np.sqrt(mean_squared_error(y_test, rfr_prd)))
```

```
RFR Score :  0.9757866419758932
RFR r2 Score :  0.8215176333064707
RFR Mean Squared Error :  585963271.651318
RFR Root Mean Squared Error :  24206.678244883537
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(rfr, x, y, cv=5)
print('\033[1m'+'Cross Validation Score :',rfr,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(y_test, rfr_prd)*100)-(score.mean()*100))
```

```
Cross Validation Score : RandomForestRegressor() :

Mean CV Score : 0.8512582201493304
Difference in R2 & CV Score: -2.974058684285964
```

Decision Tree Regressor

Decision Tree Regressor

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state= 62, test_size=0.33
dtr = DecisionTreeRegressor()
dtr.fit(x_train, y_train)
dtr_prd = dtr.predict(x_test)
print("DTR Score : ", dtr.score(x_train, y_train))
print("DTR r2 Score : ", r2_score(y_test, dtr_prd))
print("DTR Mean Squared Error : ", mean_squared_error(y_test, dtr_prd))
print("DTR Root Mean Squared Error : ", np.sqrt(mean_squared_error(y_test, dtr_prd)))
```

```
DTR Score :  1.0
DTR r2 Score :  0.5595995509329328
DTR Mean Squared Error :  1445848644.6179776
DTR Root Mean Squared Error :  38024.31649113469
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(dtr, x, y, cv=5)
print('\033[1m'+'Cross Validation Score :',dtr,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(y_test, dtr_prd)*100)-(score.mean()*100))
```

```
Cross Validation Score : DecisionTreeRegressor() :

Mean CV Score : 0.6437161074847616
Difference in R2 & CV Score: -8.411655655182877
```

Ridge Regression

Ridge Regressor

```
from sklearn.linear_model import Ridge, Lasso, ElasticNet
from sklearn.model_selection import GridSearchCV
```

```
# Prepare a range of alpha value to test

alpha_value={'alpha':[1, 0.1, 0.01, 0.001, 0.0001, 0]}

# create and fit a ridge regression model testing each alpha

model = Ridge()

grid = GridSearchCV(estimator=model, param_grid=alpha_value)
grid.fit(x, y)
print(grid)

# Summarize the results of the grid search

print('Best score : ', grid.best_score_)
print('Best estimator : ', grid.best_estimator_.alpha )
print('Best params : ', grid.best_params_)
```

```
GridSearchCV(estimator=Ridge(),
             param_grid={'alpha': [1, 0.1, 0.01, 0.001, 0.0001, 0]})
Best score :  0.8753469379406699
Best estimator :  1
Best params :  {'alpha': 1}
```

Surprise Housing - Housing Price Predication & Analysis Project

```
rdg = Ridge(alpha=1, random_state=62)
rdg.fit(x_train, y_train)
y_prd = rdg.predict(x_test)
print('Coefficient Value : ', rdg.coef_, '\n\n')
print('R2 Score : ', round(r2_score(y_test, y_prd)*100, 1))
print('Ridge Score Value : ', rdg.score(x_train, y_train))
```

```
Coefficient Value :  [-5.46471318e+03  6.35064139e+02 -3.98764495e+02  6.06960235e+03
  0.00000000e+00  1.00990959e+03 -2.08313063e+03  2.55686487e+01
  0.00000000e+00  1.62627243e+03  1.59312721e+03  0.00000000e+00
  2.44094418e+03 -8.92618043e+02  1.16301064e+04  7.43111058e+03
  9.46785924e+03  2.21344532e+03  5.88455420e+02  0.00000000e+00
 -3.65828742e+02 -6.64204590e+02  2.33291093e+03  2.75082686e+03
 -1.10728408e+03  2.42328641e+03  1.87222792e+03 -4.73771924e+02
 -4.54189295e+02 -2.13717653e+03 -9.32072667e+02  8.31446444e+02
  5.94493212e+03  0.00000000e+00 -3.59500450e+01  0.00000000e+00
  2.20733682e+01  2.54728960e+04  7.01071482e+03  0.00000000e+00
  1.03788343e+02 -6.52352451e+02 -2.77085473e+03  0.00000000e+00
 -2.76181661e+03  2.16040720e+03  3.00971864e+03  2.73918489e+03
  1.71469222e+03  3.13040606e+01 -1.09352967e+02  2.67808186e+03
  1.12248562e+03  0.00000000e+00  0.00000000e+00  1.95186489e+03
  2.69885932e+03  4.77568782e+02  1.26008412e+03  2.12340755e+02
 -5.33416819e+02  1.45283333e+03]


R2 Score :  90.3
Ridge Score Value :  0.9016034142923213
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(rdg, x, y, cv=5)
print('\033[1m'+'Cross Validation Score :',rdg,":"+'\033[0m\n')
print("Mean CV Score :",round((score.mean())*100, 1))
print('Difference in R2 & CV Score:',(r2_score(y_test, y_prd)*100)-(score.mean()*100))
```

```
Cross Validation Score : Ridge(alpha=1, random_state=62) :

Mean CV Score : 87.5
Difference in R2 & CV Score: 2.7442867617523348
```

Support Vector Regression (SVR)

Surprise Housing - Housing Price Predication & Analysis Project

Support Vector Regressor(SVR)

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state= 62, test_size=0.33)
svr = SVR()
svr.fit(x_train, y_train)
svr_prd = svr.predict(x_test)
print("SVR Score : ", svr.score(x_train, y_train))
print("SVR r2 Score : ", r2_score(y_test, svr_prd))
print("SVR Mean Squared Error : ", mean_squared_error(y_test, svr_prd))
print("SVR Root Mean Squared Error : ", np.sqrt(mean_squared_error(y_test, svr_prd)))
```

```
SVR Score :  -0.023052500256714348
SVR r2 Score :  -0.05626751468557556
SVR Mean Squared Error :  3467759757.505529
SVR Root Mean Squared Error :  58887.687656296446
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(svr, x, y, cv=5)
print('\033[1m'+'Cross Validation Score :',svr,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(y_test, svr_prd)*100)-(score.mean()*100))
```

```
Cross Validation Score : SVR() :

Mean CV Score : -0.030882789340706807
Difference in R2 & CV Score: -2.5384725344868753
```

KNN Regression (KNeighbors Regression)

Surprise Housing - Housing Price Predication & Analysis Project

KNN Regressor(KNeighbors Regressor)

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state= 62, test_size=0.33)
knn = KNeighborsRegressor()
knn.fit(x_train, y_train)
knn_prd = knn.predict(x_test)
print("KNN Score : ", knn.score(x_train, y_train))
print("KNN r2 Score : ", r2_score(y_test, knn_prd))
print("KNN Mean Squared Error : ", mean_squared_error(y_test, knn_prd))
print("KNN Root Mean Squared Error : ", np.sqrt(mean_squared_error(y_test, knn_prd)))
```

```
KNN Score :  0.8616948485554689
KNN r2 Score :  0.8215136779717507
KNN Mean Squared Error :  585976257.1406742
KNN Root Mean Squared Error :  24206.94646461371
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(knn, x, y, cv=5)
print('\033[1m'+'Cross Validation Score :',knn,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(y_test, knn_prd)*100)-(score.mean()*100))
```

```
Cross Validation Score : KNeighborsRegressor() :

Mean CV Score : 0.7963312443296825
Difference in R2 & CV Score: 2.5182433642068247
```

We can see that Ridge Regressor gives maximum R2 score of 90.3% and with cross validation score of 87.5%.

Hyper Parameter Tuning : GridSearchCV

```
from sklearn.model_selection import GridSearchCV
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=62, test_size=0.33)
```

```
print(rdg.get_params())
```

```
{'alpha': 1, 'copy_X': True, 'fit_intercept': True, 'max_iter': None, 'normalize': 'deprecated', 'positive': False, 'random_sta
te': 62, 'solver': 'auto', 'tol': 0.001}
```

```
GCV = GridSearchCV(Ridge(), alpha_value)
```

```
GCV.fit(x_train, y_train)
```

```
GridSearchCV(estimator=Ridge(),
             param_grid={'alpha': [1, 0.1, 0.01, 0.001, 0.0001, 0]})
```

```
GCV.best_params_
```

```
{'alpha': 1}
```

Surprise Housing - Housing Price Predication & Analysis Project

Final Model

```
fin_mod = Ridge()

fin_mod.fit(x_train,y_train)
pred = fin_mod.predict(x_test)
print('R2_Score :', r2_score(y_test,pred)*100)
print('mean_squared_error :', mean_squared_error(y_test,pred))
print('mean_absolute_error :', mean_absolute_error(y_test,pred))
print("root mean squared error value : ", np.sqrt(mean_squared_error(y_test, pred)))
```

```
R2_Score : 90.27898055581932
mean_squared_error : 319144152.0427069
mean_absolute_error : 13611.89917304823
root mean squared error value :  17864.60612615646
```

Saving Final Model

```
# Saving the model using .pkl

import joblib

joblib.dump(fin_mod,"surprise_House_Price_Prediction.pkl")
```

```
['surprise_House_Price_Prediction.pkl']
```

Predictions of Test Dataset Using Final Model

```
# Loading the saved model

model = joblib.load("surprise_House_Price_Prediction.pkl")

# Prediction

prediction = model.predict(xt)
prediction
```

```
array([299099.11922703, 215515.67091946, 251450.0472942 , 182995.6745008 ,
       218071.41161833, 104691.88552396, 148009.60653246, 274585.62981094,
       226885.64863844, 184252.53191569,  84074.61039901, 166723.23856119,
       123609.48400607, 204986.27009095, 277070.94153761, 147533.96455159,
       128303.53990401, 131497.28710991, 192684.17299797, 231189.4224564 ,
       176728.66073309, 158302.7482968 , 143256.05904991,  59870.59452214,
       118028.78417463, 148744.45185331, 182745.81446154, 155144.07893907,
       169861.63444817,  92446.00942847, 182555.58818726, 203429.08240707,
       235653.60931795, 182488.76907357, 129445.32537261, 174636.71405624,
       188620.17412773, 143283.45876059, 173658.63975985, 162644.47929911,
       110721.85008328, 283808.0004593 , 206948.0773732 , 205356.68005034,
       139297.1264383 , 150732.9909174 , 127330.33501038, 102984.5570463 ,
       217108.23678915, 288017.31146023, 141141.56883452, 220968.0110399 ,
       106887.6217462 ,  94695.73068444, 255571.61656683, 153900.68537179,
       168166.47831374, 189343.88595468, 153384.19287363, 237193.1751607 ,
       123431.54774231, 210078.84556903, 142197.74689374, 172017.69220067,
       229634.58098097, 102780.21866946, 186373.73881761, 232385.19173892,
       155413.41073598, 169664.86708581, 284265.08000188, 170603.81476423,
       184662.44115276, 189280.89707795, 175292.57281104, 230610.30110844,
       291492.62731013, 193790.87639142, 256433.64344079, 152389.97181111,
       199769.27878591, 154753.22999694, 171056.00668793, 165867.75043611,
       192453.75371045, 243317.26591081, 105670.59356863, 340368.34635867,
       170067.43536861, 180638.38304009, 245820.82638563, 138574.92053899,
```

Project Report on

Surprise Housing - Housing Price Predication & Analysis Project

```python
# Predicting the Item_Outlet_Sales from the feature columns of our Testing dataset

Test_data_Predication = pd.DataFrame()
Test_data_Predication['SalePrice']=prediction
Test_data_Predication.head()
```

	SalePrice
0	299099.119227
1	215515.670919
2	251450.047294
3	182995.674501
4	218071.411618

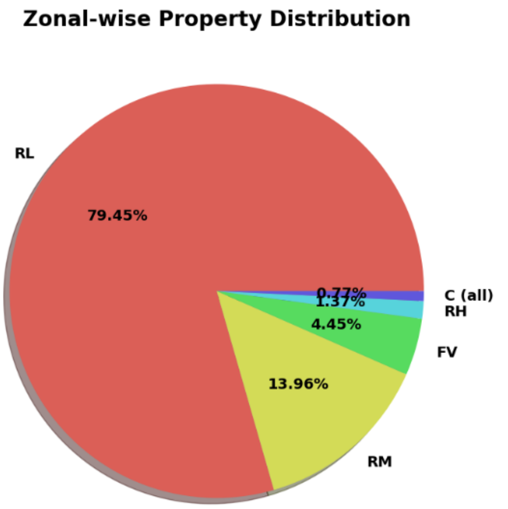
Merge Data Final Test

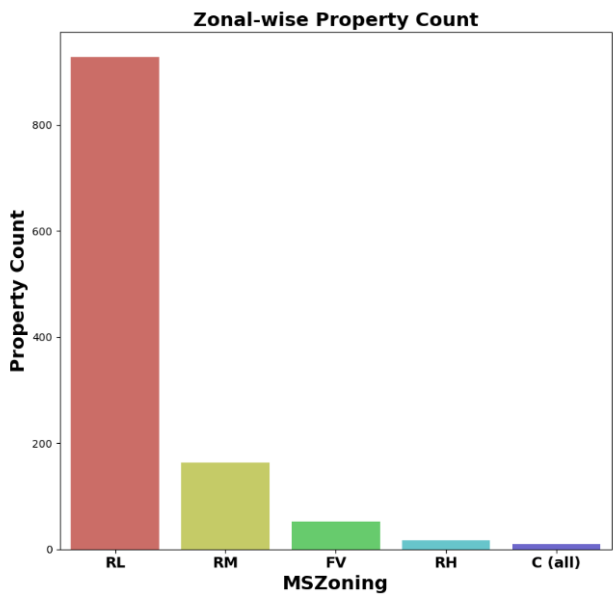
```python
Final_test_data = pd.concat([dtest, Test_data_Predication], axis=1)
Final_test_data.head()
```

geFinish	GarageCars	GarageArea	GarageQual	GarageCond	PavedDrive	WoodDeckSF	OpenPorchSF	MoSold	YrSold	SaleType	SaleCondition	SalePrice
0	3	676	4	4	2	178	51	7	2007	5	2	299099.119227
1	2	565	4	4	2	63	0	8	2009	0	0	215515.670919
1	2	522	4	4	2	202	151	6	2009	5	2	251450.047294
2	1	234	4	4	2	0	0	7	2009	5	2	182995.674501
0	3	668	4	4	2	100	18	1	2008	5	2	218071.411618

VISUALIZATIONS

Let's see the key result from EDA, starting with the zone-wise distribution of property.





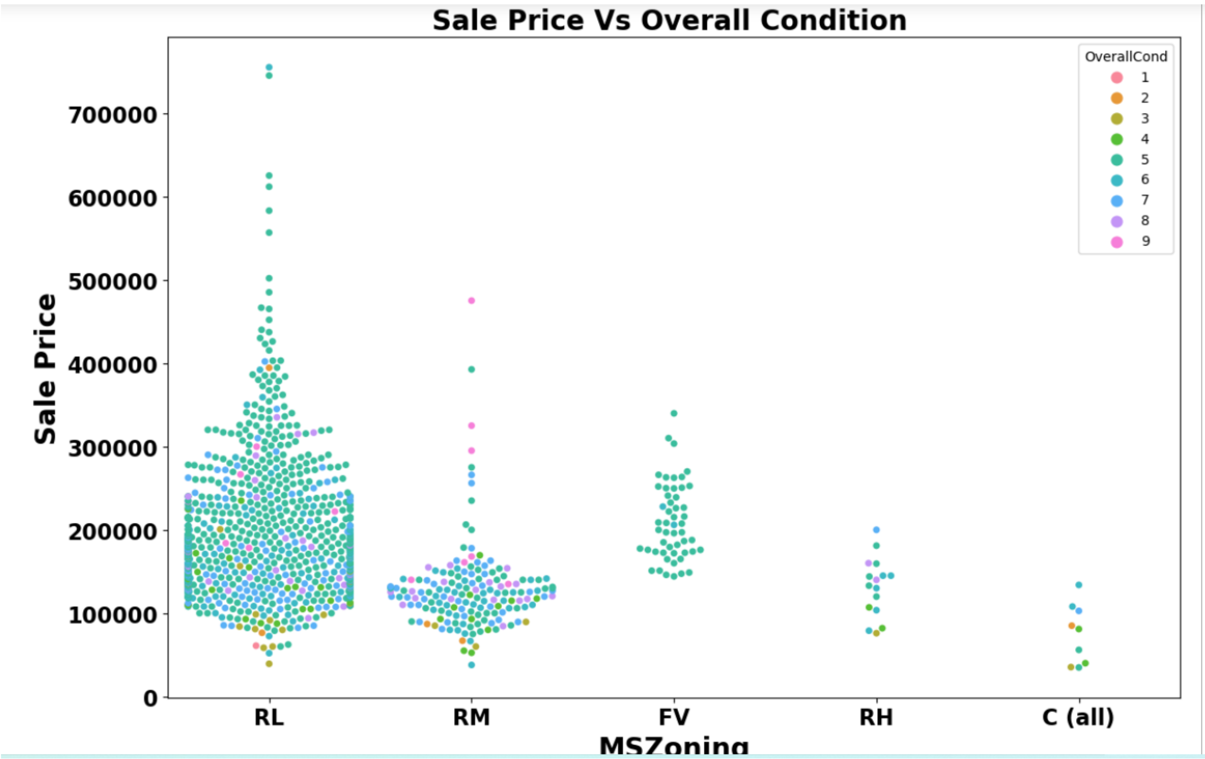
Surprise Housing - Housing Price Predication & Analysis Project

Observation:

79.45% of House properties belongs to Low Density Residential Area(RL).

13.96 % of properties belong to Medium Density Residential Area(RM).

Very Few property (0.77%) belongs to Commerical zone(C(all)).



Observation :

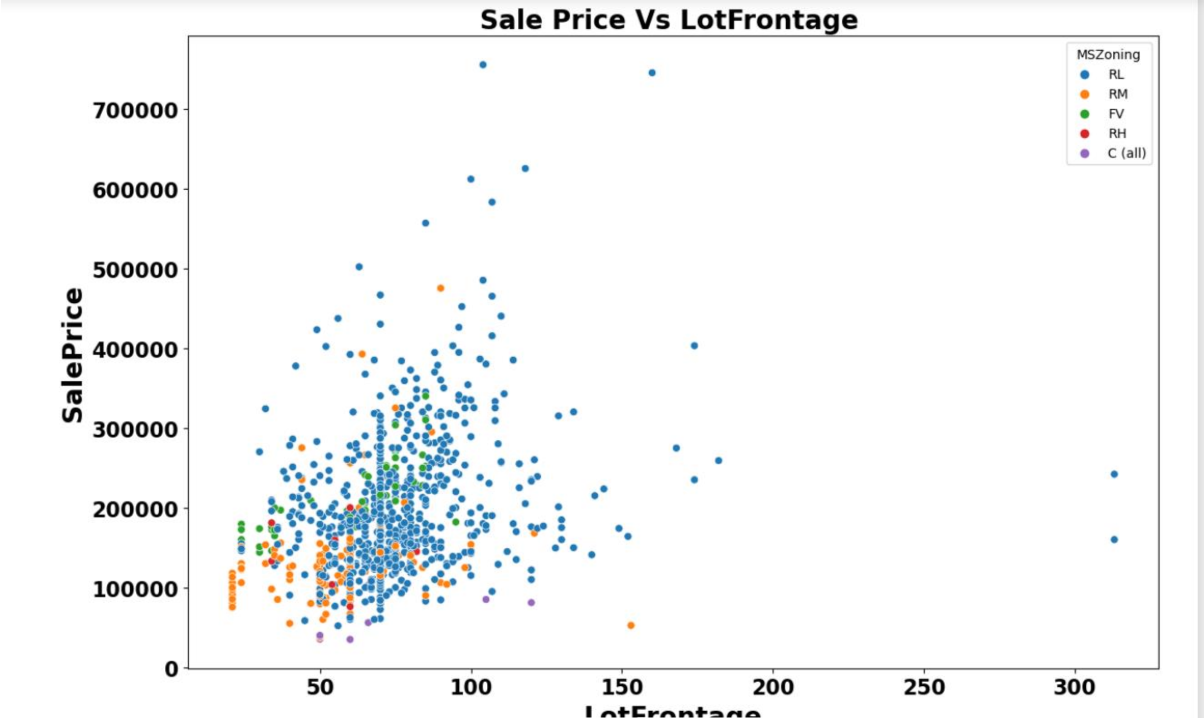
Most of property for sale have overall condition rating of either 5 or 6.

80% of housing data belongs to Low density Residential Area and Now we can see in Swramplot that Sale Price inside RL Zone is much higher than other remaining zone.

Cheapest properties are available in Commerical zone.

Project Report on

Surprise Housing - Housing Price Predication & Analysis Project

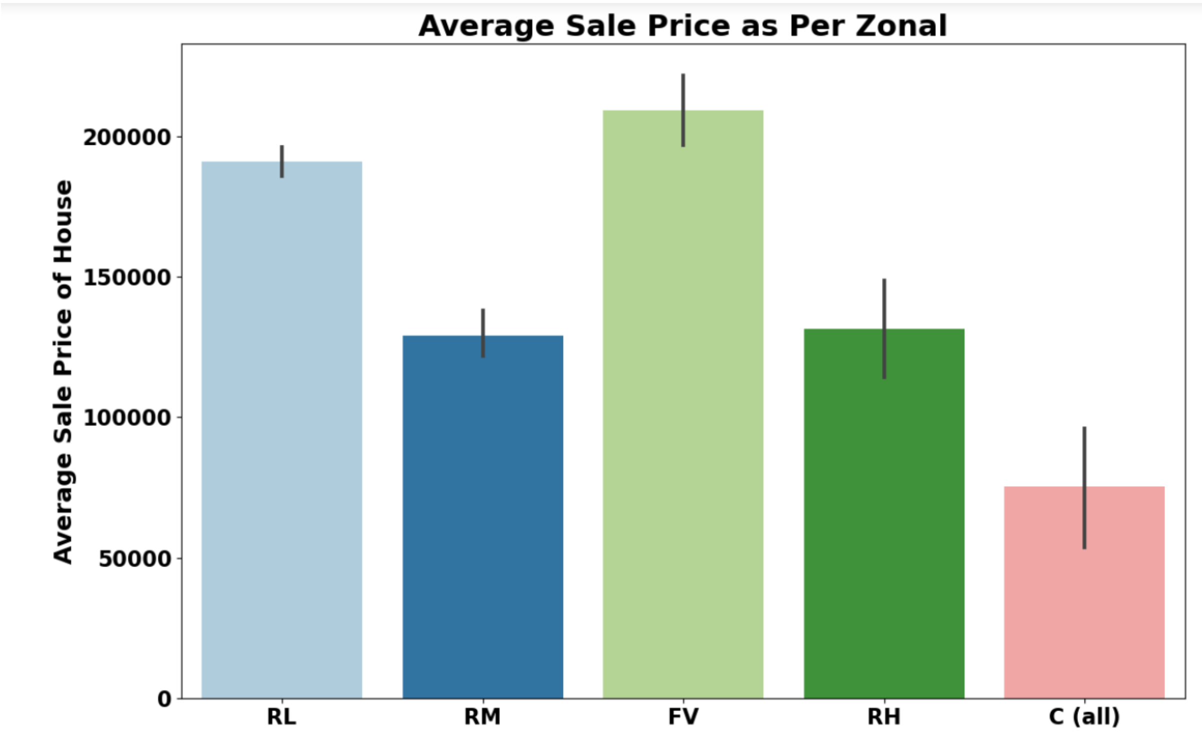


Observation:

Lot Frontage area increase and the Sale Price is also increase.

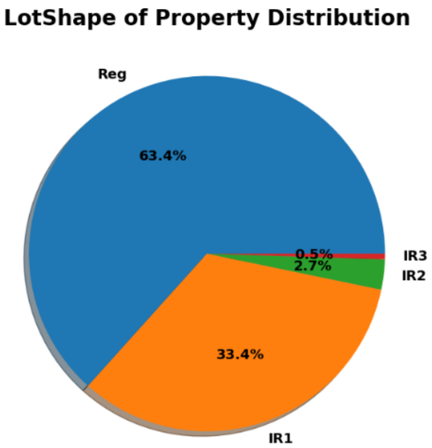
Project Report on

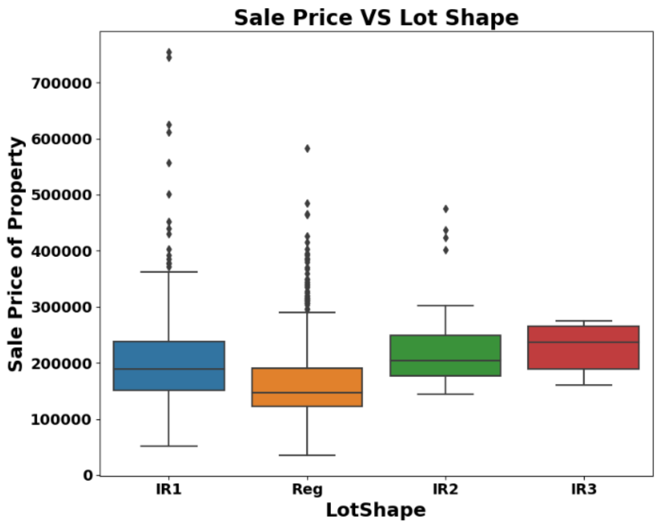
Surprise Housing - Housing Price Predication & Analysis Project



Observation :

In Average sale price of housing Floating Village Residential Zone are costiler than other

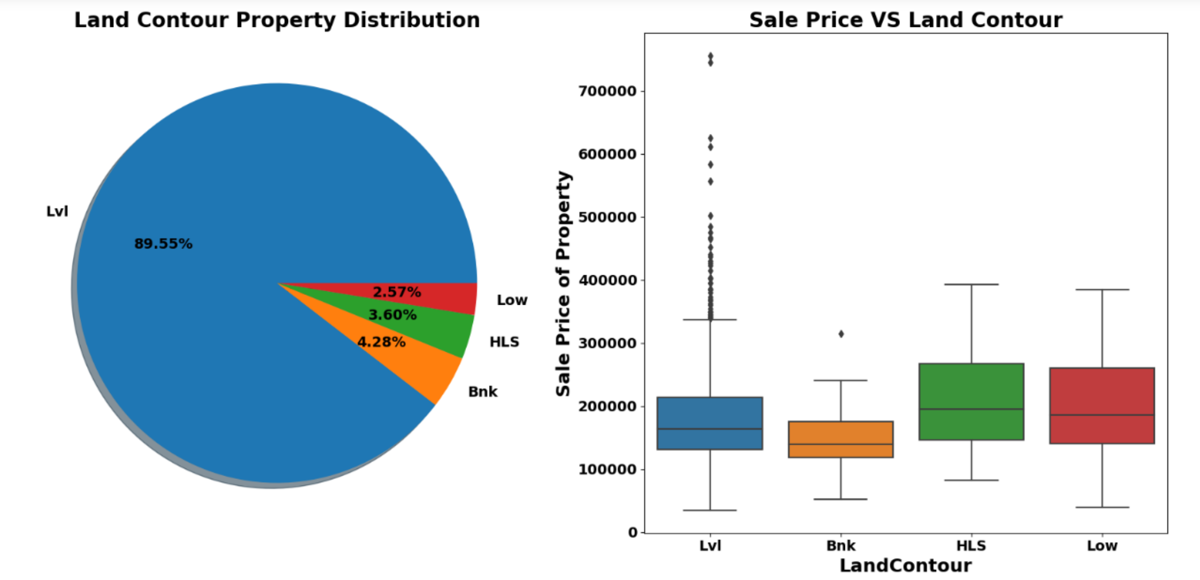




Observation :

63.4% house properties are regular in shape.

Sale Price of property with slight irregular shape is higher than regular shape.



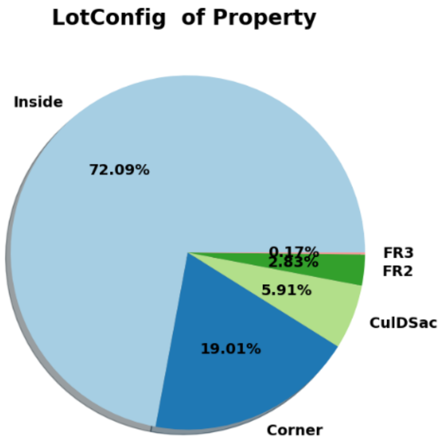
Observation :

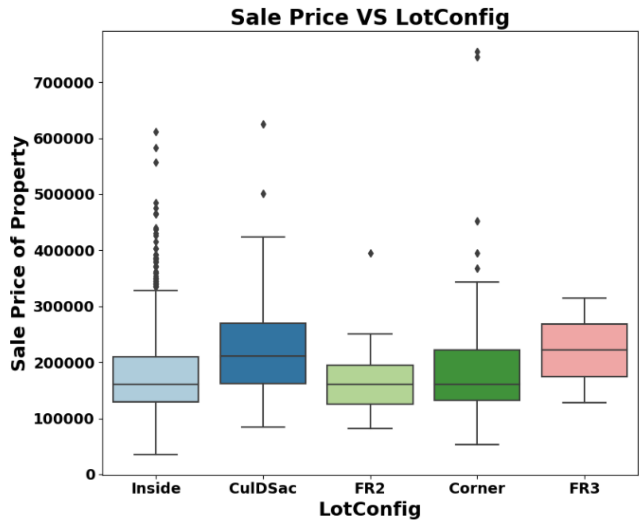
89.55% of House properties are near flat level surface.

Also price for Flat level surface house(Lvl) is much higher than Bnk, HLS, and Low.

Project Report on

Surprise Housing - Housing Price Predication & Analysis Project





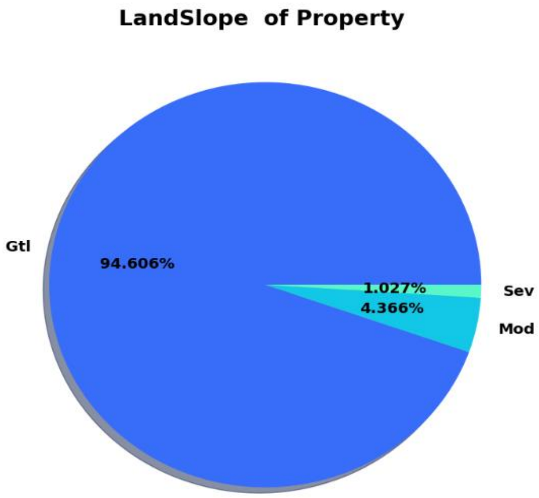
Observation :

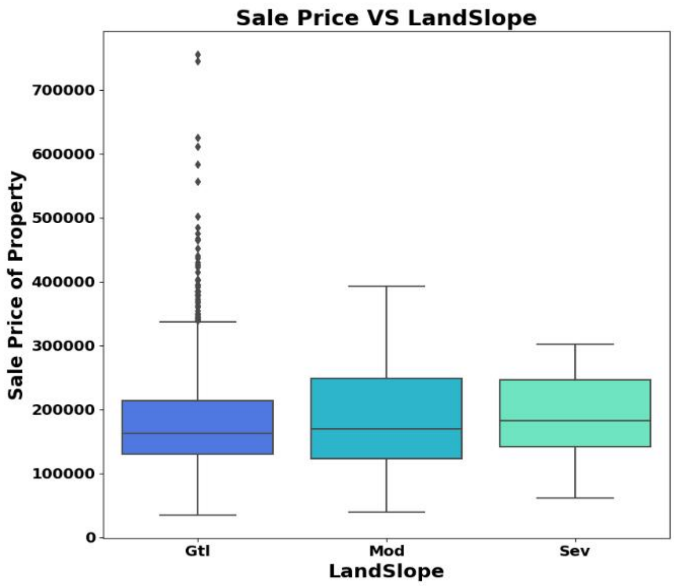
72.09 % of house comes with inside Lot configuration.

Cul-de-sac have maxmium Mean Sale Price.

Uncostly Houses belong to Inside lot configuration.

Costly Houses belongs to Corner Lot Configuration.

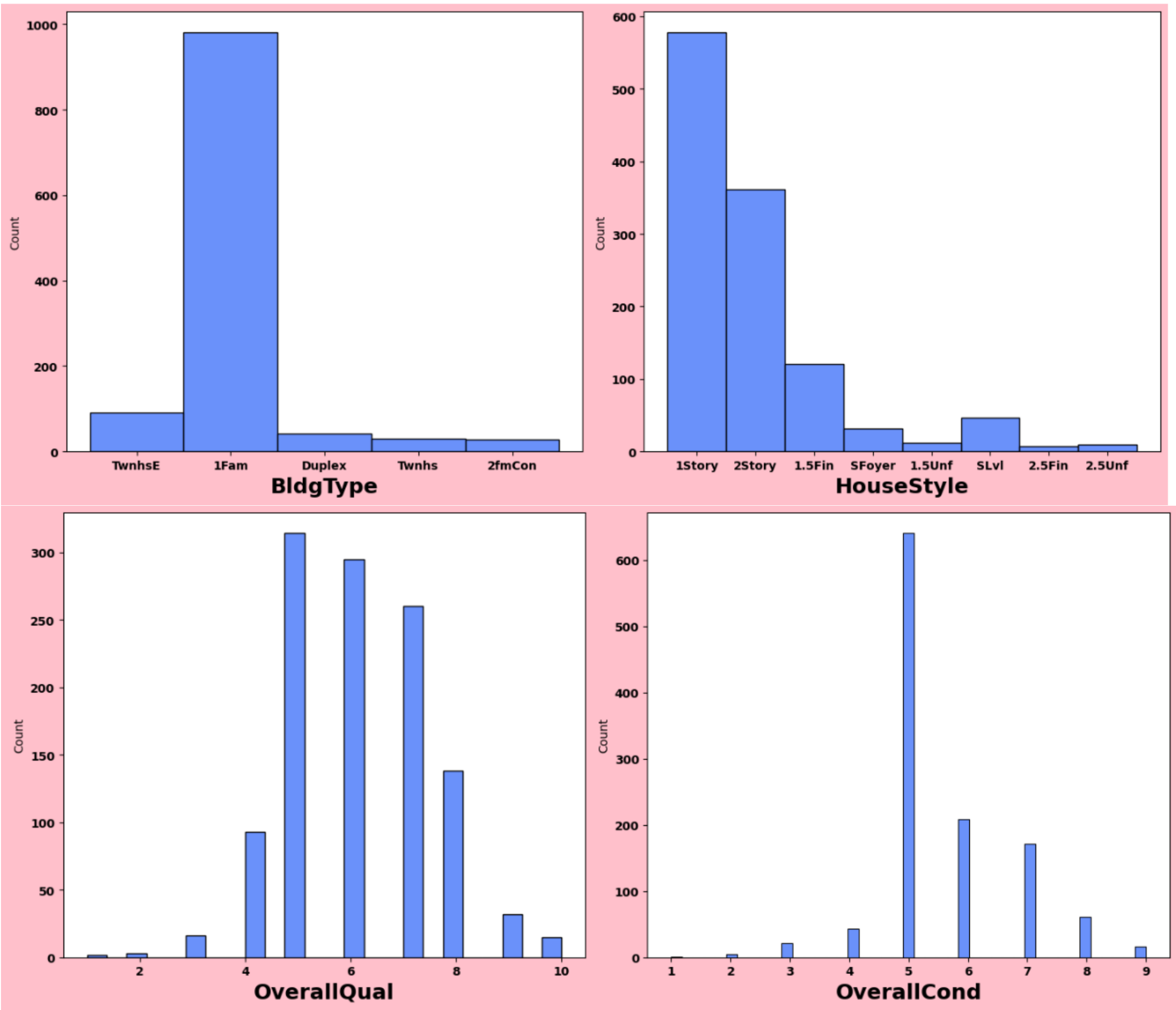




Observation :

Clearly we can see in boxplot Land slope increases the Sale price of house decreases.

1.027% properties come with severe slope and they come with low price compare to Gentle Slope properties.



Project Report on

Surprise Housing - Housing Price Predication & Analysis Project

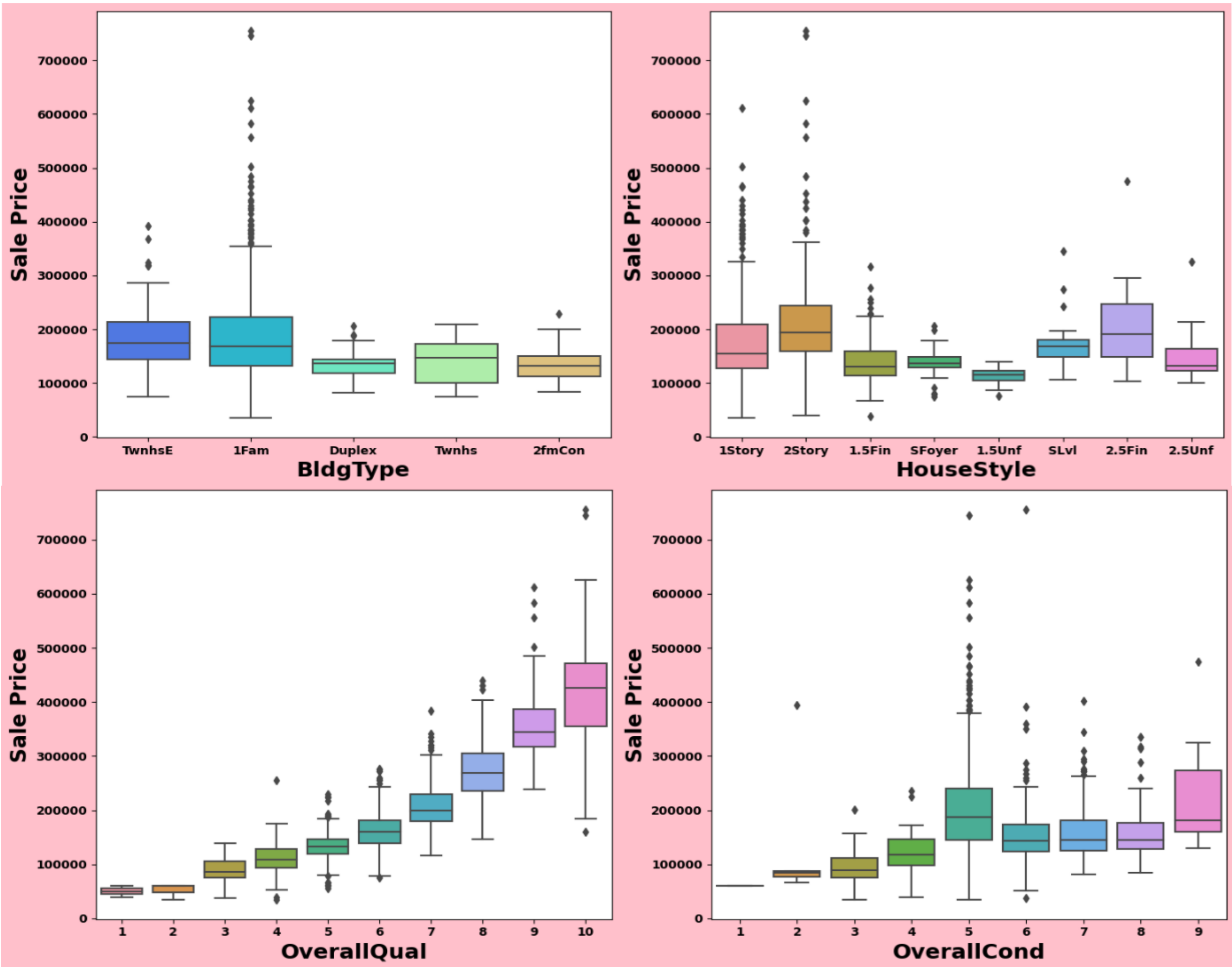
Observation :

More than 950 house properties are with building type Single-family Detached(1Fam)

Approx 50% of house properties Overall Condition Rating of 5.

Approx 50% to 70% of house properties Quality Rating between 5 to 7.

Approx 550 House Style are one story.



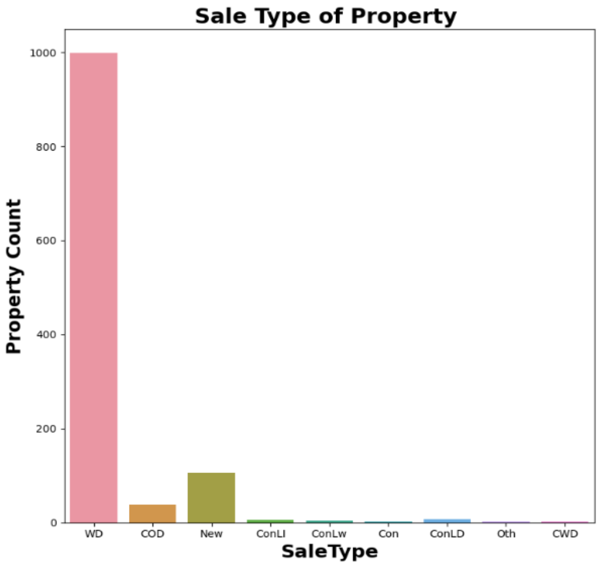
Project Report on

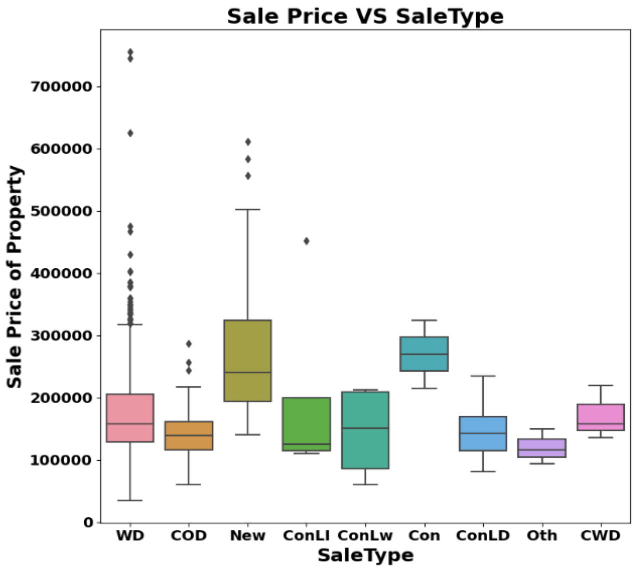
Surprise Housing - Housing Price Predication & Analysis Project

Observation:

OverallQual: Rates the overall material and finish of the house

Overall Quality Rating Is Increases and House Price Is Also Increases.



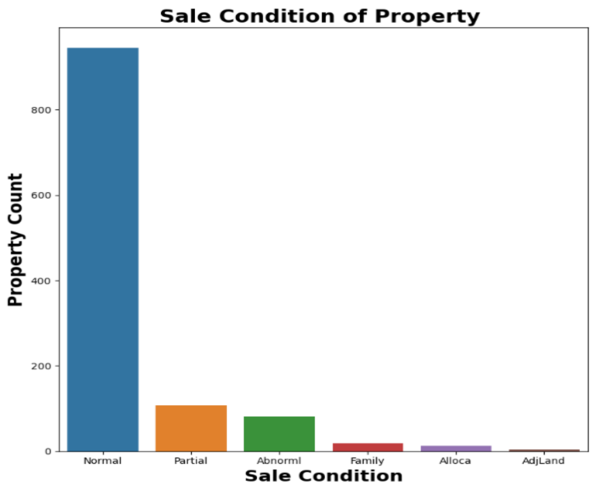


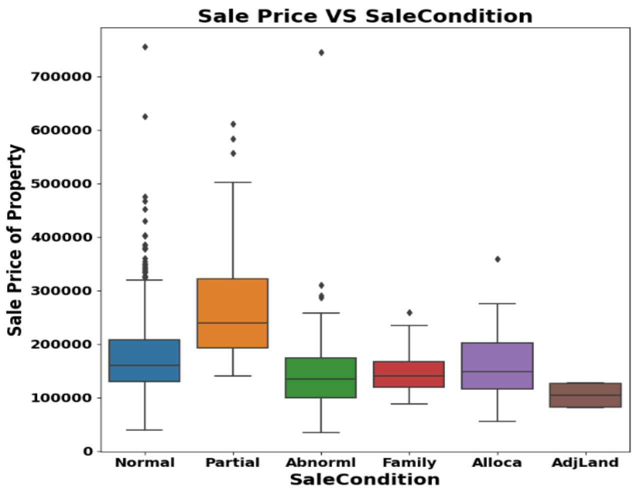
Observation :

Around 1000 sales happen by Conventional Warranty Deed.

Home just constructed and sold category are much expensive than other.

Loan based sales are below 300000.



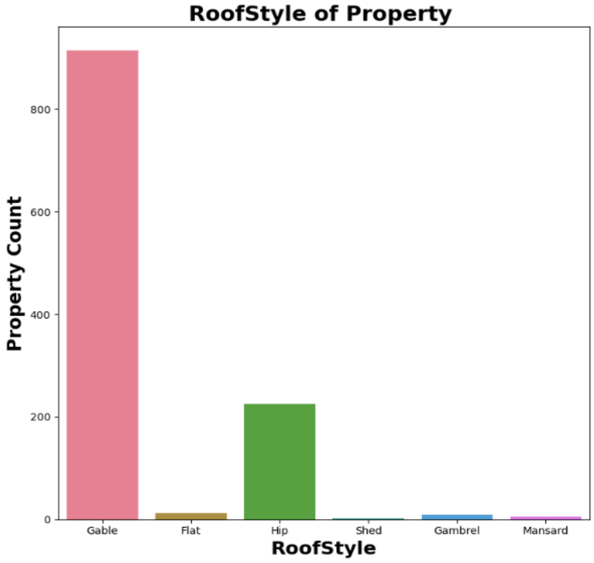


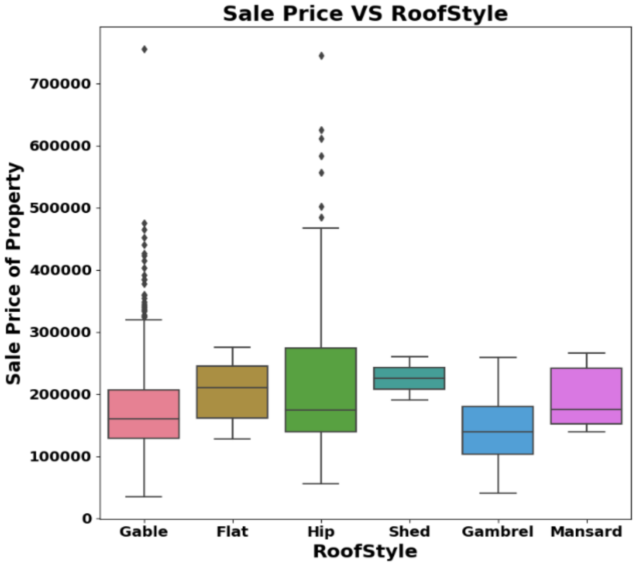
Observation :

Sale with condition like Abnorml, Family, Alloca, AdjLand are below the price of 300000.

Maximum Base Price for House comes from Partial category.

Highest property count comes from Normal sale.

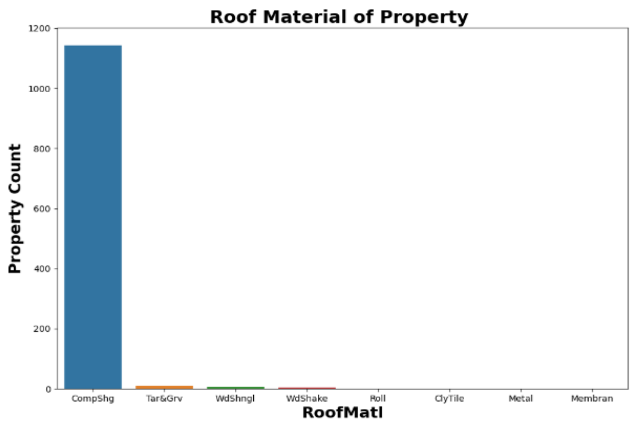


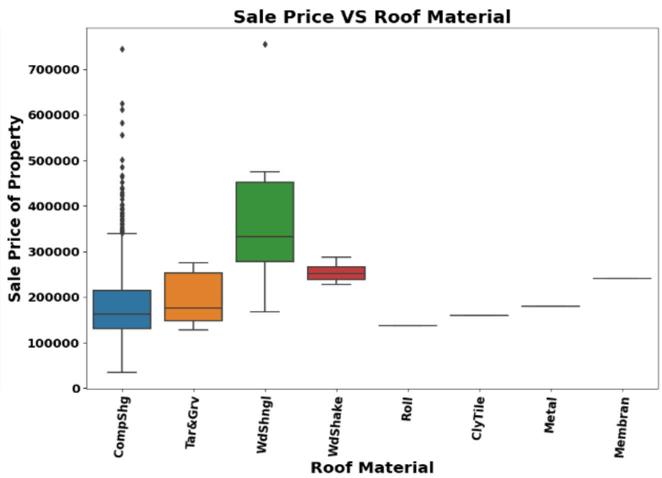


Observation :

Approx 80% House properties come with Gable Roof Style and around 20 % house properties with Hip Style.

In Boxplot Hip style Roof are much Expensive than other roof style.



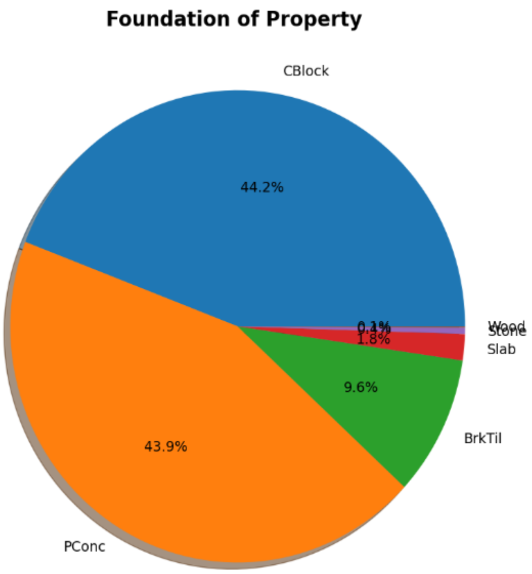


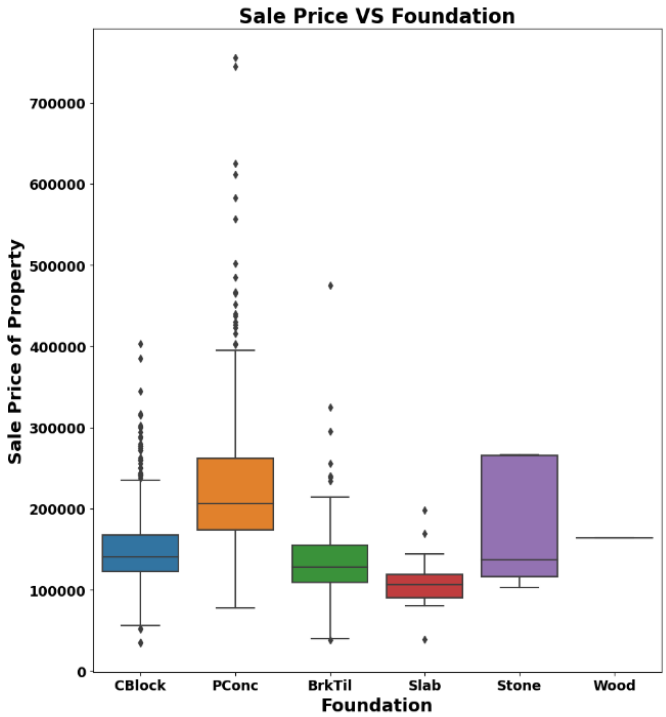
Surprise Housing - Housing Price Predication & Analysis Project

Observation :

Approx 90% Properties in Data set made with Standard (Composite) Shingle Roof Material.

Wood Shingles Roof is Very Costly compare to Other Roof Material.





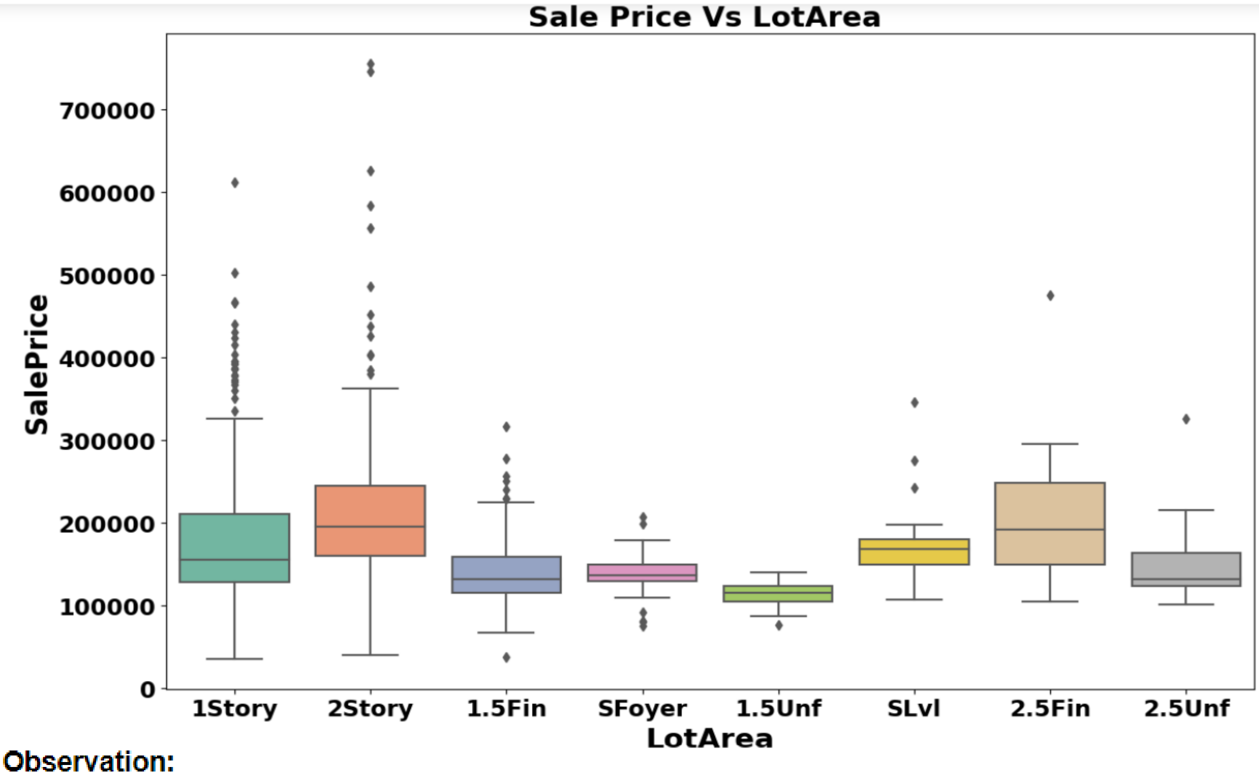
Observation :

44.2% Properties with CBlock Foundation & 43.9% housing property come with PConc Foundation.

Pconc Foundation are mostly use in costly housing properties.

Project Report on

Surprise Housing - Housing Price Predication & Analysis Project



Observation:

2Story Houses are Costly Then other Houses.

Surprise Housing - Housing Price Predication & Analysis Project

Conclusion

Key Findings and Conclusions of the Study

Algorithm	R2 Score	CV Score
Linear Regression	90.2%	87.4%
Ridge Regression	90.3%	87.5%
KNeighbors Regression	82%	79%
Support Vector Regression	-0.056	-0.030
Decision Tree Regression	60%	64%
Random Forest Regression	82%	85%
Ridge Regression	90.3%	87.5%

Ridge Regressor gives us a maximum R2 Score of 90.3%, So Ridge Regressor is selected as the best model.