PROJECT REPORT ON:

“HOUSING: PRICE PREDICTION”

SUBMITTED BY

INTERN 33

Rahul M

1.INTRODUCTION

Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company. A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below. The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

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

Business Goal:

You are required to 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.

2.Analytical Problem Framing

.

2.1 Data Preprocessing Done

- ✓ As a first step I have imported required libraries and I have imported both the datasets which were in csv format.
- ✓ Then I did all the statistical analysis like checking shape, nunique, value counts, info etc…..
- ✓ While checking the info of the datasets I found some columns with more than 80% null values, so these columns will create skewness in datasets so I decided to drop those columns.
- ✓ Then while looking into the value counts I found some columns with more than 85% zero values this also creates skewness in the model and there are chances of getting model bias so I have dropped those columns with more than 85% zero values.
- ✓ While checking for null values I found null values in most of the columns and I have used imputation method to replace those null values (mode for categorical column and mean for numerical columns).
- ✓ In Id and Utilities column the unique counts were 1168 and 1 respectively, which means all the entries in Id column are unique and ID is the identity number given for perticular asset and all the entries in Utilities column were same so these two column will not help us in model building. So I decided to drop those columns.
- ✓ Next as a part of feature extraction I converted all the year columns to there respective age. Thinking that age will help us more than year.
- ✓ And all these steps were performed to both train and test datasets separately and simultaneously.

Libraries required :-

- ✓ To run the program and to build the model we need some basic libraries

```
In [1]: #importing required libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import datetime as dt

        import warnings
        warnings.filterwarnings('ignore')
```

- ✓ from sklearn.preprocessing import OrdinalEncoder
- ✓ from statsmodels.stats.outliers_influence import variance_inflation_factor
- ✓ from sklearn.ensemble import RandomForestRegressor
- ✓ from sklearn.tree import DecisionTreeRegressor
- ✓ from xgboost import XGBRegressor
- ✓ from sklearn.ensemble import GradientBoostingRegressor
- ✓ from sklearn.ensemble import ExtraTreesRegressor
- ✓ from sklearn.metrics import classification_report
- ✓ from sklearn.model_selection import cross_val_score

With this sufficient libraries we can go ahead with our analytical skills.

3.Data Analysis and Visualization

I have used imputation method to replace null values. Due . And to remove skewness I have used yeo-johnson method. To encode the categorical columns I have use Ordinal Encoding. Use of Pearson's correlation coefficient to check the correlation between dependent and independent features. Also I have used standardization. Then followed by model building with all regression algorithms.

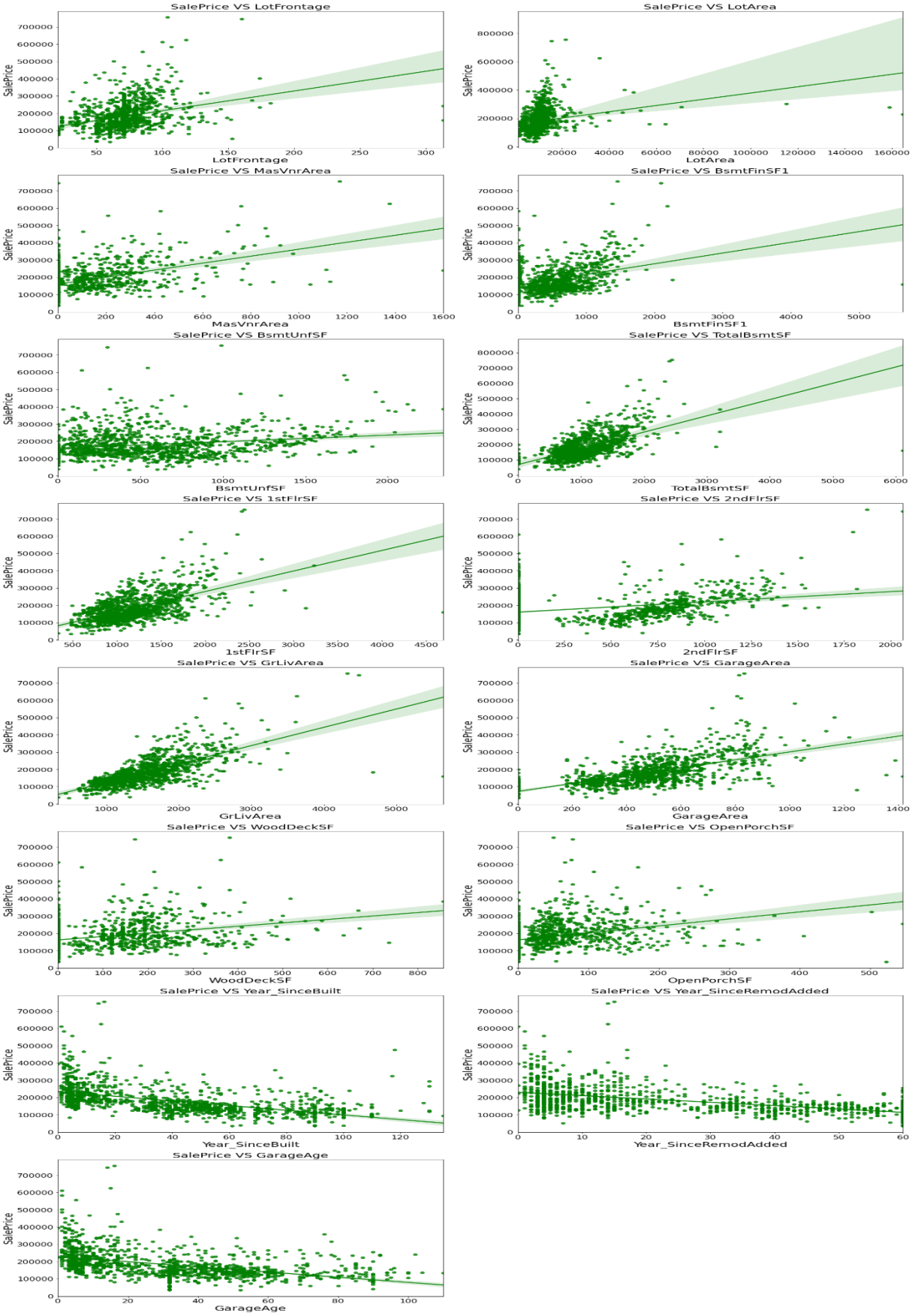
Since Saleprice was my target and it was a continuous column so this perticular problem was regression problem. And I have used all regression algorithms to build my model. By looking into the difference of r2 score and cross validation score I found ExtraTreesRegressor as a best model with least difference. Also to get the best model we have to run through multiple models and to avoid the confusion of overfitting we have go through cross validation. Below are the list of regression algorithms I have used in my project.

- ➢ RandomForestRegressor
- ➢ XGBRegressor
- ➢ ExtraTreesRegressor
- ➢ GradientBoostingRegressor
- ➢ DecisionTreeRegressor

Visualizations

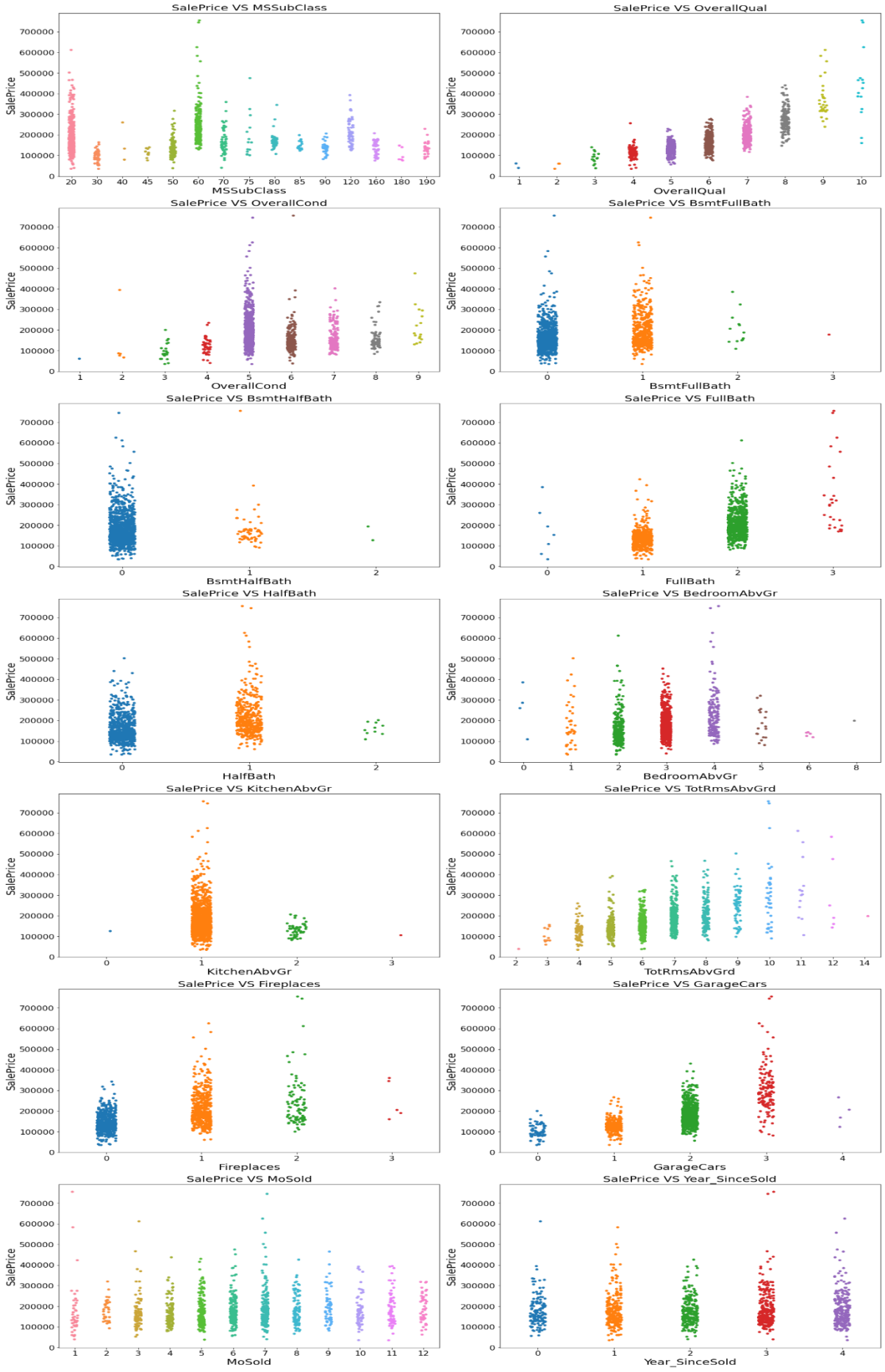
I have used bar plots to see the relation of categorical feature and I have used 2 types of plots for numerical columns one is strip plot for ordinal features and other is reg plot for continuous features.

1. **<u>Vizualization of numerical features with target:</u>**



Observations:

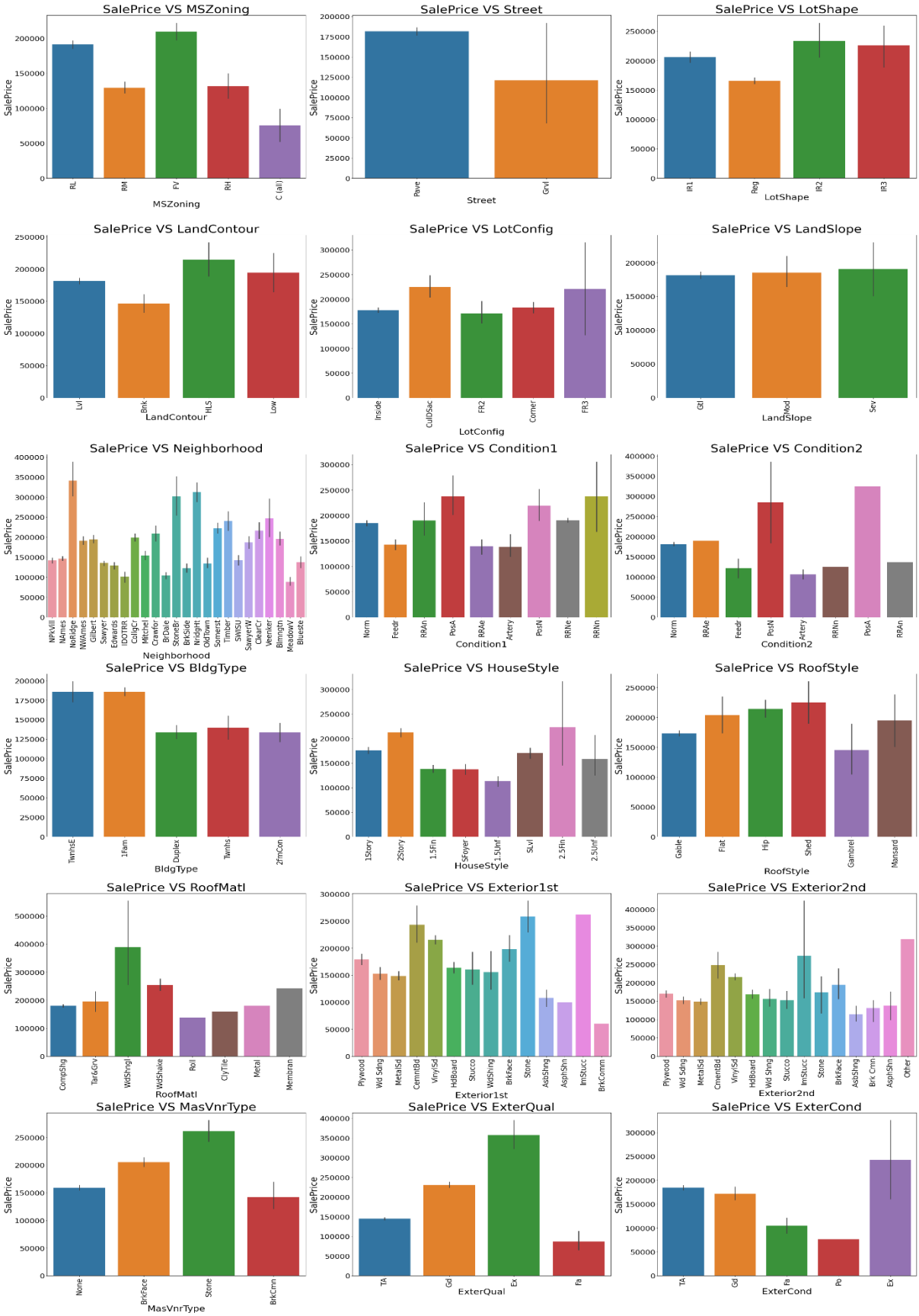
- ✓ 1.As Linear feet of street connected to property(LotFrontage) is increasing sales is decreasing and the SalePrice is ranging between 0-3 lakhs.
- ✓ 2.As Lot size in square feet(LotArea) is increasing sales is decreasing and the SalePrice is in between 0-4 lakhs.
- ✓ 3.As Masonry veneer area in square feet (MasVnrArea) is increasing sales is decreasing and SalePrice is ranging between 0-4 lakhs.
- ✓ 4.As Type 1 finished square feet(BsmtFinSF1) is increasing sales is decreasing and the SalePrice is in between 0-4 lakhs.
- ✓ 5.As Unfinished square feet of basement area (BsmtUnfSF) is increasing sales is decreasing and the SalePrice is in between 0-4 lakhs. There are some outliers also.
- ✓ 6.As Total square feet of basement area (TotalBsmtSF) is increasing sales is decreasing and the SalePrice is in between 0-4 lakhs.
- ✓ 7.As First Floor square feet(1stFlrSF) is increasing sales is decreasing and the SalePrice is in between 0-4 lakhs.
- ✓ 8.As Second floor square feet(2ndFlrSF) is increasing sales is increasing in the range 500-1000 and the SalePrice is in between 0-4 lakhs.
- ✓ 9.As Above grade (ground) living area square feet (GrLivArea) is increasing sales is decreasing and the SalePrice is in between 0-4 lakhs.
- ✓ 10.As Size of garage in square feet(GarageArea) is increasing sales is increasing and the SalePrice is in between 0-4 lakhs.
- ✓ 11.As Wood deck area in square feet(WoodDeckSF) is increasing sales is decreasing and the SalePrice is in between 0-4 lakhs.
- ✓ 12.As Open porch area in square feet (OpenPorchSF) is increasing sales is decreasing and the SalePrice is in between 0-4 lakhs.
- ✓ 13.As Year_SinceBuilt is increasing sales is decreasing and the SalePrice is high for newly built building and the sales price is in between 0-4 lakhs.
- ✓ 14.As Since Remodel date (same as construction date if no remodeling or additions)(Year_SinceRemodAdded) is increasing sales is decreasing and the SalePrice is in between 1-4 lakhs.
- ✓ 15.As Since Year garage was built(GarageAge) is increasing sales is decreasing and the SalePrice is in between 0-4 lakhs.

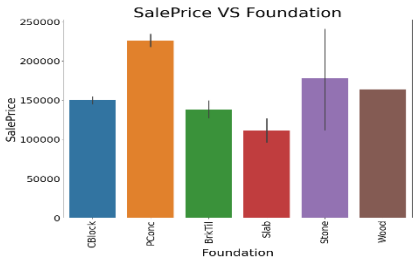

SalePrice VS MSSubClass
SalePrice
MSSubClass
SalePrice VS OverallQual
OverallQual
SalePrice VS OverallCond
OverallCond
SalePrice VS BsmtFullBath
BsmtFullBath
SalePrice VS BsmtHalfBath
BsmtHalfBath
SalePrice VS FullBath
FullBath
SalePrice VS HalfBath
HalfBath
SalePrice VS BedroomAbvGr
BedroomAbvGr
SalePrice VS KitchenAbvGr
KitchenAbvGr
SalePrice VS TotRmsAbvGrd
TotRmsAbvGrd
SalePrice VS Fireplaces
Fireplaces
SalePrice VS GarageCars
GarageCars
SalePrice VS MoSold
MoSold
SalePrice VS Year_SinceSold
Year_SinceSold

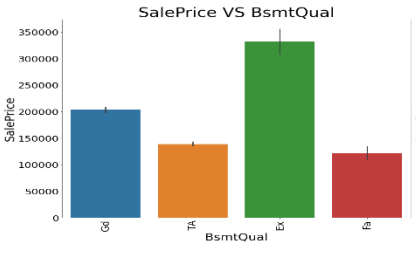
Observations:

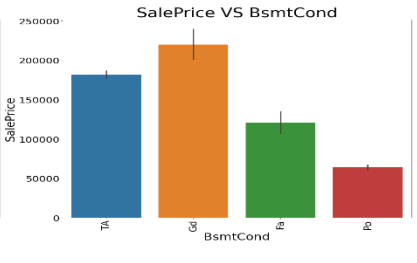
- ✓ 1.For 1-STORY 1946 & NEWER ALL STYLES (20) and 2-STORY 1946 & NEWER (60) types of dwelling (MSSuubClass) the sales is good and SalePrice is also high.
- ✓ 2.As Rates the overall material and finish of the house (OverallQual) is increasing linearly sales is also increasing And SalePrice is also increasing linearly.
- ✓ 3.For 5(Average) overall condition of the house (OverallCond) the sales is high and SalePrice is also high.
- ✓ 4.For 0 and 1 Basement full bathrooms (BsmtFullBath) the sales as well as SalePrice is high.
- ✓ 5.For 0 Basement half bathrooms (BsmtHalfBath) the sales as well as SalePrice is high.
- ✓ 6.For 1 and 2 Full bathrooms above grade (FullBath) the sales as well as SalePrice is high.
- ✓ 7.For 0 and 1 Half baths above grade (HalfBath) the sales as well as SalePrice is high.
- ✓ 8.For 2, 3 and 4 Bedrooms above grade (does NOT include basement bedrooms) (BedroomAbvGr) the sales as well as SalePrice is high.
- ✓ 9.For 1 Kitchens above grade (KitchenAbvGr) the sales as well as SalePrice is high.
- ✓ 10.For 4-9 Total rooms above grade (does not include bathrooms) (TotRmsAbvGrd) the sales as well as SalePrice is high.
- ✓ 11.For 0 and 1 Number of fireplaces (Fireplaces) the sales as well as SalePrice is high.
- ✓ 12.For 1 and 2 Size of garage in car capacity (GarageCars) the sales is high and for 3 Size of garage in car capacity (GarageCars) the SalePrice is high.
- ✓ 13.In between april to august for Month Sold (MoSold) the sales is good with SalePrice.
- ✓ 14.For all the Year_SinceSold the SalePrice and sales both are same.

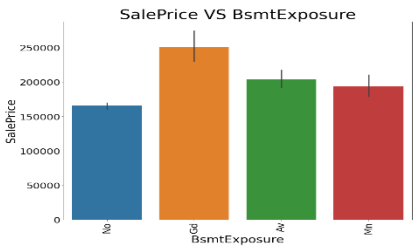
2. Visualization of categorical features with target:

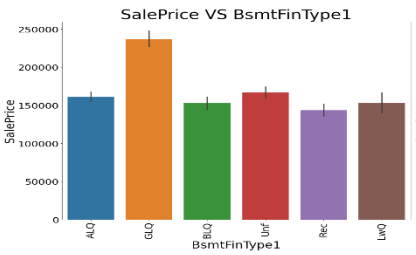


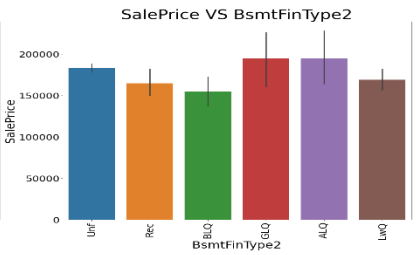

SalePrice VS Foundation
SalePrice
Foundation
CBlock
PConc
BrkTil
Slab
Stone
Wood

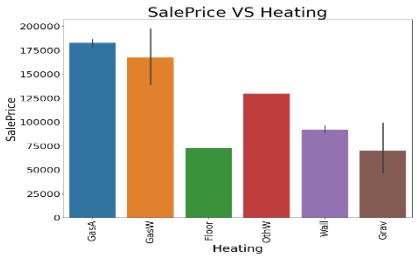

SalePrice VS BsmtQual
SalePrice
BsmtQual
Gd
TA
Ex
Fa

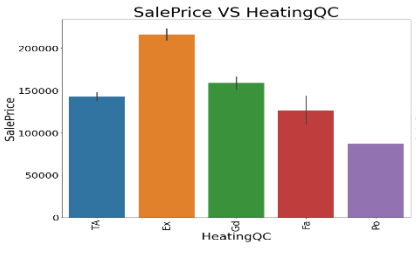

SalePrice VS BsmtCond
SalePrice
BsmtCond
TA
Gd
Fa
Po

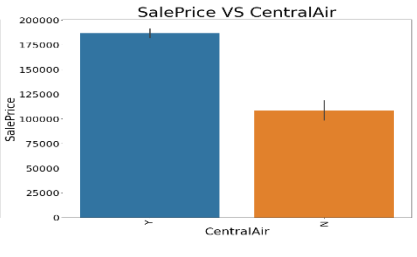

SalePrice VS BsmtExposure
SalePrice
BsmtExposure
No
Gd
Av
Mn

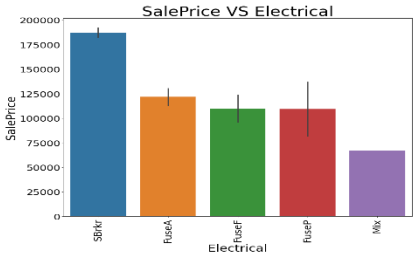

SalePrice VS BsmtFinType1
SalePrice
BsmtFinType1
ALQ
GLQ
BLQ
Unf
Rec
LwQ

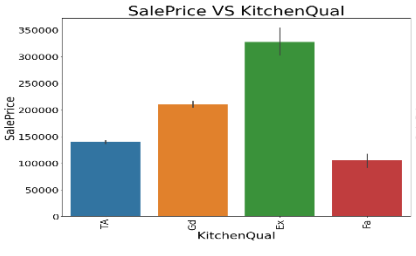

SalePrice VS BsmtFinType2
SalePrice
BsmtFinType2
Unf
Rec
BLQ
GLQ
ALQ
LwQ

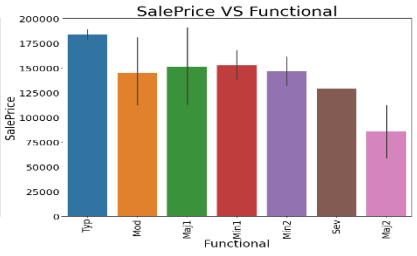

SalePrice VS Heating
SalePrice
Heating
GasA
GasW
Floor
OthW
Wall
Grav

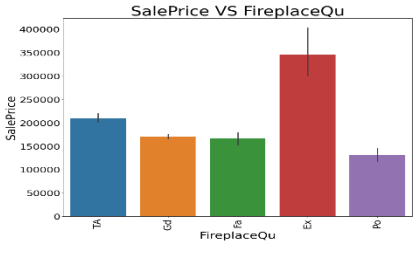

SalePrice VS HeatingQC
SalePrice
HeatingQC
TA
Ex
Gd
Fa
Po

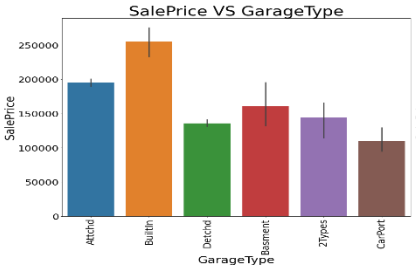

SalePrice VS CentralAir
SalePrice
CentralAir
Y
N

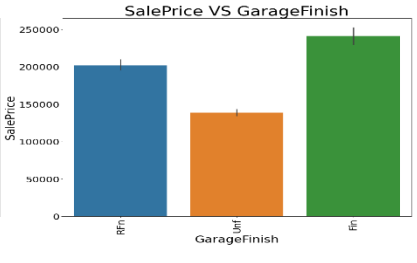

SalePrice VS Electrical
SalePrice
Electrical
SBrkr
FuseA
FuseF
FuseP
Mix

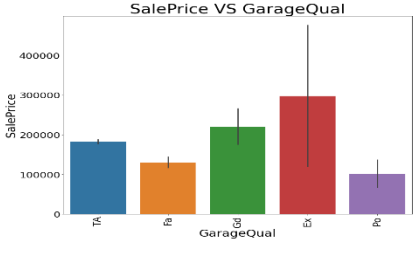

SalePrice VS KitchenQual
SalePrice
KitchenQual
TA
Gd
Ex
Fa

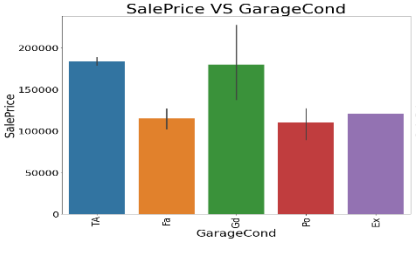

SalePrice VS Functional
SalePrice
Functional
Typ
Mod
Maj1
Min1
Min2
Sev
Maj2

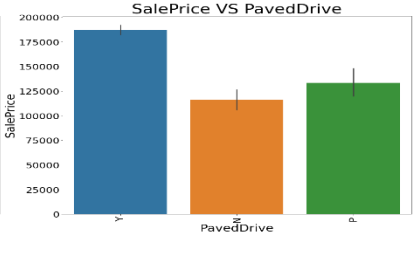

SalePrice VS FireplaceQu
SalePrice
FireplaceQu
TA
Gd
Fa
Ex
Po

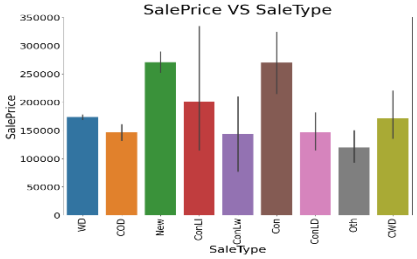

SalePrice VS GarageType
SalePrice
GarageType
Attchd
BuiltIn
Detchd
Basment
2Types
CarPort

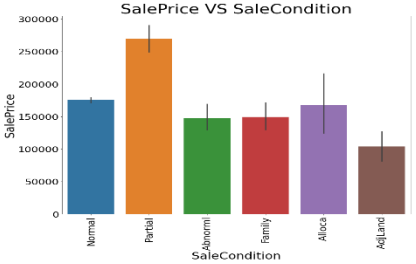

SalePrice VS GarageFinish
SalePrice
GarageFinish
RFn
Unf
Fin


SalePrice VS GarageQual
SalePrice
GarageQual
TA
Fa
Gd
Ex
Po


SalePrice VS GarageCond
SalePrice
GarageCond
TA
Fa
Gd
Po
Ex


SalePrice VS PavedDrive
SalePrice
PavedDrive
Y
N
P


SalePrice VS SaleType
SalePrice
SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD


SalePrice VS SaleCondition
SalePrice
SaleCondition
Normal
Partial
Abnorml
Family
Alloca
AdjLand

Observations:

- ✓ 1. For Floating Village Residential (FV) and Residential Low Density(RL) zoning classification of the sale(MSZoning) the SalePrice is high.
- ✓ 2. For paved type of road access to property (Street) the SalePrice is high.
- ✓ 3. For Slightly irregular (IR1), Moderately Irregular (IR2) and Irregular (IR3) shape of property (LotShape) the SalePrice is high.
- ✓ 4. For Hillside - Significant slope from side to side (HLS) Flatness of the property (LandContour) the SalePrice is High.
- ✓ 5. For Cul-de-sac (CulDSac) Lot configuration (LotConfig) the SalePrice is High.
- ✓ 6. For all types of Slope of property (LandSlope) i.e., Gentle slope (Gtl), Moderate Slope (Mod) and Severe Slope (Sev) the SalePrice is High.
- ✓ 7. For Northridge (NoRidge) locations within Ames city limits (Neighborhood) the SalePrice is High.
- ✓ 8. For Within 200' of North-South Railroad (RRNn), Adjacent to postive off-site feature (PosA) and Near positive off-site feature--park, greenbelt, etc. (PosN) Proximity to various conditions(Condition1) has the maximum SalePrice.
- ✓ 9. For Adjacent to positive off-site feature (PosA) and Near positive off-site feature--park, greenbelt, etc.(PosN) Proximity to various conditions (if more than one is present) (Condition2) has maximum SalePrice.
- ✓ 10. For Single-family Detached(1Fam) and Townhouse End Unit (TwnhsE) type of dwelling (BldgType) the SalePrice is high.
- ✓ 11. For 2Story and Two and one-half story: 2nd level finished(2.5Fin) Style of dwelling (HouseStyle) the SalePrice is high.
- ✓ 12. For Shed Type of roof (RoofStyle) the SalePrice is high.
- ✓ 13. For Wood Shingles (WdShngl) Roof material (RoofMat1) the SalePrice is high.
- ✓ 14. For Cement Board (CemntBd), Imitation Stucco (ImStucc) and Stone type of Exterior covering on house(Exterior1st) the SalePrice is high.
- ✓ 15. For Cement Board (CemntBd), Imitation Stucco (ImStucc) and other Exterior covering on house (if more than one material) (Exterior2) has maximum SalePrice.
- ✓ 16. For Stone Masonry veneer type (MasvnrType) the SalePrice is high.
- ✓ 17. For Excellent (Ex) quality of the material on the exterior(ExterQual) the SalePrice is high.

- ✓ 18. For Excellent (Ex) present condition of the material on the exterior (ExterCond) the SalePrice is high.
- ✓ 19. For Poured Contrete (PConc) Type of foundation (Foundation) the SalePrice is high.
- ✓ 20. For Excellent (100+ inches) (Ex) height of the basement (BsmtQual) the SalePrice is high.
- ✓ 21. For Good (Gd) general condition of the basement (BsmtCond) the SalePrice is high.
- ✓ 22. For Good Exposure (Gd) of walkout or garden level walls (BsmtExposure) has maximum SalePrice.
- ✓ 23. For Good Living Quarters (GLQ) of basement finished area (BsmtFinType1) has maximum SalePrice.
- ✓ 24. For Good Living Quarters (GLQ) and Average Living Quarters (ALQ) of basement finished area (if multiple types) (BsmtFinType2) has maximum SalePrice.
- ✓ 25. For Gas forced warm air furnace (GasA) and Gas hot water or steam heat (GasW) Type of heating(Heating) has high SalePrice.
- ✓ 26. For Excellent (Ex) Heating quality and condition (HeatingQC) the SalePriceis high.
- ✓ 27. For building having Central air conditioning (CentralAir) the SalePrice is high.
- ✓ 28. For Standard Circuit Breakers & Romex (Sbrkr) of Electrical system (Electrical) the SalePrice is Maximum.
- ✓ 29. For Excellent (Ex) Kitchen quality (KitchenQual) the SalePrice is high.
- ✓ 30. For Typical Functionality (Typ) type of Home functionality (Assume typical unless deductions are warranted) (Functional) the SalePrice is high.
- ✓ 31. For Excellent - Exceptional Masonry Fireplace (Ex) of Fireplace quality (FireplaceQual) has highest SalePrice.
- ✓ 32. For Built-In (Garage part of house - typically has room above garage) (BuiltIn) Garage location (GarageType) the SalePrice is maximum.
- ✓ 33. For Completely finished (Fin) Interior of the garage (GarageFinish) the SalePrice is high.
- ✓ 34. For Excellent (Ex) Garage quality (GarageQual) the SalePrice is high.
- ✓ 35. For Typical/Average (TA) and Good (Gd) Garage condition (GarageCond) the SalePrice is high.
- ✓ 36. For having Paved driveway (PavedDrive) the SalePriceis high.

- ✓ 37. For Home just constructed and sold (New) and Contract 15% Down payment regular terms (Con) of type of sale (SaleType) has highest SalePrice.
- ✓ 38. For Home was not completed when last assessed (associated with New Homes) (Partial) Condition of sale (SalesCondition) the SalePrice is maximum.

Model Building:

i) RandomForestRegressor:

```
RFR=RandomForestRegressor()
RFR.fit(X_train,y_train)
pred=RFR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))

#cross validation score
scores = cross_val_score(RFR, X, y, cv = 10).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 90.01341845801399
mean_squared_error: 683569694.940551
mean_absolute_error: 17189.93743589744
root_mean_squared_error: 26145.165804418815

Cross validation score : 83.28137603308227

R2_Score - Cross Validation Score : 6.73204242493172
```

RandomForestRegressor is giving me 90.01% r2_score.

- RandomForestRegressor has 90.01%

ii) XGBRegressor:

```
XGB=XGBRegressor()
XGB.fit(X_train,y_train)
pred=XGB.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))

#cross validation score
scores = cross_val_score(XGB, X, y, cv = 10).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 89.54760265240213
mean_squared_error: 715454235.8920137
mean_absolute_error: 17993.274227653135
root_mean_squared_error: 26747.976295264165

Cross validation score : 82.8682192056057

R2_Score - Cross Validation Score : 6.679383446796436
```

XGBRegressor is giving me 89.547% r2_score.

- XGBRegressor is giving me 89.54% accuracy.

iii) ExtraTreesRegressor:

```
ETR=ExtraTreesRegressor()
ETR.fit(X_train,y_train)
pred=ETR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))

#cross validation score
scores = cross_val_score(ETR, X, y, cv = 10).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 85.88777187931811
mean_squared_error: 965965323.6524378
mean_absolute_error: 18404.9447008547
root_mean_squared_error: 31079.982684236453

Cross validation score : 82.74943288176182

R2_Score - Cross Validation Score : 3.138338997556289
```

- ExtraTreeRegressor is giving me 85.88% accuracy.

iv) GradientBoostingRegressor:

```
|:  1  GBR=GradientBoostingRegressor()
    2  GBR.fit(X_train,y_train)
    3  pred=GBR.predict(X_test)
    4  R2_score = r2_score(y_test,pred)*100
    5  print('R2_score:',R2_score)
    6  print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
    7  print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
    8  print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))
    9
   10  #cross validation score
   11  scores = cross_val_score(GBR, X, y, cv = 10).mean()*100
   12  print("\nCross validation score :", scores)
   13
   14  #difference of accuracy and cv score
   15  diff = R2_score - scores
   16  print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 89.2464753565624
mean_squared_error: 736066043.1345657
mean_absolute_error: 16937.440324617633
root_mean_squared_error: 27130.537096315762

Cross validation score : 84.47187957531798

R2_Score - Cross Validation Score : 4.774595781244415
```

- GradientBoostingRegressor is giving me 89.24% accuracy.

v) DecisionTreeRegressor:

```
:  1  DTR=DecisionTreeRegressor()
   2  DTR.fit(X_train,y_train)
   3  pred=DTR.predict(X_test)
   4  R2_score = r2_score(y_test,pred)*100
   5  print('R2_score:',R2_score)
   6  print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
   7  print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
   8  print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))
   9
  10  #cross validation score
  11  scores = cross_val_score(DTR, X, y, cv = 10).mean()*100
  12  print("\nCross validation score :", scores)
  13
  14  #difference of accuracy and cv score
  15  diff = R2_score - scores
  16  print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 78.13917864005089
mean_squared_error: 1496347366.2478633
mean_absolute_error: 27685.239316239316
root_mean_squared_error: 38682.64942125686

Cross validation score : 68.6350858139493

R2_Score - Cross Validation Score : 9.504092826101584
```

- DecisionTreeRegressor is giving me 78.13 % accuracy.

- ✓ **By looking into the difference of model accuracy and cross validation score I found ExtraTreesRegressor as the best model.**

2. Hyper Parameter Tunning:

```
parameter = {'n_estimators':[50,100,150,200],
             'criterion':['squared_error','absolute_error','friedman_mse','poisson'],
             'min_samples_split': [1,2,3,4],
             'max_features':[None,'sqrt','log2'],
             'max_leaf_nodes':[3,5,7,9]
            }
```

```
#importing necessary libraries
from sklearn.model_selection import GridSearchCV
```

Giving ETR parameters.

```
GCV=GridSearchCV(ExtraTreesRegressor(),param_grid=parameter,cv= 5)
```

Running grid search CV for ETR.

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

```
GridSearchCV(cv=5, estimator=ExtraTreesRegressor(),
             param_grid={'criterion': ['squared_error', 'absolute_error',
                                       'friedman_mse', 'poisson'],
                         'max_features': [None, 'sqrt', 'log2'],
                         'max_leaf_nodes': [3, 5, 7, 9],
                         'min_samples_split': [1, 2, 3, 4],
                         'n_estimators': [50, 100, 150, 200]})
```

- ✓ **I have choosed all parameters of ExtraTreesRegressor, after tunning the model with best parameters the accuracy is not increasing So selected the Model with out hypertuning as the final model.**

3. Saving the model and Predicting SalePrice for test data:

- ✓ I have saved my best model using .pkl as follows.

```
# Saving the model using .pkl
import joblib
joblib.dump(ETR,"House_Price.pkl")
```

```
['House_Price.pkl']
```

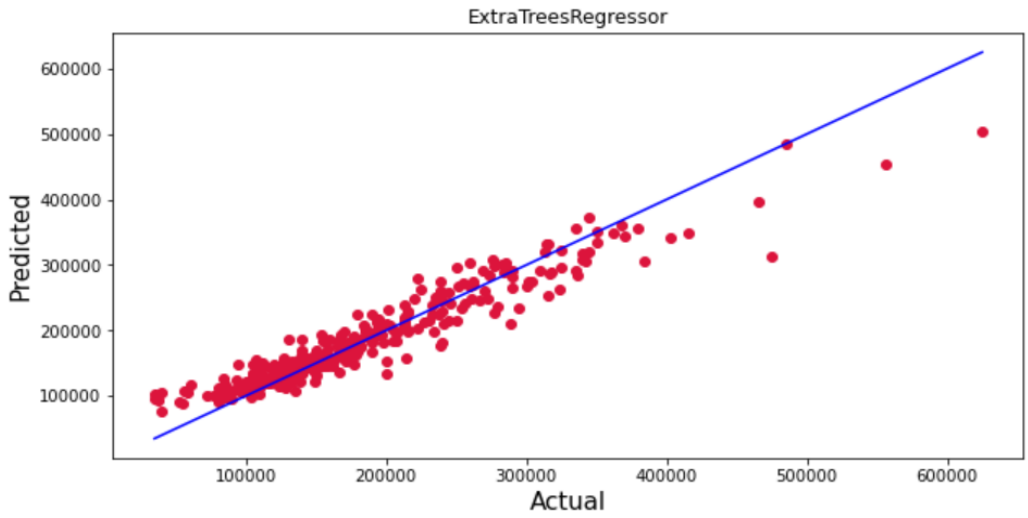
✓ Now loading my saved model and predicting the test values.

In [154]:
```
# Loading the saved model
model=joblib.load("House_Price.pkl")

#Prediction
prediction = model.predict(X_test)
prediction
```

Out[154]:
```
array([140977.27, 170414.11, 170730.  , 257451.13, 104874.26, 268361.98,
       182314.95, 125553.57, 147464.7 , 109852.54,  93960.76, 152544.97,
       260127.23, 167394.84, 139925.  , 181546.82, 193219.89, 156425.95,
       307690.31, 165631.28, 134470.  , 158503.53, 139133.47, 291347.73,
       119164.5 , 156585.87, 246917.35, 241829.36, 111747.29, 156864.24,
       179873.48, 126682.81, 193361.09, 121787.72, 162473.32, 162147.4 ,
       275202.24, 128042.33, 115563.76, 151898.  , 296053.28, 173995.27,
       454958.31,  76614.  , 104972.  , 275060.35, 139640.78, 313333.62,
       238096.5 , 137019.52, 150230.5 , 123386.84, 183441.89,  97920.78,
       215050.88, 155898.  , 224453.46, 125990.83, 357130.09, 301347.09,
       133470.52, 156223.25, 186020.3 , 147906.45, 168416.27, 199230.3 ,
       223882.06, 248581.22, 194205.47, 113470.8 , 121795.15, 108034.59,
       373882.67, 144560.58, 148657.51, 309087.46, 167947.83, 304599.52,
       319815.12, 150737.22, 119970.66, 148388.95, 122282.76, 125017.07,
       288136.17, 100110.4 , 124867.  , 197980.5 , 291299.48, 263820.12,
       169959.15,  96625.  , 167259.33,  95847.22, 110467.52, 225936.09,
        90963.59, 141005.85, 116783.4 , 303052.17, 213748.74, 116688.  ,
       137795.77, 291899.45, 116566.  , 124102.76, 248220.3 , 273213.01,
       164402.25, 213126.35, 251845.25, 133330.76, 190637.46, 102982.66,
       229813.5 , 130979.83, 157784.53, 220920.54, 288951.7 , 212893.77,
       295892.85, 130230.  , 111078.  , 103241.84, 187401.92, 148538.41,
       204257.29, 145044.09, 154383.37, 198388.1 , 344334.43, 287186.91,
```

In [156]:
```
plt.figure(figsize=(10,5))
plt.scatter(y_test, prediction, c='crimson')
p1 = max(max(prediction), max(y_test))
p2 = min(min(prediction), min(y_test))
plt.plot([p1, p2], [p1, p2], 'b-')
plt.xlabel('Actual', fontsize=15)
plt.ylabel('Predicted', fontsize=15)
plt.title("ExtraTreesRegressor")
plt.show()
```



- ✓ Plotting Actual vs Predicted, To get better insight. Bule line is the actual line and red dots are the predicted values.

Predicting SalePrice for test dataset:

```
#Predicting Sale price of house using cleaned test dataset X_1
Predicted_Sale_Price=model.predict(X_1)
Predicted_Sale_Price
```

```
array([328527.69, 261656.  , 261053.71, 166079.25, 226968.58,  93425.29,
       142303.64, 357155.77, 253491.93, 158561.78,  83660.83, 141478.5 ,
       117012.17, 164669.5 , 275048.03, 132550.  , 123199.84, 131745.5 ,
       174962.1 , 205800.24, 148466.09, 153141.  , 150755.3 , 112445.11,
       113271.04, 133659.25, 178084.43, 152632.5 , 172567.  , 112223.  ,
       136557.9 , 207016.28, 220432.27, 160613.74, 116745.7 , 180428.77,
       186829.82, 118534.34, 159914.  , 150091.33, 109494.52, 342708.5 ,
       214941.99, 200120.64, 132952.19, 123722.  , 126564.34, 108095.16,
       220549.2 , 333666.17, 142022.5 , 234959.22, 109763.26,  91196.8 ,
       270071.09, 112418.35, 148415.58, 192526.84, 114658.  , 251610.19,
        98540.  , 201366.3 , 136703.  , 143614.8 , 213815.88, 105411.59,
       153736.  , 219842.18, 139477.18, 160035.  , 275840.7 , 170553.  ,
       180692.29, 141197.  , 144451.  , 229149.03, 340554.92, 218221.9 ,
       296425.82, 146958.5 , 233614.09, 128466.8 , 133447.5 , 160785.  ,
       181522.6 , 241018.34, 125910.95, 353079.04, 149697.5 , 178349.26,
       241473.2 , 138417.  , 133100.16, 136472.  , 190238.82, 171477.37,
       272053.86, 176421.44, 339099.5 , 124985.66, 240692.71, 105796.26,
       139263.8 , 163225.52, 192296.57, 139668.27, 297304.49, 135567.  ,
       188790.29, 192369.95, 195543.4 , 164467.32, 160491.21, 245213.01,
       118891.68,  99221.12, 139018.43, 188618.1 , 138085.26, 120143.5 ,
       104502.34, 202478.93, 202451.74, 130941.92, 146029.5 , 186978.45,
       127260.84, 153558.5 ,  91052.72, 108763.6 , 153990.  , 224619.55,
       144310.  , 170938.5 , 170853.  , 330761.39, 220969.9 , 134135.  ,
       243981.67, 138484.34, 149047.64, 402108.88,  92611.78, 349023.56,
       162469.75, 225312.5 , 181578.5 , 114084.1 , 107074.81, 207486.06,
       170300.81, 144429.34, 198656.15, 107720.98, 104808.  , 183450.2 ,
       170898.  , 182053.74, 134275.5 , 180239.43, 213228.14, 141638.18,
       195480.52, 120269.15, 112762.68, 262578.28, 183000.95, 202730.44,
       134051.5 , 220167.3 , 170438.85, 126401.35, 138458.  , 282277.81,
       135163.  , 405448.17, 123120.8 , 117158.98, 163445.33, 146054.5 ,
       209993.07, 167334.5 , 251467.69, 173872.42, 448146.64, 342876.28,
       209141.09, 132475.  , 169194.33, 149353.32, 112847.58, 212234.39,
       209369.63, 107083.52, 132328.08,  67301.66, 159427.28, 211725.85,
       130190.08, 143719.5 , 147961.81, 130535.5 , 265186.65, 426001.22,
       134539.08, 130784.95, 262453.28, 149870.  , 121388.09, 139027.23,
       108132.25, 154814.  , 143372.5 , 189569.79,  96536.12, 167993.44,
       144764.3 , 128624.09, 123900.67, 174649.46, 214670.42, 227921.92,
       288909.95, 129590.42, 192991.15, 346220.44, 238001.3 , 104393.64,
       352467.4 , 187921.97, 119596.52, 161215.85, 104241.86, 138243.  ,
       124149.58, 203087.59, 178967.38, 156294.5 , 274929.73, 193827.48,
       237595.7 , 145384.18, 300590.1 ,  83816.95, 149420.82, 237089.81,
       206033.62, 276139.4 , 121132.06, 136486.34, 139648.24, 228316.51,
       159909.84, 110405.28,  99935.84, 197584.35, 119249.54, 388075.69,
       201078.3 , 121579.5 , 180159.37, 187486.86,  99622.33, 147020.37,
       208109.4 , 176041.73, 110911.26, 179151.29, 250465.8 , 226653.94,
       154037.  , 130951.25, 356626.05, 247659.64, 290817.75, 223776.93,
       186909.19, 115992.8 , 192863.12, 386625.29, 134827.  , 372277.26,
       146891.52, 338406.8 , 156120.5 , 119910.01, 173408.74, 243426.9 ,
       136237.18, 163074.5 , 182900.  , 104344.  ])
```

```
#Making dataframe for predicted SalePrice
House_Price_Predictions=pd.DataFrame()
House_Price_Predictions["SalePrice"]=Predicted_Sale_Price
House_Price_Predictions.head(10)
```

	SalePrice
0	328527.69
1	261656.00
2	261053.71
3	166079.25
4	226968.58
5	93425.29
6	142303.64
7	357155.77
8	253491.93
9	158561.78

```
#Lets save the predictions to csv
House_Price_Predictions.to_csv("House_Price_Predictions.csv",index=False)
```

4.CONCLUSION

In this project report, we have used machine learning algorithms to predict the house prices. We can select the features which are not correlated to each other and are independent in nature. These feature set were then given as an input to five algorithms and a csv file was generated consisting of predicted house prices. Hence we calculated the performance of each model using different performance metrics and compared them based on these metrics. Then we have also saved the dataframe of predicted prices of test dataset.

- ✓ we have tried best to deal with outliers, skewness, null values and zero values. So it looks quite good that we have achieved a accuracy of 85.88% even after dealing all these drawbacks.
- ✓ **This model doesn't predict future prices** of the houses mentioned by the customer. Due to this, the risk in investment in an apartment or an area increases considerably. To minimize this error, customers tend to hire an agent which again increases the cost of the process.