PFA HOUSING PROJECT

Submitted by:

Saransh Sharda

ACKNOWLEDGMENT

The research, references and data source I got for this PFA housing project is from Flip robo technologies which guided me to get complete analysis through this project.

INTRODUCTION

- ## Business Problem Framing

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

- ## 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:

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

- Review of Literature

The comprehensive summary of the research done on the project is the higher the number of utilities in the house , will be having high sale price of houses.

I was 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 and my model score is 89.9.

- Motivation for the Problem Undertaken

My objective behind to make this project is to find out the reason and utilities that are considered within to declare a house sale price at a particular places which involves number of factors ad where these all factors are co related to declare the house sale prices.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

Descriptive and prescriptive modelling is performed during the model construction with having data set of 1000 records with 80 columns. Prediction made is by filling null values with mode.

- Data Sources and their formats

  Data contains 1460 entries each having 81 variables.

  Data contains Null values.
  DATA SOURCE is from flip robo technologies where I was 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.

- Data Pre-processing Done

  Only assumptions made is that, outliers weren't removed as I was losing too much data and also null values filled with mode mostly.

  Cleaning of data done via filling/dropping null values as

```
LotFrontage 18.3219 % Missing Values
Alley 93.4075 % Missing Values
MasVnrType 0.5993 % Missing Values
MasVnrArea 0.5993 % Missing Values
BsmtQual 2.5685 % Missing Values
BsmtCond 2.5685 % Missing Values
BsmtExposure 2.6541 % Missing Values
BsmtFinType1 2.5685 % Missing Values
BsmtFinType2 2.6541 % Missing Values
FireplaceQu 47.1747 % Missing Values
GarageType 5.4795 % Missing Values
GarageYrBlt 5.4795 % Missing Values
GarageFinish 5.4795 % Missing Values
GarageQual 5.4795 % Missing Values
GarageCond 5.4795 % Missing Values
PoolQC 99.4007 % Missing Values
Fence 79.7089 % Missing Values
MiscFeature 96.2329 % Missing Values
```

  Mostly features having more than 75% data missing were removed

- Data Inputs- Logic- Output Relationships

The train dataset has 1168 rows and 81 columns. 2. The test dataset has 292 rows and 80 columns

Feature engineering

```
02]: df.columns

02]: Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

Starting from society to the end of bathroom/basement, house project has every kind of column when it comes to sale prices and sale of house and these input features affected output target in a way where every utility is defined to increase/decrease house prices in a way.

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

- Descriptive and prescriptive modelling is performed during the model construction with having data set of 1000 records with 80 columns. Prediction made is by filling null values with mode and outliers weren't removed as it got handled by regressor booster.

- Hardware and Software Requirements and Tools Used

  Listing down the hardware and software requirements along with the tools, libraries and packages used-

```
]: import numpy as np

   import pandas as pd

   import matplotlib.pyplot as plt
   import seaborn as sns

   from scipy import stats

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

These packages were used for making lot of graph to check the effect on sale prices and for data cleaning and feature engineering

Other than that standard scaler was used to scale the data and for model building -

```
# import training dependencies
from sklearn.linear_model import LinearRegression,Ridge,Lasso,ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.neighbors import  KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor,AdaBoostRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge,Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegressor
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

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

model building algorithms were used and cross validation score was checked so imported above from scikitlearn libraries.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  After data cleaning, as data was getting loss from removing the outliers so outliers weren't removed as it was taken care by boosting techniques used under model building and standard scaler was used for scaling process and for visualisation , every column was checked for the effect of input to output via violin, rel plot, bar graphs etc.

  Testing of Identified Approaches (Algorithms)

```
# import training dependencies
from sklearn.linear_model import LinearRegression,Ridge,Lasso,ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.neighbors import  KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor,AdaBoostRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge,Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegressor
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

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

R2 score was checked for the best model algorithm and also cross validation score was checked and finally randomised search cross validation was used to get 89.9 score for the model

- Run and Evaluate selected models

Linear Regression

Linear regression is been studied at great length, and there is a lot of literature on how your data must be structured to make best use of the model.

As such, there is a lot of sophistication when talking about these requirements and expectations which can be intimidating. In practice, you can uses these rules more as rules of thumb when using Ordinary Least Squares Regression, the most common implementation of linear regression.

Try different preparations of your data using these heuristics and see what works best for your problem.

- **Linear Assumption.** Linear regression assumes that the relationship between your input and output is linear. It does not support anything else. This may be obvious, but it is good to remember when you have a lot of attributes. You may need to transform data to make the relationship linear (e.g. log transform for an exponential relationship).
- **Remove Noise.** Linear regression assumes that your input and output variables are not noisy. Consider using data cleaning operations that let you better expose and clarify the signal in your data. This is most important for the output variable and you want to remove outliers in the output variable (y) if possible.
- **Remove Collinearity.** Linear regression will over-fit your data when you have highly correlated input variables. Consider calculating pairwise correlations for your input data and removing the most correlated.
- **Gaussian Distributions.** Linear regression will make more reliable predictions if your input and output variables have a Gaussian distribution. You may get some benefit using transforms (e.g. log or BoxCox) on you variables to make their distribution more Gaussian looking.
- **Rescale Inputs:** Linear regression will often make more reliable predictions if you rescale input variables using standardization or normalization.

```
test_pred = lin_reg.predict(X_test)
train_pred = lin_reg.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('====================================')
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:

_____________________________________
MAE: 5654298215795037.0
MSE: 1.9309287541672116e+33
RMSE: 4.394233441872668e+16
R2 Square -3.314273227648219e+23
====================================
Train set evaluation:

_____________________________________
MAE: 13417.914575595196
MSE: 437466009.19350743
RMSE: 20915.688111881653
R2 Square 0.9299659449536839
```

Random Sample Consensus - RANSAC

Random sample consensus (`RANSAC`) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates. Therefore, it also can be interpreted as an outlier detection method.

A basic assumption is that the data consists of "inliers", i.e., data whose distribution can be explained by some set of model parameters, though may be subject to noise, and "outliers" which are data that do not fit the model. The outliers can come, for example, from extreme values of the noise or from erroneous measurements or incorrect hypotheses about the interpretation of data. RANSAC also assumes that, given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data.

```
model = RANSACRegressor(base_estimator=LinearRegression(), max_trials=100)
model.fit(X_train, y_train)

test_pred = model.predict(X_test)
train_pred = model.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('====================================')
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:
_____________________________________
MAE: 8465553132291403.0
MSE: 9.440647768585423e+32
RMSE: 3.0725637126974964e+16
R2 Square -1.6204060395057984e+23
====================================
Train set evaluation:
_____________________________________
MAE: 5332891476664644.0
MSE: 7.006293238125499e+32
RMSE: 2.64694035409291e+16
R2 Square -1.1216394325449587e+23
```

3. Ridge Regression

Source: [scikit-learn]

Ridge regression addresses some of the problems of **Ordinary Least Squares** by imposing a penalty on the size of coefficients. The ridge coefficients minimize a penalized residual sum of squares,

$$\min_w ||Xw-y||_2^2+\alpha||w||_2^2 \quad \text{minw||Xw−y||22+α||w||22}$$

$\alpha>=0$α>=0 is a complexity parameter that controls the amount of shrinkage: the larger the value of αα, the greater the amount of shrinkage and thus the coefficients become more robust to collinearity.

Ridge regression is an L2 penalized model. Add the squared sum of the weights to the least-squares cost function.

```
model = Ridge(alpha=100, solver='cholesky', tol=0.0001, random_state=42)
model.fit(X_train, y_train)
pred = model.predict(X_test)

test_pred = model.predict(X_test)
train_pred = model.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('====================================')
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:

_____________________________________
MAE: 21627.91548382347
MSE: 1163748353.7067196
RMSE: 34113.75607737617
R2 Square 0.8002525984407258
====================================
Train set evaluation:

_____________________________________
MAE: 15795.191384070411
MSE: 629916269.2662129
RMSE: 25098.132784456553
R2 Square 0.8991565293091238
```

4. LASSO Regression

A linear model that estimates sparse coefficients.
Mathematically, it consists of a linear model trained with ℓ_1ℓ1 prior as regularizer. The objective function to minimize is:

$$\min_w 12n_{samples}||||Xw-y||||_{22}+\alpha||||w||||_1$$minw12nsamples||Xw−y||22+α||w||1

The lasso estimate thus solves the minimization of the least-squares penalty with $\alpha||||w||||_1$α||w||1 added, where αα is a constant and $||||w||||_1$||w||1 is the ℓ_1-normℓ1−norm of the parameter vector.

```
print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('====================================')
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:

_____________________________________
MAE: 26349.369376440187
MSE: 3092313620.6137357
RMSE: 55608.57506368722
R2 Square 0.46923094794803977
====================================
Train set evaluation:

_____________________________________
MAE: 16212.931559747412
MSE: 579806600.3916348
RMSE: 24079.173581990617
R2 Square 0.9071786001319168
```

5. Elastic Net

A linear regression model trained with L1 and L2 prior as regularizer.

This combination allows for learning a sparse model where few of the weights are non-zero like Lasso, while still maintaining the regularization properties of Ridge.

Elastic-net is useful when there are multiple features which are correlated with one another. Lasso is likely to pick one of these at random, while elastic-net is likely to pick both.

A practical advantage of trading-off between Lasso and Ridge is it allows Elastic-Net to inherit some of Ridge's stability under rotation.

The objective function to minimize is in this case

$$\min_w 12n_{samples}||||Xw-y||||22+\alpha\rho||||w||||1+(1-\rho)2||||w||||22$$

```
test_pred = model.predict(X_test)
train_pred = model.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('==================================')
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:

_____________________________________
MAE: 22576.10603137892
MSE: 1350162374.5061197
RMSE: 36744.555712460584
R2 Square 0.7682562341491717
==================================
Train set evaluation:

_____________________________________
MAE: 14641.191552730044
MSE: 486102318.23398584
RMSE: 22047.728187593068
R2 Square 0.92217974471639
```

6. Polynomial Regression

Source: scikit-learn

One common pattern within machine learning is to use linear models trained on nonlinear functions of the data. This approach maintains the generally fast performance of linear methods, while allowing them to fit a much wider range of data.

For example, a simple linear regression can be extended by constructing polynomial features from the coefficients. In the standard linear regression case, you might have a model that looks like this for two-dimensional data:

$$\hat{y}(w,)=w0+w1x1+w2x2 \quad \text{y^(w,x)=w0+w1x1+w2x2}$$

If we want to fit a paraboloid to the data instead of a plane, we can combine the features in second-order polynomials, so that the model looks like this:

$\hat{y}(w,)=w0+w1x1+w2x2+w3x1x2+w4x21+w5x22$y^(w,x)=w0+w1x1+w2x2+w3x1x2+w4x12+w5x22

The (sometimes surprising) observation is that this is still a linear model: to see this, imagine creating a new variable

$z=[x1,x2,x1x2,x21,x22]$z=[x1,x2,x1x2,x12,x22]

With this re-labeling of the data, our problem can be written

$\hat{y}(w,)=w0+w1z1+w2z2+w3z3+w4z4+w5z5$y^(w,x)=w0+w1z1+w2z2+w3z3+w4z4+w5z5

We see that the resulting polynomial regression is in the same class of linear models we'd considered above (i.e. the model is linear in w) and can be solved by the same techniques. By considering linear fits within a higher-dimensional space built with these basis functions, the model has the flexibility to fit a much broader range of data.

```
print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('==================================')
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:

_____________________________________
MAE: 32914.05664322205
MSE: 4735609419.52736
RMSE: 68815.76432422559
R2 Square 0.1871733495155612
==================================
Train set evaluation:

_____________________________________
MAE: 4.844429592291514e-10
MSE: 4.2738797888473384e-19
RMSE: 6.537491712306286e-10
R2 Square 1.0
```

7. Stochastic Gradient Descent

Gradient Descent is a very generic optimization algorithm capable of finding optimal solutions to a wide range of problems. The general idea of Gradient Sescent is to tweak parameters iteratively in order to minimize a cost function. Gradient Descent measures the local gradient of the error function with regards to the parameters vector, and it goes in the direction of descending gradient. Once the gradient is zero, you have reached a minimum.

In [1159]:

```
sgd_reg = SGDRegressor(n_iter_no_change=250, penalty=None, eta0=0.0001, max_iter=100000)
sgd_reg.fit(X_train, y_train)

test_pred = sgd_reg.predict(X_test)
train_pred = sgd_reg.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('==================================')
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:

_____________________________________
MAE: 22982.31926266606
MSE: 1517171772.5197318
RMSE: 38950.88923913974
R2 Square 0.7395905065604349
==================================
Train set evaluation:

_____________________________________
MAE: 14335.478307396384
MSE: 465669682.926806
RMSE: 21579.38096718268
R2 Square 0.9254508109838754
```

8. Random Forest Regressor¶

```
test_pred = rf_reg.predict(X_test)
train_pred = rf_reg.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)

print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:

_____________________________________
MAE: 16110.326798136646
MSE: 654194742.5022765
RMSE: 25577.23093890886
R2 Square 0.8877130957802414
Train set evaluation:

_____________________________________
MAE: 7274.464641333333
MSE: 160131841.20233497
RMSE: 12654.321048651127
R2 Square 0.97436444901832
```

9. Support Vector Machine

```
print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)

print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:
_____________________________________
MAE: 23022.693730527375
MSE: 2006035548.9099205
RMSE: 44788.78820541945
R2 Square 0.6556812415209905
Train set evaluation:
_____________________________________
MAE: 81.20204053326226
MSE: 2256900.975644557
RMSE: 1502.2985640825716
R2 Square 0.9996386920952927
```

From bagging/boosting techniques→

```
------------------------------------ DecisionTreeRegressor -------------------------------
DecisionTreeRegressor(random_state=97)
R2_score =  0.7577812933420962
Mean_Squared_Error = 1411190428.0590062
Mean_Absolute_Error = 25566.785714285714
Root_Mean_Squared_Error = 37565.814619930796
Cross_Val_Score =  0.6161743682116995


------------------------------------ AdaBoostRegressor -------------------------------
AdaBoostRegressor(random_state=97)
R2_score =  0.8268036876291739
Mean_Squared_Error = 1009059050.6621058
Mean_Absolute_Error = 22740.511610121084
Root_Mean_Squared_Error = 31765.68983450707
Cross_Val_Score =  0.7445859088244098
```

```
------------------------------------ RandomForestRegressor -------------------------------
RandomForestRegressor(random_state=97)
R2_score =  0.877013200353193
Mean_Squared_Error = 716533404.1862764
Mean_Absolute_Error = 16525.841925465837
Root_Mean_Squared_Error = 26768.141590074505
Cross_Val_Score =  0.8191990269962733


------------------------------------ GradientBoostingRegressor -------------------------------
GradientBoostingRegressor(random_state=97)
R2_score =  0.8920246067710987
Mean_Squared_Error = 629075447.9410919
Mean_Absolute_Error = 15769.524540409799
Root_Mean_Squared_Error = 25081.37651607447
Cross_Val_Score =  0.8530801858328034
```

Finally we got gradient boosting technique with r2 score of 89.9.

- Key Metrics for success in solving problem under consideration

  Randomised search cv was used over gradient boosting regressor which was in effective of the outliers in many columns.

```
print(cross_val_score(GBR,X,y,cv=10).mean())
```

```
0.8530801858328034
```

```
print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```

```
MAE: 15452.582876741406
MSE: 654696364.6316432
RMSE: 25587.035088724977
```

```
best_model = GradientBoostingRegressor(random_state=97)
```

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

```
GradientBoostingRegressor(random_state=97)
```

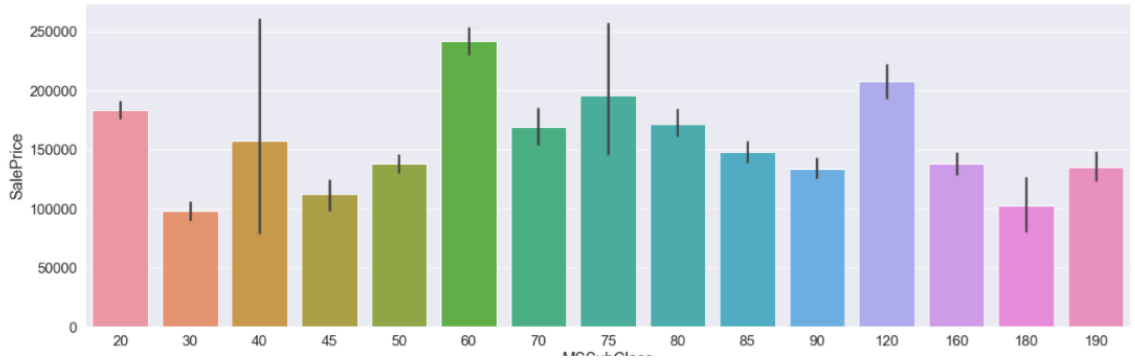
```
best_model.score(X_test,y_test)
```

```
0.8920246067710987
```

- Visualizations

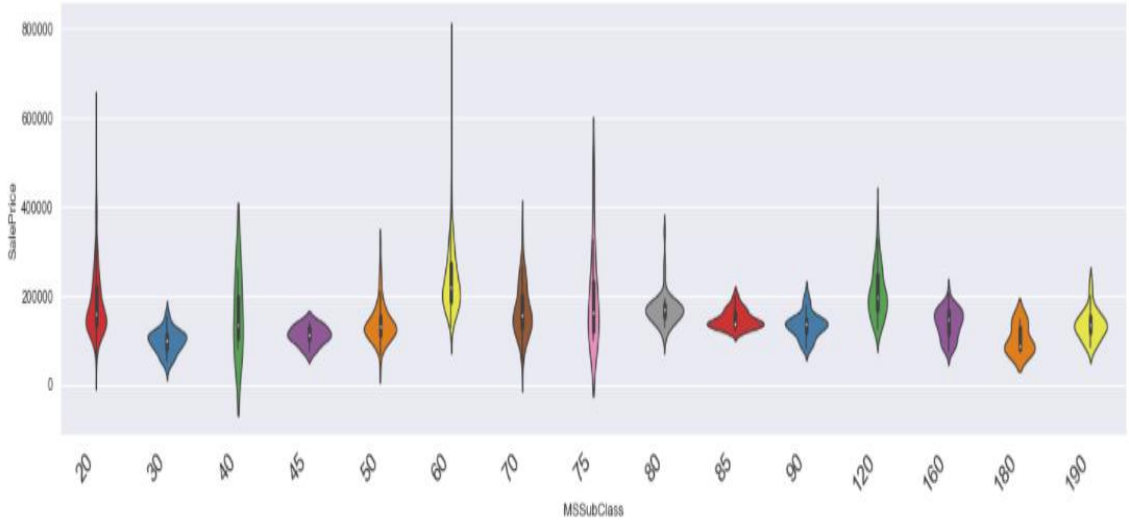
As there were 80 columns so lot of graphs been used so summarising the graphs I used for categorical data/Numerical data,

```
7]: sns.catplot(y='SalePrice',x='MSSubClass',data= df.sort_values('SalePrice',ascending=False),kind="bar",height=6, aspect=3)
plt.show()
```



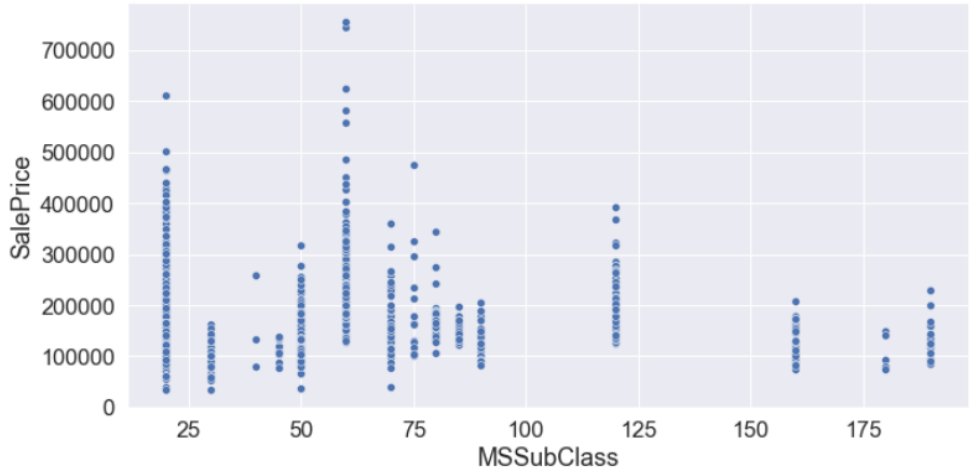
Graphs were made via matplotlib and seaborn, where sale price was checked with effect of inputs.

```
Text(13, 0, '180'),
Text(14, 0, '190')])
```

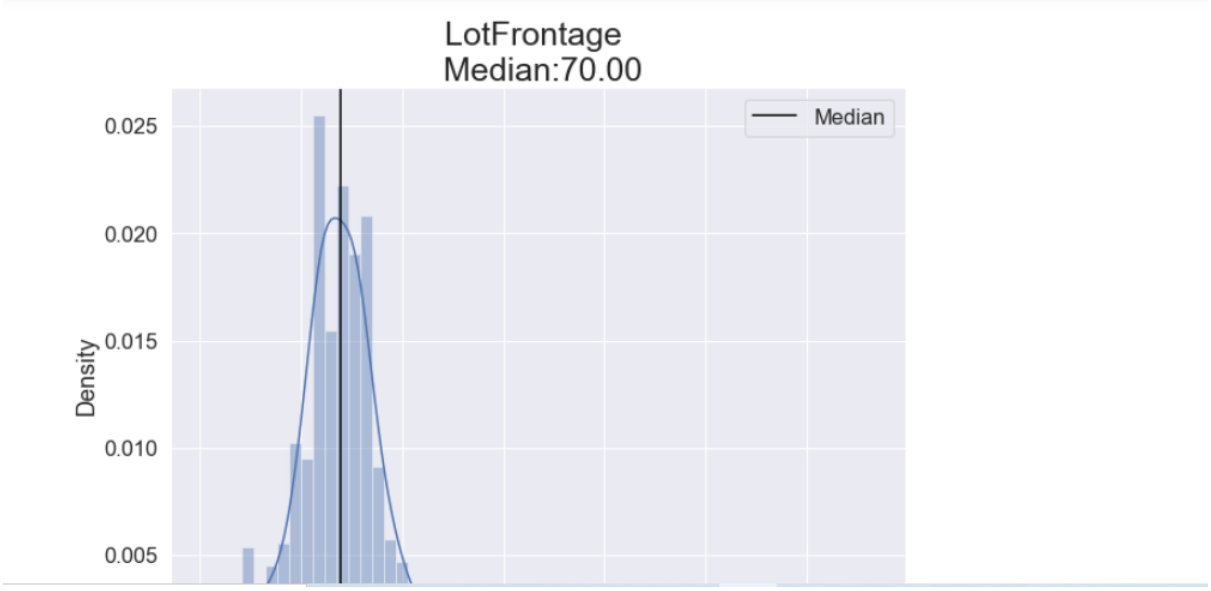


Again sale prices are checked with input features and effect was checked positively and negatively.

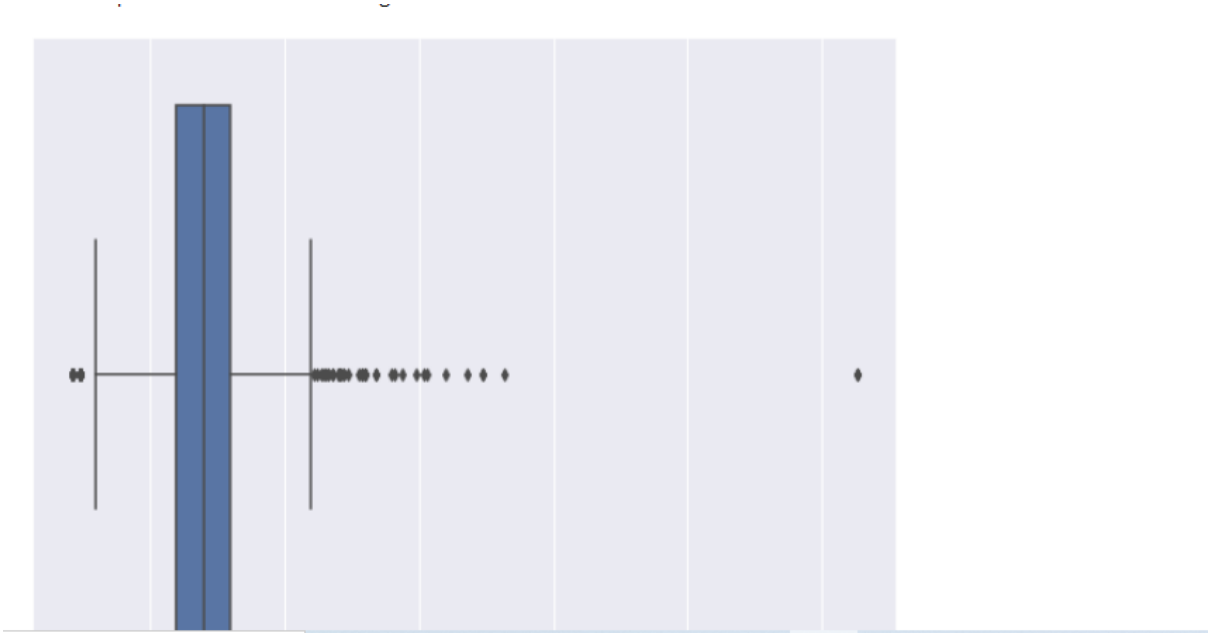
```
<seaborn.axisgrid.FacetGrid at 0x22d7764de80>
```



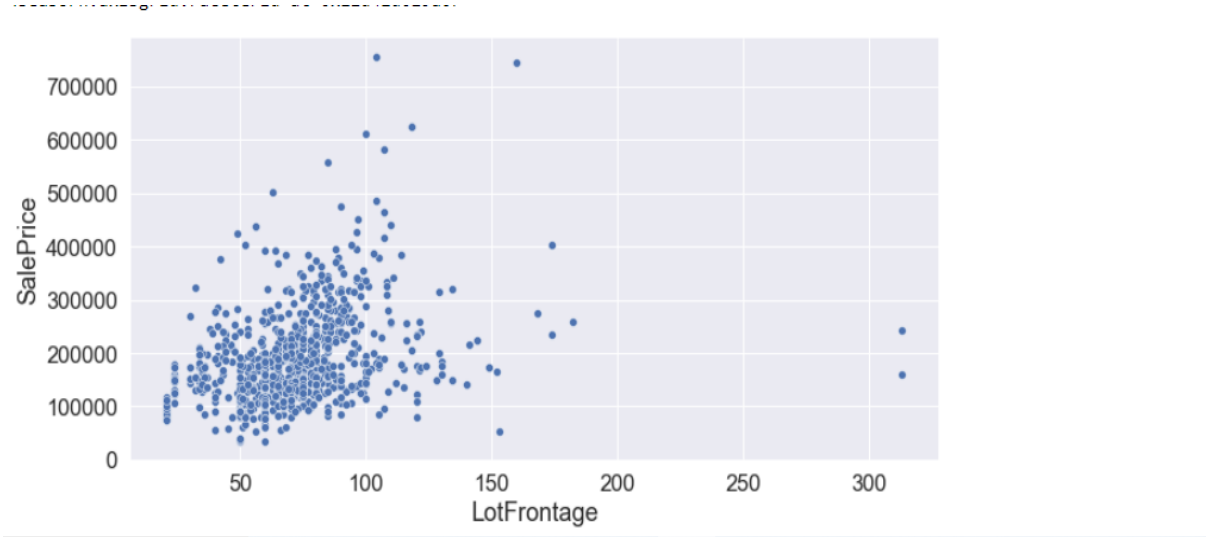
Count of maximum density was checked for each column taking sale prices into actions.



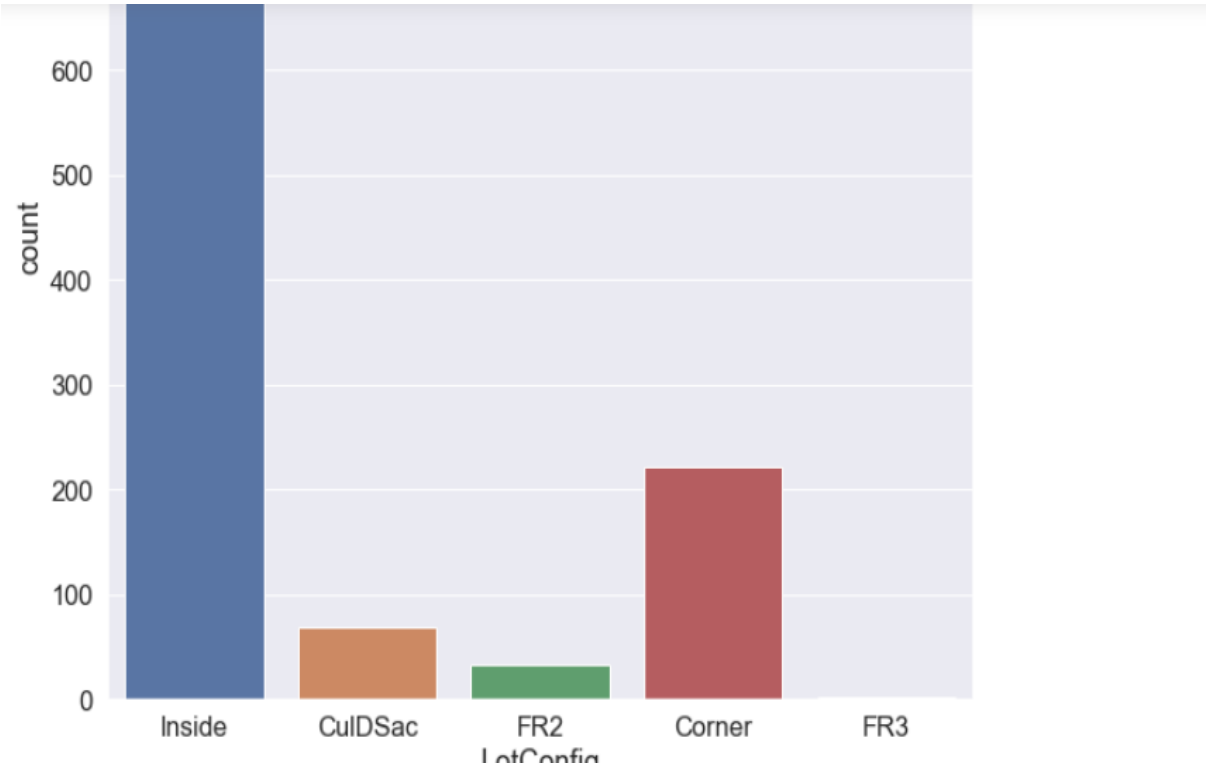
Median and skewness was checked for numerical data via these graphs.



Outliers were checked for maximum via box plots .



Now here from these graphs, I checked density where maximum houses lies in between for numerical features taking into.



Via count plots, count was checked for categorical features.

Interpretation of the Results

Given all data into account, sale prices are effected by taking on number of utilities, for example having furnished garage and 3 fire exit places will eventually increase the house sale prices .

CONCLUSION

- Key Findings and Conclusions of the Study

  From the whole problem statement, my key findings are as company can look to buy those houses with key variables as 1 kitchen , 4-5 bedrooms, basement bathroom, and the conditions variable of the houses which will cost them at median prices, company should not buy houses having numbers of bedrooms and bathroom because that just increase prices.

- Learning Outcomes of the Study in respect of Data Science

  From the power of visualization, data cleaning and various algorithms used, company can look to buy those houses with key variables as 1 kitchen , 4-5 bedrooms, basement bathroom, and the conditions variable of the houses which will cost them at median prices, company should not buy houses having numbers of bedrooms and bathroom because that just increase prices.

  And in this housing project, I calculated that GRADIENT BOOST REGRESSOR works awesome with r2 value of 86 and cross val score 89, linear regression is prone to outliers so won't work in this project.

- Limitations of this work and Scope for Future Work

  The limitations of the solution provided are that there are few houses that have basic features and excellent features as well but prices are either very low or very high with basic features , that can

imbalance the cost of the company. The steps/technique to be used is to reduce the redundant features and increase the size of data will help to predict more.

THANKYOU