HOUSING PRICE PREDICTION

Submitted by:

ABHISHEK JAIN

ACKNOWLEDGMENT

I have taken the use of internet and some websites to study and analyze this dataset and done research on it.

- Kaggle: Your Machine Learning and Data Science Community
- ResearchGate | Find and share research

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

  Basic concepts of machine learning and python is required for solving this problem.

- ## Review of Literature

  The two datasets are provided which are train.csv and test.csv. All the data is taken from these two datasets only.

- ## Motivation for the Problem Undertaken

  House price prediction can help determine the selling price of a house and can help to arrange the right time to purchase a house. There are various factors that influence the price of a house which include physical conditions, concept and location.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

  For doing analysis of this project , I have kept in mind many things about that how the problem can be visualized and represented in a simple way so I firstly I have imported all the necessary libraries required for data cleaning and viewing .

  Secondly I have done analysis of each column separately so that I can find out what exactly each column of this dataset is pointing towards and what is the correlation between them.

  Thirdly, I have checked that if any null value is present in the dataset or not as it can effect our model processing.

  Finally, after all the data processing and EDA steps I have use various models for Machine Learning and selected the best model afterwards.

  And in the last I saved the best predicted model to be available for further uses.

- ## Data Sources and their formats

  The data sources which were provided to me were in form of csv files so it was easy for me to import it and then do further calculations on it.

  The two csv files used in the project are train.csv and test.csv

- ## Data Pre-processing Done

  There were many steps used in Data cleaning / pre-processing of data in this project by me.

  1. Finding out null data in dataset
  2. Dropping the columns which have large amount of null data

3. Identified the columns which are having only categorical data or numeric data and applied encoding methods to them.
4. Tried to visualize each and specific column using matplotlib library and tried to find a correlation between all.
5. Finding out the outliers and removing them which can effect the performance of our model.
6. Removing the skewness after working on outliers.
7. In the last using standardization techniques to further clean data and then saving it for machine learning processes.

- ## Data Inputs- Logic- Output Relationships

  When I used the dataset train.csv and applied EDA methods on it and found out the correlation between each column , I saw that some columns are highly correlated to each other while some are not. Some columns have many null values because of which they need to be dropped while I have to changed some of the categorical columns into continuous columns so that EDA steps can be performed on it.

  In the beginning we had 1168 rows and 81 columns.

  After all the EDA steps –

  removing null values, outlier removal, standardization of data

  we get the final shape of dataset as 1168 rows and 195 columns.

  Then ,splitting the dataset into two parts i.e independent(input) and dependent(output) variables we get new dimension of data as (1168,194) as x and (1168,1) as y to be used for **machine learning**.

  So we have 1 column for output data and rest columns for input data to be used for ML.

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

  No assumptions were made by me in this project for model prediction.

- Hardware and Software Requirements and Tools Used

  Python 3.8 or above is used.

  Jupyter notebook for coding work.

  Libraries which are used in this project –

  - ✓ **Pandas** for importing data.
  - ✓ **Numpy** used in EDA steps.
  - ✓ **Matplotlib** and **Seaborn** for visualization of data.
  - ✓ **Sklearn** for standardization techniques, Machine learning.
  - ✓ **Warning** for ignoring warnings in our code.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Used **Regression model** as have both dependent and independent variables in our dataset.

- Testing of Identified Approaches (Algorithms)

  - Linear Regression
  - Ridge Regression
  - Lasso Regression
  - SVR Regressor
  - KNeighbors Regressor
  - RandomForest Regressor
  - AdaBoost Regressor
  - GradientBoosting Regressor

- Run and Evaluate selected models

All the screenshots are shown below in which we can see all the different types of models used for machine learning.

In [209]:
```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import SGDRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

In [210]:
```
LR_model= LinearRegression()
RD_model= Ridge()
LS_model= Lasso()
DT_model= DecisionTreeRegressor()
SV_model= SVR()
KNR_model= KNeighborsRegressor()
RFR_model= RandomForestRegressor()
XGB_model= XGBRegressor()
Elastic_model= ElasticNet()
SGH_model= SGDRegressor()
Bag_model=BaggingRegressor()
ADA_model=AdaBoostRegressor()
GB_model= GradientBoostingRegressor()

model=[LR_model,RD_model,LS_model,DT_model,SV_model,KNR_model,RFR_model,XGB_model,Elastic_model,SGH_model,Bag_model,ADA_model,GB_
```

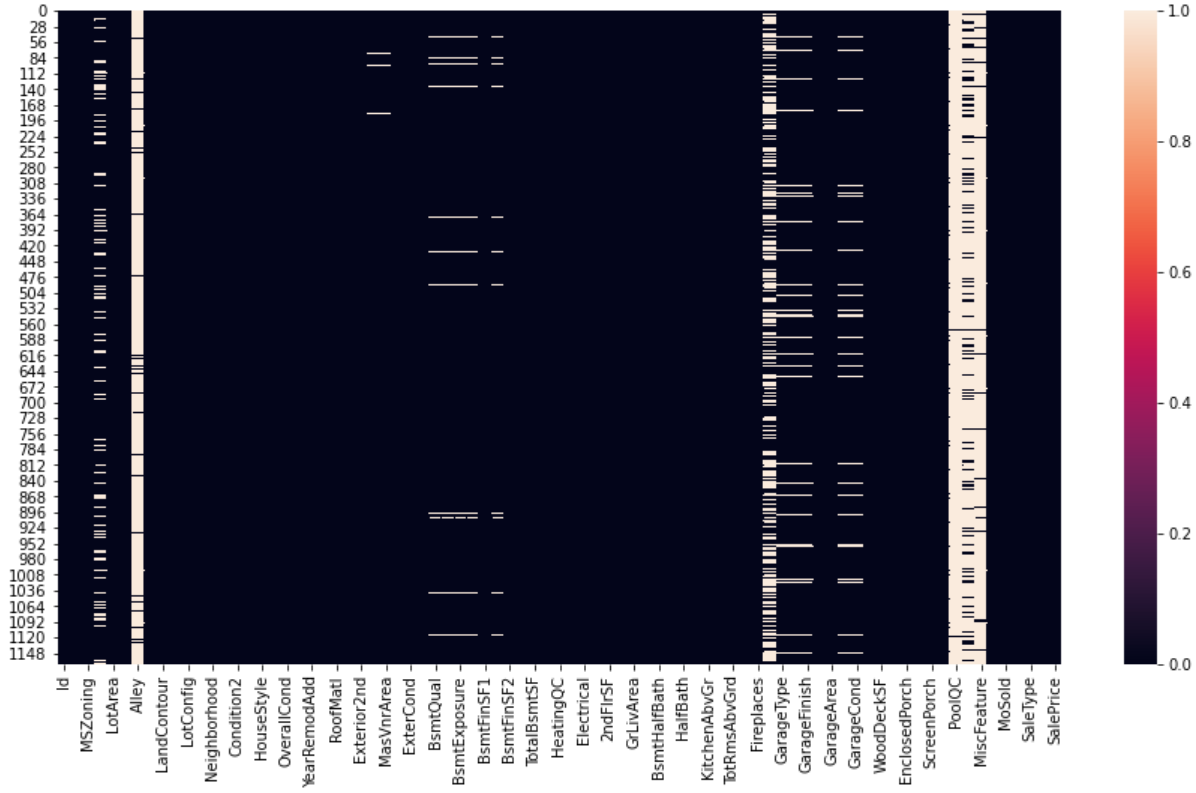
In [211]:
```
for m in model:
    m.fit(x_train,y_train)
    print('mean_absolute_error of ',m ,'model', mean_absolute_error(y_test,m.predict(x_test)))
    print('mean_square_error of',m,'model' , mean_squared_error(y_test,m.predict(x_test)))
    print('R2 Score of',m,'model', r2_score(y_test,m.predict(x_test) )*100)
    print('X' * 50, '\n\n')
```

```
mean_absolute_error of  LinearRegression() model 537326017308136.4
mean_square_error of LinearRegression() model 4.287995170695581e+31
R2 Score of LinearRegression() model -6.423966887766676e+23
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


mean_absolute_error of  Ridge() model 20349.730675895775
mean_square_error of Ridge() model 1049834275.6994458
R2 Score of Ridge() model 84.27213568050323
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


mean_absolute_error of  Lasso() model 21276.998037010835
mean_square_error of Lasso() model 1015079076.0952758
R2 Score of Lasso() model 84.79281315924835
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


mean_absolute_error of  DecisionTreeRegressor() model 27992.46153846154
mean_square_error of DecisionTreeRegressor() model 1583964078.2450142
R2 Score of DecisionTreeRegressor() model 76.27018598435772
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


mean_absolute_error of  SVR() model 58231.33310022471
mean_square_error of SVR() model 6951466788.437316
R2 Score of SVR() model -4.141890773368506
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

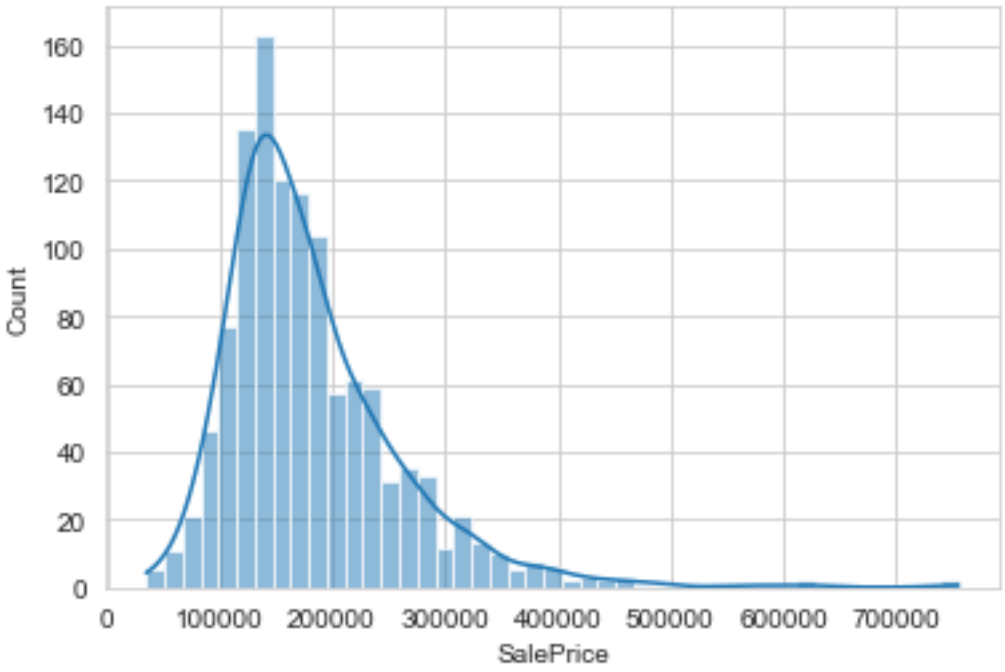
```
mean_absolute_error of  KNeighborsRegressor() model 28790.854131054133
mean_square_error of KNeighborsRegressor() model 2342335136.4532194
R2 Score of KNeighborsRegressor() model 64.9088145913489
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


mean_absolute_error of  RandomForestRegressor() model 17984.349914529914
mean_square_error of RandomForestRegressor() model 750417313.1235048
R2 Score of RandomForestRegressor() model 88.75778591250082
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


mean_absolute_error of  XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...) model 18395.702501780626
mean_square_error of XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...) model 731787368.3079476
R2 Score of XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
```

```
mean_absolute_error of  ElasticNet() model 31852.28142778712
mean_square_error of ElasticNet() model 2694860674.23156
R2 Score of ElasticNet() model 59.62752976795006
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


mean_absolute_error of  SGDRegressor() model 19606.400976130542
mean_square_error of SGDRegressor() model 1055829542.1760465
R2 Score of SGDRegressor() model 84.18231889714436
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


mean_absolute_error of  BaggingRegressor() model 19749.11025641026
mean_square_error of BaggingRegressor() model 872970921.2598577
R2 Score of BaggingRegressor() model 86.92177563426033
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


mean_absolute_error of  AdaBoostRegressor() model 23951.516450442108
mean_square_error of AdaBoostRegressor() model 1065711326.5002017
R2 Score of AdaBoostRegressor() model 84.03427709027794
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


mean_absolute_error of  GradientBoostingRegressor() model 16875.817742910294
mean_square_error of GradientBoostingRegressor() model 653768812.2707772
R2 Score of GradientBoostingRegressor() model 90.20570444905434
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

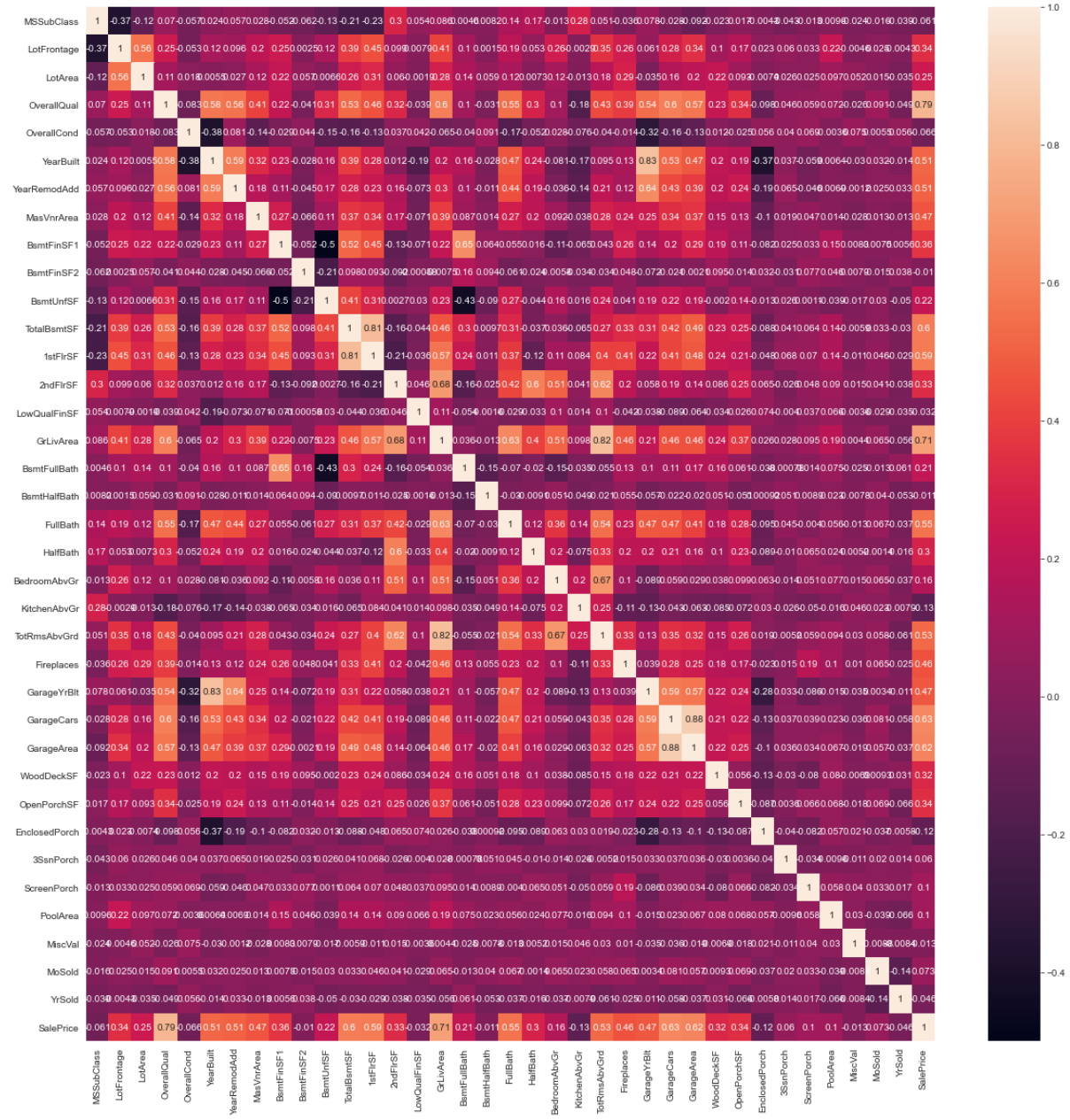
- Visualizations



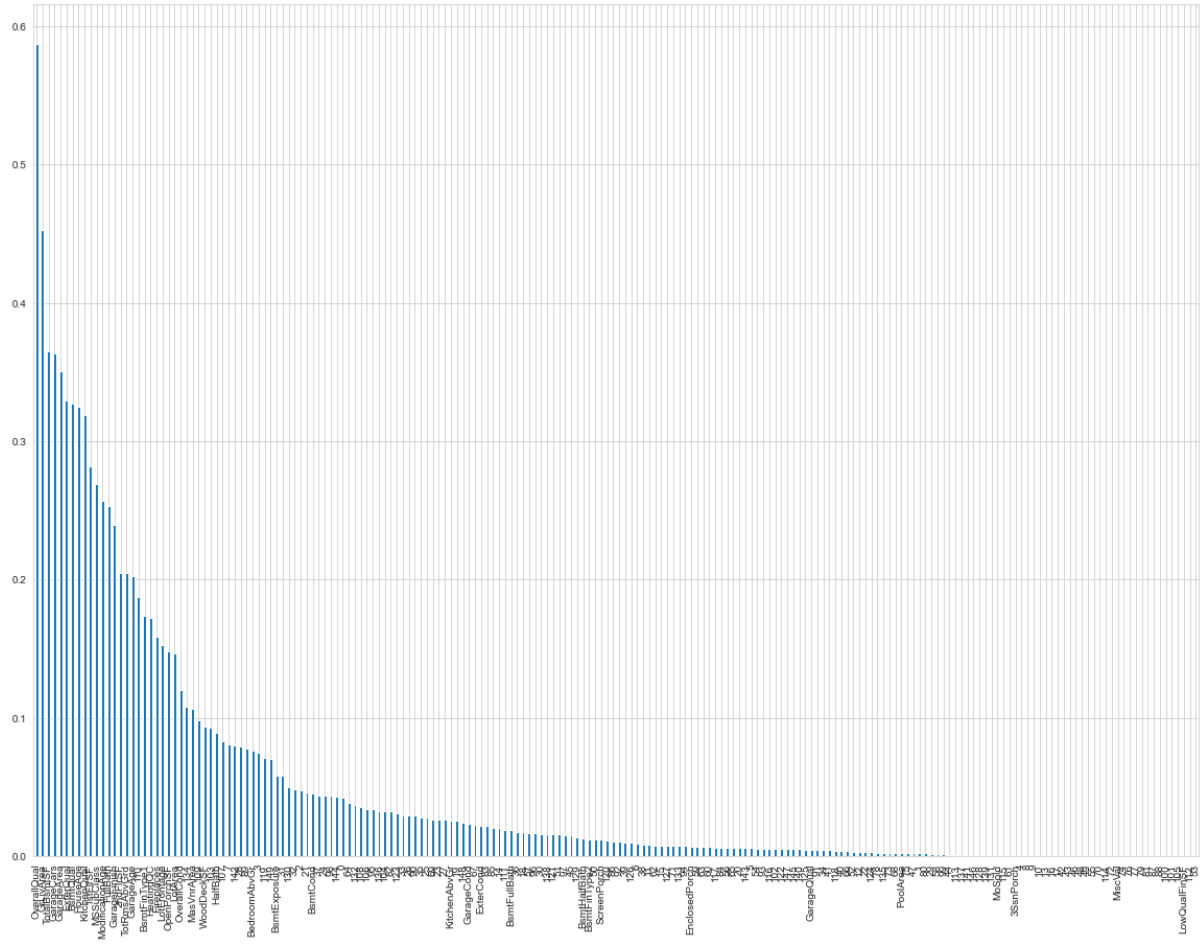
Null values Graph



Histoplot showing salesprice



Correlation graph



Feature Selection Graph showing all values

- Interpretation of the Results

After preprocessing data the final shape of dataset is

1168 rows

195 columns

Target variable is salesprice(y) – 1 column and 1168 rows

Other columns are input variables (x) – 194 columns and 1164 rows

Data modelling

Amongst all the models used Ridge model is the best . So we use it for hyperparameter tuning using GridSearchCV.

CONCLUSION

- Key Findings and Conclusions of the Study

Using train.csv data the final shape of our dataset is 1168 rows and columns size is 195

But, when we used test.csv the rows size is 293 and columns size is reduced to 147 after feature selection.

We have to use regression model for machine learning as the output variable has continuous data. We have done hypertuning of our model also to improve accuracy ,but sometimes it does not work i.e hypertuning can reduce the accuracy of our model.

Feature Selection

```
In [255]: test=final[selectedfeatures]

In [256]: test.shape

Out[256]: (292, 147)

In [257]: for feature in skew_feature:
              test[feature]= np.log(test[feature])

In [258]: test.shape

Out[258]: (292, 147)

In [259]: test.head()
```

Out[259]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	MasVnrArea	ExterQual	ExterCond	BsmtQual	BsmtCond	BsmtExposure	BsmtFinType1	Bsı
0	20	4.454347	9.557964	9	5	200.0	4	3	5	3	5	7	
1	120	4.094345	8.668024	8	5	0.0	4	3	4	3	4	7	
2	20	4.094345	9.379070	8	5	0.0	4	3	4	3	4	2	
3	70	4.317488	9.392662	7	7	0.0	3	3	3	3	2	4	
4	60	4.454347	9.588640	6	5	74.0	4	3	4	3	3	2	

- Limitations of this work and Scope for Future Work

    - The dataset provided to us is very small in size. l so our predictions can be wrong when working with a large dataset.
    - There were large amount of null values in our dataset which was necessary to filter. If no null values were present, then we may have a chance to improve the accuracy of our model and there should be no necessity to drop many columns then.