FLIP ROBO

PFA HOUSING PROJECT

Submitted by:

Shailza firoz

ACKNOWLEDGMENT

The sources I have used to accomplish this report are Medium, TowardsDataScience, StackOverflow.

INTRODUCTION

- Business Problem Framing

  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.

- Conceptual Background of the Domain Problem

  The company is looking at prospective properties to buy houses to enter the market. We 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.

- Review of Literature

  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.

  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.

  We 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.

- Motivation for the Problem Undertaken

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.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

We first look into the statistics of data shown in fig 1.

```
#checking out the statistical summary of our dataset
df.describe()
```

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000
mean	724.136130	56.767979	70.807363	10484.749144	6.104452	5.595890	1970.930651	1984.758562	101.696918	444.726027	46.647260
std	416.159877	41.940650	22.440317	8957.442311	1.390153	1.124343	30.145255	20.785185	182.218483	462.664785	163.520016
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	0.000000
25%	360.500000	20.000000	60.000000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	0.000000
50%	714.500000	50.000000	70.000000	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	0.000000
75%	1079.500000	70.000000	79.250000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	0.000000
max	1460.000000	190.000000	313.000000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.000000

```
#checking out the statistical summary of our dataset
df.describe()
```

BsmtUnfSF	TotalBsmtSF	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	BedroomAbvGr
1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000
569.721747	1061.095034	1169.860445	348.826199	6.380137	1525.066781	0.425514	0.055651	1.562500	0.388699	2.884418
449.375525	442.272249	391.161983	439.696370	50.892844	528.042957	0.521615	0.236699	0.551882	0.504929	0.817229
0.000000	0.000000	334.000000	0.000000	0.000000	334.000000	0.000000	0.000000	0.000000	0.000000	0.000000
216.000000	799.000000	892.000000	0.000000	0.000000	1143.250000	0.000000	0.000000	1.000000	0.000000	2.000000
474.000000	1005.500000	1096.500000	0.000000	0.000000	1468.500000	0.000000	0.000000	2.000000	0.000000	3.000000
816.000000	1291.500000	1392.000000	729.000000	0.000000	1795.000000	1.000000	0.000000	2.000000	1.000000	3.000000
2336.000000	6110.000000	4692.000000	2065.000000	572.000000	5642.000000	3.000000	2.000000	3.000000	2.000000	8.000000

```
#checking out the statistical summary of our dataset
df.describe()
```

KitchenAbvGr	TotRmsAbvGrd	Fireplaces	GarageCars	GarageArea	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	
1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	
1.045377	6.542808	0.617295	1.776541	476.860445	96.206336	46.559932	23.015411	3.639555	15.051370	3.448630	
0.216292	1.598484	0.650575	0.745554	214.466769	126.158988	66.381023	63.191089	29.088867	55.080816	44.896939	
0.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
1.000000	5.000000	0.000000	1.000000	338.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
1.000000	6.000000	1.000000	2.000000	480.000000	0.000000	24.000000	0.000000	0.000000	0.000000	0.000000	
1.000000	7.000000	1.000000	2.000000	576.000000	171.000000	70.000000	0.000000	0.000000	0.000000	0.000000	
3.000000	14.000000	3.000000	4.000000	1418.000000	857.000000	547.000000	552.000000	508.000000	480.000000	738.000000	1

```
#checking out the statistical summary of our dataset
df.describe()
```

Cars	GarageArea	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	MiscVal	MoSold	YrSold	SalePrice
0000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000
6541	476.860445	96.206336	46.559932	23.015411	3.639555	15.051370	3.448630	47.315068	6.344178	2007.804795	181477.005993
5554	214.466769	126.158988	66.381023	63.191089	29.088867	55.080816	44.896939	543.264432	2.686352	1.329738	79105.586863
0000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	2006.000000	34900.000000
0000	338.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	5.000000	2007.000000	130375.000000
0000	480.000000	0.000000	24.000000	0.000000	0.000000	0.000000	0.000000	0.000000	6.000000	2008.000000	163995.000000
0000	576.000000	171.000000	70.000000	0.000000	0.000000	0.000000	0.000000	0.000000	8.000000	2009.000000	215000.000000
0000	1418.000000	857.000000	547.000000	552.000000	508.000000	480.000000	738.000000	15500.000000	12.000000	2010.000000	755000.000000

Fig 1 Statistical description of data

From this statistical analysis we make some of the interpretations that,

1. Maximum standard deviation of 8957.44 is observed in the LotArea column.

2. Maximum SalePrice of a house observed is 755000 and minimum is 34900.

3. In the columns Id, MSSubclass, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfsF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath, HalfBath, TotRmsAbvGrd, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, Miscval, salePrice mean is considerably greater than median so the columns are positively skewed.

4. In the columns FullBath, BedroomAbvGr, Fireplaces, Garagecars, GarageArea, YrSold Median is greater than mean so the columns are negatively skewed.

5. In the columns Id, MSSubClass, LotFrontage, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtHalfBath, BedroomAbvGr, ToRmsAbvGrd, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, MiscVal, SalePrice there is considerable difference between the 75 percentile and maximum so outliers are present.

- Data Sources and their formats

The variable features of this problem statement are,

MSSubClass: Identifies the type of dwelling involved in the sale

MSZoning: Identifies the general zoning classification of the sale

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

Alley: Type of alley access to property

LotShape: General shape of property

LandContour: Flatness of the property

Utilities: Type of utilities available

LotConfig: Lot configuration

LandSlope: Slope of property

Neighborhood: Physical locations within Ames city limits

Condition1: Proximity to various conditions

Condition2: Proximity to various conditions (if more than one is present)

BldgType: Type of dwelling

HouseStyle: Style of dwelling

OverallQual: Rates the overall material and finish of the house

OverallCond: Rates the overall condition of the house

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

RoofMatl: Roof material

Exterior1st: Exterior covering on house

Exterior2nd: Exterior covering on house (if more than one material)

MasVnrType: Masonry veneer type

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

ExterCond: Evaluates the present condition of the material on the exterior

Foundation: Type of foundation

BsmtQual: Evaluates the height of the basement

BsmtCond: Evaluates the general condition of the basement

BsmtExposure: Refers to walkout or garden level walls

BsmtFinType1: Rating of basement finished area

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

HeatingQC: Heating quality and condition

CentralAir: Central air conditioning

Electrical: Electrical system

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality (Assume typical unless deductions are warranted)

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

GarageType: Garage location

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

GarageCond: Garage condition

PavedDrive: Paved driveway

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

Fence: Fence quality

MiscFeature: Miscellaneous feature not covered in other categories

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

SaleCondition: Condition of sale

The data types of features are shown in fig 4,

```
#checking data types of columns
df.dtypes
```

```
Id                 int64
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
                  ...
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
SalePrice          int64
Length: 81, dtype: object
```

Fig 4 Data types of features

- Data Preprocessing Done

We first did data cleaning. We first looked at the percentage of values missing in columns then we imputed missing values .

	Missing Values	% of Total Values
PoolQC	1161	99.4
MiscFeature	1124	96.2
Alley	1091	93.4
Fence	931	79.7
FireplaceQu	551	47.2
LotFrontage	214	18.3
GarageType	64	5.5
GarageYrBlt	64	5.5
GarageFinish	64	5.5
GarageQual	64	5.5
GarageCond	64	5.5
BsmtExposure	31	2.7
BsmtFinType2	31	2.7
BsmtCond	30	2.6
BsmtFinType1	30	2.6
BsmtQual	30	2.6
MasVnrArea	7	0.6
MasVnrType	7	0.6

Fig 5 Missing values

We then explored categorical variables as shown in fig 6.

Exploring categorical columns

```
#exploring categorical columns
for column in df.columns:
    if df[column].dtypes == object:
        print(str(column) + ' : ' + str(df[column].unique()))
        print(df[column].value_counts())
        print('*********************************************************************************************************')
        print('\n')
```

```
MSZoning : ['RL' 'RM' 'FV' 'RH' 'C (all)']
RL         928
RM         163
FV          52
RH          16
C (all)      9
Name: MSZoning, dtype: int64
*********************************************************************************************************


Street : ['Pave' 'Grvl']
Pave    1164
Grvl       4
Name: Street, dtype: int64
*********************************************************************************************************


Alley : [nan 'Grvl' 'Pave']
Grvl    41
Pave    36
```

Fig 6 Exploring categorical variables

We observed that there is only one unique value present in Utilities so will be dropping this column. Then we encoded all the categorical columns into numerical columns using dummy variables.

Encoding categorical columns

```
categorical_cols = ['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'LotConfig', 'LandSlope', 'Neighborhood', 'Conditic

df = pd.get_dummies(df, columns = categorical_cols, drop_first=True)
```

```
df
```

	Id	MSSubClass	LotFrontage	LotArea	Utilities	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnf
0	127	120	70.0	4928	AllPub	6	5	1976	1976	0.0	120	0	9
1	889	20	95.0	15865	AllPub	8	6	1970	1970	0.0	351	823	10
2	793	60	92.0	9920	AllPub	7	5	1996	1997	0.0	862	0	2
3	110	20	105.0	11751	AllPub	6	6	1977	1977	480.0	705	0	11
4	422	20	70.0	16635	AllPub	6	7	1977	2000	126.0	1246	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	289	20	70.0	9819	AllPub	5	5	1967	1967	31.0	450	0	4
1164	554	20	67.0	8777	AllPub	4	5	1949	2003	0.0	0	0	
1165	196	160	24.0	2280	AllPub	6	6	1976	1976	0.0	566	0	2
1166	31	70	50.0	8500	AllPub	4	4	1920	1950	0.0	0	0	6
1167	617	60	70.0	7861	AllPub	6	5	2002	2003	0.0	457	0	3

1168 rows × 259 columns

Fig 7 Encoding categorical columns

Then we checked the correlation with the help of heatmap as shown in fig 8,

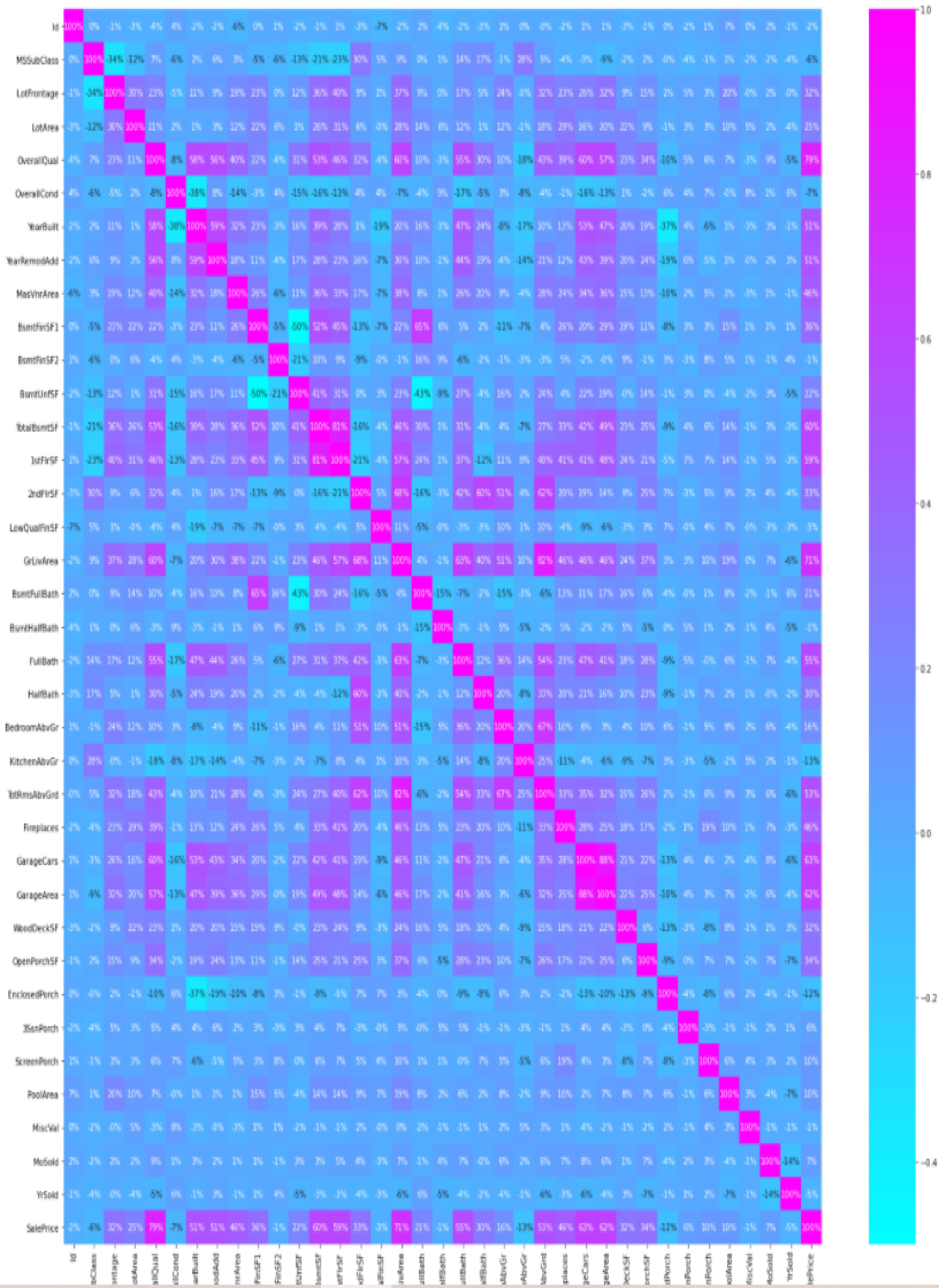


Fig 8 Heatmap of correlation

While checking the heatmap of correlation we observed that,

1. SalePrice is highly positively correlated with the columns OverallQual, YearBuilt, YearRemodAdd, TotalBsmtSF, 1stFlrSF, GrLivArea, FullBath, TotRmsAbvGrd, GarageCars, GarageArea.

2. SalePrice is negatively correlated with OverallCond, KitchenAbvGr, Encloseporch, YrSold.

3. We observe multicollinearity in between columns so we will be using Principal Component Analysis(PCA).

4. No correlation has been observed between the column Id and other columns so we will be dropping this column.

- Data Inputs- Logic- Output Relationships

Here we check the correlation between all our feature variables with target variable labels as shown in fig 10.

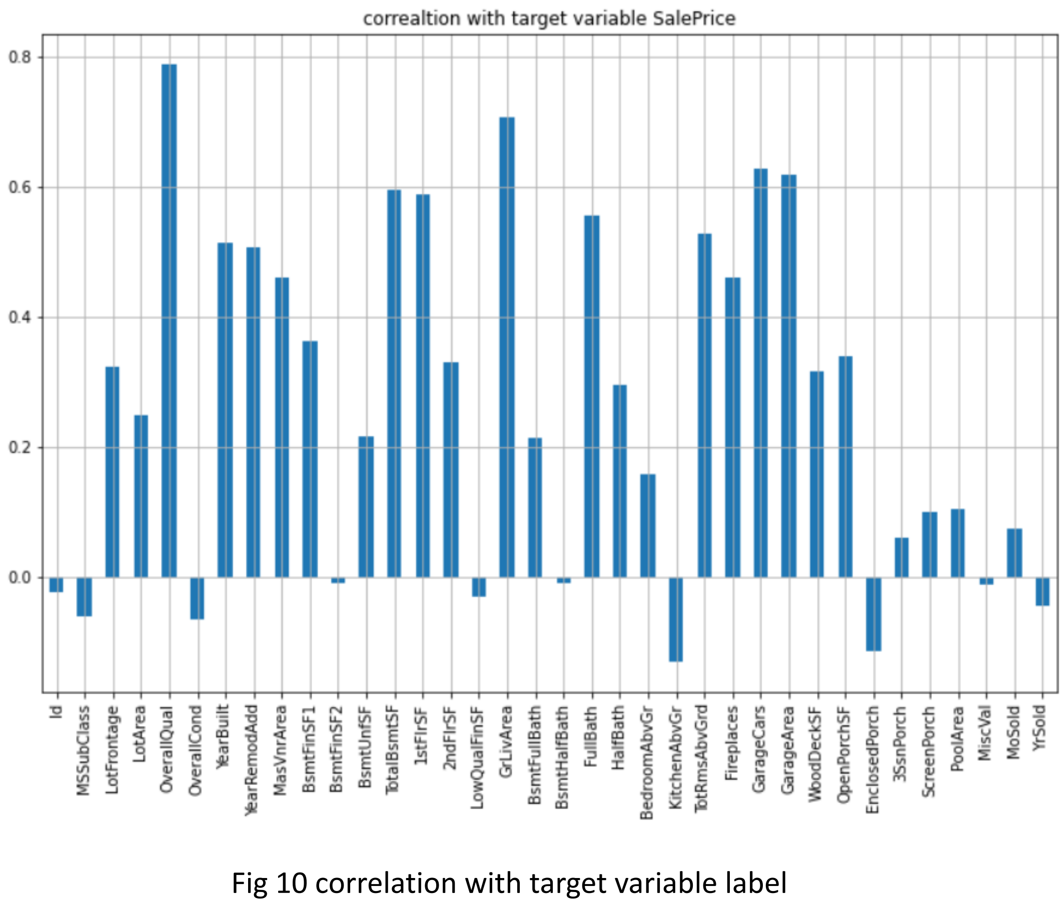


Fig 10 correlation with target variable label

1. The column OverallQual is most positively correlated with SalePrice.

2. The column KitchenAbvGrd is most negatively correlated with SalePrice.

- Set of assumptions related to the problem under consideration

By looking into the target variable label we assumed that it was a Regression type of problem.

We observed multicollinearity in between columns so we assumed that we will be using Principal Component Analysis (PCA).

We also observed that only one single unique value was present in Utilities column so we assumed that we will be dropping these columns.

- Hardware and Software Requirements and Tools Used

This project was done on a laptop with i5 processor with quad cores and eight threads with 8gb of ram and the latest GeForce GTX 1650 GPU on Anaconda, jupyter notebook.

The tools, libraries and packages we used for accomplishing this project are pandas, numpy, matplotlib, seaborn, scipy stats, sklearn.decomposition pca, sklearn standardscaler, GridSearchCV, joblib.

Through the pandas library we loaded our csv file 'Data file' into dataframe and performed data manipulation and analysis.

With the help of numpy we worked with arrays.

With the help of matplotlib and seaborn we did plot various graphs and figures and did data visualization.

With scipy stats we treated outliers through winsorization technique.

With sklearn.decomposition's pca package we reduced the number of feature variables from 256 to 100 by plotting scree plot with their Eigenvalues and chose the number of columns on the basis of their nodes.

With sklearn's standardscaler package we scaled all the feature variables onto a single scale.

Through GridSearchCV we were able to find the right parameters for hyperparameter tuning.

Through joblib we saved our model in csv format.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  We first converted all our categorical variables to numeric variables with the help of dummy variables to checkout and dropped the columns which we felt were unnecessary.

  We observed skewness in data so we tried to remove the skewness through treating outliers with winsorization technique.

  The data was improperly scaled so we scaled the feature variables on a single scale using sklearn's StandardScaler package.

  There were too many (256) feature variables in the data so we reduced it to 100 with the help of Principal Component Analysis(PCA) by plotting Eigenvalues and taking the number of nodes as our number of feature variables.

- ## Testing of Identified Approaches (Algorithms)

The algorithms we used for the training and testing are as follows:-

- Linear Regression
- Lasso
- Ridge
- Elastic Net
- SVR
- KNeighborsRegressor
- Decision Tree Regressor
- Random Forest Regressor
- AdaBoostRegressor
- Gradient Boosting Regressor

- ## Run and Evaluate selected models

The algorithms we used are shown in fig 11,

```
#Importing all model library
from sklearn.linear_model import LinearRegression,Lasso,Ridge,ElasticNet
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor

#Importing Boosting models
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor

#importing error metrics
from sklearn.model_selection import GridSearchCV,cross_val_score
```

Fig 11 Algorithms used

The results observed over different evaluation metrics are shown in fig 12,

```
score of LinearRegression() is: 0.8224023067822429
Error:
Mean absolute error: 21983.03594681287
Mean squared error: 1016181146.2848227
Root Mean Squared Error: 31877.59630657278
r2_score: 0.8451431350165133
********************************************************************************************


score of DecisionTreeRegressor() is: 1.0
Error:
Mean absolute error: 33349.05128205128
Mean squared error: 2904893311.905983
Root Mean Squared Error: 53897.062182515874
r2_score: 0.5573203920994878
********************************************************************************************


score of KNeighborsRegressor() is: 0.7910630500200235
Error:
Mean absolute error: 26847.836752136755
Mean squared error: 1671882262.554359
Root Mean Squared Error: 40888.65689349992
r2_score: 0.7452201836776653
********************************************************************************************


score of SVR() is: -0.04563664106634713
Error:
Mean absolute error: 58255.16893502842
Mean squared error: 6883309069.209987
Root Mean Squared Error: 82965.71020132345
r2_score: -0.04895437891886911
********************************************************************************************
```

```
Model: Lasso()
Score: [0.85207898 0.74649293 0.78624285 0.69359244 0.81790264 0.69908843
 0.79772316 0.69620109 0.60174926 0.83774268]
Mean score: 0.7528814469884274
Standard deviation: 0.07542969515426799
******************************************************************************************************
*********************


Model: Ridge()
Score: [0.85208142 0.74653129 0.78638215 0.69365996 0.8179455  0.69913248
 0.7978861  0.69675913 0.6026382  0.83781317]
Mean score: 0.7530829395860547
Standard deviation: 0.07522890383822077
******************************************************************************************************
*********************


Model: ElasticNet()
Score: [0.84352472 0.74457611 0.81451183 0.71347987 0.82780095 0.68914429
 0.84265308 0.78494133 0.79472646 0.85876943]
Mean score: 0.7914128054295206
Standard deviation: 0.05524013251954638
******************************************************************************************************
*********************


Model: RandomForestRegressor()
Score: [0.78642659 0.70841927 0.80088874 0.77416544 0.78435831 0.5748967
 0.79620818 0.80767199 0.85233838 0.80521004]
Mean score: 0.7690583639721766
Standard deviation: 0.07308999923937654
******************************************************************************************************
*********************
```

```
Model: AdaBoostRegressor()
Score: [0.67687313 0.63623881 0.67534235 0.68152578 0.61651457 0.54811785
 0.6737292  0.72426054 0.67222986 0.69305946]
Mean score: 0.6597891543469266
Standard deviation: 0.046385992712606065
**********************************************************************************************************
********************


Model: GradientBoostingRegressor()
Score: [0.79435779 0.7301687  0.81540851 0.75602916 0.78039711 0.67290297
 0.80468041 0.78554125 0.84626604 0.77022326]
Mean score: 0.7755975195869904
Standard deviation: 0.045738804971296516
**********************************************************************************************************
********************
```

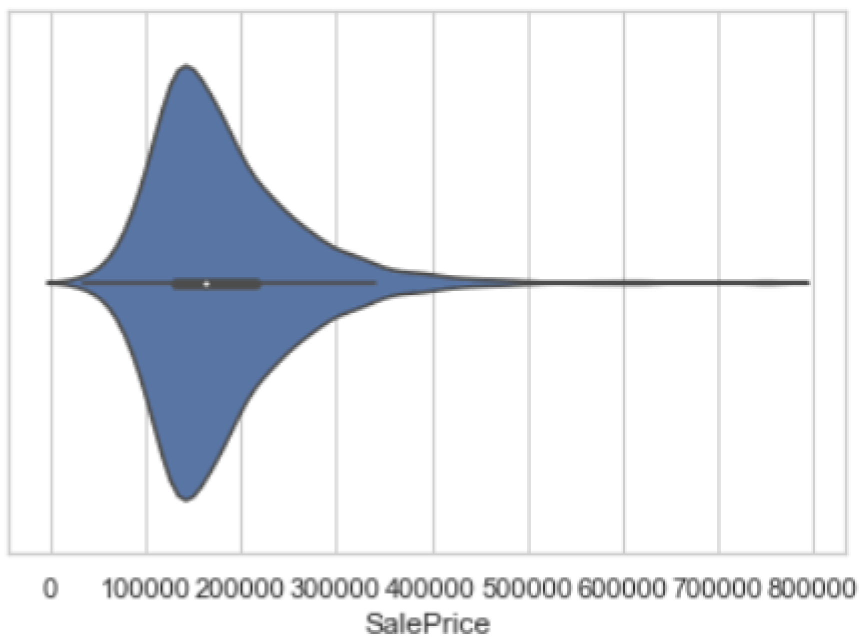
Fig 12 Results observed

- Key Metrics for success in solving problem under consideration

  We used the metric Root Mean Squared Error by selecting the Ridge Regressor model which was giving us the best(minimum) RMSE score.

- Visualizations

  ViolinPlot of SalePrice :-



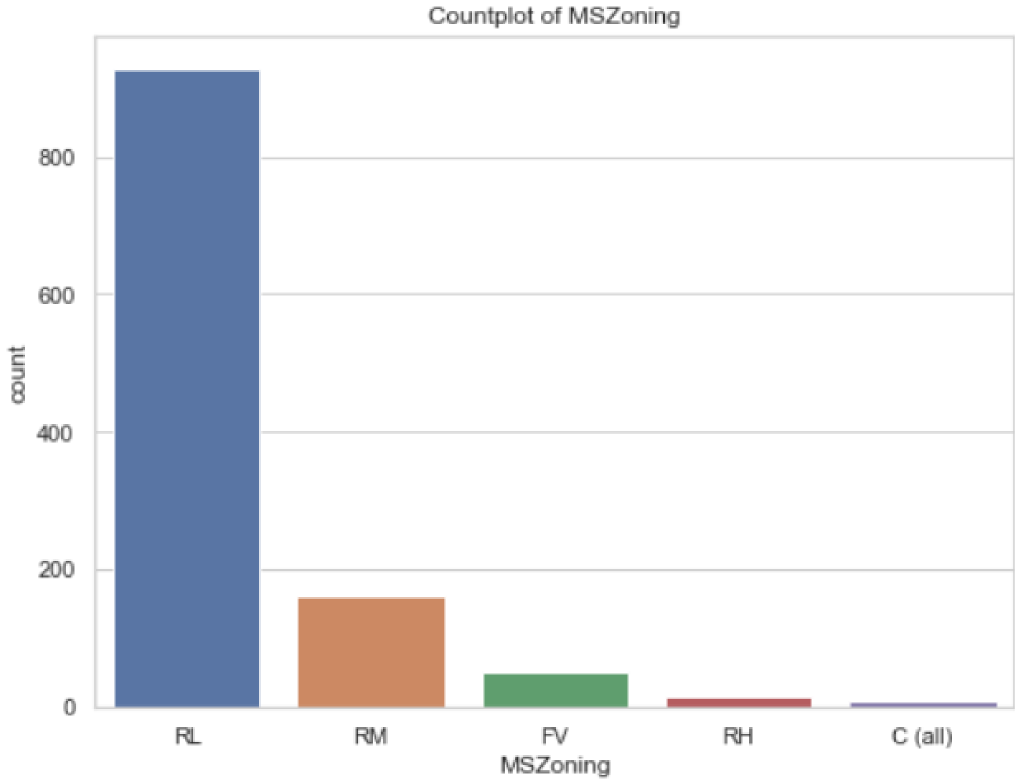

```
140000    18
135000    16
155000    12
139000    11
160000    11
          ..
126175     1
204000     1
186000     1
369900     1
105500     1
Name: SalePrice, Length: 581, dtype: int64
```

Fig 13 VioilinPlot of SalePrice

Observation:

Maximum number of SalePrice lies between 140000 and 230000.

Countplot of MSZoning:-



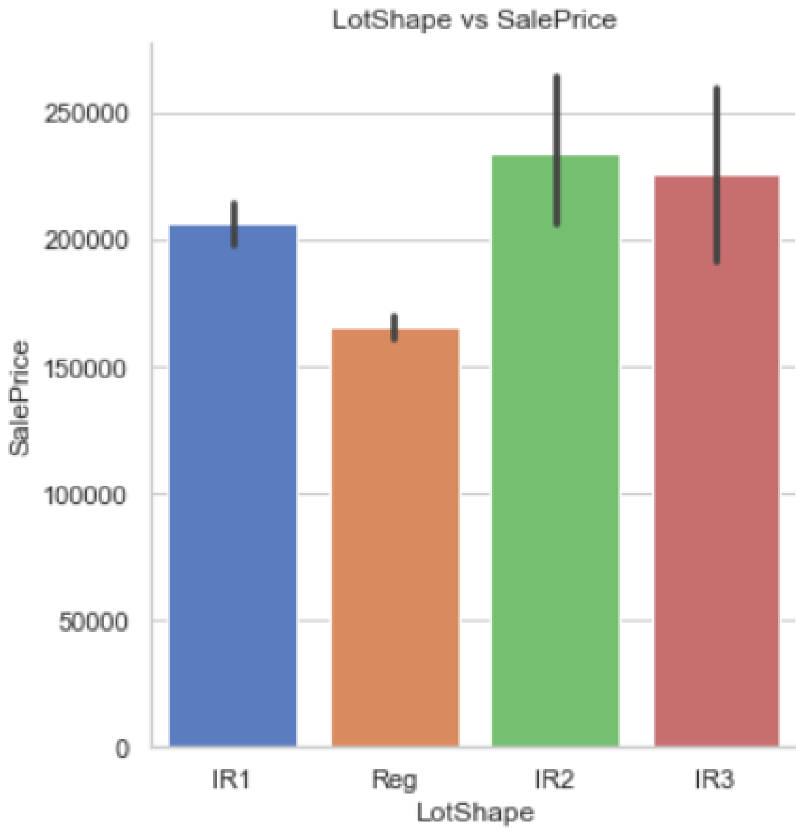

```
RL         928
RM         163
FV          52
RH          16
C (all)      9
Name: MSZoning, dtype: int64
```

Fig 14 Countplot of MSZoning

Observation:

Maximum, 928 number of MSZoning are RL.

Checking column LotShape with SalePrice:-



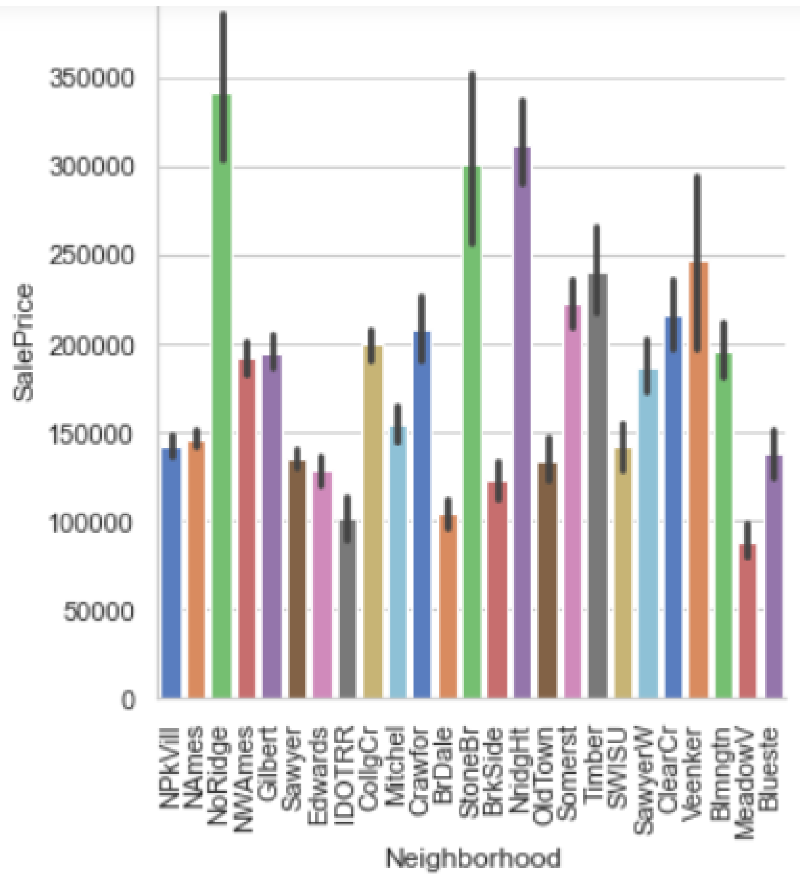

```
SalePrice  LotShape
34900      Reg         1
35311      Reg         1
37900      Reg         1
39300      Reg         1
40000      Reg         1
                      ..
582933     Reg         1
611657     IR1         1
625000     IR1         1
745000     IR1         1
755000     IR1         1
Name: LotShape, Length: 733, dtype: int64
```

Fig 15 LotShape vs SalePrice

Observation:

SalePrice is maximum with IR2 LotShape.

Checking column Neighborhood with SalePrice:-



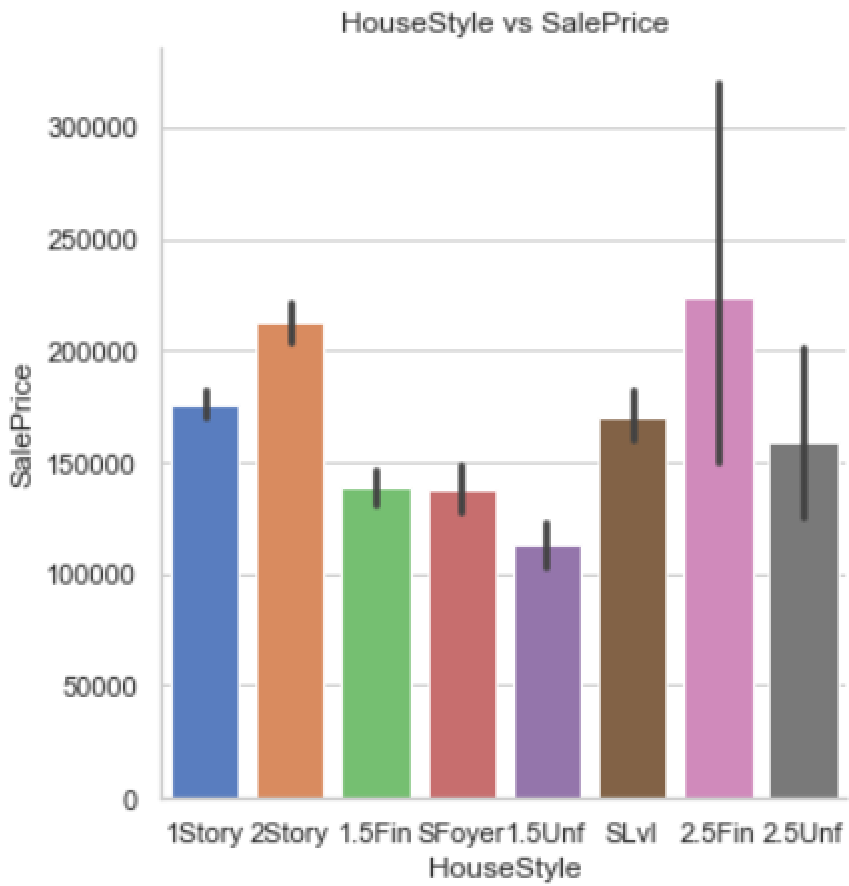

```
SalePrice  Neighborhood
34900      IDOTRR          1
35311      IDOTRR          1
37900      OldTown         1
39300      BrkSide         1
40000      IDOTRR          1
                          ..
582933     NridgHt         1
611657     NridgHt         1
625000     NoRidge         1
745000     NoRidge         1
755000     NoRidge         1
Name: Neighborhood, Length: 1013, dtype: int64
```

Fig 16 Neighborhood vs SalePrice

Observation:

SalePrice is maximum with NoRidge Neighborhood.

Checking the column HouseStyle with SalePrice:-




```
SalePrice  HouseStyle
34900      1Story        1
35311      1Story        1
37900      1.5Fin        1
39300      1Story        1
40000      2Story        1
                        ..
582933     2Story        1
611657     1Story        1
625000     2Story        1
745000     2Story        1
755000     2Story        1
Name: HouseStyle, Length: 840, dtype: int64
```

Fig 17 HouseStyle vs SalePrice

Observation:

SalePrice is maximum with 2.5Fin HouseStyle.

Checking KitchenQual and CentralAir with SalePrice:-

```
#checking GarageType and GarageCond with respect to SalePrice
sns.factorplot(x='KitchenQual',y='SalePrice',hue='CentralAir',data=df,kind='violin',size=5,palette='muted',aspect=2)
plt.title('SalePrice according to KitchenQual and CentralAir')
plt.xticks()
plt.ylabel('SalePrice')
plt.show()
```

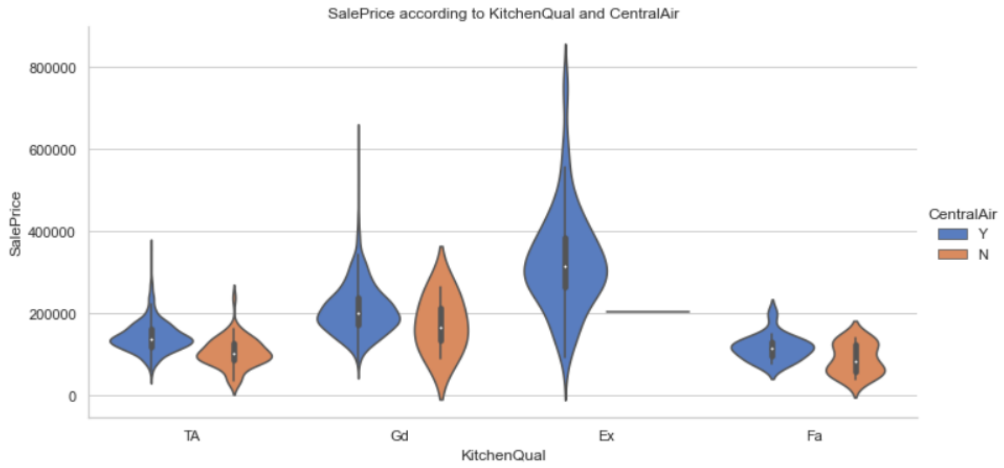


Fig 18 ViolinPlot between KitchenQual and CentralAir with respect to SalePrice

Observation:

SalePrice is maximum with Ex kitchenQual and CentralAir.

Checking SalePrice, OverallQual, YearBuilt, GrLivArea, GarageCars:-

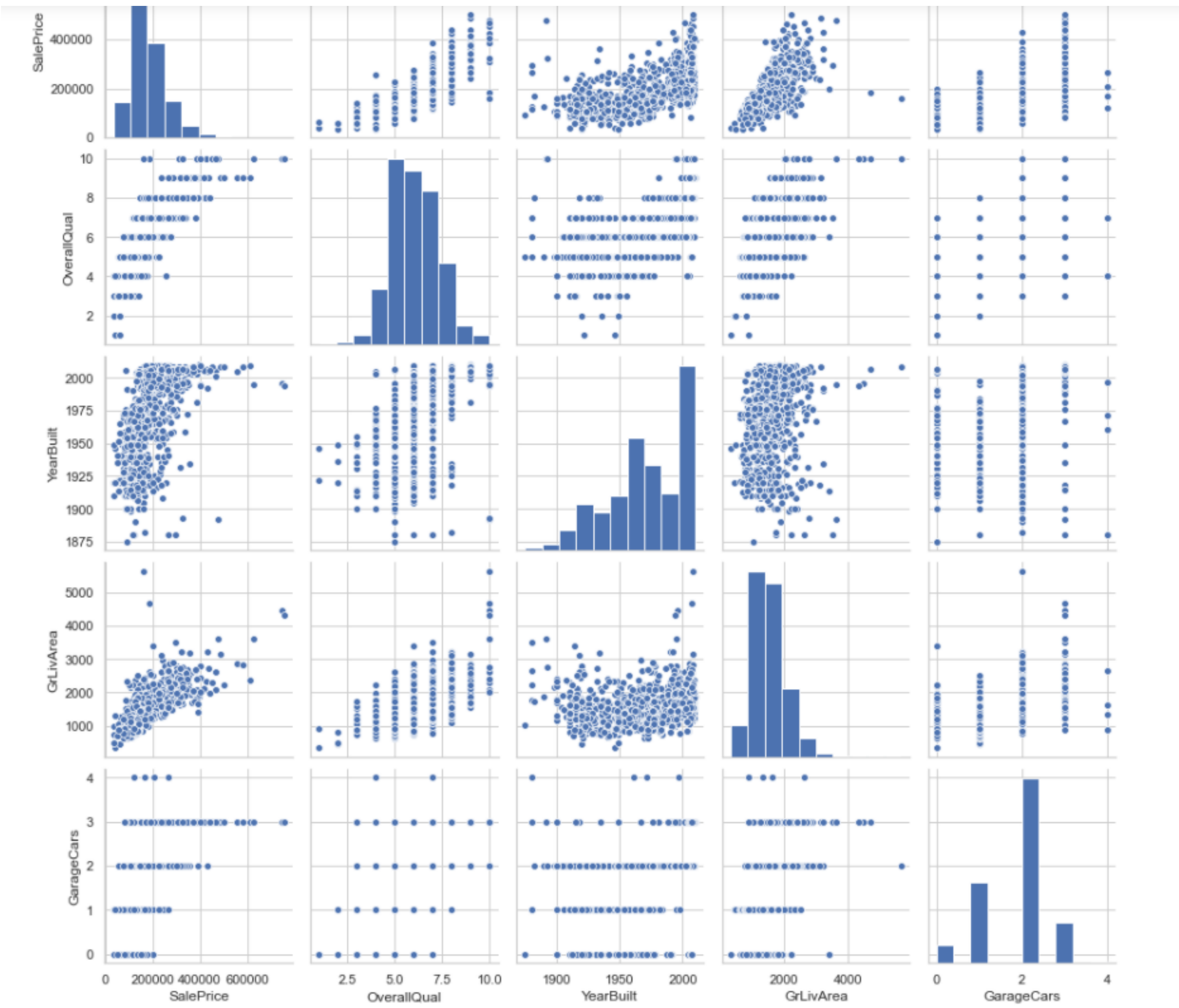


Fig 19 pairplot

Observation:

SalePrice is highly positively correlated with GrLivArea and OverallQual.

- Interpretation of the Results

From the visualization we interpreted that the target variable SalePrice was highly positively correlated with the columns GrLivArea, YearBuilt, OverallQual, GarageCars, GarageArea.

From the preprocessing we interpreted that data was improper scaled.

From the modeling we interpreted that after hyperparameter tuning Ridge Regressor works best with respect to our model with minimum RMSE of 31806 as shown in fig 20

```
RG=Ridge(alpha=25)
RG.fit(x_train,y_train)
print('Score:',RG.score(x_train,y_train))
y_pred=RG.predict(x_test)
print('\n')
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('\n')
print("r2_score:",r2_score(y_test,y_pred))
print('\n')
```

```
Score: 0.8223601918721507


Mean absolute error: 21831.129709253644
Mean squared error: 1011636781.6056582
Root Mean Squared error: 31806.238092639283


r2_score: 0.8458356553118661
```

Fig 20 score of Ridge after Hyperparameter tuning.

CONCLUSION

- Key Findings and Conclusions of the Study

  In this project we have tried to show how the house prices vary and what are the factors related to the changing of house prices.The best(minimum) RMSE score was achieved using the best parameters of Ridge Regression through GridSearchCV though the Lasso Regression model performed well too.

- Learning Outcomes of the Study in respect of Data Science

  This project has demonstrated the importance of sampling effectively, modelling and predicting data.

  Through different powerful tools of visualization we were able to analyse and interpret different hidden insights about the data.

  Through data cleaning we were able to remove unnecessary columns and outliers from our dataset due to which our model would have suffered from overfitting or underfitting.

  The few challenges while working on this project where:-

  - Improper scaling
  - Too many features
  - Missing values
  - Skewed data due to outliers

  The data was improperly scaled so we scaled it to a single scale using sklearns's package StandardScaler.

  There were too many(256) features present in the data so we applied Principal Component Analysis(PCA) and found out the Eigenvalues and on the basis of number of nodes we were able to reduce our features upto 90 columns.

There were a lot of missing values present in different columns which we imputed on the basis of our understanding.

The columns were skewed due to the presence of outliers which we handled through winsorization technique.

- ## Limitations of this work and Scope for Future Work

While we couldn’t reach out goal of minimum RMSE in house price prediction without letting the model to overfit, we did end up creating a system that can with enough time and data get very close to that goal. As with any project there is room for improvement here. The very nature of this project allows for multiple algorithms to be integrated together as modules and their results can be combined to increase the accuracy of the final result. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and vesatility to the project.