PFA HOUSING PROJECT

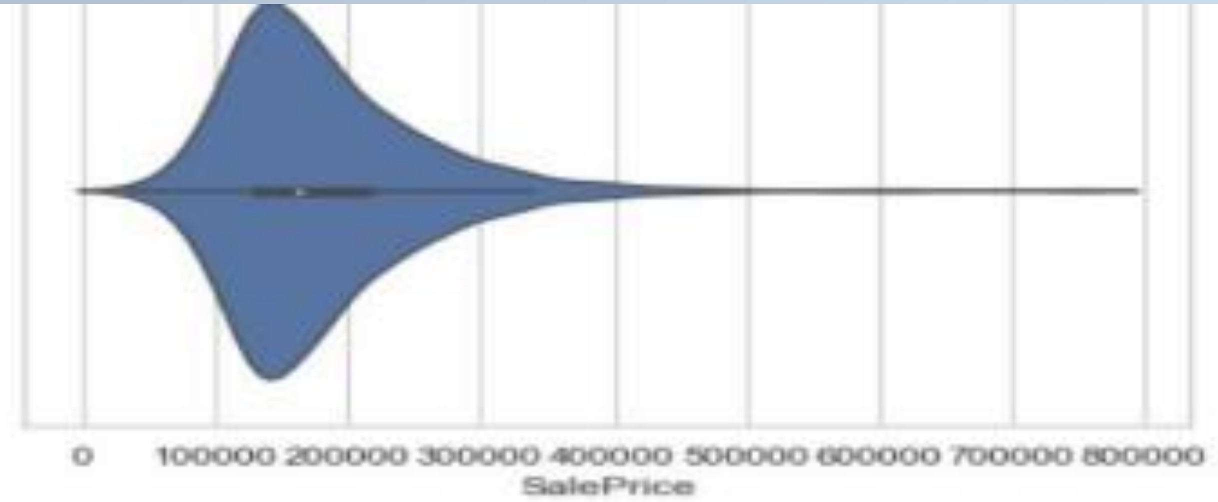
Submitted by : ANUM YUSUF

Contents :

INTRODUCTION

Problem statement and understanding

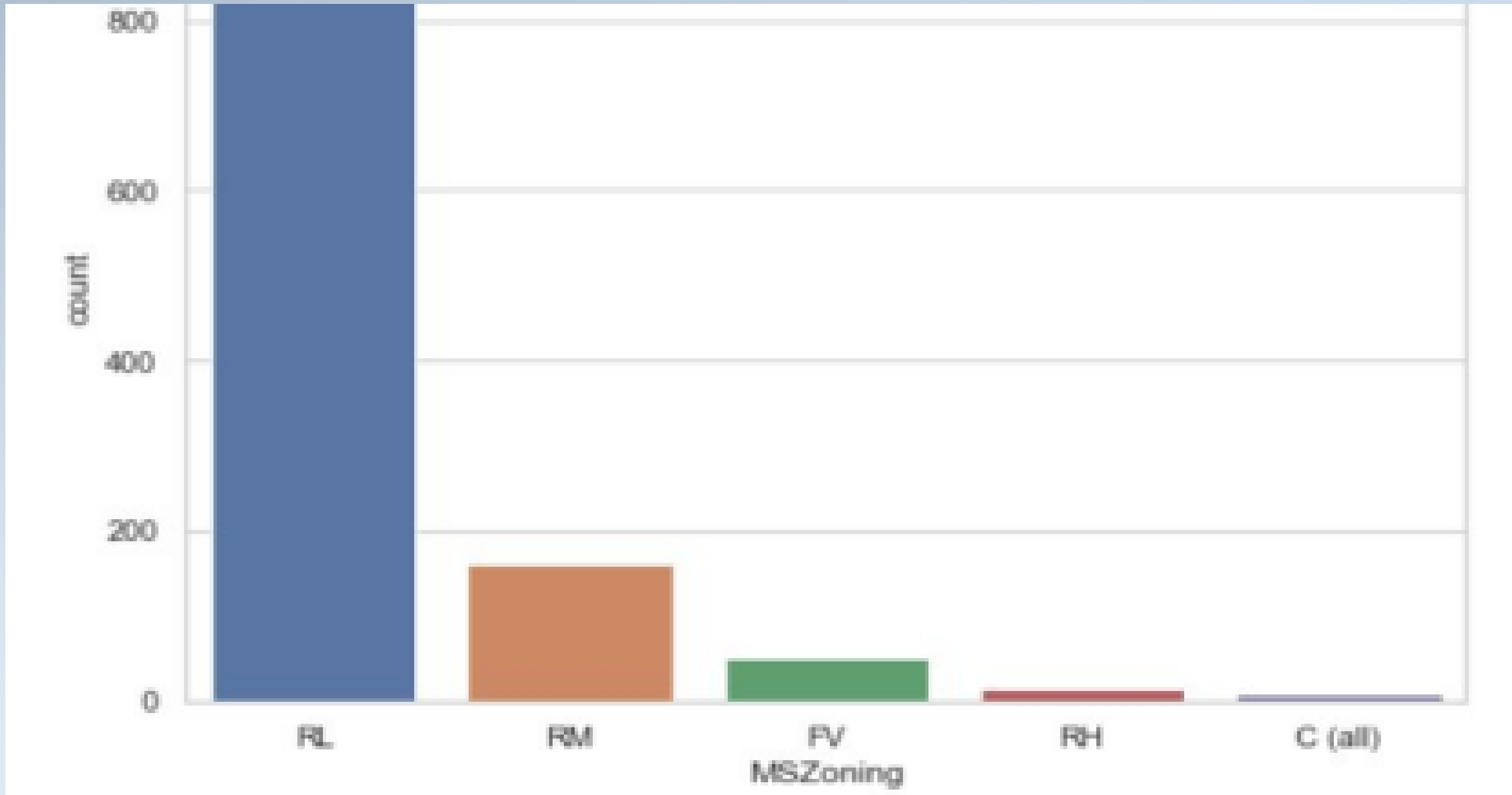
- A US-based housing company named Surprise Housing has decided to enter the Australian market.
- The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia.
- 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.



140000	18
135000	16
155000	12
139000	11
160000	11
	..
126175	1
204000	1
186000	1
369900	1
105500	1

EDA steps and Visualization

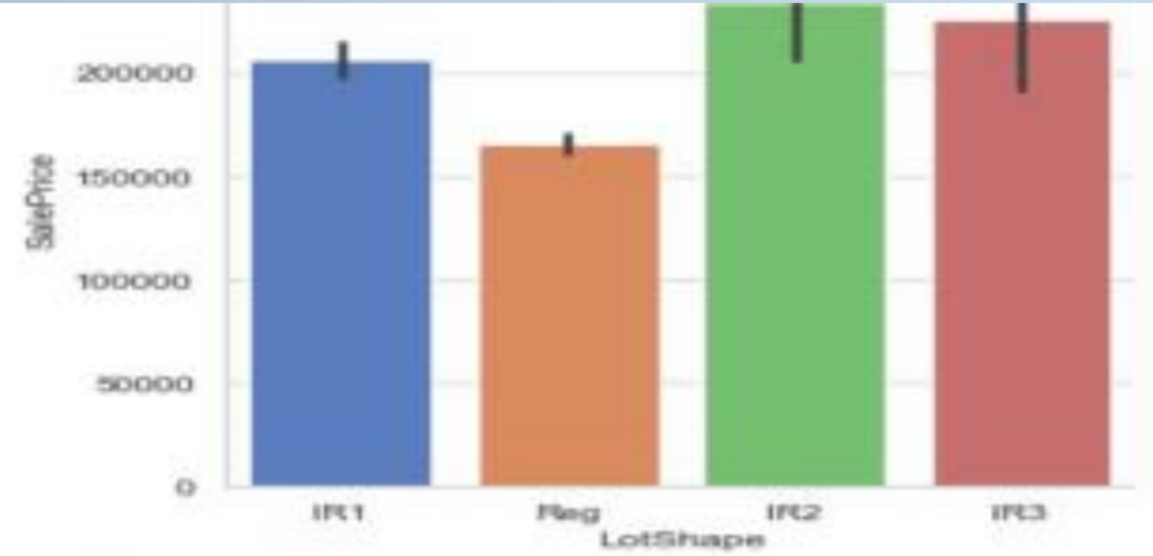
Observation: Maximum number of Sale Price lies between 140000 and 230000.



RL 928
RM 163

Observation:

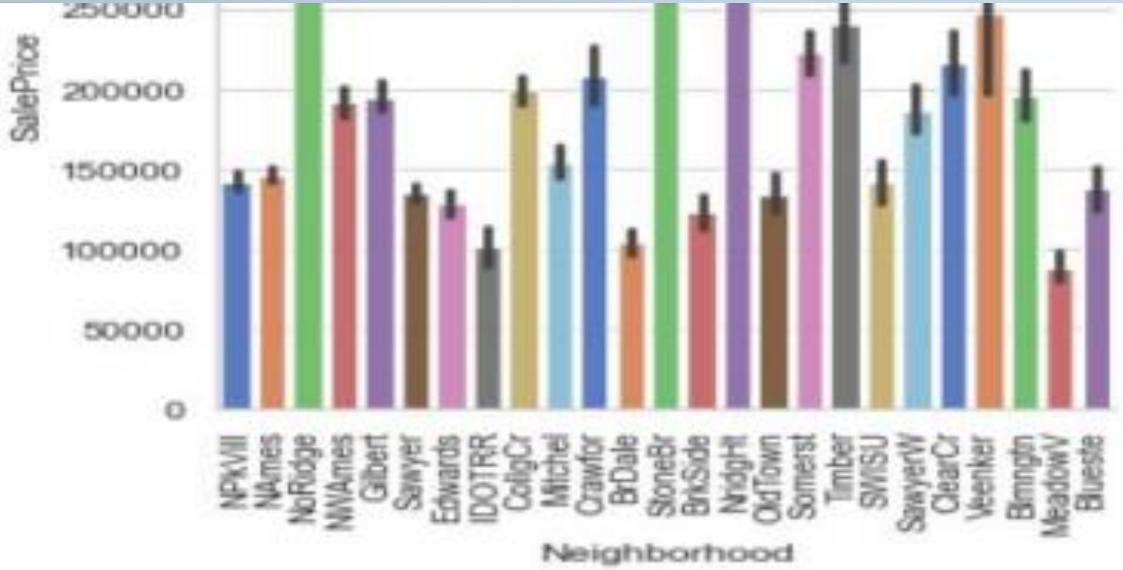
Maximum, 928 number of MSZoning are RL.



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

Observation:

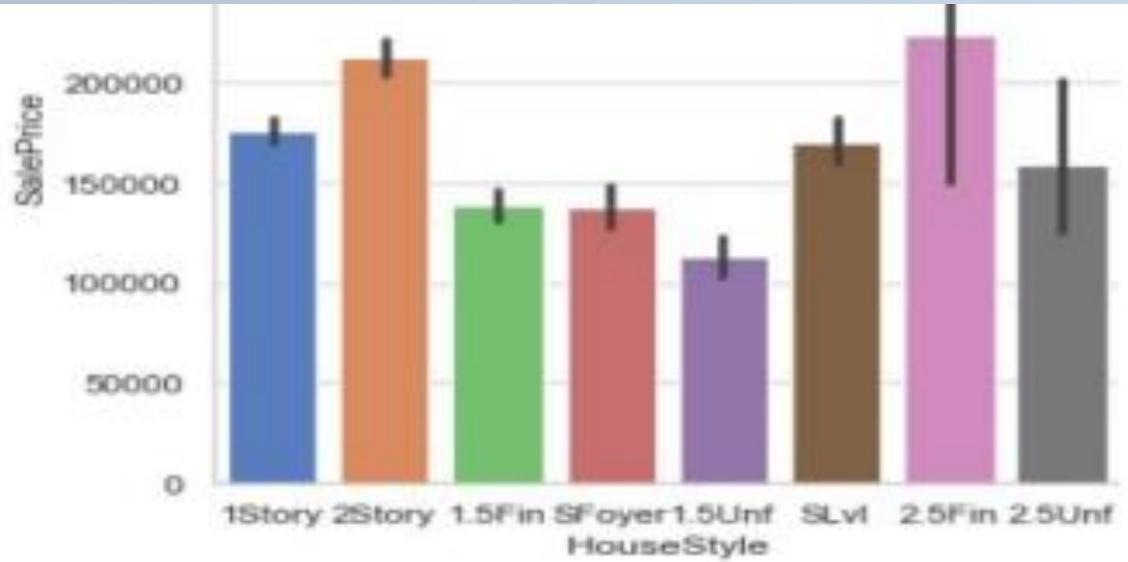
Sale Price is maximum with IR2 Lot Shape.



SalePrice	Neighborhood	
34900	IDOTRR	1
35311	IDOTRR	1
37900	OldTown	1
39300	BrkSide	1
40000	IDOTRR	1
		..
582933	NridgHt	1
611657	NridgHt	1

Observation:

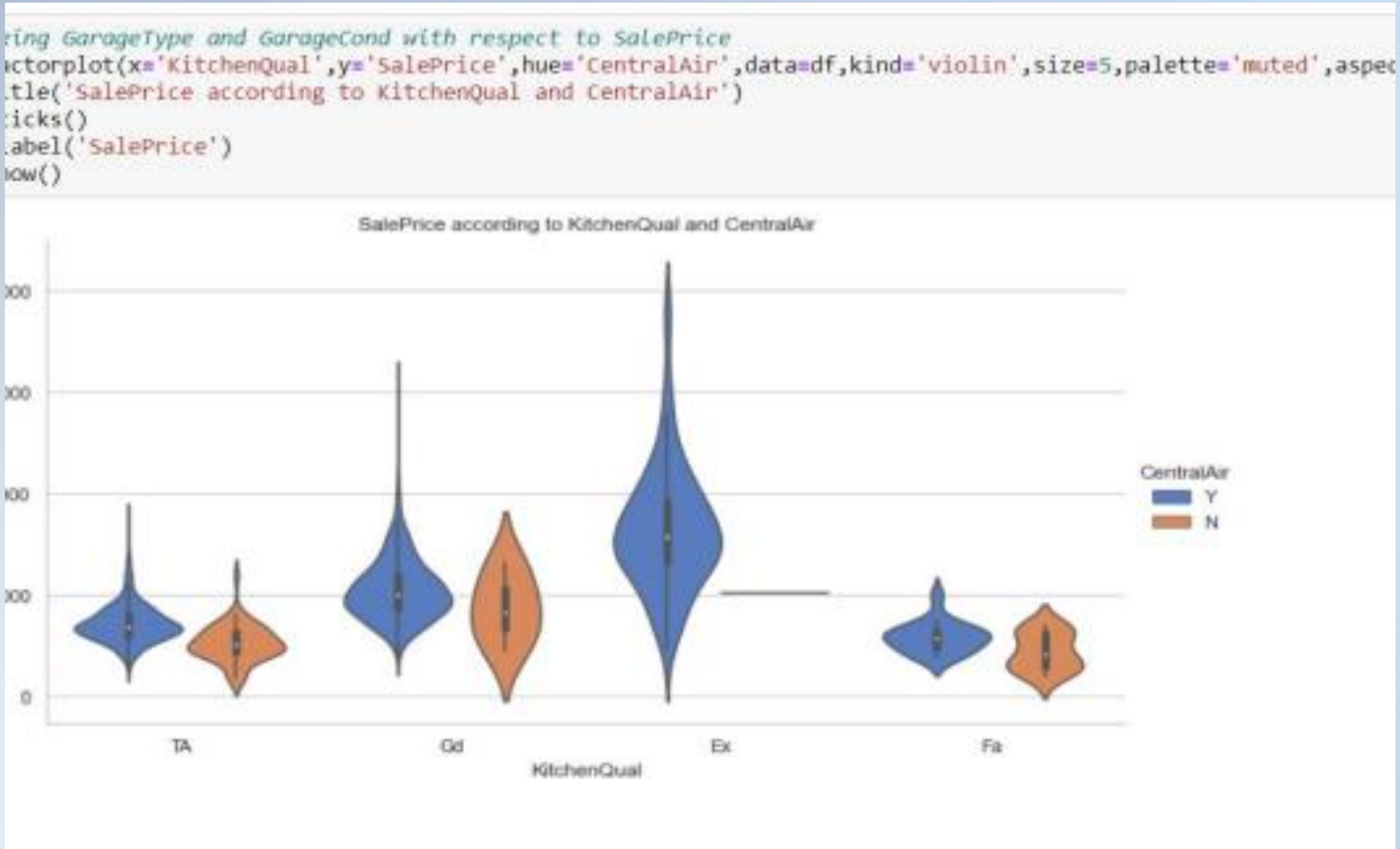
Sale Price is maximum with No Ridge Neighborhood.




```
SalePrice  HouseStyle
34900      1Story        1
35311      1Story        1
37900      1.5Fin        1
39300      1Story        1
40000      2Story        1
                        ..
582933     2Story        1
611657     1Story        1
```

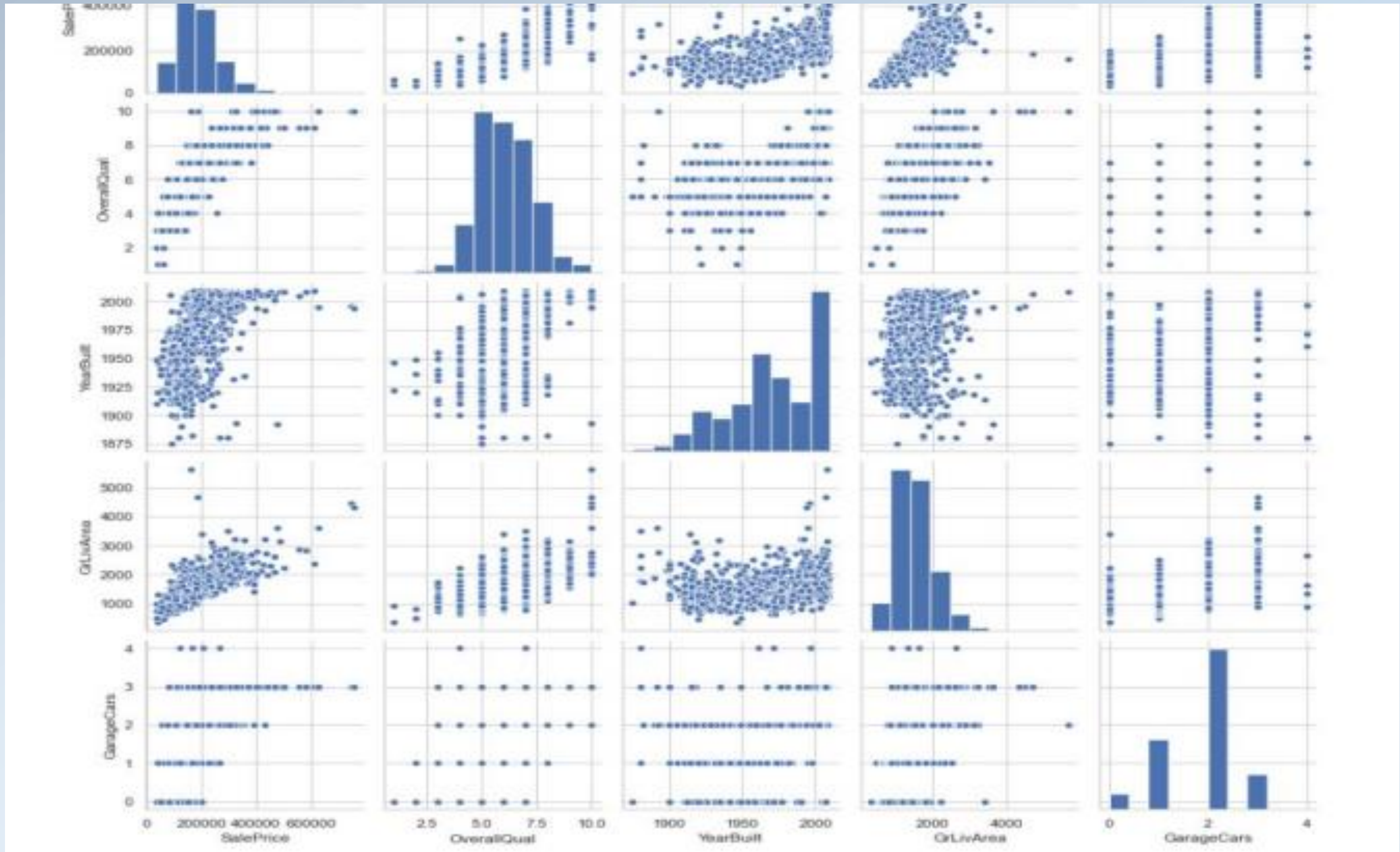
Observation:

Observation: Sale Price is maximum with 2.5Fin House Style.



Observation:

Sale Price is maximum with Ex kitchen Qual and Central Air.



Observation:

Sale Price is highly positively correlated with Gr Liv Area and Overall Qual.

Steps and assumptions used to complete the project

Data Pre-processing Done

We first done data cleaning. We first looked 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

We then explored categorical variables.

We observed that there is only one unique value present in Utilities so will be dropping this column

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
```

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
```

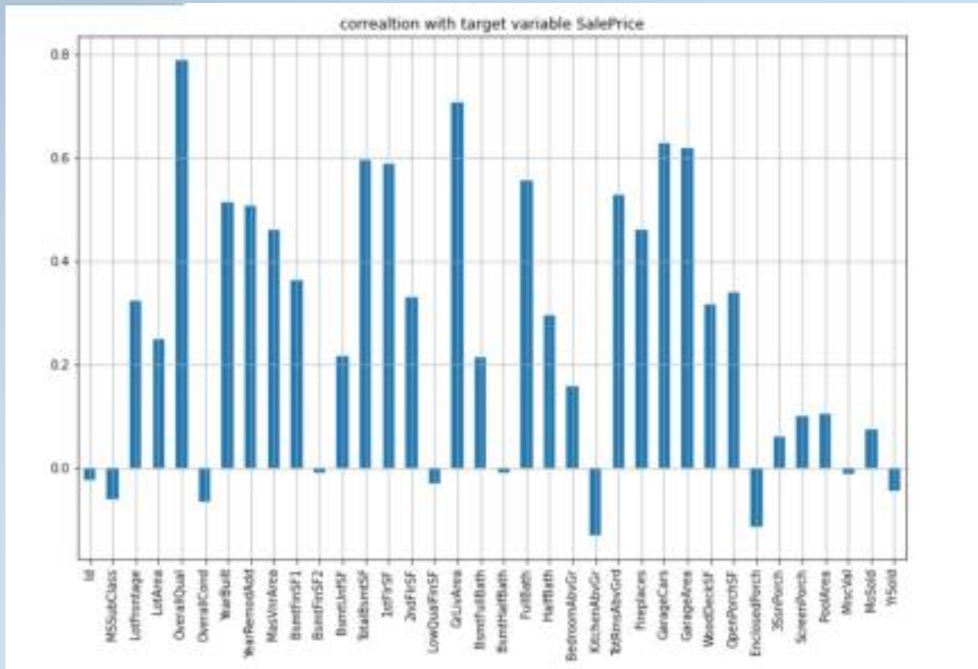
	Id	MSSubClass	LotFrontage	LotArea	Utilities	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnf!
0	127	120	70.0	4928	AllPub	6	5	1976	1976	0.0	120	0	9
1	889	20	95.0	15865	AllPub	8	6	1970	1970	0.0	351	823	10
2	793	60	92.0	9920	AllPub	7	5	1996	1997	0.0	862	0	2
3	110	20	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

While checking the heatmap of correlation we observed that,

- 1. Sale Price is highly positively correlated with the columns Overall Qual, Year Built, Year Remod Add, Total BsmtSF, 1stFlrSF, Gr Liv Area, Full Bath, Tot Rms Abv Grd, Garage Cars, Garage Area.
- 2. Sale Price is negatively correlated with Overall Cond, Kitchen Abv Gr, Enclose porch, Yr Sold.
- 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.

Here we check the correlation between all our feature variables with target variable label.



1. The column Overall Qual is most positively correlated with Sale Price.

2. The column Kitchen Abv Grd is most negatively correlated with Sale Price.

Problem-solving approaches

- ✓ 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 improper 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

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 this columns.

Model Dashboard

```python
#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
```

```
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
********************************************************************************************************
*********************
```

Finalized Model

```
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
```

Conclusion

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 Regressor through GridSearchCV though Lasso Regressor model performed well too. 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.

Acknowledgement

I would like to express my special thanks of grattitude to the sources Medium, Towards DataScience, Stack Overflow, Campus X and KrishNaik's youtube channel which helped me to accomplish this project.