HOUSING: PRICE PREDICTION

Submitted by:

Sampada Sambrani

Internship batch 34
e-mail:sampada.sambrani@gmail.com

ACKNOWLEDGMENT

I would like to express my deep and sincere gratitude to FLIP ROBO for giving me the opportunity to do this project. As a great bridge between academic and industry, this program educated me how to perform theoretical methodology in real life. I would like to express my sincere thankfulness to our assigned mentors for the continuous support of our queries, for their patience, enthusiasm, motivation and immense knowledge.

INTRODUCTION

The real estate sector is an important industry with many stakeholders ranging from regulatory bodies to private companies and investors. Among these stakeholders, there is a high demand for a better understanding of the industry operational mechanism and driving factors. Today there is a large amount of data available on relevant statistics as well as on additional contextual factors, and it is natural to try to make use of these in order to improve our understanding of the industry. Notably,

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 main steps in this project are:

• Exploratory Data Analysis (EDA). By conducting explanatory data analysis, we obtain a better understanding of our data. This yields insights that can be helpful later when building a model, as well as insights that are independently interesting.

• Feature Selection In order to avoid overfitting issues.

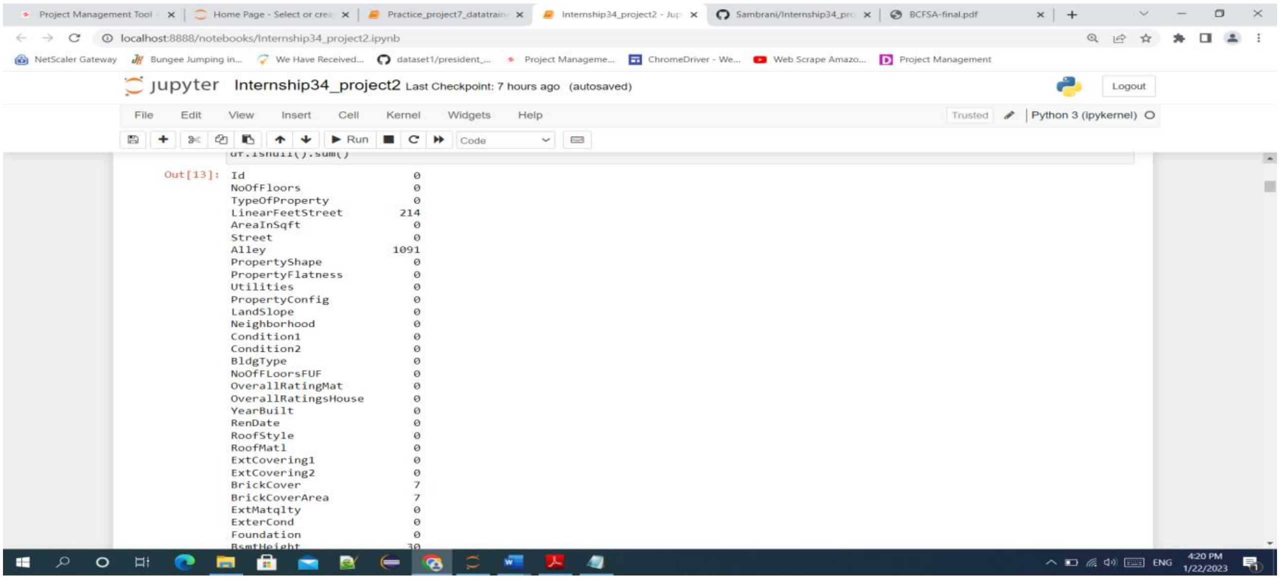
• Modeling We apply LinearRegression and check with the r2score.

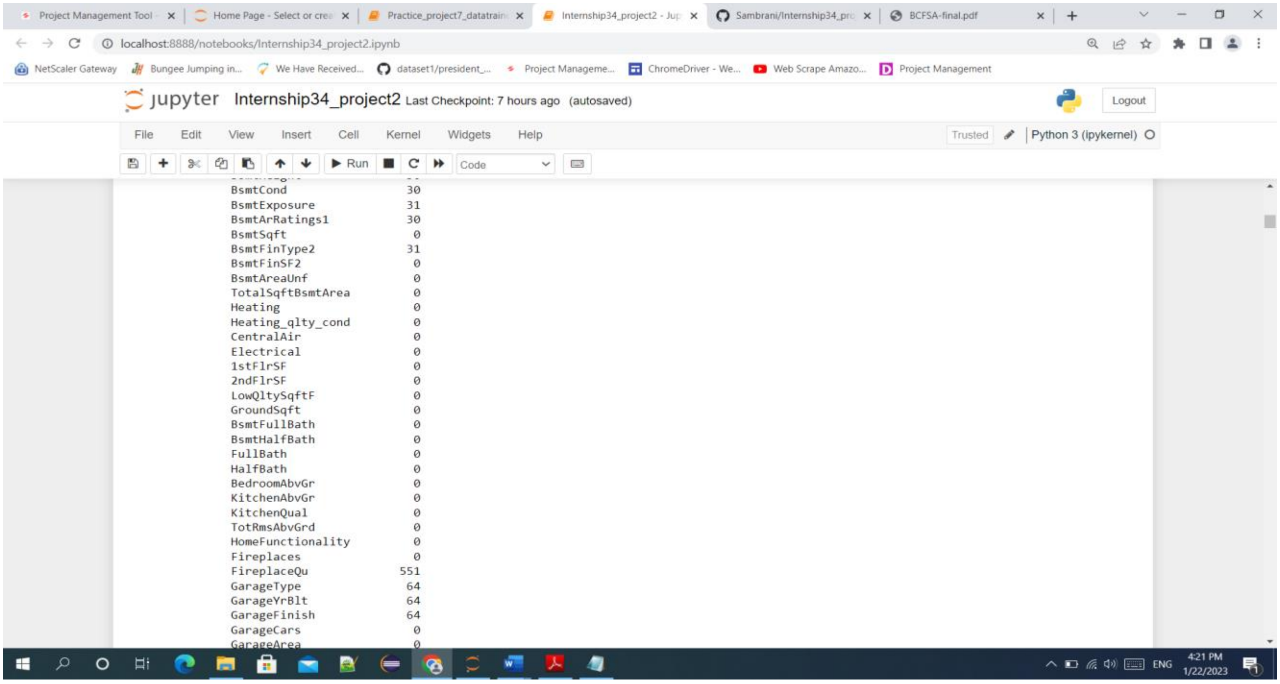
We are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

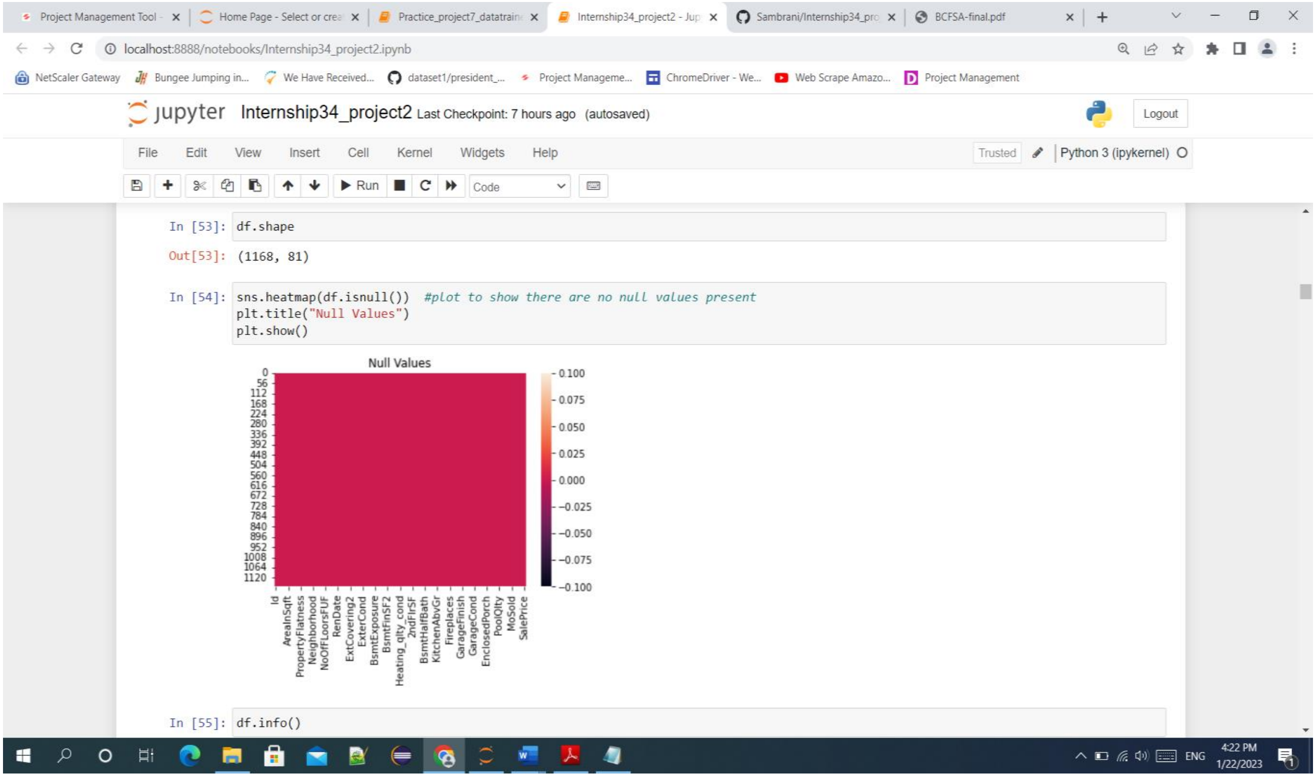
Analytical Problem Framing

- Mathematical/Analytical Modelling of the Problem

As our dataset had numerous null values , we first used fillna() to fill the null/nan values in all the columns , by using mean for numeric data columns and mode for categorical data columns as shown below







- The statistical summary was obtained by using describe (), which gives some idea of the percentile, mean ,count,standard deviation , min and max values , so that we can get idea whether our data is skewed ,our data is highly spreaded and some knowledge of outliers are present or not / may be present.
- The correlation w.r.t the target variable was checked, to get the features which are positively and negatively correlated with the target variable.We have checked for multicollinearity exist or not using the corr().The acceptable range is <+/-0.7
- The skewness was checked for the feature variables only ,using the skew(). If high skewness is present then we using various transformation techniques to reduce the skewness, and even after transformation if not reduced then we can drop the columns. The acceptable range for skewness is +/-0.5

- Data Sources and their formats

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.

- Two datasets are being provided (test.csv, train.csv). We have to train on train.csv dataset and predict on test.csv file

```
df=pd.read_csv("train.csv")
df
```

0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN

1168 rows × 81 columns

```
df.columns
```

```
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')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 81 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Id                    1168 non-null   int64
 1   NoOfFloors            1168 non-null   int64
 2   TypeOfProperty        1168 non-null   object
 3   LinearFeetStreet      1168 non-null   float64
 4   AreaInSqft            1168 non-null   int64
 5   Street                1168 non-null   object
 6   Alley                 1168 non-null   object
 7   PropertyShape         1168 non-null   object
 8   PropertyFlatness      1168 non-null   object
 9   Utilities             1168 non-null   object
 10  PropertyConfig        1168 non-null   object
 11  LandSlope             1168 non-null   object
 12  Neighborhood          1168 non-null   object
 13  Condition1            1168 non-null   object
 14  Condition2            1168 non-null   object
 15  BldgType              1168 non-null   object
 16  NoOfFLoorsFUF         1168 non-null   object
 17  OverallRatingMat      1168 non-null   int64
 18  OverallRatingsHouse   1168 non-null   int64
 19  YearBuilt             1168 non-null   int64
 20  RenDate               1168 non-null   int64
```

```
20   RenDate               1168 non-null    int64
21   RoofStyle             1168 non-null    object
22   RoofMatl              1168 non-null    object
23   ExtCovering1          1168 non-null    object
24   ExtCovering2          1168 non-null    object
25   BrickCover            1168 non-null    object
26   BrickCoverArea        1168 non-null    float64
27   ExtMatqlty            1168 non-null    object
28   ExterCond             1168 non-null    object
29   Foundation            1168 non-null    object
30   BsmtHeight            1168 non-null    object
31   BsmtCond              1168 non-null    object
32   BsmtExposure          1168 non-null    object
33   BsmtArRatings1        1168 non-null    object
34   BsmtSqft              1168 non-null    int64
35   BsmtFinType2          1168 non-null    object
36   BsmtFinSF2            1168 non-null    int64
37   BsmtAreaUnf           1168 non-null    int64
38   TotalSqftBsmtArea     1168 non-null    int64
39   Heating               1168 non-null    object
40   Heating_qlty_cond     1168 non-null    object
41   CentralAir            1168 non-null    object
```

```
39   Heating               1168 non-null    object
40   Heating_qlty_cond     1168 non-null    object
41   CentralAir            1168 non-null    object
42   Electrical            1168 non-null    object
43   1stFlrSF              1168 non-null    int64
44   2ndFlrSF              1168 non-null    int64
45   LowQltySqftF          1168 non-null    int64
46   GroundSqft            1168 non-null    int64
47   BsmtFullBath          1168 non-null    int64
48   BsmtHalfBath          1168 non-null    int64
49   FullBath              1168 non-null    int64
50   HalfBath              1168 non-null    int64
51   BedroomAbvGr          1168 non-null    int64
52   KitchenAbvGr          1168 non-null    int64
53   KitchenQual           1168 non-null    object
54   TotRmsAbvGrd          1168 non-null    int64
55   HomeFunctionality     1168 non-null    object
56   Fireplaces            1168 non-null    int64
57   FireplaceQu           1168 non-null    object
58   GarageType            1168 non-null    object
59   GarageYrBlt           1168 non-null    float64
60   GarageFinish          1168 non-null    object
61   GarageCars            1168 non-null    int64
62   GarageArea            1168 non-null    int64
63   GarageQual            1168 non-null    object
64   GarageCond            1168 non-null    object
65   PavedDrive            1168 non-null    object
66   WoodDeckSF            1168 non-null    int64
67   OpenPorchSF           1168 non-null    int64
68   EnclosedPorch         1168 non-null    int64
69   3SsnPorch             1168 non-null    int64
70   ScreenPorch           1168 non-null    int64
```

```
 52   KitchenAbvGr         1168 non-null   int64
 53   KitchenQual          1168 non-null   object
 54   TotRmsAbvGrd         1168 non-null   int64
 55   HomeFunctionality    1168 non-null   object
 56   Fireplaces           1168 non-null   int64
 57   FireplaceQu          1168 non-null   object
 58   GarageType           1168 non-null   object
 59   GarageYrBlt          1168 non-null   float64
 60   GarageFinish         1168 non-null   object
 61   GarageCars           1168 non-null   int64
 62   GarageArea           1168 non-null   int64
 63   GarageQual           1168 non-null   object
 64   GarageCond           1168 non-null   object
 65   PavedDrive           1168 non-null   object
 66   WoodDeckSF           1168 non-null   int64
 67   OpenPorchSF          1168 non-null   int64
 68   EnclosedPorch        1168 non-null   int64
 69   3SsnPorch            1168 non-null   int64
 70   ScreenPorch          1168 non-null   int64
 71   PoolArea             1168 non-null   int64
 72   PoolQlty             1168 non-null   object
 73   FenceQlty            1168 non-null   object
 74   MiscFeature          1168 non-null   object
 75   MiscVal              1168 non-null   int64
 76   MoSold               1168 non-null   int64
 77   YrSold               1168 non-null   int64
 78   SaleType             1168 non-null   object
 79   SaleCondition        1168 non-null   object
 80   SalePrice            1168 non-null   int64
dtypes: float64(3), int64(35), object(43)
```

- Data Preprocessing Done

  For data cleaning we have firstly filled all the null values with respective mean,median or mode depending on the type of data.

  After checking with the skewness and applying power transformation to reduce the skewness , even if the columns had high skewness, we have to drop that columns (very high negative and positive skewed columns).

```
#checking the skewness only for feature columns
import warnings
warnings.filterwarnings('ignore')
x.skew().sort_values(ascending=False)
```

```
MiscVal              23.065943
PoolArea             13.243711
Condition2           11.514458
AreaInSqft           10.659285
Heating              10.103609
3SsnPorch             9.770611
LowQltySqftF          8.666142
RoofMatl              7.577352
Alley                 5.436187
LandSlope             4.812568
BsmtFinSF2            4.365829
KitchenAbvGr          4.365259
BsmtHalfBath          4.264403
ScreenPorch           4.105741
EnclosedPorch         3.043610
Condition1            3.008289
BrickCoverArea        2.834658
LinearFeetStreet      2.710383
OpenPorchSF           2.410840
```

```
TypeOfProperty        -1.796785
ExtMatqlty            -1.810843
ExterCond             -2.516219
SaleCondition         -2.671829
Electrical            -3.104209
PropertyFlatness      -3.125982
FenceQlty             -3.185107
PavedDrive            -3.274035
BsmtCond              -3.293554
CentralAir            -3.475188
BsmtFinType2          -3.615783
SaleType              -3.660513
HomeFunctionality     -3.999663
GarageQual            -4.582386
GarageCond            -5.422472
Street               -17.021969
MiscFeature          -17.238424
PoolQlty             -19.401558
dtype: float64
```

As we can see skewness in most of the columns , we will remove the skewness by using power_transform() and try to bring it as close to 1

```
PoolQlty             -17.021969
Street               -17.021969
dtype: float64
```

From the above observation we can see that, the skewness still needs to be reduced.

```
#Drop the columns, which are highly skewed
df.drop(['PoolArea','Street','PoolQlty','MiscFeature','3SsnPorch','LowQltySqftF'],axis=1,inplace=True)
```

- Data Inputs- Logic- Output Relationships

The data inputs and Output relationship in our project was w.r.t the data set provided .

- (test.csv, train.csv). We have to train on train.csv dataset and predict on test.csv file.
- From the input (feature variables) we need to determine:
  - Which variables are important to predict the price of variable?
  - How do these variables describe the price of the house?
- If any of the columns are least important in determining the house prices, we can drop from the dataset
- The output data is our predictions made for the house price based on the input data(feature columns), as test.csv consist of only feature columns.
- Upon training the data for feature and target columns in train.csv , based on these assumptions , the machine has predicted the output(house price) on the test.csv.

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

  Only assumption made here is , based on the high skewness values for certain column, we have dropped the columns

- Hardware and Software Requirements and Tools Used

Hardware specification

Processor: intel CORE i3 (10th gen) minimum

RAM: 2GB and above

Hard disc capacity: Minimum of 100GB

Display type: Standard VGA

Software specification

Operating system: Windows 10

Front end : Jupyter framework(anaconda)

Programming tool: Anaconda

Internet browser: Google chrome

Libraries

import numpy as np – for numeric algebra
import pandas as pd- for data representation
import sklearn
import matplotlib.pyplot as plt- for data visualization
import seaborn as sns
from sklearn.model_selection import train_test_split

- Scikit-learn is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit-learn can also be used for data-mining and data-analysis, which makes it a great tool who is starting out with ML.
- SciPy is a very popular library among Machine Learning enthusiasts as it contains different modules for optimization, linear algebra, integration and statistics.
- NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-

level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra

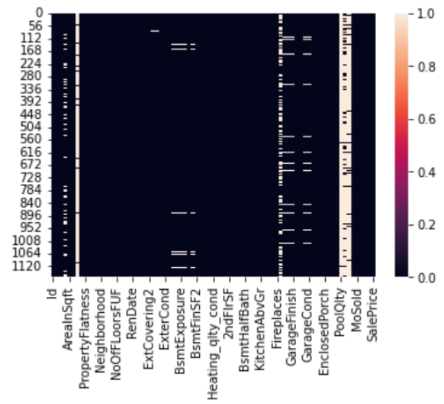
- Pandas is a popular Python library for data analysis.
- Matplotlib is a very popular Python library for data visualization

Model/s Development and Evaluation

- Our dataset had some null values, as shown in the fig below, these null values were filled by fillna().

```
sns.heatmap(df.isnull())
```

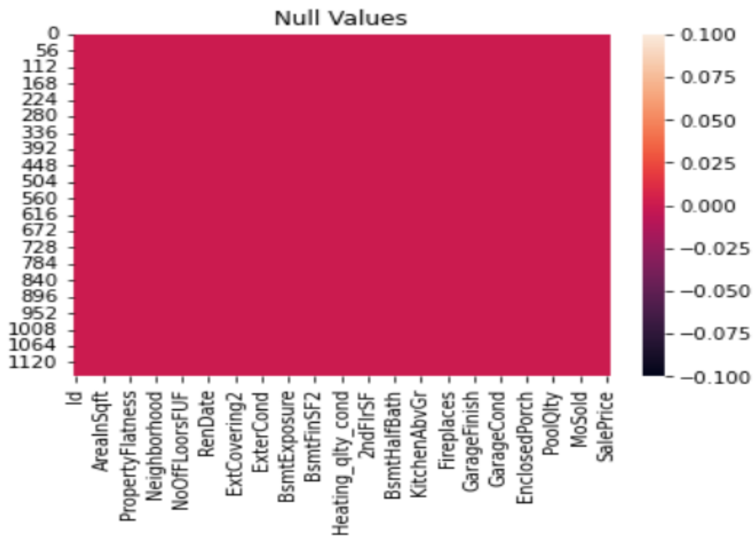
<AxesSubplot:>



From the above details : the total null values present in each columns are: LinearFeetStreet:214 Alley:1091 BrickCover:7 BrickCoverArea:7 BsmtHeight:30 BsmtCond:30 BsmtExposure:31 BsmtArRatings1:30 BsmtFinType2:31 FireplaceQu:551 GarageType:64 GarageYrBlt:64 GarageFinish:64 GarageQual:64 GarageCond:64 PoolQlty:1161 FenceQlty:931 MiscFeature :1124

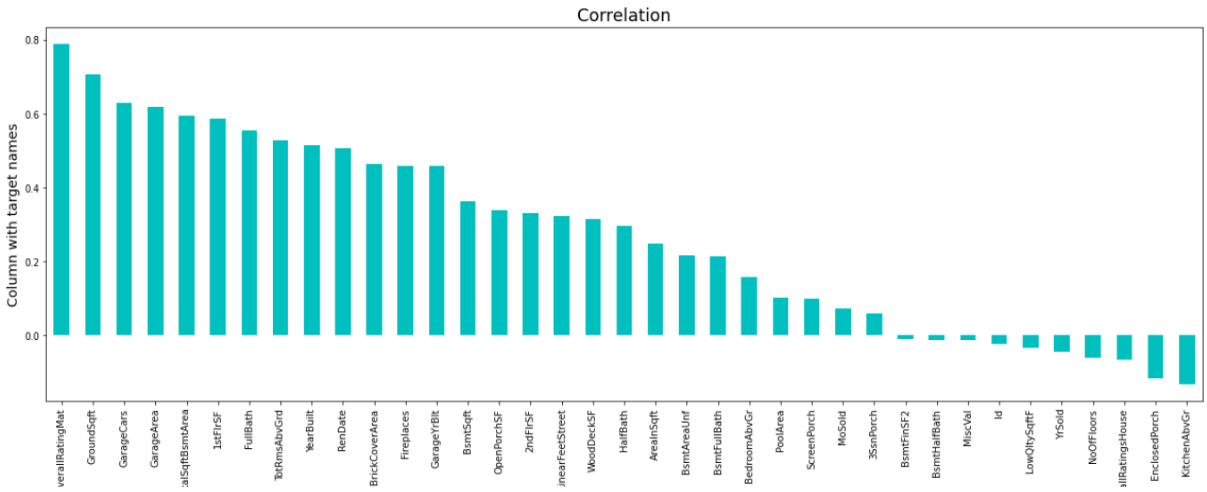
- After the null/nan values were replaced with mean/mode depending on the columns datatype, the fig below shows that there are no null values present.

```
sns.heatmap(df.isnull())  #plot to show there are no null values present
plt.title("Null Values")
plt.show()
```



- The correlation w.r.t the target variable was found using corr(), the below fig shows which columns are positively and negatively correlated, which can show us that which features are affecting the house price.

```
#checking columns which are positively and negatively corelated with target columns:
plt.figure(figsize=(22,7))
df.corr()['SalePrice'].sort_values(ascending=False).drop(['SalePrice']).plot(kind='bar',color='c')
plt.xlabel('Feature',fontsize=14)
plt.ylabel('Column with target names',fontsize=14)
plt.title('Correlation',fontsize=18)
plt.show()
```



- Since the target variable has continuous numeric values , its see that the problem is based on Linear Regression . The below figure shows the test results and predicted results.

```
#scaling the data using min-max scaler
from sklearn.preprocessing import MinMaxScaler
mns=MinMaxScaler()
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
for i in range(0,100):
    x_train,x_test,y_train,y_test=train_test_split(x1,y1,test_size=0.2,random_state=i)
    lr.fit(x_train,y_train)
    pred_train=lr.predict(x_train)
    pred_test=lr.predict(x_test)
    print(f"At randomstate{i},the training accuracy is:-{r2_score(y_train,pred_train)}")
    print(f"At randomstate{i},the testing accuracy is:-{r2_score(y_test,pred_test)}")
    print("\n")
```

```
At randomstate48,the training accuracy is:-0.8267336551098403
At randomstate48,the testing accuracy is:-0.8593751289526096


At randomstate49,the training accuracy is:-0.8235924005673824
At randomstate49,the testing accuracy is:-0.8764363124697591


At randomstate50,the training accuracy is:-0.8290116526510287
At randomstate50,the testing accuracy is:-0.8503956989264774


At randomstate51,the training accuracy is:-0.8323267637034968
At randomstate51,the testing accuracy is:-0.8051017934685027
```

```
x_train,x_test,y_train,y_test=train_test_split(x1,y1,test_size=0.3,random_state=48)
lr.fit(x_train,y_train)
pred_test=lr.predict(x_test)
print(r2_score(y_test,pred_test))
```

```
0.8528141394765885
```

approx: 86%

- The predictions to be made on the test.csv dataset to determine the house prices.

```
Final_result_pred=pd.DataFrame()
Final_result_pred['ID']=df1['Id']
Final_result_pred['SalePrice']=ans
```

```
Final_result_pred
```

	ID	SalePrice
0	337	339253.582459
1	1018	230640.399044
2	929	269113.359184
3	1148	158820.870119
4	1227	248250.779481
...	...	...
287	83	261759.164024
288	1048	141088.700667
289	17	172003.963514
290	523	209034.483827
291	1379	107562.293039

292 rows × 2 columns

CONCLUSION

- ## Key Findings and Conclusions of the Study

Our Dataset contained numerical as well as categorical variables.
You need to find important features which affect the price positively or negatively

- ## Learning Outcomes of the Study in respect of Data Science

According to the results, the Linear Regression Model obtained the most remarkable accuracy. The outcome of training the given file was seen by the results obtained at the test file , where the predictions made were approx. 85%
.

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

People will be able to utilize this program in the future to acquire the most accurate pricing of a home.

This application may be converted into a Flutter application to get support for Android and iOS devices, allowing it to be used everywhere. It can also be used as an external or internal service for apps that display property for rent.

Users may apply this methodology to various fields, such as tuition costs in a specific location, swimming pool rates, and data science-type models.