Industry practioner's suggested assignment

Topic: Bangalore house price prediction

Meet Tank – Smit vora

Link of implementation: https://github.com/Meet200/Machine_Learning-/blob/main/Industry_ML_work.ipynb

Dataset explanation : Dataset contains area-type(super built up, plot area , build-up area) of flat , availability of flat/home (ready to move or any specific date), location of house , total square-feet area , total bath and balcony available in house, size of house in BHK and price of house. Among this much of columns there are some columns which will not help to build our model which we have not taken in count while building our model. Same columns are available with testing dataset

Data loading and understanding

1st step is to load data: (here we can also see the size of data)

```
dataset = pd.read_csv("Train.csv") # Assigning data to a variable called "d
print(dataset)#printing dataset
```

```
                area_type   availability  ... balcony   price
0      Super built-up  Area         19-Dec  ...     1.0   39.07
1                Plot  Area  Ready To Move  ...     3.0  120.00
2            Built-up  Area  Ready To Move  ...     3.0   62.00
3      Super built-up  Area  Ready To Move  ...     1.0   95.00
4      Super built-up  Area  Ready To Move  ...     1.0   51.00
...                     ...            ...  ...     ...     ...
13315        Built-up  Area  Ready To Move  ...     0.0  231.00
13316  Super built-up  Area  Ready To Move  ...     NaN  400.00
13317        Built-up  Area  Ready To Move  ...     1.0   60.00
13318  Super built-up  Area         18-Jun  ...     1.0  488.00
13319  Super built-up  Area  Ready To Move  ...     1.0   17.00

[13320 rows x 9 columns]
```

2nd step we need to find information available in data for that,

```
dataset.describe() #finding information from dat
```

	bath	balcony	price
count	13247.000000	12711.000000	13320.000000
mean	2.692610	1.584376	112.565627
std	1.341458	0.817263	148.971674
min	1.000000	0.000000	8.000000
25%	2.000000	1.000000	50.000000
50%	2.000000	2.000000	72.000000
75%	3.000000	2.000000	120.000000
max	40.000000	3.000000	3600.000000

Industry practioner's suggested assignment

We can use .describe() function in dataset and it will give result for numerical columns we can see that for bath, balcony, price it is giving us different values and we can also see that Min price of house in Bangalore is 8 Lakh and Maximum price is 3600 Lakh.

In next step will try to find out what are the columns available in dataset and it contains which type of different classes.

Foe example: Below image shows what are different values available under area_type column and what is number of frequency of data in different classes.

```
# count of flats with different areas type
dataset['area_type'].value_counts()
```

```
Super built-up  Area    8790
Built-up  Area          2418
Plot  Area              2025
Carpet  Area              87
Name: area_type, dtype: int64
```

Data cleaning and processing

Now to build model we need to remove some of the column which is not needed and also convert datatype of someof the column and add some new column which will consider under data cleaning process

next step is to remove unnecessary we are having column like society , availability , number of balcony and bath(easy countable from BHK).

```
# removing unnecessary columns like , area type, availability ,balcony type
dataset2 = dataset.drop(['area_type','society','balcony','availability'],axis='columns')
```

We are also having some row and columns containing NULL values which we should remove columns having NULL values are,

```
#first step in data-clea
dataset2.isnull().sum()
```

```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

So will remove those row and clean our dataset after removing that we can see in below image that there are no NULL values available and we can also see the size of data after removing NULL values,

Industry practioner's suggested assignment

```
# there will be no null
dataset3.isnull().sum()
```

```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

```
#dataset shape before an
print(dataset2.shape)
print(dataset3.shape)
```

```
(13320, 5)
(13246, 5)
```

Size of house/flat is in form of 3BHK- 1BHK-2BHK and so on we are using lambda function to convert it into the only integer format like 3-2-1…. We have used concept of splitting and lambda function for the same. Below images shows code and output for the same

```
dataset3['size_bhk'] = dataset3['size'].apply(lambda x: int(x.split(' ')[0]))
dataset3.size_bhk.unique()
```

```
array([ 2,  4,  3,  6,  1,  8,  7,  5, 11,  9, 27, 10, 19, 16, 43, 14, 12,
       13, 18])
```

In dataset columns like total_sqft , price of house are stored in string format which we need to convert to Float so we can use it in our model for prediction in below image we can see function which convert string to float.

```
# function to convert string to float
def str_to_float(x):
    try:
        float(x)
    except:
        return False
    return True

# converting total_sqft area from string to float
dataset3[~dataset3['total_sqft'].apply(str_to_float)]
```

Total_sqft column is having some value present in range form i.e., 1200-1300 , 2000-2100 so we need to remove ‘-’ separator from that range value and assign new values which will average value of those 2 values

```
# function which will remove '-'  from dataset
def remove_for_sqft(x):
    tokens = x.split('-')
    if len(tokens) ==2:
        return (float(tokens[0]) + float(tokens[1]))/2
    try:
        return float(x)
    except:
        return None
```

Industry practioner's suggested assignment

In next step will try to find-out price of house per square feet which will allow to find property price range in different are in each segments

```
# finding price/sqft for any house just for information
dataset5 = dataset4.copy()
dataset5['price_per_sqft'] = dataset5['price']*100000  # price/sqft considring
dataset5['price_per_sqft'] = dataset5['price_per_sqft']/dataset5['total_sqft']
dataset5.head()
```

Example:

	location	size	total_sqft	bath	price	size_bhk	price_per_sqft
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2	3699.810606
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615.384615
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305.555556
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245.890861
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250.000000

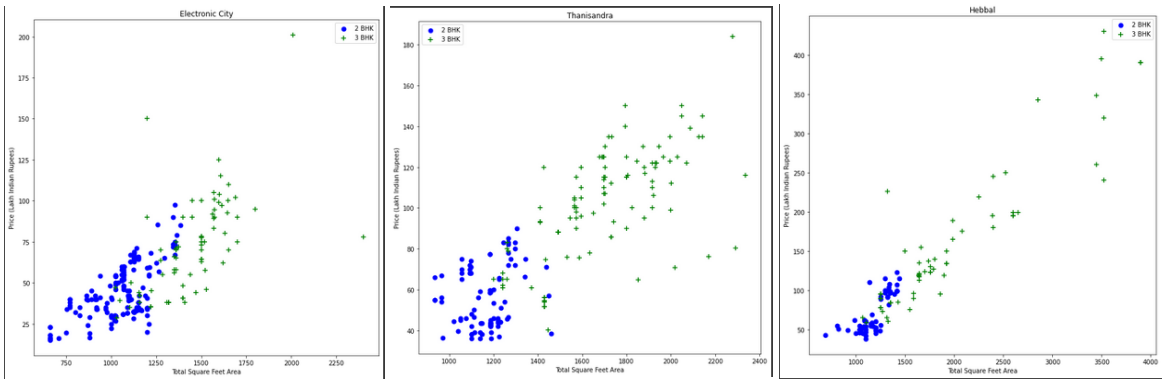
Now in dataset some house location are very rarely use below image show the frequency of different location.

```
location
Whitefield              533
Sarjapur  Road          392
Electronic City         304
Kanakpura Road          264
Thanisandra             235
                       ...
Kumbhena Agrahara         1
Kudlu Village,            1
Konappana Agrahara        1
Kodanda Reddy Layout      1
1 Annasandrapalya         1
Name: location, Length: 1287, dtype: int64
```

Here as we can see some location are sue 1 time same as that there are many location available so for that location having < 10 frequency will give it common name as ‘Other’ which will help to reduce complexity in one hot encoding technique.

Now next step is to build-up our model and for that we need to plot some graph to visualize shape of data below 3 graph shows that data is in linear form and we can use techniques like Linear regression or Lasso regression or ridge regression,

Industry practioner's suggested assignment



Here we can also see that price of 3BHK house (green dots) is > 2BHK in any area.so now will start to build our model and before that we need to apply one hot encoding to classify the value of house

```
dummies = pd.get_dummies(dataset6.location)
dummies.head(3)
```

```
#concatanating not hot endoing  values
dataset6 = pd.concat([dataset6,dummies],axis='columns')
dataset6.head()
```

After one hot encoding data may look like this,

location	size	total_sqft	bath	price	size_bhk	price_per_sqft	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	7th Phase JP Nagar	8th Phase JP Nagar
Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2	3699.810606	0	0	0	0	0	0	0	0	0
Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615.384615	0	0	0	0	0	0	0	0	0
Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305.555556	0	0	0	0	0	0	0	0	0
Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245.890861	0	0	0	0	0	0	0	0	0
Kothanur	2 BHK	1200.0	2.0	51.00	2	4250.000000	0	0	0	0	0	0	0	0	0

Model building

First step in model building is to divide Target variable (price) and features.

```
# defining target varibles and features
X = dataset7.drop(['price'],axis='columns')
y = dataset7.price
print("X_shape", X.shape)
print("y_shape", y.shape)
```

After that will be splitting into training and testing data in 20-80%.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=10)
```

Industry practioner's suggested assignment

We will be applying all 3 regression Linear, ridge and lasso to find out which method has best accuracy.

```
# Linear regresson
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
regression.fit(X_train,y_train)
regression.score(X_test,y_test)
```

```
0.7926222597511906
```

Linear regression is giving 79% accuracy same as that we can see accuracy for lasso and ridge they are almost same just change in fraction only

```
#Lasso regression
from sklearn import linear_model
regression2 = linear_model.Lasso(alpha=0.05)
regression2.fit(X_train,y_train)
regression2.score(X_test,y_test)
```

```
0.7923245728236419
```

```
#ridge regression
from sklearn.linear_model import Ridge
regression3 = Ridge(alpha=0.1)
regression3.fit(X_train,y_train)
regression3.score(X_test,y_test)
```

```
0.7926987079682799
```

Now next step is to predict price of house for test data where we can manually type constrains for house or we can directly select from test data and it will give result in Lakhs,

```
#Testing on Test dataset
def predict_price(location,sqft,bath,bhk):
    loc_index = np.where(X.columns==location)[0][0]

    x = np.zeros(len(X.columns))
    x[0] = sqft
    x[1] = bath
    x[2] = bhk
    if loc_index >= 0:
        x[loc_index] = 1

    return regression.predict([x])[0]
```

Industry practioner's suggested assignment

Example: in below image it is showing house price for row 1 which is of 9BHK house and 2400 sqft with 9 Bathrooms it is predicting 157Lakh rupees for this house

```
predict_price(test_dataset.location[1],test_dataset.total_sqft[1], test_dataset.bath[1], te

157.3887448310852
```

While in below image I have randomly chosen all value and we can see the price different for not specified location it is giving price of 10Lakh while in Electronic city it is giving value of 13 Lakh rupees.

```
predict_price("other",1800, 3, 3) #predicting value from random data

10.552270889282227

predict_price("Electronic City",1800, 3, 3) #predicting value from random data

13.58362865447998
```

We can randomly select house same as done in above image and predict price of house.

Conclusion : Information which I can find out is that price of house mainly depend on 2 things size of house (BHK) and area of house (sqft) this 2 things will affect value too much where in some case Location also matters because of high demand value can be increase. In this work main part is of cleaning of data set because it is too much random data and having too much null and random value like in some case number of bathroom are too much then number of BHK so it is not easy to say it whether it is true data or not. And also data correction play measure role in this assignment.