Banglore\_House\_Price\_prediction

~by Abhishek Mishra, Batch 1826

(***statements in red are codes)***

Problem Sratement: Here we have to predict the price of a house in banglore given different attributes.

First we will import all the necessary packages needed for our dataset.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

import warnings

warnings.simplefilter("ignore")

Now we will upload our dataset using pandas

df=pd.read\_csv("F:\DataTrained project datasets\project dynamic m20/Bengaluru\_House\_Data.csv")

we will now se the first 5 rows of our dataset using : df.head()

df.info()

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 area\_type 13320 non-null object

1 availability 13320 non-null object

2 location 13319 non-null object

3 size 13304 non-null object

4 society 7818 non-null object

5 total\_sqft 13320 non-null object

6 bath 13247 non-null float64

7 balcony 12711 non-null float64

8 price 13320 non-null float64

dtypes: float64(3), object(6)

memory usage: 936.7+ KB

for column in df.columns:

print(df[column].value\_counts())

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

By above code we check te value counts of all the columns

Checking the Null data in our dataset:

df.isna().sum()

area\_type 0

availability 0

location 1

size 16

society 5502

total\_sqft 0

bath 73

balcony 609

price 0

dtype: int64

Replacing the Nan values by suitable values:

df["location"]=df["location"].fillna("Whitefield")

df["size"]=df["size"].fillna("2 BHK")

df["bath"]=df["bath"].fillna(df["bath"].median())

df["balcony"]=df["balcony"].fillna(df["balcony"].median())

df.isna().sum()

area\_type 0

availability 0

location 0

size 0

society 5502

total\_sqft 0

bath 0

balcony 0

price 0

dtype: int64

Dropping the society column as it has many Nan values:

df.drop("society",axis=1,inplace=True)

As size column ha sdata in the form of 2 BHK ,we need to remove bhk and convert thr object type to int type :

df["size"]=df["size"].str.split(" ").str.get(0).astype(int)

df["total\_sqft"].unique()

array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],

dtype=object)

AS the total\_sqft column has some values which are separated by “-“ so we need to mange those and convert them to mean value of range:

def convert\_range(x):

num=x.split("-")

if len(num)==2:

return(float(num[0])+float(num[1]))/2

try:

return float(x)

except:

return None

df["total\_sqft"]=df["total\_sqft"].apply(convert\_range)

df["location"].value\_counts()

Whitefield 541

Sarjapur Road 399

Electronic City 302

Kanakpura Road 273

Thanisandra 234

...

Satyasaibaba Layout 1

Indira Nagar 3rd Stage 1

Kodanda Reddy Layout 1

Asthagrama Layout 1

Shivanagar 1

Name: location, Length: 1305, dtype: int64

Removing the extra spaces before and after the name in rows:

df["location"]=df["location"].apply(lambda x:x.strip())

As the “location” column is very mess we will consider only those location which occurs more than 10 times :

location\_count=df["location"].value\_counts()

location\_count\_less\_10=location\_count[location\_count<=10]

df["location"]=df["location"].apply(lambda x: 'other' if x in location\_count\_less\_10 else x)

Now checking the value counts:

df["location"].value\_counts()

other 2885

Whitefield 542

Sarjapur Road 399

Electronic City 304

Kanakpura Road 273

...

Banjara Layout 11

Narayanapura 11

Marsur 11

Kodigehalli 11

Thyagaraja Nagar 11

Name: location, Length: 242, dtype: int64

We have changed the categories occurring less than 10 times by “other”

Checking for skewness of data:

df.skew()

size 4.823510

total\_sqft 15.267336

bath 4.237801

balcony -0.058782

price 8.064469

dtype: float64

checking for Outliers:

sns.boxplot(df["price"])



So there are many outliers in “price” column we need to remove them

q3=df["price"].quantile(0.75)

q1=df["price"].quantile(0.25)

iqr=q3-q1

upper=q3+1.5\*iqr

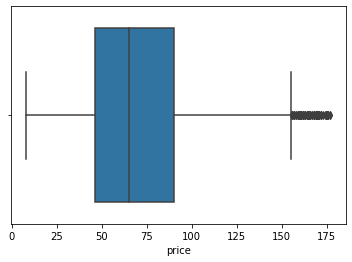
lower=q1-1.5\*iqr

q3,q1,iqr,upper,lower

(100.0, 48.0, 52.0, 178.0, -30.0)

df=df[(df["price"]>0)&(df["price"]<178)]

sns.boxplot(df["price"])



Now we will check outliers for the the total\_sqft column:

sns.boxplot(df["total\_sqft"])



Now we will remove outliers from this :

Q3=df["total\_sqft"].quantile(0.75)

Q1=df["total\_sqft"].quantile(0.25)

IQR=Q3-Q1

upperlimit=Q3+1.5\*IQR

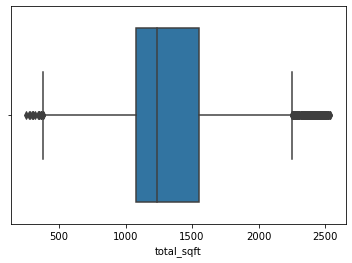
lowerlimit=Q1-1.5\*IQR

upperlimit,lowerlimit,IQR,Q3,Q1

(2545.0, 233.0, 578.0, 1678.0, 1100.0)

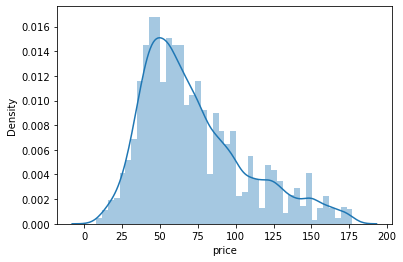
df=df[(df["total\_sqft"]>233)&(df["total\_sqft"]<2545)]

sns.boxplot(df["total\_sqft"])



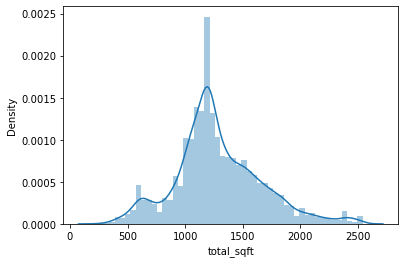
Now we will see the distribution of our columns :

sns.distplot(df["price"])



From graph we can see data is fine not much skewed.

sns.distplot(df["total\_sqft"])

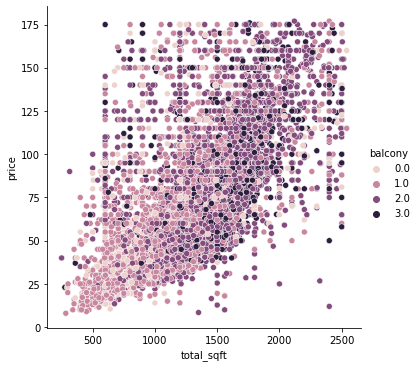


Dropping area\_type and availability column as it is of no use to us

df.drop(columns=["area\_type","availability"],inplace=True)

Seeing relation between total\_sqft and price

sns.relplot(x="total\_sqft",y="price",hue="balcony",data=df)

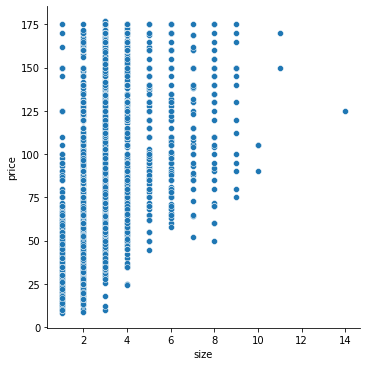


“balcony” column is also of no use so we will drop that as well

df.drop("balcony",axis=1,inplace=True)

relation between “size” and “price”

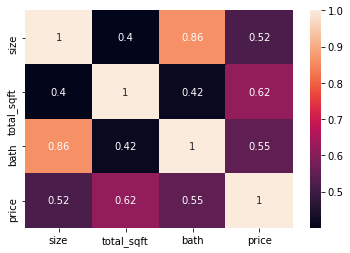
sns.relplot(x="size",y="price",data=df)



Checking the correlation between all the columns

corr=df.corr()

sns.heatmap(corr,annot=True)



Now we will build a model for prediction thus we need to import the packages for same:

from sklearn.linear\_model import LinearRegression,Lasso,Ridge

from sklearn.metrics import mean\_squared\_error,mean\_absolute\_error

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import r2\_score

from sklearn.preprocessing import OneHotEncoder,StandardScaler,LabelEncoder

AS location column is object type we need to convert into numerical or int type to operate on that column

le=LabelEncoder()

df["location"]=le.fit\_transform(df["location"])

Now we will separate our dependent and independent data

Independent data:

x=df.drop("price",axis=1)

dependent data:

y=df["price"]

Now we need to standardize data :

sc=StandardScaler()

X=sc.fit\_transform(x)

We will split the data into train ,test datasets

x\_train,x\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.20,random\_state=48)

x\_train.shape,x\_test.shape,y\_train.shape,y\_test.shape

((9000, 4), (2250, 4), (9000,), (2250,))

Above is the shape of our train and test datasets

Passing the instance of LinearRegression

lm=LinearRegression()

lm.fit(x\_train,y\_train)

pred=lm.predict(x\_test)

lm.score(x\_train,y\_train)

Lasso instance

ls=Lasso()

ls.fit(x\_train,y\_train)

pred\_lasso=ls.predict(x\_test)

ls.score(x\_train,y\_train)

Ridge Instance

rd=Ridge()

rd.fit(x\_train,y\_train)

pred\_ridge=rd.predict(x\_test)

rd.score(x\_train,y\_train)