Loading packages

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import plotly.express as px
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [4]:
```
project_path = "/content/drive/My Drive/Colab Notebooks/"
```

Loading the dataset

In [5]:
```
air=pd.read_csv(project_path+"ABNYC2019.csv")
```

In [6]:
```
air.shape
```

Out[6]:
```
(48895, 16)
```

Checking the attributes of data

In [7]:
```
air.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48895 non-null  int64  
 1   name                            48879 non-null  object 
 2   host_id                         48895 non-null  int64  
 3   host_name                       48874 non-null  object 
 4   neighbourhood_group             48895 non-null  object 
 5   neighbourhood                   48895 non-null  object 
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object 
 9   price                           48895 non-null  int64  
 10  minimum_nights                  48895 non-null  int64  
 11  number_of_reviews               48895 non-null  int64  
 12  last_review                     38843 non-null  object 
 13  reviews_per_month               38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64  
 15  availability_365                48895 non-null  int64  
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

In [8]:
```
air.head(5)
```

Out[8]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	nu
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	

In [9]:
```
air.isnull().values.any()
```

Out[9]:
```
True
```

In [10]:
```
air.describe()
```

Out[10]:

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculate
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466	1.373221	
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	1.680442	
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	0.010000	
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	0.190000	
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	0.720000	
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	2.020000	
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500000	

In [11]:
```
air.mean()
```

Out[11]:
```
id                                1.901714e+07
host_id                           6.762001e+07
latitude                          4.072895e+01
longitude                        -7.395217e+01
price                             1.527207e+02
minimum_nights                    7.029962e+00
number_of_reviews                 2.327447e+01
reviews_per_month                 1.373221e+00
calculated_host_listings_count    7.143982e+00
availability_365                  1.127813e+02
dtype: float64
```

Cleaning the dataset before analysis

In [12]:
```
air.drop(['id','host_id'],axis=1,inplace=True)
```

In [13]:
```
air['reviews_per_month'].fillna(value=air['reviews_per_month'].mean(), inplace=True)
```

In [14]:
```
air['last_review'] = pd.to_datetime(air['last_review'])
```

In [15]:
```
air['last_review'].mean()
```

Out[15]:
```
Timestamp('2018-10-04 01:47:23.910202624')
```

In [16]:
```
air['last_review'].fillna(value=air['last_review'].mean(), inplace=True)
```

In [17]:
```
air.fillna({'name':"Noname"}, inplace=True)
```

In [18]:
```
air.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype         
---  ------                          --------------  -----         
 0   name                            48895 non-null  object        
 1   host_name                       48874 non-null  object        
 2   neighbourhood_group             48895 non-null  object        
 3   neighbourhood                   48895 non-null  object        
 4   latitude                        48895 non-null  float64       
 5   longitude                       48895 non-null  float64       
 6   room_type                       48895 non-null  object        
 7   price                           48895 non-null  int64         
 8   minimum_nights                  48895 non-null  int64         
 9   number_of_reviews               48895 non-null  int64         
 10  last_review                     48895 non-null  datetime64[ns]
 11  reviews_per_month               48895 non-null  float64       
 12  calculated_host_listings_count  48895 non-null  int64         
 13  availability_365                48895 non-null  int64         
dtypes: datetime64[ns](1), float64(3), int64(5), object(5)
memory usage: 5.2+ MB
```

Univariate analysis

In [19]:
```
fig=px.histogram(air,x="room_type",template="plotly_dark",title="Counts of different room type")
fig.show()
```

In [20]:
```
fig=px.histogram(air, x="neighbourhood_group",color="room_type",template="plotly_dark",title="Share of room types in d
ifferent neighbourhood")
fig.show()
```

In [21]:
```
fig = px.histogram(air, x="number_of_reviews",range_x=(0,100),template="plotly_dark",title="Number of reviews recieved by each apartment")
fig.show()
```

In [22]:
```
fig = px.histogram(air, x="availability_365",template="plotly_dark",title="Avialability of apartments")
fig.show()
```

In [23]:
```
fig = px.histogram(air, x="minimum_nights",range_x=(0,50),template="plotly_dark",title="Minimum nights stayed")
fig.show()
```

Bivariate analysis

In [24]:
```
fig = px.scatter(air, x="neighbourhood", y="price", color="room_type",template="plotly_dark", facet_col="neighbourhood
_group",title="Areas with room are available in diffrent neigbourhood")
fig.show()
```

In [25]:
```
fig = px.scatter(air,x="latitude", y="longitude", color="room_type", template="plotly_dark", title="Spread of room typ
es in NYC",width=600,height=400)
fig.show()
```

In [26]:
```
fig = px.pie(air, values='price', names='room_type',template="plotly_dark",title="Share of rooms based on price")
fig.show()
```

In [27]:
```
fig = px.strip(air, x="price", y="minimum_nights",color="room_type",facet_col="neighbourhood_group",template="plotly_d
ark",title="Prices in different neighbourhood")
fig.show()
```

In [28]:
```
fig = px.pie(air, values='price', names='neighbourhood_group',template="plotly_dark",title="Share of neighbourhoods ba
sed on price")
fig.show()
```

In [29]:
```
fig = px.scatter(air, x='last_review', y='price', range_x=['2019-01-01','2019-12-31'],color="room_type",facet_col="neighbourhood_group",template="plotly_dark")
fig.show()
fig = px.scatter(air, x='last_review', y='price', range_x=['2018-01-01','2018-12-31'],color="room_type",facet_col="neighbourhood_group",template="plotly_dark")
fig.show()
fig = px.scatter(air, x='last_review', y='price', range_x=['2017-01-01','2017-12-31'],color="room_type",facet_col="neighbourhood_group",template="plotly_dark")
fig.show()
fig = px.scatter(air, x='last_review', y='price', range_x=['2016-01-01','2016-12-31'],color="room_type",facet_col="neighbourhood_group",template="plotly_dark")
fig.show()
```

Checking for top five number of listing based on host name

In [30]:
```
grouped = air.groupby('host_name').sum().reset_index()
grouped.sort_values('calculated_host_listings_count', ascending=False).head(5)
```

Out[30]:

	host_name	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	avai
9781	Sonder (NYC)	13316.25823	-24198.18856	82795	4353	1281	562.346572	106929	
1356	Blueground	9451.60418	-17166.13165	70331	7470	29	286.177172	53824	
5336	Kara	5827.31420	-10579.90583	36723	3839	324	130.170600	14679	
5471	Kazuya	4197.55601	-7612.96008	4514	3090	87	94.087514	10609	
9780	Sonder	3909.55849	-7103.58856	20451	2784	43	100.655836	9216	

In [31]:
```
air["name_length"]=air['name'].map(str).apply(len)
```

Removing unwanted features for the module

In [32]:
```
air.drop(['host_name','name','last_review','latitude','longitude'], axis = 1,inplace=True)
```

Correlation between variables

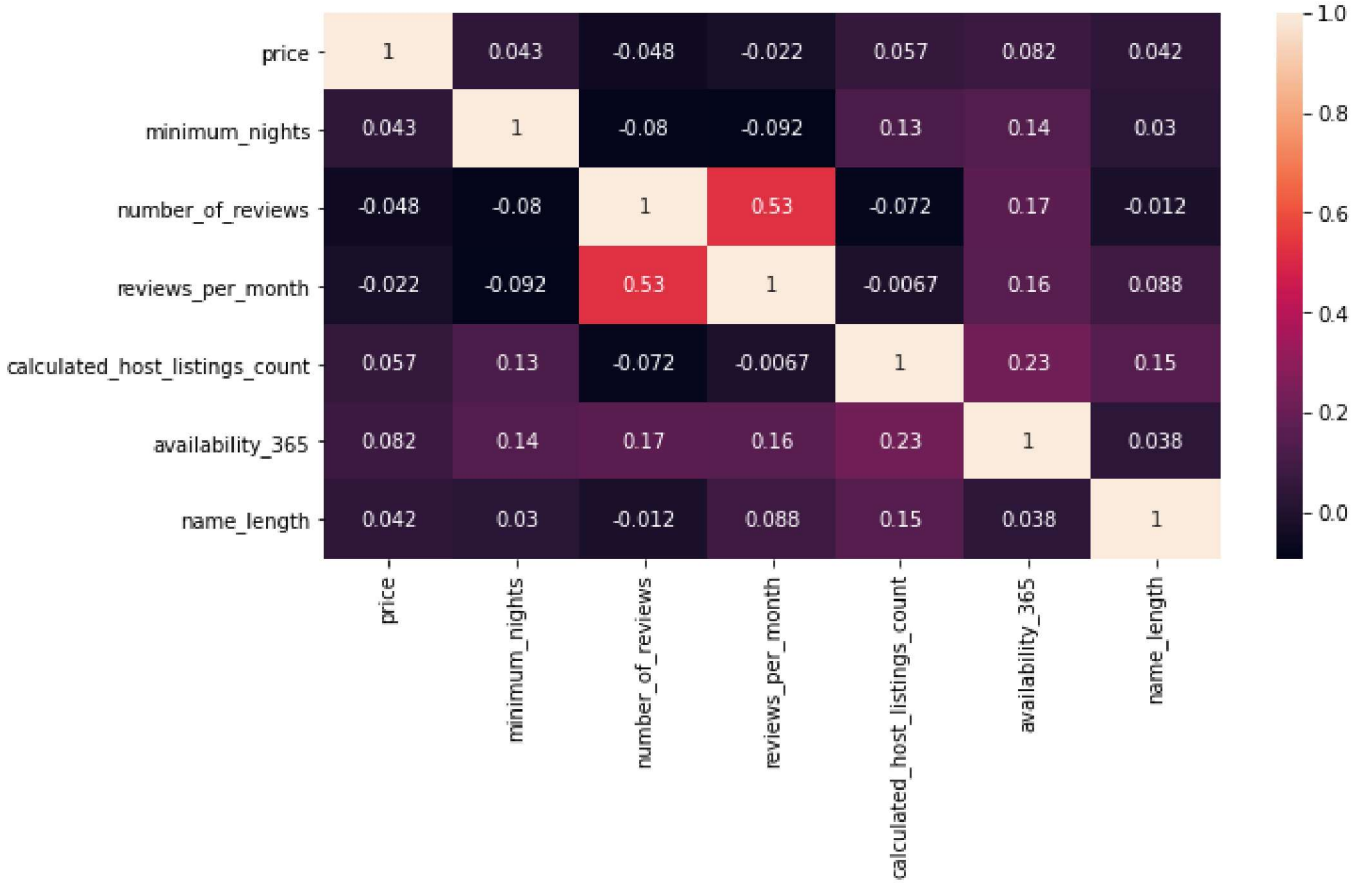
In [33]:
```
corre=air.corr()
air.corr()
```

Out[33]:

	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
price	1.000000	0.042799	-0.047954	-0.022373	0.057472	0.081829
minimum_nights	0.042799	1.000000	-0.080116	-0.091942	0.127960	0.144303
number_of_reviews	-0.047954	-0.080116	1.000000	0.530093	-0.072376	0.172028
reviews_per_month	-0.022373	-0.091942	0.530093	1.000000	-0.006701	0.162980
calculated_host_listings_count	0.057472	0.127960	-0.072376	-0.006701	1.000000	0.225701
availability_365	0.081829	0.144303	0.172028	0.162980	0.225701	1.000000
name_length	0.042219	0.029754	-0.012488	0.087683	0.152267	0.037850

In [34]:
```
plt.figure(figsize=(10,5))
sns.heatmap(corre,annot=True)
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x7f44b06caef0>



In [35]:
```
## Creating dummy variable of categorical variables for creating regression model
air_1 = pd.get_dummies(air, columns=['neighbourhood_group',"neighbourhood","room_type"], prefix = ['NG',"N","R"],drop_
first=True)
```

In [36]:
```
## making sure that the columns are according to the required number
air_1.shape
```

Out[36]: (48895, 233)

In [37]:
```
## Separating target column
x= air_1.loc[:, air_1.columns != 'price']
y= air_1["price"]
```

In [38]:
```
x.shape
```

Out[38]: (48895, 232)

In [39]:
```
y.shape
```

Out[39]: (48895,)

Train-test split

In [40]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=200)
x_train.head()
```

Out[40]:

	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365	name_length	NG_Brooklyn	NG_M
7625	1	24	0.470000	2	80	35	1	
22341	2	18	0.650000	1	95	34	0	
15704	1	0	1.373221	1	0	24	0	
10803	30	1	0.040000	121	273	42	0	
2759	4	7	0.100000	1	0	35	0	

5 rows × 232 columns

Linear regression model

In [41]:
```
LR = LinearRegression().fit(x_train, y_train)
```

In [42]:
```
y_pred = LR.predict(x_test)
```

In [43]:
```
print('Coefficients: \n', LR.coef_)
```

```
Coefficients:
 [-7.33459020e-02 -3.26782992e-01  2.00431985e+00 -2.25484048e-01
  1.94032339e-01  4.19606415e-01 -4.10497122e+10  1.72804980e+11
 -1.94551773e+10 -1.30031110e+10  1.30031109e+10  1.30031110e+10
  1.94551774e+10  1.94551774e+10  4.10497122e+10 -1.72804979e+11
  4.10497122e+10  1.94551774e+10  1.30031110e+10  3.63542380e+01
  1.94551774e+10  1.94551773e+10  4.10497123e+10  1.94551774e+10
  1.94551773e+10  9.24982274e+00  4.10497122e+10  4.10497122e+10
  4.10497123e+10  4.10497122e+10  1.94551775e+10  1.94551774e+10
  4.10497123e+10 -4.07155399e+01  4.10497123e+10  4.10497122e+10
  1.30031110e+10  4.10497122e+10  1.94551773e+10  4.10497122e+10
  4.10497123e+10 -4.68576078e+01  1.30031110e+10 -1.72804979e+11
 -1.72804979e+11  1.24353033e+02 -1.72804979e+11  3.59867096e-01
  7.76497364e+00  1.30031110e+10  4.10497123e+10 -6.32804607e+09
  4.10497123e+10  1.94551773e+10  4.10497123e+10  1.30031110e+10
 -9.09781744e+00 -4.33417797e-01  4.10497123e+10  1.94551773e+10
  4.10497123e+10  4.10497123e+10  4.10497123e+10  1.94551773e+10
  1.30031110e+10  1.94551773e+10  4.10497123e+10  4.10497122e+10
  1.94551773e+10  4.10497122e+10 -1.72804979e+11  5.96408844e-01
  4.10497122e+10 -1.72804979e+11 -2.45698185e+01 -2.68494854e+01
  1.94551773e+10  1.94551773e+10  1.30031109e+10  1.30031110e+10
  1.94551774e+10 -4.71465111e+00 -1.72804979e+11  4.10497122e+10
 -1.72804979e+11  4.10497122e+10  1.94551774e+10  9.67915058e-01
  1.94551774e+10  4.10497123e+10  4.10497122e+10  4.31169334e+07
  1.94551773e+10  1.94551773e+10  4.10497123e+10 -1.72804979e+11
  1.30031109e+10  1.30031109e+10  4.10497122e+10  1.30031110e+10
  4.10497123e+10 -1.72804979e+11  1.30031110e+10 -1.72804979e+11
 -1.72804979e+11  3.85530424e+00  1.94551773e+10  1.94551774e+10
  1.94551773e+10  1.30031110e+10  1.30031109e+10 -2.64866695e+01
 -1.72804980e+11  1.94551773e+10  1.94551773e+10  1.94551774e+10
  1.94551774e+10  4.10497122e+10  1.94551773e+10  1.94551773e+10
 -2.31660366e-01 -1.72804979e+11  1.94551773e+10  1.30031110e+10
 -1.72804979e+11  1.94551774e+10  1.94551774e+10  1.53862834e+01
 -1.72804979e+11  4.10497122e+10 -1.72804980e+11  1.30031110e+10
  1.94551773e+10 -2.18473701e+01  1.94551773e+10  1.30031109e+10
 -1.72804979e+11  4.10497122e+10  4.10497122e+10 -1.72804979e+11
  4.95265579e+00  2.61718249e+00  1.84215877e+01 -7.73505592e+00
  1.94751240e+01 -1.73155708e+01 -1.72804979e+11  4.10497123e+10
  1.94551774e+10  1.30031110e+10  1.30031109e+10  1.30031109e+10
  1.30031110e+10 -1.72804979e+11 -1.72804979e+11  1.44842529e+00
  1.04258840e+01  1.30031110e+10 -9.58752441e+00  1.94551773e+10
```

```
 4.10497123e+10 -5.79373360e-01 -1.74995747e+01 -1.26965134e+01
 7.98090720e+00  1.30031110e+10  1.30031113e+10  4.10497123e+10
 4.10497123e+10  1.94551773e+10  1.30031109e+10  4.10497123e+10
 1.94551773e+10  1.94551773e+10  2.38418579e-07  1.94551773e+10
 5.08800603e+02  1.94551773e+10 -1.72804979e+11  1.30031109e+10
 1.94551773e+10  1.30031109e+10 -1.65032043e+01  4.10497126e+10
 4.10497123e+10  1.30031111e+10  1.30031110e+10 -1.72804979e+11
-3.54664202e+01  1.30031110e+10  1.94551773e+10  4.10497123e+10
 1.94551773e+10  2.69624138e+01  1.94551773e+10  1.30031110e+10
 1.30031110e+10 -1.72804979e+11  1.94551773e+10  4.10497123e+10
-1.72804979e+11 -6.87346745e+00  1.30031109e+10  1.30031110e+10
 1.30031110e+10  5.53803837e+00 -1.72804979e+11 -1.72804979e+11
-3.56582404e+01  1.39191775e+01 -1.72804979e+11 -1.72804979e+11
 2.97803842e+01  4.10497123e+10 -9.38843608e+00 -1.72804980e+11
 1.30031110e+10  1.91949933e+01 -1.72804979e+11  5.37095814e+01
 1.30031110e+10  1.94551774e+10 -7.45364904e+00  4.10497123e+10
 1.30031110e+10  4.10497123e+10  1.94551773e+10 -2.35628915e+00
 0.00000000e+00  1.94551773e+10 -9.80383759e+01 -1.28567581e+02]
```

In [44]:
```
print('Mean squared error:{:.5f}'.format(mean_squared_error(y_test, y_pred)))
```

```
Mean squared error:39962614951181760.00000
```

In [45]:
```
print('Coefficient of determination:{:.5f}'.format( r2_score(y_test, y_pred)))
```

```
Coefficient of determination:-664107546751.53162
```

Ridge regression

In [46]:
```
Rg= Ridge()
Rg.fit(x_train, y_train)

y_pred_R=Rg.predict(x_test)
```

In [47]:
```
print('Coefficients: \n', Rg.coef_)
```

```
Coefficients:
 [-7.27308322e-02 -3.26520199e-01  1.99014197e+00 -2.24712172e-01
  1.93717540e-01  4.19195192e-01  3.04115024e+01  1.00897364e+02
  8.30733187e+00 -1.68561071e+01 -3.41139731e+01  3.82819545e+00
  2.86618721e+01  2.40024707e+01 -5.50943625e+01  1.55897431e+02
 -2.61470947e+01  2.84278685e+01  1.49300045e+01  1.43757789e+01
  6.90584088e+01 -2.54906771e+01 -1.09194723e+01  3.04028685e+01
 -7.09955820e+00  1.60993031e+00 -4.81367749e+01 -6.10850782e+01
  2.34632543e+01 -4.34900648e+01  9.37980611e+01  4.57312889e+00
  1.49084009e+01 -4.39052569e+01  5.69653137e+01 -2.30914613e+01
  1.23035430e+00 -2.07535898e+01  1.64081746e+00 -4.45876292e+01
  3.08609800e+01 -4.53562971e+01  7.56833028e+00  3.09714189e+01
 -4.16022322e+01  1.07041034e+02 -1.08085724e+01 -6.88034488e+00
  1.99830900e-01 -1.21082546e+01  4.33930907e+01  0.00000000e+00
  7.50310728e+01 -3.82605985e+01 -1.40009215e+01 -1.37922197e+01
 -1.61874072e+01 -7.69295193e+00 -1.81997276e+01 -4.07663147e+01
 -1.48661704e+01  1.66500263e+01  5.42427276e+01 -4.04297993e+00
 -1.89739025e+01 -2.66376195e+01  2.34426387e+01 -4.47379724e+01
 -4.88266385e+00 -3.83570965e+01 -5.31586012e+01 -6.17036647e+00
 -4.53730763e+01 -1.98847507e+01 -2.81051043e+01 -3.15296322e+01
 -1.93498723e+01 -1.98617095e+00 -3.99480904e+01 -8.46709500e+00
  8.55700232e+01 -1.11954634e+01 -5.37235094e+00 -2.83385131e+01
  1.16345790e+02 -2.05754463e+01  7.00188426e+00 -6.47316616e+00
  1.21830322e+01  1.29037446e+01 -4.51725780e+01  0.00000000e+00
  5.58872752e-01 -2.91037958e+01  2.50272574e+01  1.42389325e+01
 -3.25403364e+01 -5.94246715e+01 -4.46886431e+01  1.58800227e+01
  1.93262591e+01  4.42255933e+01  2.80032421e+01 -6.93526421e+01
 -9.07129906e+00 -3.55237841e+00 -1.25784607e+01  5.78085570e+01
 -3.83087934e+01 -1.58059993e+01 -4.55229948e+01 -3.14025504e+01
 -9.80892045e+01 -1.39216675e+01 -1.01751679e+01  6.40750634e+01
  5.75155341e+00 -2.80871735e+01 -1.46852106e+01 -2.16697188e+01
 -7.64471167e+00 -1.82219421e+01 -3.33233066e+01  7.22657032e+00
 -2.52064435e+01  1.34910361e+01  2.58755193e+01  7.66339111e+00
 -1.95788958e+01 -5.21193306e+01 -9.07718784e+01  8.23044475e+00
 -2.57341775e+01 -2.52167548e+01 -3.41905825e+01 -4.13035013e+01
  5.16443393e+01 -4.69199314e+01 -3.86440987e+01 -6.64402668e+01
 -2.42082783e+00 -4.51652880e+00  1.01154408e+01 -1.49926211e+01
  9.90724469e+00 -2.34332719e+01 -1.34713287e+01  5.32871117e+00
  6.40139943e+01  2.58431485e+01 -2.76333453e+01 -3.25044725e+01
 -5.75207824e+00  5.95314935e+01 -3.07587747e+00 -5.45686755e+00
  2.71450499e+00  1.32810605e+00 -8.63741324e+00 -2.88231234e+01
```

```
 3.48667708e+01 -7.89639437e+00 -2.32764056e+01 -1.92489713e+01
 3.66142661e-01  9.78967431e+00  2.35971350e+02  6.07333006e+01
-9.69639622e+00 -2.68595089e+01 -3.28144357e+01 -9.43094785e+00
-1.30228520e+01 -2.26004167e+01  0.00000000e+00 -1.47453600e+01
 4.17674149e+02 -5.07060132e+00 -4.58427860e+01 -3.23907583e+01
-2.44977324e+01 -2.43060010e+01 -2.16450869e+01  2.97679182e+02
-4.17912307e+00  5.78659901e+01  2.74923965e+01  5.03709747e+01
-3.76369158e+01  6.52887266e+00 -2.82033489e+01  8.08574086e+00
-1.32706306e+01  1.45311608e+01 -8.80811120e+00  2.01177882e+01
 1.90527124e+01  6.43578721e-01 -1.38591877e+01 -7.19932528e+00
 3.51325986e+01 -1.37035134e+01 -3.19314727e+01 -4.82116284e+00
-4.55067227e+00 -1.77194554e+00  2.88988736e+02 -5.46419286e+01
-3.59926833e+01  5.91588778e+00 -3.03389252e+01 -1.03151634e+01
 1.99699657e+01  6.02651515e+01 -1.63599334e+01 -9.69832171e+01
-7.39894213e+00  5.81715110e+00  3.51347834e+01  4.10236664e+01
-1.28198084e+01  2.34045976e+01 -1.45365115e+01  2.13199171e+01
 3.11808781e+01 -1.01900383e+01 -2.72279960e+01 -9.03703540e+00
 0.00000000e+00 -2.79609216e+00 -9.80688385e+01 -1.28499530e+02]
```

In [48]:
```
print('Mean squared error: {:.5f}'.format(mean_squared_error(y_test, y_pred_R)))
```

```
Mean squared error: 53649.20341
```

In [49]:
```
print('Coefficient of determination:{:.5f}'.format(r2_score(y_test, y_pred_R)))
```

```
Coefficient of determination:0.10845
```

Lasso regression

In [50]:
```
Lo = Lasso(alpha=10,max_iter = 10000)
Lo.fit(x_train, y_train)
y_pred_L=Lo.predict(x_test)
```

In [51]:
```
print('Coefficients: \n', Lo.coef_)
```

```
Coefficients:
 [ 6.90532739e-02 -3.04373058e-01 -0.00000000e+00  4.01906117e-02
  1.59967179e-01  6.34026817e-01 -0.00000000e+00  2.32864078e+01
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
 -0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00  0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00  0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00  0.00000000e+00 -0.00000000e+00  0.00000000e+00
 -0.00000000e+00 -0.00000000e+00  0.00000000e+00  0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00  0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00  0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
```

```
 0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
-0.00000000e+00 -0.00000000e+00  0.00000000e+00  0.00000000e+00
-0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
-0.00000000e+00 -0.00000000e+00  0.00000000e+00 -0.00000000e+00
 0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
-0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
-0.00000000e+00  0.00000000e+00 -0.00000000e+00  0.00000000e+00
-0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
-0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
-0.00000000e+00  0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
-0.00000000e+00 -0.00000000e+00  0.00000000e+00 -0.00000000e+00
-0.00000000e+00 -0.00000000e+00  0.00000000e+00  0.00000000e+00
-0.00000000e+00  0.00000000e+00 -0.00000000e+00 -0.00000000e+00
-0.00000000e+00 -0.00000000e+00  0.00000000e+00  0.00000000e+00
-0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
 0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 0.00000000e+00 -0.00000000e+00 -6.85250812e+01 -0.00000000e+00]
```

In [52]:
```
print('Mean squared error: {:.5f}'.format(mean_squared_error(y_test, y_pred_L)))
```

```
Mean squared error: 56027.12505
```

In [53]:
```
print('Coefficient of determination: {:.5f}'.format(r2_score(y_test, y_pred_L)))
```

```
Coefficient of determination: 0.06893
```

Cross validation score

In [54]:
```
print('Cross validation score of linear regression ={:.5f} '.format(cross_val_score(LR, x, y, cv = 10).mean()))
print('Cross validation score of ridge regression = {:.5f}'.format(cross_val_score(Rg, x, y, cv = 10).mean()))
print('Cross validation score of lasso regression = {:.5f}'.format(cross_val_score(Lo, x, y, cv = 10).mean()))
```

```
Cross validation score of linear regression =-492005747090.12537
Cross validation score of ridge regression = 0.10921
Cross validation score of lasso regression = 0.07121
```

Citation

- Yashvi Patel(Kaggle)
- Amit Jadhav (Kaggle)
- Scikit-learn
- Seaborn documentation
- Plotly documentation

In [54]: