```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
#to load dataset
df=pd.read_csv("AB_NYC_2019.csv")
df.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	l
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	4
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	4
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	4
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	4
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	4

```
df.shape
```

```
(48895, 16)
```

```
#drop unwanted columns
df.drop(["id","name","host_id","host_name","calculated_host_listings_count"],axis=1,inplac
df.head()
```

	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimu
0	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	

```
#check null values
df.isnull().sum()
```

```
neighbourhood_group        0
neighbourhood              0
latitude                   0
longitude                  0
room_type                  0
price                      0
minimum_nights             0
number_of_reviews          0
last_review            10052
reviews_per_month      10052
availability_365           0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   neighbourhood_group  48895 non-null  object 
 1   neighbourhood        48895 non-null  object 
 2   latitude             48895 non-null  float64
 3   longitude            48895 non-null  float64
 4   room_type            48895 non-null  object 
 5   price                48895 non-null  int64  
 6   minimum_nights       48895 non-null  int64  
 7   number_of_reviews    48895 non-null  int64  
 8   last_review          38843 non-null  object 
 9   reviews_per_month    38843 non-null  float64
 10  availability_365     48895 non-null  int64  
dtypes: float64(3), int64(4), object(4)
memory usage: 4.1+ MB
```

```
#check the percentage of null vallues
df.isnull().sum()/df.shape[0]*100
```

```
neighbourhood_group     0.000000
neighbourhood           0.000000
latitude                0.000000
longitude               0.000000
room_type               0.000000
price                   0.000000
minimum_nights          0.000000
number_of_reviews       0.000000
last_review            20.558339
reviews_per_month      20.558339
availability_365        0.000000
dtype: float64
```

```
#drop the last_review table
df.drop("last_review",axis=1,inplace=True)
#check
df.isnull().sum()
```

```
neighbourhood_group        0
neighbourhood              0
latitude                   0
longitude                  0
room_type                  0
price                      0
minimum_nights             0
number_of_reviews          0
reviews_per_month      10052
availability_365           0
dtype: int64
```

```
#fill null values with mean
m=df["reviews_per_month"].mean()
print("Mean of reviews_per_month",m)
df["reviews_per_month"].fillna(m,inplace=True)
```

```
Mean of reviews_per_month 1.3732214298586884
```

```
#check
df.isnull().sum()
```

```
neighbourhood_group    0
neighbourhood          0
latitude               0
longitude              0
room_type              0
price                  0
minimum_nights         0
number_of_reviews      0
reviews_per_month      0
availability_365       0
dtype: int64
```

```
#visualize
sns.heatmap(df.isnull())
plt.show()
```


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   neighbourhood_group  48895 non-null  object 
 1   neighbourhood        48895 non-null  object 
 2   latitude             48895 non-null  float64
 3   longitude            48895 non-null  float64
 4   room_type            48895 non-null  object 
 5   price                48895 non-null  int64  
 6   minimum_nights       48895 non-null  int64  
 7   number_of_reviews    48895 non-null  int64  
 8   reviews_per_month    48895 non-null  float64
 9   availability_365     48895 non-null  int64  
dtypes: float64(3), int64(4), object(3)
memory usage: 3.7+ MB
```

```
#seperate the object and numerical value
df_num=df.select_dtypes(['int64','float64'])
df_cat=df.select_dtypes(object)
```

```
#applylabel encoder to categorical value
from sklearn.preprocessing import LabelEncoder
```

```
for col in df_cat:
  le=LabelEncoder()
  df_cat[col]=le.fit_transform(df_cat[col])
```

```
df_cat
```

	neighbourhood_group	neighbourhood	room_type
0	1	108	1
1	2	127	0
2	2	94	1
3	1	41	0
4	2	61	0

```
#concat df_cat & df_num
df_new=pd.concat([df_num,df_cat],axis=1)
df_new.head()
```

	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month
0	40.64749	-73.97237	149	1	9	0.210000
1	40.75362	-73.98377	225	1	45	0.380000
2	40.80902	-73.94190	150	3	0	1.373221
3	40.68514	-73.95976	89	1	270	4.640000
4	40.79851	-73.94399	80	10	9	0.100000

```
#seperate input and output variable
X=df_new.drop("price",axis=1)
Y=df_new["price"]
```

```
#use train test split
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=1)
```

```
#apply sclaing oninput data
from sklearn.preprocessing import MinMaxScaler
#create object
ms=MinMaxScaler()
X_train=ms.fit_transform(X_train)
X_test=ms.transform(X_test)
```

```
#create neural network
import tensorflow as tf
#create object
model=tf.keras.Sequential([
      tf.keras.layers.Dense(128,activation='relu',input_shape=(X.shape[1],)),   #hidden la
      tf.keras.layers.Dense(64,activation='relu'),        #hidden layer 2
      tf.keras.layers.Dense(1)
])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 128)               1280      
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256      
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65        
=================================================================
Total params: 9,601
Trainable params: 9,601
Non-trainable params: 0
_________________________________________________________________
```

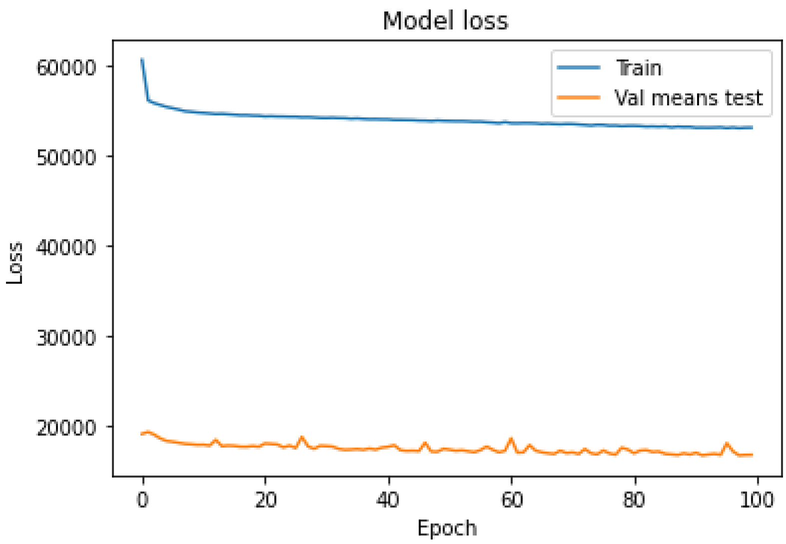
```
#compile model
model.compile(optimizer='adam',loss='mse',metrics=['mae'])
```

```
#train model
trained_model=model.fit(X_train,Y_train,epochs=100,validation_split=0.1)
```

```
Epoch 72/100
963/963 [==============================] - 1s 1ms/step - loss: 53531.1055 - mae:
Epoch 73/100
963/963 [==============================] - 1s 1ms/step - loss: 53479.6172 - mae:
Epoch 74/100
963/963 [==============================] - 1s 1ms/step - loss: 53423.9883 - mae:
Epoch 75/100
963/963 [==============================] - 1s 1ms/step - loss: 53495.8594 - mae:
Epoch 76/100
963/963 [==============================] - 1s 1ms/step - loss: 53477.6680 - mae:
Epoch 77/100
963/963 [==============================] - 1s 1ms/step - loss: 53405.4453 - mae:
Epoch 78/100
963/963 [==============================] - 1s 1ms/step - loss: 53415.0664 - mae:
Epoch 79/100
963/963 [==============================] - 1s 1ms/step - loss: 53353.0469 - mae:

Epoch 80/100
963/963 [==============================] - 1s 1ms/step - loss: 53387.0586 - mae:
Epoch 81/100
963/963 [==============================] - 1s 1ms/step - loss: 53391.5000 - mae:
Epoch 82/100
963/963 [==============================] - 1s 1ms/step - loss: 53357.8672 - mae:
Epoch 83/100
963/963 [==============================] - 1s 1ms/step - loss: 53294.0352 - mae:
Epoch 84/100
963/963 [==============================] - 1s 1ms/step - loss: 53318.4609 - mae:
Epoch 85/100
963/963 [==============================] - 1s 1ms/step - loss: 53280.9375 - mae:
Epoch 86/100
963/963 [==============================] - 1s 1ms/step - loss: 53326.4883 - mae:
Epoch 87/100
963/963 [==============================] - 1s 1ms/step - loss: 53215.4648 - mae:
Epoch 88/100
963/963 [==============================] - 1s 1ms/step - loss: 53295.0352 - mae:
Epoch 89/100
963/963 [==============================] - 1s 1ms/step - loss: 53255.2344 - mae:
Epoch 90/100
```

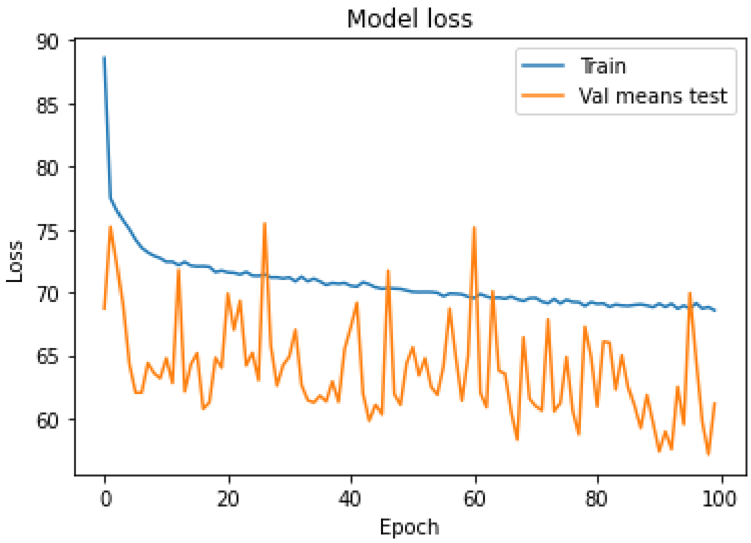
```
Epoch 90/100
963/963 [==============================] - 1s 1ms/step - loss: 53260.9219 - mae:
Epoch 91/100
963/963 [==============================] - 1s 1ms/step - loss: 53176.8672 - mae:
Epoch 92/100
963/963 [==============================] - 1s 1ms/step - loss: 53187.9766 - mae:
Epoch 93/100
963/963 [==============================] - 1s 1ms/step - loss: 53177.0039 - mae:
Epoch 94/100
963/963 [==============================] - 1s 1ms/step - loss: 53192.3164 - mae:
Epoch 95/100
963/963 [==============================] - 1s 1ms/step - loss: 53218.0898 - mae:
Epoch 96/100
963/963 [==============================] - 1s 1ms/step - loss: 53137.5117 - mae:
Epoch 97/100
963/963 [==============================] - 1s 1ms/step - loss: 53201.5273 - mae:
Epoch 98/100
963/963 [==============================] - 1s 1ms/step - loss: 53124.8672 - mae:
Epoch 99/100
963/963 [==============================] - 1s 1ms/step - loss: 53173.9453 - mae:
Epoch 100/100
963/963 [==============================] - 1s 1ms/step - loss: 53182.9883 - mae:
```

```
#visualise training error and testing error
plt.plot(trained_model.history['loss']) #training's loss means error
plt.plot(trained_model.history['val_loss']) #testing's loss means error
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val means test'], loc='upper right')#loc means location
plt.show()
```




```
#visualise training error and testing error
plt.plot(trained_model.history['mae']) #training's loss means error
plt.plot(trained_model.history['val_mae']) #testing's loss means error
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val means test'], loc='upper right')#loc means location
plt.show()
```

```
plt.show()
```

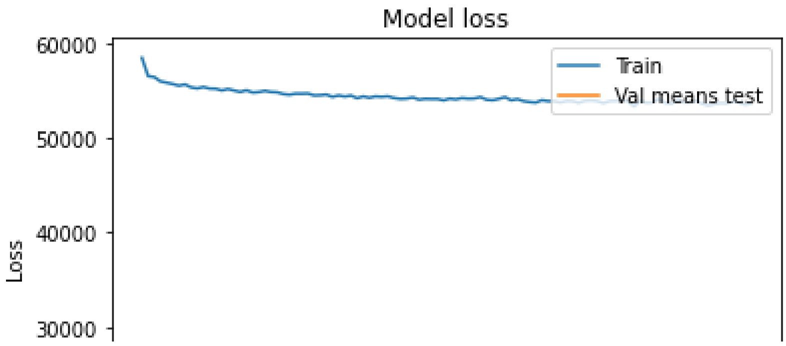



```
from keras.layers import Dropout
from keras import regularizers


model_2 = tf.keras.Sequential([
tf.keras.layers.Dense(128, activation='relu', kernel_regularizer=regularizers.l1(0.01), ir
    Dropout(0.3),
    tf.keras.layers.Dense(128, activation='relu', kernel_regularizer=regularizers.l1(0.01)
    Dropout(0.3),
    tf.keras.layers.Dense(128, activation='relu', kernel_regularizer=regularizers.l1(0.01)
    Dropout(0.3),
    tf.keras.layers.Dense(128, activation='relu', kernel_regularizer=regularizers.l1(0.01)
    Dropout(0.3),
    tf.keras.layers.Dense(1, kernel_regularizer=regularizers.l2(0.01))
])


#compile model
model_2.compile(optimizer='adam',loss='mse',metrics=['mae'])


#train model
trained_model1=model_2.fit(X_train,Y_train,epochs=100,validation_split=0.1)
```

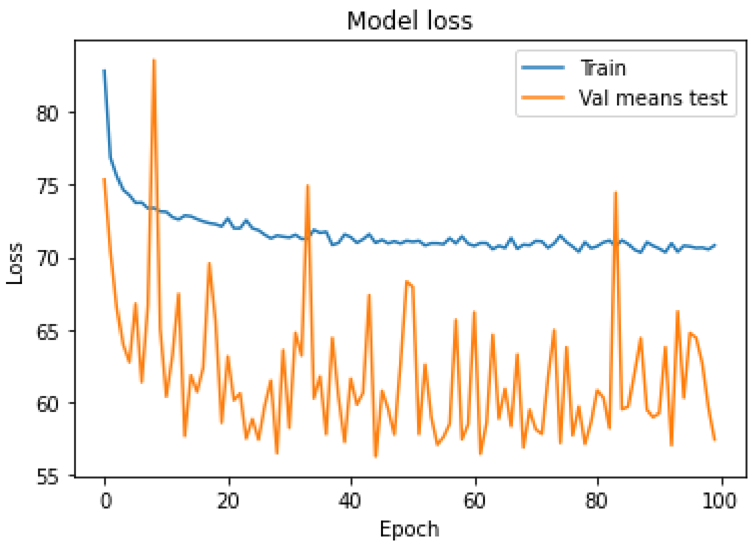
```
963/963 [==============================] - 2s 2ms/step - loss: 53649.3672 - mae:
Epoch 73/100
963/963 [==============================] - 2s 2ms/step - loss: 53875.7656 - mae:
Epoch 74/100
963/963 [==============================] - 2s 2ms/step - loss: 53912.0234 - mae:
Epoch 75/100
963/963 [==============================] - 2s 2ms/step - loss: 53843.2383 - mae:
Epoch 76/100
963/963 [==============================] - 2s 2ms/step - loss: 53614.8203 - mae:
Epoch 77/100
963/963 [==============================] - 2s 2ms/step - loss: 53835.1836 - mae:
Epoch 78/100
963/963 [==============================] - 2s 2ms/step - loss: 53855.6641 - mae:
Epoch 79/100
963/963 [==============================] - 2s 2ms/step - loss: 53951.5469 - mae:
Epoch 80/100
963/963 [==============================] - 2s 2ms/step - loss: 53799.7148 - mae:
```

```
Epoch 81/100
963/963 [==============================] - 2s 2ms/step - loss: 53277.3984 - mae:
Epoch 82/100
963/963 [==============================] - 2s 2ms/step - loss: 53873.2266 - mae:
Epoch 83/100
963/963 [==============================] - 2s 2ms/step - loss: 53665.2070 - mae:
Epoch 84/100
963/963 [==============================] - 2s 2ms/step - loss: 53697.4727 - mae:
Epoch 85/100
963/963 [==============================] - 2s 2ms/step - loss: 54044.9492 - mae:
Epoch 86/100
963/963 [==============================] - 2s 2ms/step - loss: 53646.2930 - mae:
Epoch 87/100
963/963 [==============================] - 2s 2ms/step - loss: 53554.1797 - mae:
Epoch 88/100
963/963 [==============================] - 2s 2ms/step - loss: 54020.0195 - mae:
Epoch 89/100
963/963 [==============================] - 2s 2ms/step - loss: 53710.9766 - mae:
Epoch 90/100
963/963 [==============================] - 2s 2ms/step - loss: 53684.2891 - mae:
Epoch 91/100
963/963 [==============================] - 2s 2ms/step - loss: 53885.3438 - mae:
Epoch 92/100
963/963 [==============================] - 2s 2ms/step - loss: 53560.2188 - mae:
Epoch 93/100
963/963 [==============================] - 2s 2ms/step - loss: 53377.5000 - mae:
Epoch 94/100
963/963 [==============================] - 2s 2ms/step - loss: 53515.1328 - mae:
Epoch 95/100
963/963 [==============================] - 2s 2ms/step - loss: 53635.8711 - mae:
Epoch 96/100
963/963 [==============================] - 2s 2ms/step - loss: 53596.1172 - mae:
Epoch 97/100
963/963 [==============================] - 2s 2ms/step - loss: 53792.0625 - mae:
Epoch 98/100
963/963 [==============================] - 2s 2ms/step - loss: 53698.3008 - mae:
Epoch 99/100
963/963 [==============================] - 2s 2ms/step - loss: 53449.9180 - mae:


Epoch 100/100
963/963 [==============================] - 2s 2ms/step - loss: 53583.6758 - mae:
```

```
#visualise training error and testing error
plt.plot(trained_model1.history['loss']) #training's loss means error
plt.plot(trained_model1.history['val_loss']) #testing's loss means error
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val means test'], loc='upper right')#loc means location
plt.show()
```




```
#visualise training error and testing error
plt.plot(trained_model1.history['mae']) #training's loss means error
plt.plot(trained_model1.history['val_mae']) #testing's loss means error
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val means test'], loc='upper right')#loc means location
plt.show()
```




```
#test model
Y_pred=model_2.predict(X_test)
```

```
mse,mae =model_2.evaluate(X_test,Y_test)
print("Mean Squared Error : ",mse)
print("Mean Absolute Error : ",mae)

from sklearn.metrics import r2_score
print("R2-Score : ",r2_score(Y_test,Y_pred))
```

```
459/459 [==============================] - 0s 872us/step - loss: 52166.6953 - mae: 6
Mean Squared Error :  52166.6953125
Mean Absolute Error :  64.15260314941406
R2-Score :  0.1164465947110811
```

0s completed at 4:43 PM