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

df = pd.read_csv("D:/PYTHON/ML/housePrice.csv")

df.shape
```

```
(3479, 8)
```

```
df.head()
```

```
  Area  Room  Parking  Warehouse  Elevator         Address
Price  \
0   63     1     True       True      True         Shahran
1.850000e+09
1   60     1     True       True      True         Shahran
1.850000e+09
2   79     2     True       True      True          Pardis
5.500000e+08
3   95     2     True       True      True   Shahrake Qods
9.025000e+08
4  123     2     True       True      True  Shahrake Gharb
7.000000e+09

   Price(USD)
0    61666.67
1    61666.67
2    18333.33
3    30083.33
4   233333.33
```

```
df.tail()
```

```
     Area  Room  Parking  Warehouse  Elevator             Address  \
3474   86     2     True       True      True  Southern Janatabad
3475   83     2     True       True      True            Niavaran
3476   75     2    False      False     False              Parand
3477  105     2     True       True      True              Dorous
3478   82     2    False       True      True              Parand

             Price  Price(USD)
3474  3.500000e+09   116666.67
3475  6.800000e+09   226666.67
3476  3.650000e+08    12166.67
```

```
3477  5.600000e+09   186666.67
3478  3.600000e+08    12000.00
```

```
df.describe().transpose()
```

```
             count          mean           std        min
25%  \
Room        3479.0  2.079908e+00  7.582753e-01        0.0
2.000000e+00
Price       3479.0  5.359023e+09  8.099935e+09  3600000.0
1.418250e+09
Price(USD)  3479.0  1.786341e+05  2.699978e+05      120.0
4.727500e+04

                     50%           75%           max
Room        2.000000e+00  2.000000e+00  5.000000e+00
Price       2.900000e+09  6.000000e+09  9.240000e+10
Price(USD)  9.666667e+04  2.000000e+05  3.080000e+06
```

```
df.columns
```

```
Index(['Area', 'Room', 'Parking', 'Warehouse', 'Elevator', 'Address',
'Price',
       'Price(USD)'],
      dtype='object')
```

```
df.isnull().sum()
```

```
Area           0
Room           0
Parking        0
Warehouse      0
Elevator       0
Address       23
Price          0
Price(USD)     0
dtype: int64
```

```
df=df.dropna()
```

```
df.isnull().sum()
```

```
Area          0
Room          0
Parking       0
Warehouse     0
Elevator      0
Address       0
Price         0
Price(USD)    0
dtype: int64
```

```
df['Room'].unique()
```

```
array([1, 2, 3, 0, 4, 5], dtype=int64)
```

```
df[['Parking','Warehouse','Elevator']] =
df[['Parking','Warehouse','Elevator']].astype(int)
df.head()
```

```
  Area  Room  Parking  Warehouse  Elevator         Address
Price  \
0   63     1        1          1         1         Shahran
1.850000e+09
1   60     1        1          1         1         Shahran
1.850000e+09
2   79     2        1          1         1          Pardis
5.500000e+08
3   95     2        1          1         1   Shahrake Qods
9.025000e+08
4  123     2        1          1         1  Shahrake Gharb
7.000000e+09

   Price(USD)
0    61666.67
1    61666.67
2    18333.33
3    30083.33
4   233333.33
```

```
df.tail()
```

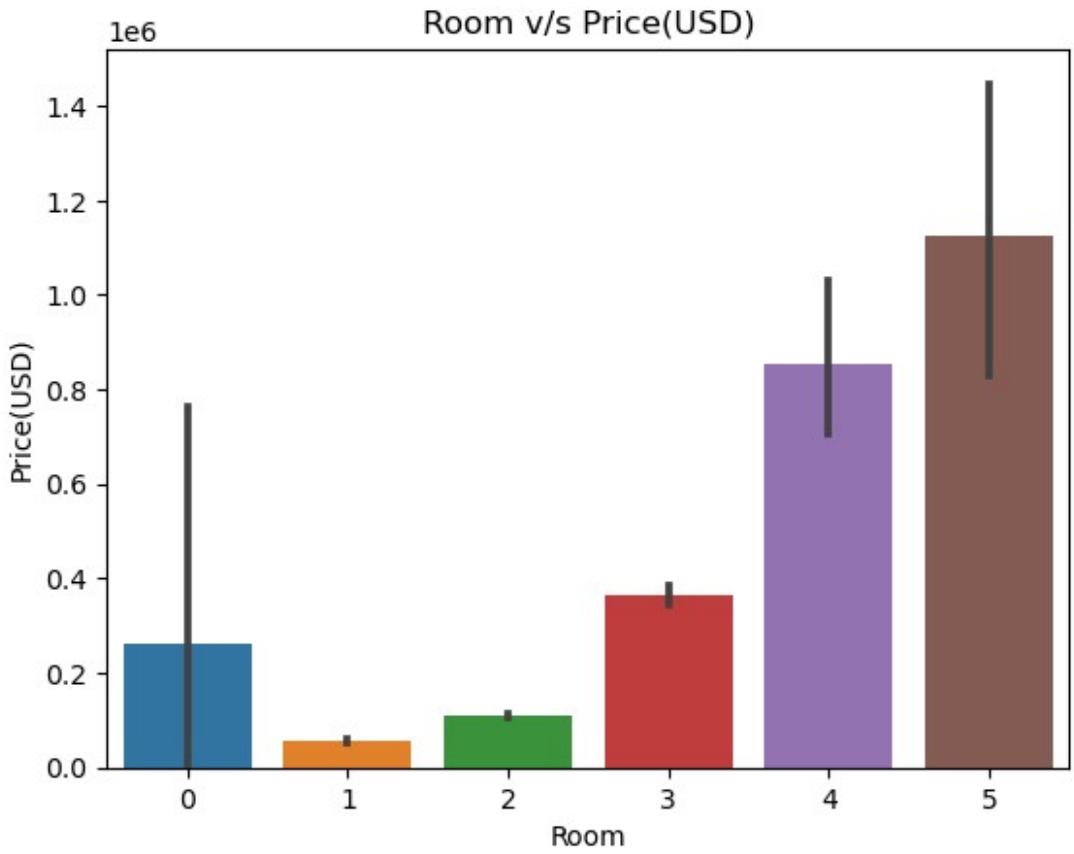
```
     Area  Room  Parking  Warehouse  Elevator             Address  \
3474   86     2        1          1         1  Southern Janatabad
3475   83     2        1          1         1            Niavaran
3476   75     2        0          0         0              Parand
3477  105     2        1          1         1              Dorous
3478   82     2        0          1         1              Parand

             Price  Price(USD)
3474  3.500000e+09   116666.67
3475  6.800000e+09   226666.67
3476  3.650000e+08    12166.67
3477  5.600000e+09   186666.67
3478  3.600000e+08    12000.00
```

```
plt.title('Room v/s Price(USD)')
```

```
sns.barplot(x="Room", y="Price(USD)", data=df)
```

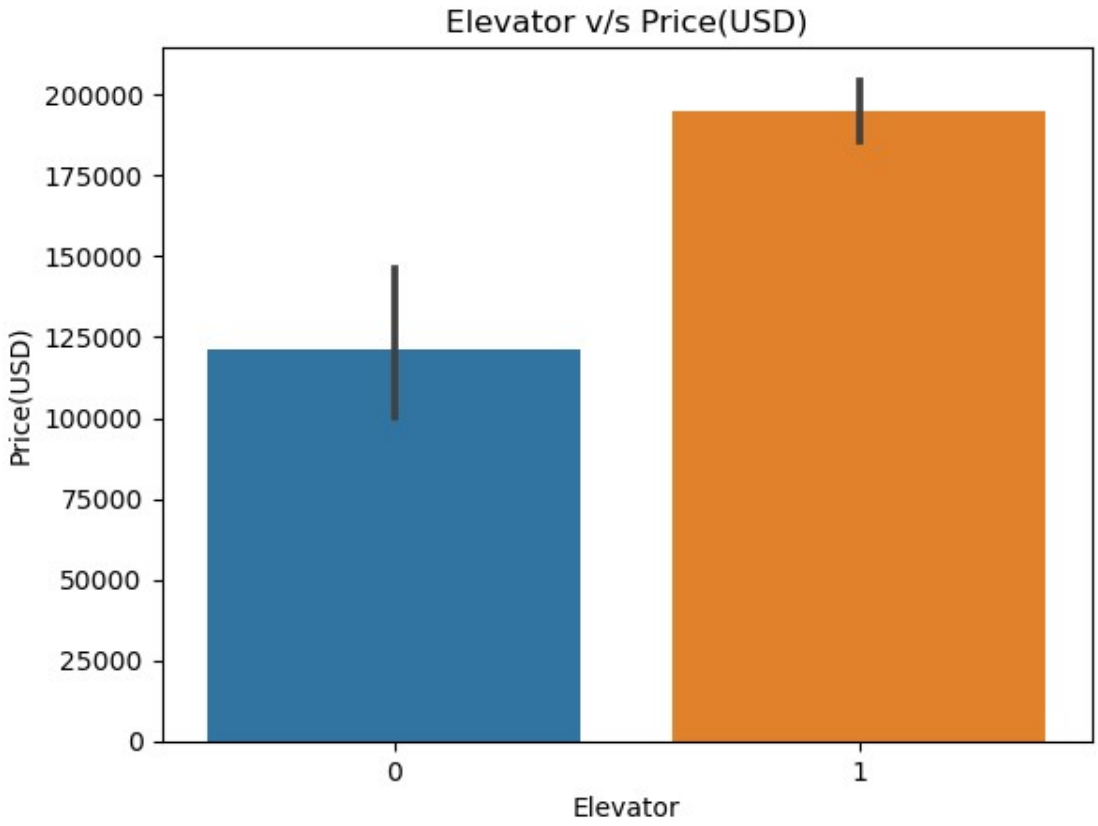
```
<AxesSubplot:title={'center':'Room v/s Price(USD)'}, xlabel='Room',
ylabel='Price(USD)'>
```




```
plt.title('Elevator v/s Price(USD)')

sns.barplot(x="Elevator", y="Price(USD)", data=df)
```

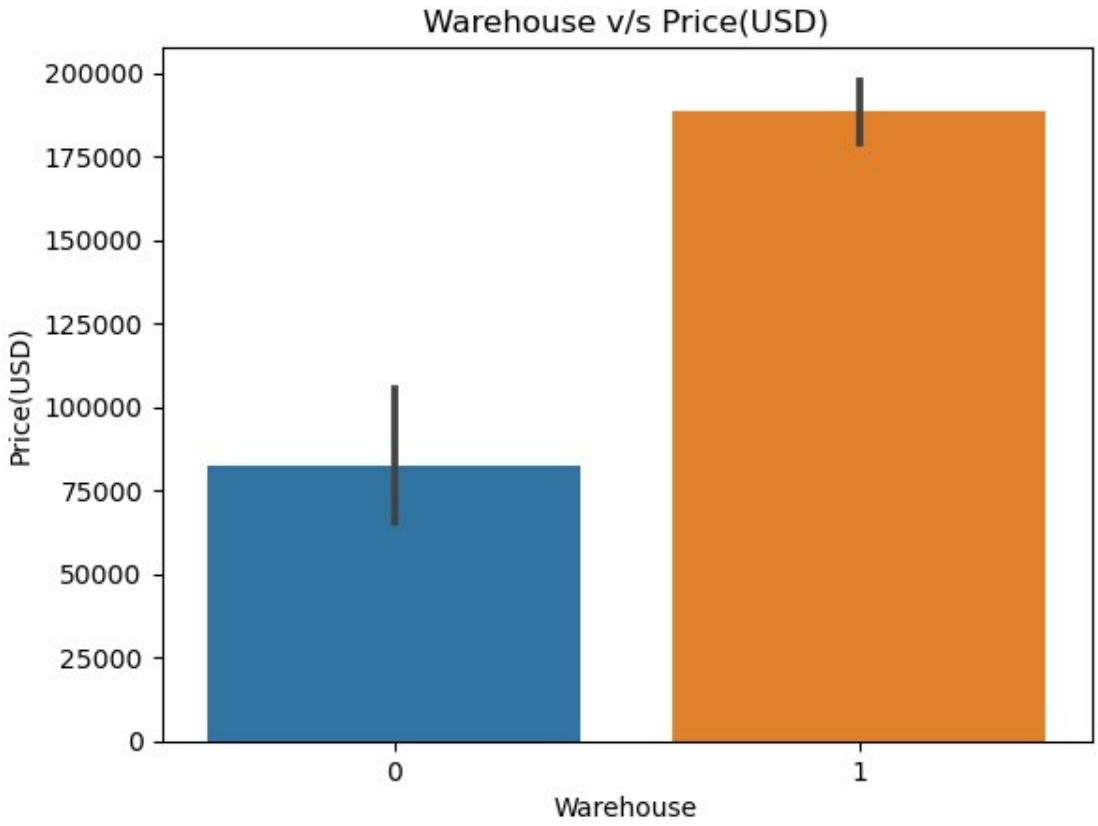
```
<AxesSubplot:title={'center':'Elevator v/s Price(USD)'},
xlabel='Elevator', ylabel='Price(USD)'>
```



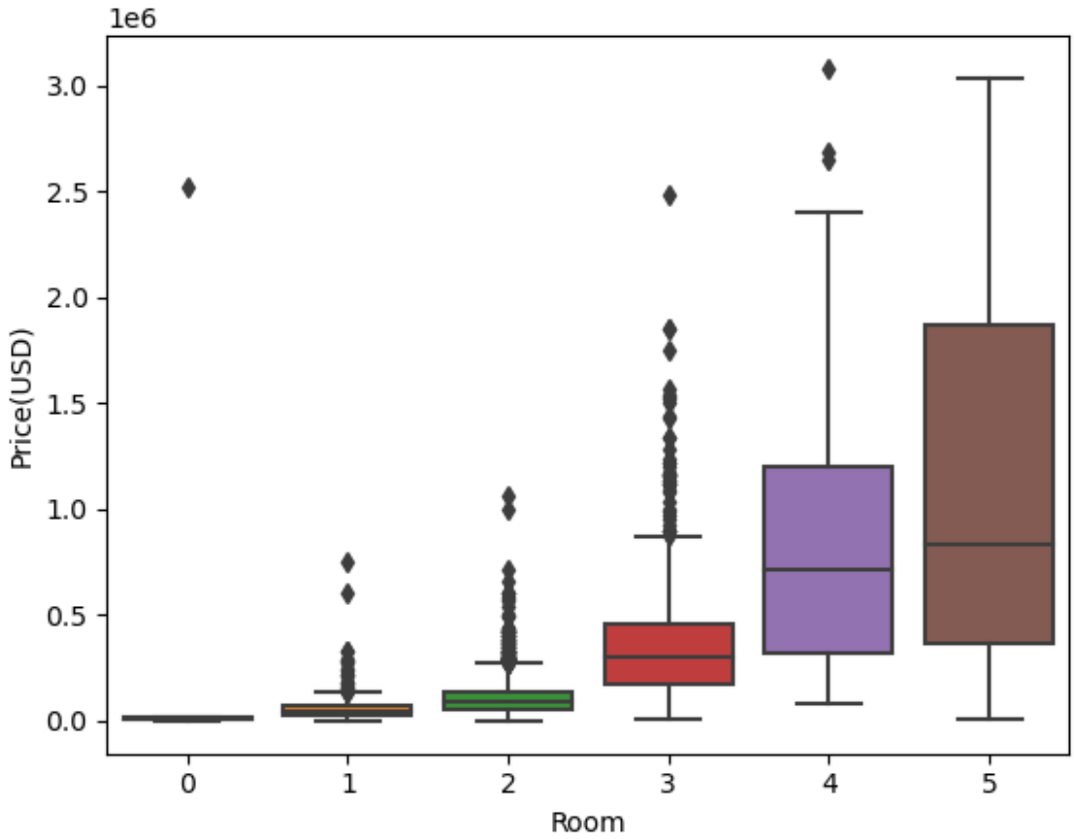

```
plt.title('Warehouse v/s Price(USD)')

sns.barplot(x="Warehouse", y="Price(USD)", data=df)
```

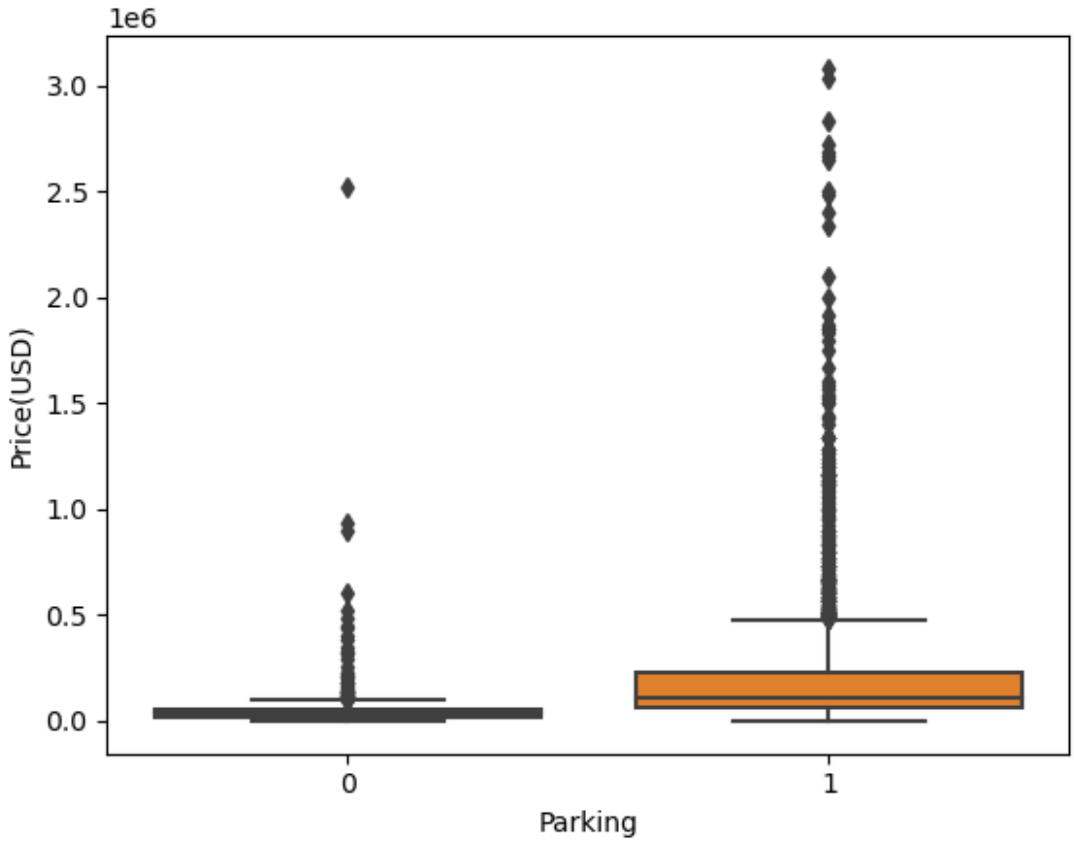
```
<AxesSubplot:title={'center':'Warehouse v/s Price(USD)'},
xlabel='Warehouse', ylabel='Price(USD)'>
```




```
ax = sns.boxplot(x="Room", y="Price(USD)", data=df)
```

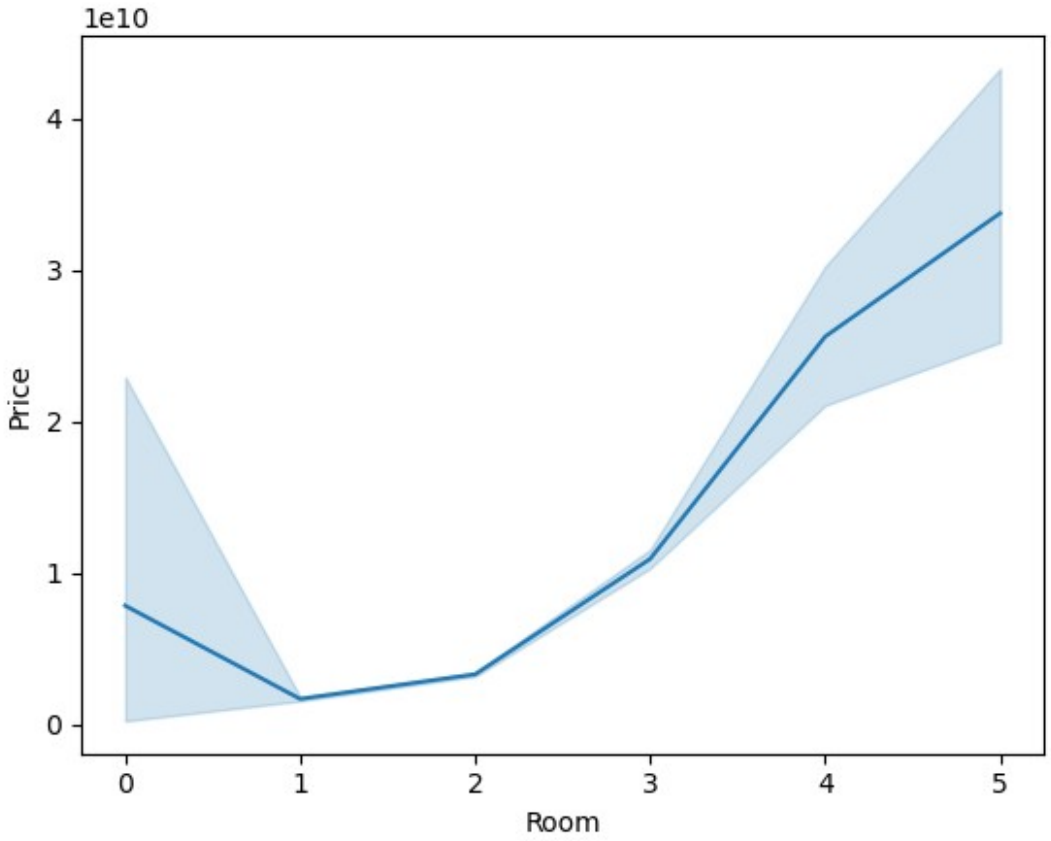



```
ax = sns.boxplot(x="Parking", y="Price(USD)", data=df)
```




```
sns.lineplot(data = df, x = 'Room', y ='Price' )
```

```
<AxesSubplot:xlabel='Room', ylabel='Price'>
```




```
correlation=df.corr()
plt.figure(figsize=(10,5))
sns.heatmap(correlation, annot = True, cmap = 'Purples')
```

```
<AxesSubplot:>
```

```
df.drop(['Address',"Price"], axis=1, inplace=True)
df.head()
```

```
  Area  Room  Parking  Warehouse  Elevator  Price(USD)
0   63     1        1          1         1    61666.67
1   60     1        1          1         1    61666.67
2   79     2        1          1         1    18333.33
3   95     2        1          1         1    30083.33
4  123     2        1          1         1   233333.33
```

```
df['Price(USD)'] = df['Price(USD)'].astype(int)
df.info('Price(USD)')
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3456 entries, 0 to 3478
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Area        3456 non-null   object
 1   Room        3456 non-null   int64 
 2   Parking     3456 non-null   int32 
 3   Warehouse   3456 non-null   int32 
 4   Elevator    3456 non-null   int32 
 5   Price(USD)  3456 non-null   int32 
dtypes: int32(4), int64(1), object(1)
memory usage: 264.0+ KB
```

```
df['Area'] = df['Area'].apply(lambda x: re.sub(',', '', x))
df["Area"] = pd.to_numeric(df["Area"] , errors='coerce')
```

in below import multiple scikit_learn library.

```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
from sklearn.model_selection import train_test_split
```

Linear Regression :

```
lr = LinearRegression()
```

```
X=df.drop("Price(USD)", axis=1,)
y=df['Price(USD)']
```

in this section train the model on the dataset

```
lr.fit(X, y)
```

```
LinearRegression()
```

```
 x_train, x_test,y_train,y_test=train_test_split(X,y ,test_size=0.2,
random_state=8)
```

Evaluate the model

```
y_pred = lr.predict(x_test)
```

in the below cell we compute the evaluation metrics.

```
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)
lr_squared = lr.score(x_test, y_pred)
```

Below cell print the evaluation metrics

```
print('Mean Squared Error: ',lr_mse)
print('root Mean Squared Error: ',lr_rmse)
print('root mean squred error: ', lr_squared)
```

```
Mean Squared Error:  27658121568.421448
root Mean Squared Error:  166307.31062831078
root mean squred error:  1.0
```

RMSE indicates better performance on the linear regression model.

Random forest Regression :

Fit the model for Random Forest Regression.

```
model = RandomForestRegressor(n_estimators=100, random_state=0)
model.fit(x_train, y_train)
```

```
RandomForestRegressor(random_state=0)
```

in the below we predict the target variable for the test set.

```
y_pred = model.predict(x_test)
```

in the below cell we compute the evaluation metrics.

```
rfr_mse = mean_squared_error(y_test, y_pred)
rfr_rmse = np.sqrt(rfr_mse)
rfr_mae = mean_absolute_error(y_test, y_pred)
rfr_r2 = r2_score(y_test, y_pred)
```

Below cell print the evaluation metrics.

```
print('Mean Squared Error: ', rfr_mse)
print('Root Mean Squared Error: ', rfr_rmse)
print('Mean Absolute Error: ', rfr_mae)
print('R2 Score: ', rfr_r2)
```

```
Mean Squared Error:  13624958328.467304
Root Mean Squared Error:  116725.99679791689
Mean Absolute Error:  66300.97571233594
R2 Score:  0.6461944486196864
```

SVR(support vector machine regression :

fit the model for svr

```
model = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=.1)
model.fit(X, y)
```

```
SVR(C=100, gamma=0.1)
```

```
y_pred = model.predict(x_test)
```

in the below cell we compute the evaluation metrics.

```
svr_mse = mean_squared_error(y_test, y_pred)
svr_rmse = np.sqrt(svr_mse)
svr_mae = mean_absolute_error(y_test, y_pred)
svr_r2 = r2_score(y_test, y_pred)
```

Below cell print the evaluation metrics.

```
print('Mean Squared Error: ', svr_mse)
print('Root Mean Squared Error: ', svr_rmse)
print('Mean Absolute Error: ', svr_mae)
print('R2 Score: ', svr_r2)
```

```
Mean Squared Error:  40756685066.4255
Root Mean Squared Error:  201882.84985710276
Mean Absolute Error:  104622.57934121475
R2 Score:  -0.05834756222572546
```

knn Regression :

below knn model fit

```
model = KNeighborsRegressor(n_neighbors=5)
model.fit(x_train, y_train)
```

```
KNeighborsRegressor()
```

Predict the target variable for the test set

```
y_pred = model.predict(x_test)
```

in the below cell we compute the evaluation metrics.

```
knnr_mse = mean_squared_error(y_test, y_pred)
knnr_rmse = np.sqrt(knnr_mse)
knnr_mae = mean_absolute_error(y_test, y_pred)
knnr_r2 = r2_score(y_test, y_pred)
```

Below cell print the evaluation metrics

```
print('Mean Squared Error: ', knnr_mse)
print('Root Mean Squared Error: ', knnr_rmse)
print('Mean Absolute Error: ', knnr_mae)
print('R2 Score: ', knnr_r2)
```

```
Mean Squared Error:  17434634973.220055
Root Mean Squared Error:  132040.27784437616
Mean Absolute Error:  70925.66329479769
R2 Score:  0.5472668252550521
```

below create dataframe for all models that we applied with RMSE (Root Mean Squared Error).

```
m_score = pd.DataFrame({'RMSE': [lr_rmse,svr_rmse, knnr_rmse,
rfr_rmse]},
                       index =
['LinearRegression','RandomForestRegression','SVR',
'KNeighborsRegression'])
```

```
m_score
```

```
                                RMSE
LinearRegression        166307.310628
RandomForestRegression  201882.849857
SVR                     132040.277844
KNeighborsRegression    116725.996798
```

END