Project No #5

Housing Price prediction

Using Machine Learning Algorithms Regression

```
data.head()
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	79545.45857	5.682861	7.009188	4.09	23086.80050	1.059034e+06
1	79248.64245	6.002900	6.730821	3.09	40173.07217	1.505891e+06
2	61287.06718	5.865890	8.512727	5.13	36882.15940	1.058988e+06
3	63345.24005	7.188236	5.586729	3.26	34310.24283	1.260617e+06
4	59982.19723	5.040555	7.839388	4.23	26354.10947	6.309435e+05

Linear Regression

KNN Regression

Random Forest Regression

Stacking Regression

IMPORTS LIBRARY

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,cross_val_score,GridSearchCV,KFold
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from mlxtend.regressor import StackingRegressor
from sklearn.metrics import mean_squared_error
```

Data Cleaning

```
data.isnull().sum()
```

```
Avg. Area Income                0
Avg. Area House Age             0
Avg. Area Number of Rooms       0
Avg. Area Number of Bedrooms    0
Area Population                 0
Price                           0
dtype: int64
```

```
data.duplicated().sum()
```

```
0
```

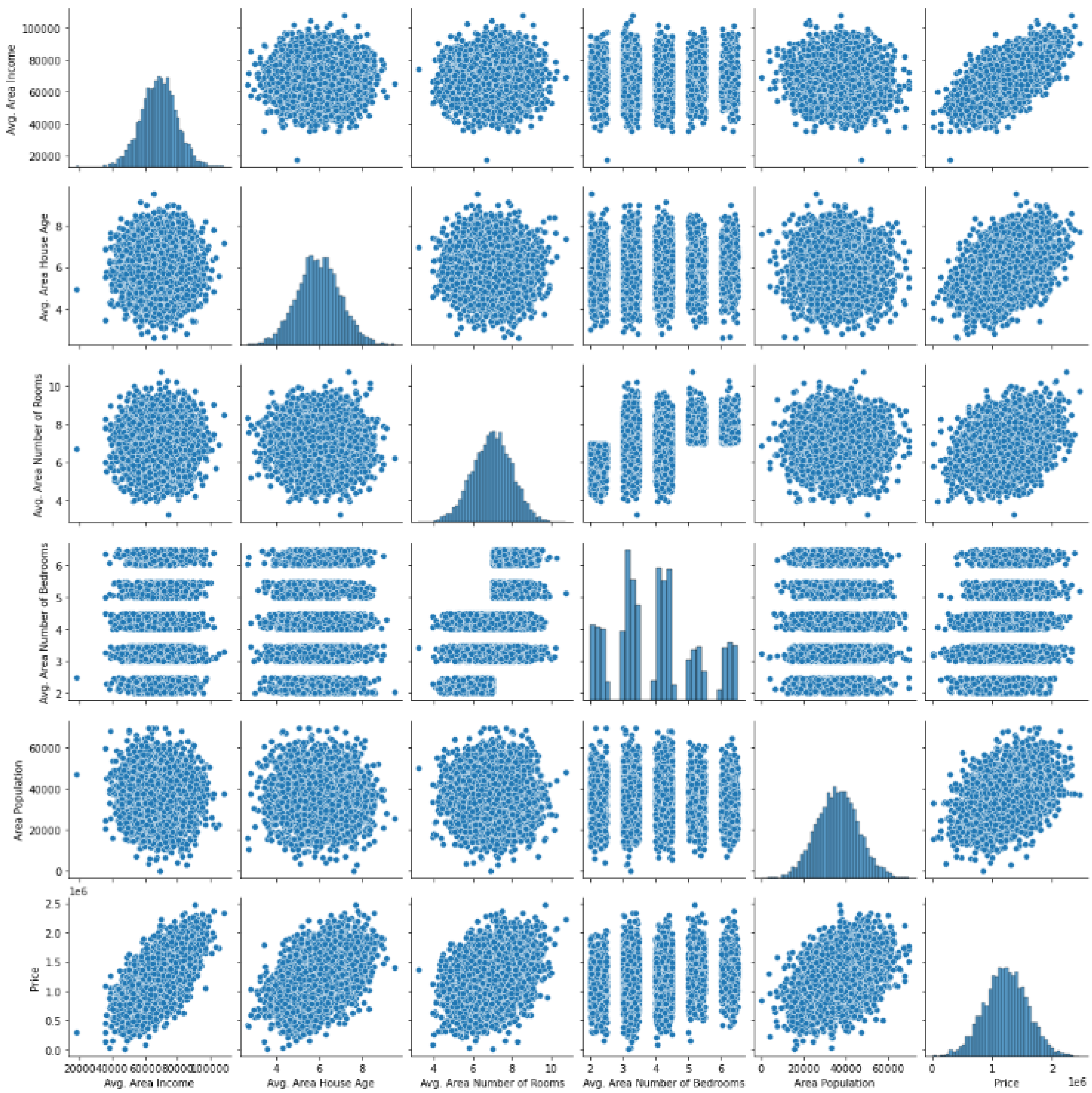
Exploratory Analysis

```
data.describe()
```

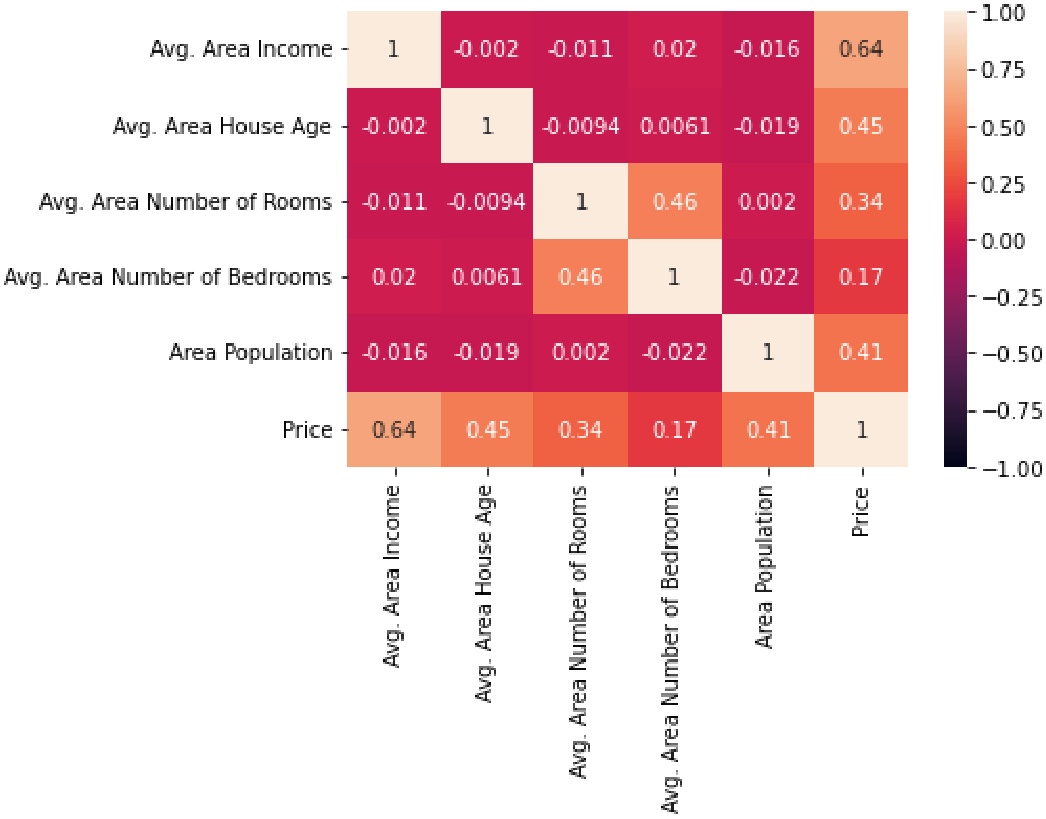
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562390	5.322283	6.299250	3.140000	29403.928700	9.975771e+05
50%	68804.286405	5.970429	7.002902	4.050000	36199.406690	1.232669e+06
75%	75783.338665	6.650808	7.665871	4.490000	42861.290770	1.471210e+06
max	107701.748400	9.519088	10.759588	6.500000	69621.713380	2.469066e+06

```
sns.pairplot(data)
plt.plot()
```

[]



```
sns.heatmap(data.corr(),vmin=-1,vmax=1,annot=True)
plt.show()
```



Machine Learning

```
data.head()
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	79545.45857	5.682861	7.009188	4.09	23086.80050	1.059034e+06
1	79248.64245	6.002900	6.730821	3.09	40173.07217	1.505891e+06
2	61287.06718	5.865890	8.512727	5.13	36882.15940	1.058988e+06
3	63345.24005	7.188236	5.586729	3.26	34310.24283	1.260617e+06
4	59982.19723	5.040555	7.839388	4.23	26354.10947	6.309435e+05

```
x=data.iloc[:,:5].values
y=data.iloc[:,5].values
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

1. Linear Regression

```
lm=LinearRegression()
```

```
lm.fit(x_train,y_train)
```

```
LinearRegression()
```

```
lm.coef_
```

```
array([2.16604083e+01, 1.65809651e+05, 1.20329408e+05, 2.19309558e+03,
       1.52858855e+01])
```

```
data.columns[0:5]
```

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population'],
      dtype='object')
```

```
pd.DataFrame(lm.coef_,index=data.columns[0:5],columns=["Coefficient"])
```

	Coefficient
Avg. Area Income	21.660408
Avg. Area House Age	165809.651152
Avg. Area Number of Rooms	120329.407878
Avg. Area Number of Bedrooms	2193.095578
Area Population	15.285885

```python
lm.intercept_
```

```
-2646630.531087137
```

```python
lm.score(x_train,y_train)
```

```
0.9188401140943028
```

```python
y_pred=lm.predict(x_test)
```

```python
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
102711.83810005663
```

Inference in Regression

```python
x_with_constant=sm.add_constant(x_train)
```

```python
lm_sm=sm.OLS(y_train,x_with_constant)
```

```python
result=lm_sm.fit()
```

```python
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.919
Model:                            OLS   Adj. R-squared:                  0.919
Method:                 Least Squares   F-statistic:                     9044.
Date:                Fri, 17 Nov 2023   Prob (F-statistic):               0.00
Time:                        06:59:43   Log-Likelihood:                -51755.
No. Observations:                4000   AIC:                         1.035e+05
Df Residuals:                    3994   BIC:                         1.036e+05
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -2.647e+06   1.91e+04   -138.228      0.000   -2.68e+06   -2.61e+06
x1            21.6604      0.149    144.946      0.000      21.367      21.953
x2          1.658e+05   1598.673    103.717      0.000    1.63e+05    1.69e+05
x3          1.203e+05   1779.180     67.632      0.000    1.17e+05    1.24e+05
x4          2193.0956   1461.592      1.500      0.134    -672.440    5058.631
x5            15.2859      0.161     94.837      0.000      14.970      15.602
==============================================================================
Omnibus:                        4.735   Durbin-Watson:                   2.016
Prob(Omnibus):                  0.094   Jarque-Bera (JB):                4.353
Skew:                          -0.034   Prob(JB):                        0.113
Kurtosis:                       2.854   Cond. No.                     9.42e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.42e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Correlation between variables

check VIF

```
print(variance_inflation_factor(x_train,0))
print(variance_inflation_factor(x_train,1))
print(variance_inflation_factor(x_train,2))
print(variance_inflation_factor(x_train,3))
print(variance_inflation_factor(x_train,4))
```

⇒ more correlated

```
29.518898716616043
27.14474538095936
44.50881222623392
14.51216193025586
12.896484451106032
```

Solutions:

Remove variables one by one which are having VIF>10 and fit regressions.
Regularization or Dimensionality Reduction.

Do again

```
new_data=data.drop("Avg. Area Number of Bedrooms",axis=1)
new_data
x_new=new_data.iloc[:,:4].values
y_new=new_data.iloc[:,4].values
x_new_train,x_new_test,y_new_train,y_new_test=train_test_split(x_new,y_new,test_size=0.2,random_state=0)
lm=LinearRegression()
lm.fit(x_new_train,y_new_train)
y_new_pred=lm.predict(x_new_test)
np.sqrt(mean_squared_error(y_new_test,y_new_pred))
```

```
102671.05426024446
```

```
x_with_constant=sm.add_constant(x_new_train)
lm_sm=sm.OLS(y_new_train,x_with_constant)
result=lm_sm.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.919
Model:                            OLS   Adj. R-squared:                  0.919
Method:                 Least Squares   F-statistic:                 1.130e+04
Date:                Mon, 20 Nov 2023   Prob (F-statistic):               0.00
Time:                        10:26:06   Log-Likelihood:                -51756.
No. Observations:                4000   AIC:                         1.035e+05
Df Residuals:                    3995   BIC:                         1.036e+05
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -2.647e+06   1.91e+04   -138.230      0.000   -2.68e+06   -2.61e+06
x1            21.6681      0.149    145.060      0.000      21.375      21.961
x2          1.658e+05   1598.857    103.719      0.000    1.63e+05    1.69e+05
x3          1.216e+05   1578.423     77.015      0.000    1.18e+05    1.25e+05
x4            15.2785      0.161     94.821      0.000      14.963      15.594
==============================================================================
Omnibus:                        4.614   Durbin-Watson:                   2.015
Prob(Omnibus):                  0.100   Jarque-Bera (JB):                4.261
Skew:                          -0.035   Prob(JB):                        0.119
Kurtosis:                       2.856   Cond. No.                     9.42e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.42e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
print(variance_inflation_factor(x_new_train,0))
print(variance_inflation_factor(x_new_train,1))
print(variance_inflation_factor(x_new_train,2))
print(variance_inflation_factor(x_new_train,3))
```

```
29.484395570334836
27.144248713809294
31.57863327968842
12.879726751429272
```

VIF > 10

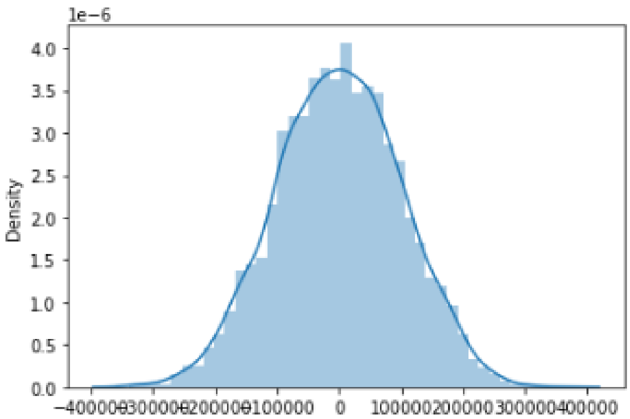
Solutions:

Remove variables one by one which are having VIF>10 and fit regressions.
Regularization or Dimensionality Reduction.

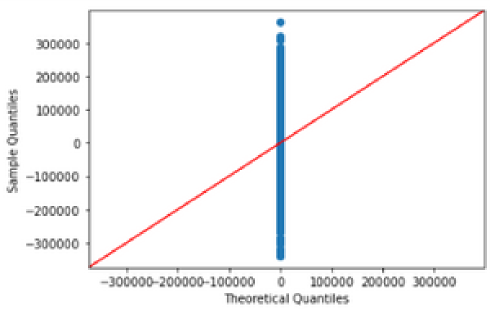
Normality of Residuals

```
resid_new=y_new_train-lm.predict(x_new_train)
sns.distplot(resid_new)
plt.show()
```

```
C:\Users\Prasa\anaconda3\new\lib\site-packages\seaborn\distributions.py:2619: F
n and will be removed in a future version. Please adapt your code to use either
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
import pylab as py
sm.qqplot(resid_new,line="45")
py.show()
```



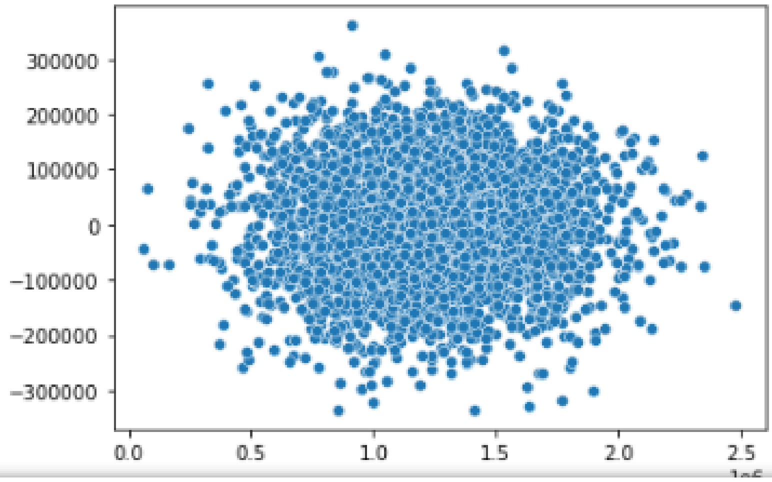
Solutions:

Data transformation with different approaches

Homoscedasticity & Residual Independency

```
sns.scatterplot(lm.predict(x_train),resid)
plt.show()
```

```
C:\Users\Prasa\anaconda3\new\lib\site-packages\seaborn\_decorators.py
ord args: x, y. From version 0.12, the only valid positional argument
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Assumption is not necessary like stats

2.KNN Regression

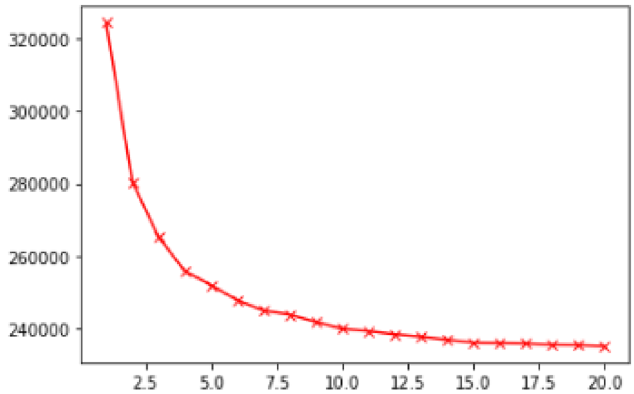
Selecting the optimal k value

Validation set approach

```
Errors=[]
K=np.arange(1,21)

for k in K:
    model=KNeighborsRegressor(n_neighbors=k)
    cvals=np.sqrt(-cross_val_score(model,x_train,y_train,cv=10,scoring="neg_mean_squared_error"))
    Errors.append(cvals.mean())
```

```
plt.plot(K,Errors,"rx-")
plt.show()
```




```
knn = KNeighborsRegressor(n_neighbors=7)
```

```
knn.fit(x_train, y_train)
```

```
KNeighborsRegressor(n_neighbors=7)
```

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

```
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
239881.84240633072
```

3.Random Forest Regression

Optimizing hyper parameters

```
params={"n_estimators":[100,200,300,400,500]}
model=RandomForestRegressor()
cval=KFold(n_splits=5)
```

```
gsearch=GridSearchCV(model,params,cv=cval)
```

```
results=gsearch.fit(x_train,y_train)
results.best_params_
```

```
{'n_estimators': 400}
```

```
rf = RandomForestRegressor(n_estimators=500)
```

```
rf.fit(x_train, y_train)
```

```
RandomForestRegressor(n_estimators=500)
```

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

```
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
122016.85344638201
```

```
rf.feature_importances_
```

```
array([0.43426751, 0.23458556, 0.12542725, 0.01680957, 0.18891012])
```

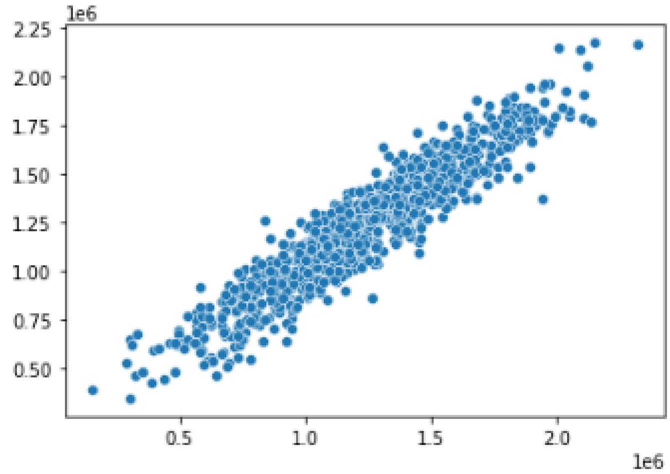
```
data.columns[:5]
```

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population'],
      dtype='object')
```

```
sns.scatterplot(y_test,y_pred)
plt.plot()
```

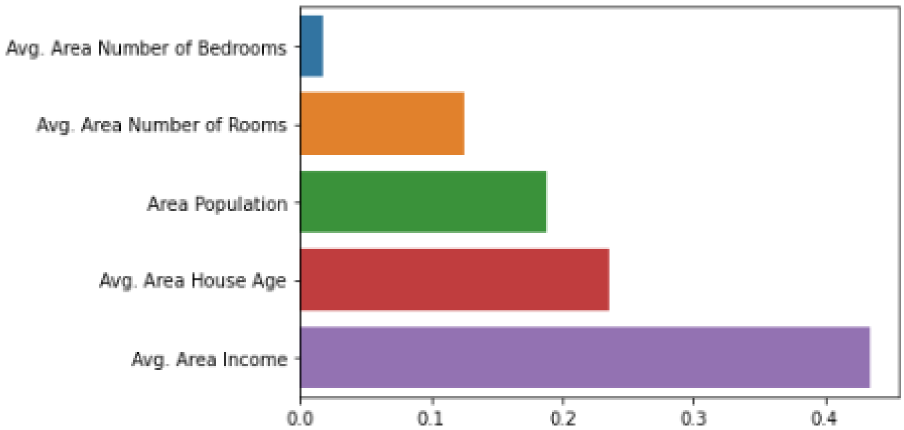
```
C:\Users\Prasa\anaconda3\new\lib\site-packages\seaborn\_decor
d args: x, y. From version 0.12, the only valid positional ar
icit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
[]
```




```
idx=np.argsort(rf.feature_importances_)
```

```
sns.barplot(x=rf.feature_importances_[idx],y=data.columns[:5][idx])
plt.show()
```



4.Stacking Regression

```
bmodel1=LinearRegression()
bmodel2=KNeighborsRegressor(n_neighbors=6)
```

```
metamodel=RandomForestRegressor(n_estimators=500)
```

```
st=StackingRegressor(regressors=[bmodel1,bmodel2],meta_regressor=metamodel)
```

```
st.fit(x_train, y_train)
```

```
StackingRegressor(meta_regressor=RandomForestRegressor(n_estimators=500),
                  regressors=[LinearRegression(),
                              KNeighborsRegressor(n_neighbors=6)])
```

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

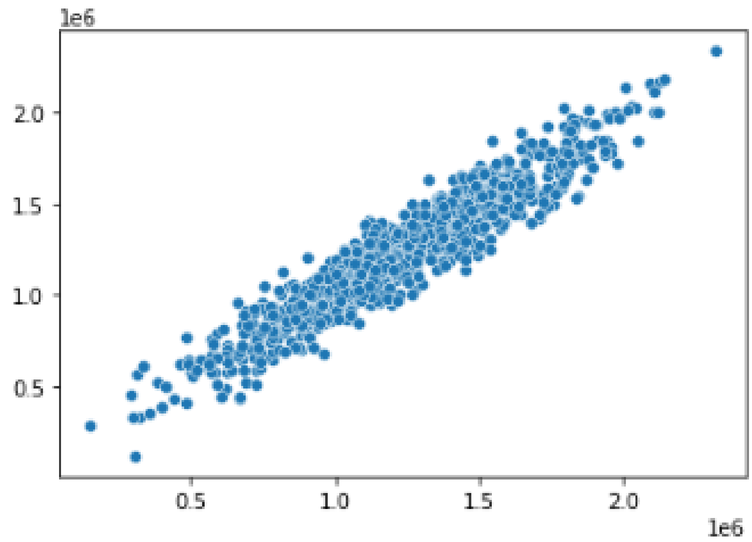
```
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
113621.43814581452
```

```
sns.scatterplot(y_test,y_pred)
plt.plot()
```

```
C:\Users\Prasa\anaconda3\new\lib\site-packages\seaborn\_decorator
d args: x, y. From version 0.12, the only valid positional argume
icit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
[]
```



Comparing performance

```
y_pred=lm.predict(x_new_test)
lm_rmse=np.sqrt(mean_squared_error(y_pred,y_test))
```

```
y_pred=knn.predict(x_test)
knn_rmse=np.sqrt(mean_squared_error(y_pred,y_test))
```

```
y_pred=rf.predict(x_test)
rf_rmse=np.sqrt(mean_squared_error(y_pred,y_test))
```

```
y_pred=st.predict(x_test)
st_rmse=np.sqrt(mean_squared_error(y_pred,y_test))
```

```
pd.DataFrame({"Model":["Linear Regression","KNN","Random Forest","Stacking"],"RMSE":[lm_rmse,knn_rmse,rf_rmse,st_rmse]})
```

	Model	RMSE
0	Linear Regression	102671.054260
1	KNN	239881.842406
2	Random Forest	121567.989545
3	Stacking	113621.438146