In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df = pd.read_csv("USA_Housing.csv")
df.head()
```

Out[2]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Avg. Area Income              5000 non-null   float64
 1   Avg. Area House Age           5000 non-null   float64
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
 6   Address                       5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

In [4]:
```
df.describe()
```

Out[4]:

	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%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

In [5]:
```
df.corr().style.background_gradient() #to check the correalation through heatmap
```

Out[5]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
Avg. Area Income	1.000000	-0.002007	-0.011032	0.019788	-0.016234	0.639734
Avg. Area House Age	-0.002007	1.000000	-0.009428	0.006149	-0.018743	0.452543
Avg. Area Number of Rooms	-0.011032	-0.009428	1.000000	0.462695	0.002040	0.335664
Avg. Area Number of Bedrooms	0.019788	0.006149	0.462695	1.000000	-0.022168	0.171071
Area Population	-0.016234	-0.018743	0.002040	-0.022168	1.000000	0.408556
Price	0.639734	0.452543	0.335664	0.171071	0.408556	1.000000

In [6]:
```
df.corr()["Price"].sort_values()
```

Out[6]:
```
Avg. Area Number of Bedrooms    0.171071
Avg. Area Number of Rooms       0.335664
Area Population                 0.408556
Avg. Area House Age             0.452543
Avg. Area Income                0.639734
Price                           1.000000
Name: Price, dtype: float64
```

Dividing the dataset into x & y

In [7]:
```
x = df.iloc[:,:-2].values
y = df.iloc[:, -2].values
```

In [8]:
```
x
```

Out[8]:
```
array([[7.95454586e+04, 5.68286132e+00, 7.00918814e+00, 4.09000000e+00,
        2.30868005e+04],
       [7.92486425e+04, 6.00289981e+00, 6.73082102e+00, 3.09000000e+00,
        4.01730722e+04],
       [6.12870672e+04, 5.86588984e+00, 8.51272743e+00, 5.13000000e+00,
        3.68821594e+04],
       ...,
       [6.33906869e+04, 7.25059061e+00, 4.80508098e+00, 2.13000000e+00,
        3.32661455e+04],
       [6.80013312e+04, 5.53438842e+00, 7.13014386e+00, 5.44000000e+00,
        4.26256202e+04],
       [6.55105818e+04, 5.99230531e+00, 6.79233610e+00, 4.07000000e+00,
        4.65012838e+04]])
```

In [9]:
```
y
```

Out[9]:
```
array([1059033.55787012, 1505890.91484695, 1058987.98787608, ...,
       1030729.58315229, 1198656.87240769, 1298950.48026696])
```

dividing the model (Training and testing model)

In [10]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.2, random_state=
```

Performing LinearRegression on the dataset

In [11]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(xtest, ytest)
ypred = linreg.predict(xtest)
```

Displaying interception

In [12]:
```
linreg.intercept_
```

Out[12]:
```
-2639118.359220582
```

Displaying coefficient

In [13]:
```
linreg.coef_
```

Out[13]:
```
array([2.11974663e+01, 1.68646275e+05, 1.20151740e+05, 1.90404420e+03,
       1.54397830e+01])
```

To interpret the Coefficient more easily we need to convert this data into a Dataframe

In [14]:
```
x = df.iloc[:,:-2]
y = df.iloc[:,-2]
```

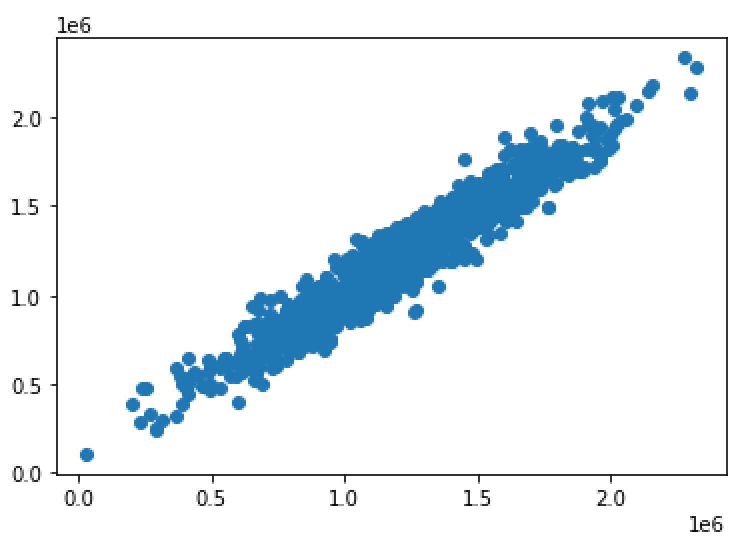
In [15]:
```
coef_df = pd.DataFrame(linreg.coef_,x.columns, columns = ["Coefficient"])
coef_df
```

Out[15]:

	Coefficient
Avg. Area Income	21.197466
Avg. Area House Age	168646.274595
Avg. Area Number of Rooms	120151.740076
Avg. Area Number of Bedrooms	1904.044197
Area Population	15.439783

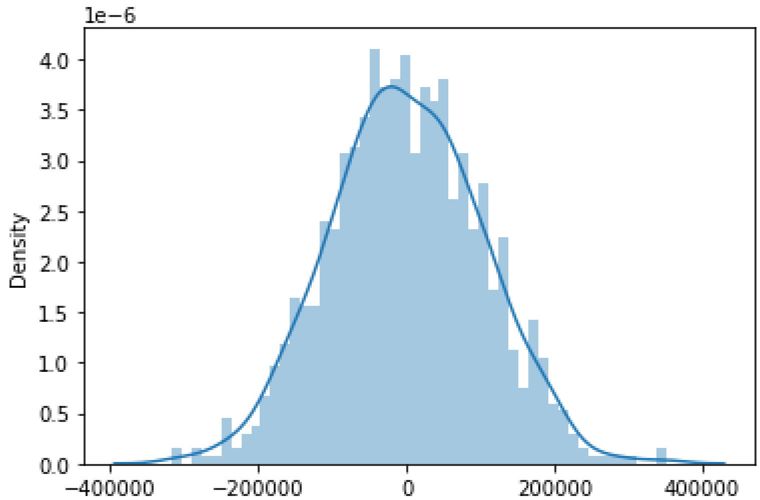
In [16]:
```
plt.scatter(ytest, ypred)
```

Out[16]: <matplotlib.collections.PathCollection at 0x24602281f0>



In [17]:
```
sns.distplot((ytest-ypred), bins=50)
```

Out[17]:
```
<AxesSubplot:ylabel='Density'>
```



Displaying evaluation Metrics

- It is used to evaluate the the performance and the accuracy of the model
- we need to import some modules for evaluation

In [18]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error,r2_score
```

In [19]:
```
print(f"MeanAbsoluteError :-  {mean_absolute_error(ytest,ypred)}")
print(f"MeanSquaredError :-  {mean_squared_error(ytest,ypred)}")
print(f"MeanSquaredError :-  {mean_squared_error(ytest,ypred)}")
print(f"rSquared :-  {(r2_score(ytest,ypred))}")
```

```
MeanAbsoluteError :-  82069.79610021316
MeanSquaredError :-  10478307802.663448
MeanSquaredError :-  10478307802.663448
rSquared :-  0.9220790430417267
```

- r2_score helps to determine the accuracy of the model
- Root Mean Squared Error or RMSE represents the squareroot Mean Squared Error.It measures the Standard Deviation of the residuals
- Mean Squared Error or MSE represents the average of the squared difeerence between the actual and the predicted values in the given Dataset

- MAE or Mean Absolute error represents the average of the absolute difference between the actual values and the predicted values in the given Dataset

CONCLUSION

- After evaluating this Dataset we can say that this model is 92% accurate and can be used for future predictions
- All the evaluation metrics has been displayed in the above Dataset