Since price is a continuous variable, linear regression may be a good place to start from

it's always a good step to use **describe()** and **info()** to get a better sense of the data and see if we have any missing values.

When you do data.info() and if it gives all 5,000 entries and 5,000 non-null then there are no missing values in this dataset.

**Warning: The more features in our dataset, the harder our pair plot will be to interpret.**

**Sns . histplot ( housing \_ data [ ' Price ' ] )**

**Plt . show( )**

A scatterplot of Price vs. Avg. Area Income shows a strong positive linear relationship between the two.

Sns . scatterplot ( x = ' Price ' , y = ' Avg. Area Income ' , data = housing\_data )

Plt . show( )

**Boxplot** -> it will line it is minimum and the last line is the max. Also, the line in the middle of the box is the median. Also When the box starts it is 25% and when the box is over it is 75%

BOXPLOT ==OUTLIER (MIN ,25%,50%,75% and MAX)

***it is strongly advised to solve the issue if severe collinearity issue exists(e.g. correlation >0.8 between 2 variables)***

Sns . heatmap(DATA . corr()) -> it will give us HeatMAP without numbers only colors

Sns **.** heatmap(DATA**.** corr(), annot**=True**)

Plt **.** show()

We use HeatMAp to check whether is data clean and to make sure there are **no severe collinearity issues.**

``` **THE FIRST THING begin creating and training our model**

We first need to split our data into training and testing sets. This can be done using sklearn's ***train\_test\_split(X, y, test\_size)*** function.

We'll now import sklearn's LinearRegression model and begin training it using the ***fit(train\_data, train\_data\_labels)*** method. Then, after it is trained, the model can be used to make predictions, usually with a ***predict(test\_data)*** method call

To get a rough idea of how well the model is predicting, we can make a scatterplot with the true test labels (y\_test) on the x-axis, and our predictions on the y-axis. Ideally, we'd like a **45-degree line.** The straighter the line, the better our predictions are.

**loss functions**

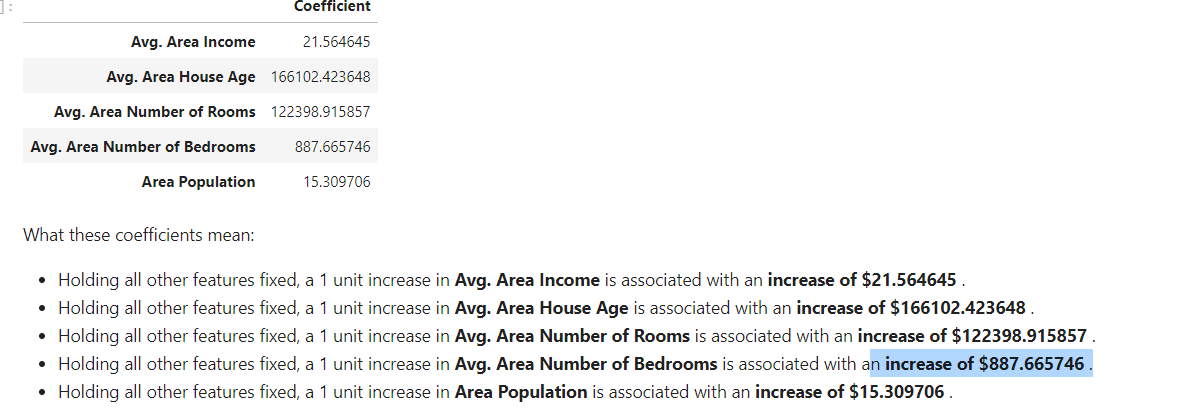
1. **Mean Absolute Error** (MAE) is the mean of the absolute value of the errors, it is the easiest to understand because it's the average error.
2. **Mean Squared Error** (MSE) is the mean of the squared errors, it is the easiest to understand because it's the average error.
3. **Root Mean Squared Error** (RMSE) is the square root of the mean of the squared errors, it is even more popular than MSE because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them

Luckily, **sklearn** can calculate all of these metrics for us. Luckily, sklearn can calculate all of these metrics for us

the coefficient of determination (R^2), which is the percentage of variation in y explained by all the x variables together. Usually, an R^2 of .70 is considered good.

**Dosbol.DATA= pd . DataFrame ( lm . coef \_ , X . columns , columns = [ ' Coefficient ' ] )**

****