How I proceeded ?

> Firstly find a dataset from Kaggle's discussion zone and got a dataset for houses in California.

> Then I use Pandas library to store the dataset for manipulation.

> Then I plotted some graphs, actually learned to plot graphs via matplotlib and seaborn libraries.

Data Cleaning →

> Then I came to know that I had some NULL values in some columns , so I replaced the null values of the column with median value so that it is easy for me to judge.

Preparing Target Variable & Input Features variable

> Then I loaded my input features into a variable X and loaded what I desire out from this model in 'y' (Target).

> As the data set was large and diverse, I used

→ As the data set was large

Z- Score Normalisation on the features which scales the features based on :

$$X\_Scaled = \frac{X-\mu}{\sigma}$$

Here
μ: Mean
σ: Standard Deviation

I did this with sklearn.preprocess library by importing StandardScaler ( ) function for standardization

Splitting The Data :

→ Before training the model, we divide our dataset into two or three parts, majorily two. Those are :

1) Training Set
- Study material for your model.

2) Testing Set
- This is a new data that the model has not seen before & will be used to evaluate the model.

→ Will import train test split from sklearn.model_selection

* will import train_test_split from

which returns 4 values, splitted data, arguments passed are

(features, output, test size, random state (keeps splitting random))

Train the Model :

For training via linear regression :

from sklearn.linear_model import LinearRegression.

* So we put our training data in fit function of linear regression.

What it does is that it takes input features and known output to find the coefficients and bias term for the eq^n.

$$y = \Sigma w_i x_i + b$$

Contributing for the learning part.

model.intercept_ → Bias term

model.coeff_ → weights for feature

Make Prediction →

* Use model's score or prediction function to make

predictions: The score functions calculate an R^2-score which determines how much variancy is my model regarding instead of returning a mean valued answer on a scale of 0 to 1.