Linear regression: The term regression is used when we try to find out the relationship betn variables.

Simple Linear Regression: Simple linear regression is an approach for predicting a response using a single feature. It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).

Multiple linear regression: Multiple linear regression attempts to model the relationship between two or more features and a response by fitting a linear equation to the observed data. Clearly, it is nothing but an extension of simple linear regression

```
import pandas as pd
df = pd.read_csv('/content/generated_data.csv')
```

```
df
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b
0	1	0.00113	6.53584	0	0.98542	4.90081	60.12242	4.99054	3	453	21.76784	0.51276
1	1	0.87241	3.51304	0	0.95924	4.72511	90.09379	8.78739	0	362	15.02063	0.14970
2	1	0.11271	5.14592	0	0.91430	6.89531	4.76557	9.34756	4	95	12.92292	0.68093
3	0	0.63364	7.05673	0	0.07704	7.96250	26.67127	1.82544	3	317	23.80440	0.10022
4	1	0.52053	7.57258	1	0.49227	6.22471	20.87046	2.83895	6	99	17.26946	0.77099
...	...	...	...	...	...	...	...	...	...	...	...	...
95	0	0.50093	7.19273	0	0.74008	4.23408	36.86495	3.36242	2	190	18.97723	0.68740
96	0	0.48375	1.29044	1	0.79745	7.06991	41.75136	3.80451	0	431	12.04406	0.57572
97	1	0.40590	7.65708	1	0.49692	6.09458	1.51823	1.96137	2	250	20.83845	0.42445
98	1	0.74086	7.06802	0	0.66626	4.12948	56.97681	3.18546	0	338	11.72099	0.03314
99	1	0.68672	1.91820	0	0.59536	7.75170	48.88824	7.58830	6	239	23.99861	0.47168

100 rows × 14 columns

Next steps: Generate code with df View recommended plots

```
df.head()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b
0	1	0.00113	6.53584	0	0.98542	4.90081	60.12242	4.99054	3	453	21.76784	0.51276
1	1	0.87241	3.51304	0	0.95924	4.72511	90.09379	8.78739	0	362	15.02063	0.14970
2	1	0.11271	5.14592	0	0.91430	6.89531	4.76557	9.34756	4	95	12.92292	0.68093
3	0	0.63364	7.05673	0	0.07704	7.96250	26.67127	1.82544	3	317	23.80440	0.10022
4	1	0.52053	7.57258	1	0.49227	6.22471	20.87046	2.83895	6	99	17.26946	0.77099

Next steps: Generate code with df View recommended plots

```
#import data
x= df.drop('medv',axis =1)

#output data
y=df['medv']
```

```
x.shape
```

```
(100, 13)
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=0, test_size = 0.25)
```

```
x_train
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat
48	1	0.27093	5.03084	0	0.23786	5.16015	80.90721	5.32975	7	50	12.27553	0.43788	4.96307
6	1	0.77656	4.91400	1	0.77006	5.83755	89.89566	9.74740	4	284	24.14012	0.31361	0.62316
99	1	0.68672	1.91820	0	0.59536	7.75170	48.88824	7.58830	6	239	23.99861	0.47168	6.83101
82	1	0.01709	4.99852	1	0.90537	7.05769	93.98113	2.01682	2	195	18.53793	0.21353	2.07044
76	1	0.47774	3.78368	1	0.33582	7.29698	84.99978	1.09028	9	279	21.84224	0.10476	0.99331
...	...	...	...	...	...	...	...	...	...	...	...	...	...
96	0	0.48375	1.29044	1	0.79745	7.06991	41.75136	3.80451	0	431	12.04406	0.57572	6.46749
67	0	0.27031	0.55094	0	0.76693	4.88248	81.01061	2.17339	3	22	20.11344	0.88895	9.76879
64	1	0.44358	7.85018	0	0.62577	7.76048	17.73404	8.26618	9	491	13.80653	0.58644	8.94991
47	0	0.05119	4.27666	1	0.16354	7.52369	75.76188	9.91105	2	320	22.03903	0.97471	2.79910
44	1	0.71106	4.76615	0	0.34802	5.99920	48.08007	0.02521	8	433	19.41444	0.14229	1.46110

75 rows × 13 columns

Next steps: Generate code with x_train View recommended plots

```
x_train.head()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat
48	1	0.27093	5.03084	0	0.23786	5.16015	80.90721	5.32975	7	50	12.27553	0.43788	4.96307
6	1	0.77656	4.91400	1	0.77006	5.83755	89.89566	9.74740	4	284	24.14012	0.31361	0.62316
99	1	0.68672	1.91820	0	0.59536	7.75170	48.88824	7.58830	6	239	23.99861	0.47168	6.83101
82	1	0.01709	4.99852	1	0.90537	7.05769	93.98113	2.01682	2	195	18.53793	0.21353	2.07044
76	1	0.47774	3.78368	1	0.33582	7.29698	84.99978	1.09028	9	279	21.84224	0.10476	0.99331

Next steps: Generate code with x_train View recommended plots

```
x_train.shape
```

```
(75, 13)
```

```
x_test.shape
```

```
(25, 13)
```

```
from sklearn.linear_model import LinearRegression
#import the class
```

```
#creating the object
regressor = LinearRegression()

#train the algorithm
regressor.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
regressor.coef_
```

```
array([-2.71003320e+00,  1.29984189e+01,  1.02411410e+00, -3.88307299e+00,
        6.04021992e+00, -1.20263841e+00, -9.75640680e-02,  1.20346894e+00,
        8.73869406e-01, -3.76959988e-03, -1.45949662e-01, -3.64317015e+00,
       -7.60195696e-01])
```

```
regressor.intercept_
```

```
24.929589144519866
```

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

```
y_pred.shape
```

```
(25,)
```

```
result = pd.DataFrame({'Actual':y_test,'Producted':y_pred})
```

```
result
```

	Actual	Producted
26	46.40363	37.534715
86	42.04326	19.458876
2	44.73570	35.695929
55	15.75017	27.158367
75	24.23296	23.734345
93	14.08896	30.856696
16	5.90187	24.119602
73	14.79583	15.719729
54	6.48275	23.576746
95	11.55914	29.325739
53	27.08367	23.073161
92	0.33963	40.359969
78	15.83577	11.128520
13	45.25552	33.106660
7	1.25507	32.315001
30	48.29485	31.575741
22	48.43831	33.263169
24	21.83203	33.118340
33	31.64006	30.092000
8	22.26504	34.235618
43	42.15528	23.488630
62	1.19969	26.916610
3	4.54992	23.528648
71	7.45144	21.062237
45	49.69206	32.315804

Next steps: Generate code with `result` View recommended plots

```
residual_error = abs(y_test- y_pred)
residual_error
```

```
26     8.868915
86    22.584384
2      9.039771
55    11.408197
75     0.498615
93    16.767736
16    18.217732
73     0.923899
54    17.093996
95    17.766599
53     4.010509
92    40.020339
78     4.707250
13    12.148860
7     31.059931
30    16.719109
22    15.175141
24    11.286310
33     1.548060
8     11.970578
43    18.666650
62    25.716920
3     18.978728
71    13.610797
45    17.376256
Name: medv, dtype: float64
```

```
#mean absolute error
sum(residual_error)/len(residual_error)
```

```
14.646611200979779
```

```
from sklearn.linear_model import LinearRegression
import numpy as np
```

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_pred, y_test)
```

```
14.646611200979779
```

```
from sklearn.metrics import mean_absolute_percentage_error
mean_absolute_percentage_error(y_test,y_pred)
```

```
7.338982149081252
```

```
regressor.score(x_test,y_test)
```

```
-0.013475047353684655
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

```
-0.013475047353684655
```

```
new = [[0,0.14455,12.5,7.87,0,0.524,6.172,96.1,5.9505,5,311,0,0.573,5.856,85.9,2.5979]]
```

```
new
```

```
[[0,
  0.14455,
  12.5,
  7.87,
  0,
  0.524,
  6.172,
  96.1,
  5.9505,
  5,
  311,
  0,
  0.573,
  5.856,
  85.9,
  2.5979]]
```

```
new = np.array(new)
print(new.shape)
```

```
(1, 16)
```

```
print(regressor.n_features_in_)
```

```
13
```

```
new = np.array(new).reshape(1, -1)
```

```
# Remove the extra features
new = new[:, : regressor.n_features_in_]

# Add missing features with default values (e.g., 0)
new = np.pad(new, (0, regressor.n_features_in_ - new.shape[1]), mode='constant', constant_values=0)
```

```
prediction = regressor.predict(new)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted with
  warnings.warn(
```

```
prediction
```

```
array([82.82634401])
```