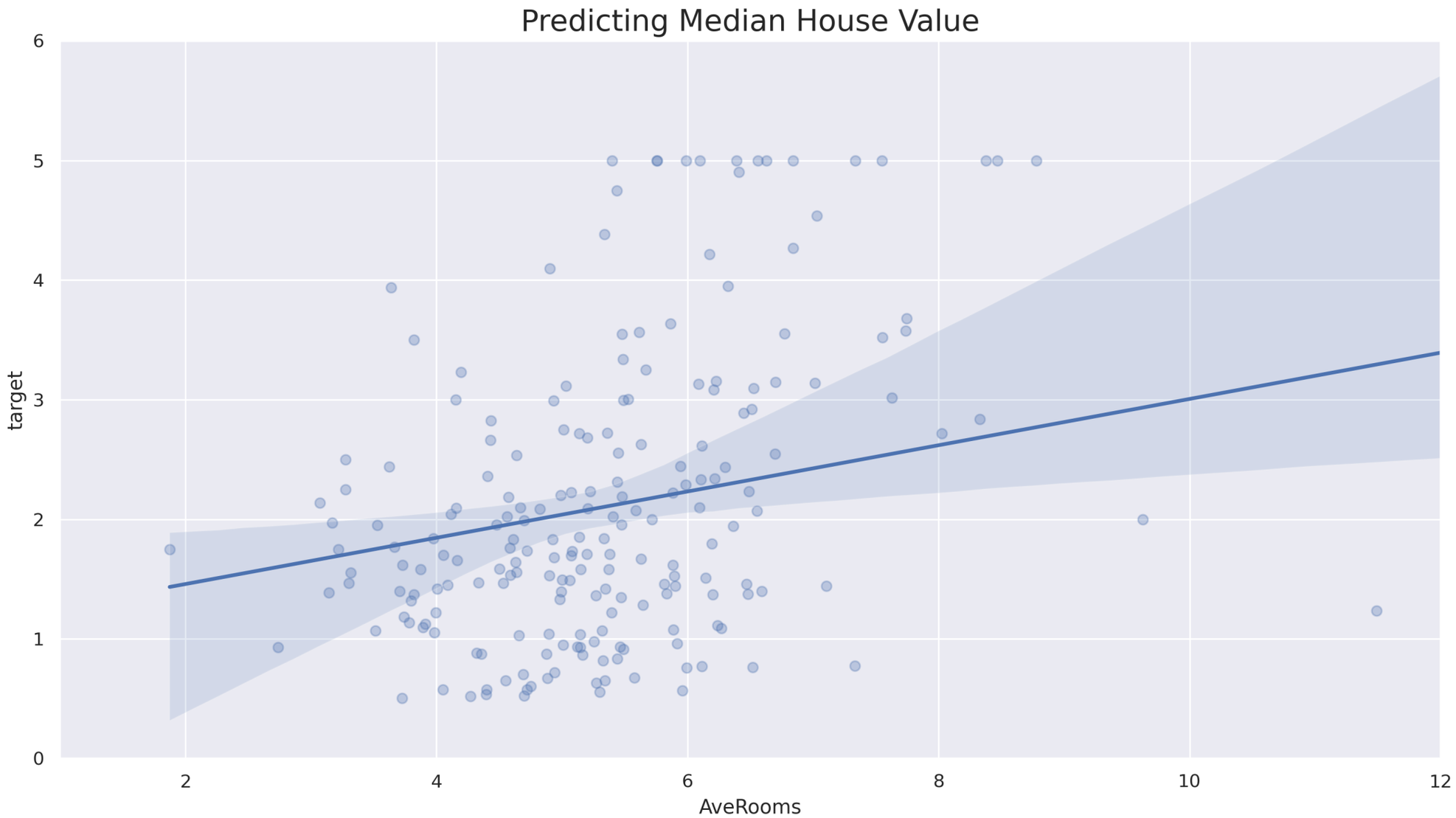
TOPIC - LINEAR REGRESSION

LINEAR REGRESSION

Given a set of data points, Linear Regression gives a straight line that minimizes the square of the distance between the points and the line.

Traditionally the least squares regression line was found using calculus.

LINEAR REGRESSION EXAMPLE



LINEAR REGRESSION IS MACHINE LEARNING!

- Machine learning requires data, and a column to make predictions.
- Linear Regression predicts 1 column (y-value) from another column (x-value).
- Machine learning extends Linear Regression from 2-dimensions to an arbitrary number of dimensions.
- It's called machine learning because the model learns from new data.
- More data results in more accurate models and a better regression line.

LINEAR REGRESSION IN 2-DIMENSIONS

`y = m*x + b`

In math, `x` is the input, `m` is the slope, `b` is the y-intercept, and `y` is the output.

Note that `x` may take on any value (like data points in a column).

The key to finding the line of least squares is to find the right `m`, the slope, and `b`, the y-intercept.

LINEAR REGRESSION IN 2-DIMENSIONS

$$y = mx + b$$

Input	Weights	Output
X	M	Y
	B	

LINEAR REGRESSION IN 2-DIMENSIONS

$$y = mx + b$$

Input	Weights	Output
X	M=2	Y
	B=3	

$$y = 2x + 3$$

LINEAR REGRESSION IN N-DIMENSIONS

$y = mx + b$

$y = b + mx$

$y = B_0 + B_1X_1$

} 2D

LINEAR REGRESSION IN N-DIMENSIONS

$y = mx + b$

$y = b + mx$

$y = B_0 + B_1X_1$

} 2D

$y = B_0 + B_1X_1 + B_2X_2$: 3D

LINEAR REGRESSION IN N-DIMENSIONS

$y = mx + b$

$y = b + mx$

$y = B_0 + B_1X_1$

} 2D

$y = B_0 + B_1X_1 + B_2X_2$: 3D

$y = B_0 + B_1X_1 + B_2X_2 + B_3X_3$: 4D

LINEAR REGRESSION IN N-DIMENSIONS

$y = mx + b$

$y = b + mx$

$y = B_0 + B_1X_1$

} 2D

$y = B_0 + B_1X_1 + B_2X_2$: 3D

$y = B_0 + B_1X_1 + B_2X_2 + B_3X_3$: 4D

$y = B_0 + B_1X_1 + B_2X_2 + B_3X_3 + \ldots$: ND

LINEAR REGRESSION IN N-DIMENSIONS

$$y = mx + b$$

$$y = b + mx$$

$$y = B_0 + B_1X_1$$

2D

$$y = B_0 + B_1X_1 + B_2X_2 \quad : 3D$$

$$y = B_0 + B_1X_1 + B_2X_2 + B_3X_3 \quad : 4D$$

$$y = B_0 + B_1X_1 + B_2X_2 + B_3X_3 + \dots$$

$$y = B_0X_0 + B_1X_1 + B_2X_2 + B_3X_3 + \dots$$

$$y = \sum BX$$

ND

$$y = BX$$

LINEAR REGRESSION IN N-DIMENSIONS

$y = BX$

In machine learning, X is the input, the columns, and B are the weights, the slopes.

X may take on any number of dimensions. If X has 100 columns, then X contains 100 dimensions.

Now B provides a unique weight for each column. If there are 100 columns, then B is a list of 100 different values.

For the final result, each value in a column X is multiplied by the weight B and summed.

LINEAR REGRESSION WITH DATA

X_1 total_bedrooms	X_2 population	X_3 households	X_4 median_income	y median_house_value
1283.0	1015.0	472.0	1.4936	66900.0
1901.0	1129.0	463.0	1.8200	80100.0
174.0	333.0	117.0	1.6509	85700.0
337.0	515.0	226.0	3.1917	73400.0
326.0	624.0	262.0	1.9250	65500.0

LINEAR REGRESSION WITH DATA

	B_1	B_2	B_3	B_4	B_0
	X_1	X_2	X_3	X_4	y
	total_bedrooms	population	households	median_income	median_house_value
	1283.0	1015.0	472.0	1.4936	66900.0
	1901.0	1129.0	463.0	1.8200	80100.0
	174.0	333.0	117.0	1.6509	85700.0
	337.0	515.0	226.0	3.1917	73400.0
	326.0	624.0	262.0	1.9250	65500.0

The key is to find the right B's!

CHOOSING THE WEIGHTS

$y = BX$

The weights, also known as the coefficients, are the range of values for B.

At first try, B is completely random.

After comparing the random predictions to the results, B is adjusted on whether it's too high or too low using gradient descent (calculus).

During each subsequent iteration, B is adjusted until the error is minimized.

After the error has been minimized for B, the final model is selected.

HOW LINEAR REGRESSION WORKS WITH CODE

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()

model.fit(X,y)

model.score(X,y)

model.predict(new_X)
```

HAPPY CODING!