Augustine Joseph

Statistics for Data Science

Linear Regression 101

What is Linear Regression?

Linear regression is a machine learning algorithm used to predict a number (continuous outcome) based on one or more input factors (features). It's widely used in data science for tasks like predicting sales, pricing, and forecasting.

Why is it Important?

Linear regression is easy to understand and implement, making it one of the first algorithms data scientists learn. It helps you see how changes in one or more features impact the outcome, which is useful in many real-world scenarios.

Our Example:

In this guide, we will use linear regression to predict house prices based on features like square footage, lot size, number of rooms, and the city/cost of living.

Supervised Learning

Definition and Overview:

In supervised learning, the machine is given data where the correct answers (labels) are already known. The model's job is to learn the pattern between the input data and the output labels, so it can predict the right answers for new, unseen data.

Example - House Price Prediction:

Imagine you have historical data on house prices.

- You know the square footage, lot size, number of rooms, and cost of living for each house, and you also know what the actual sale price was.
- You feed this data to the model (supervised learning), and the model learns how these features (inputs) relate to the house price (output).
- After training, it can predict the price for a new house with similar features.

Supervised Learning

Before diving into Linear Regression, it's important to first understand its broader category: Supervised Learning. Linear regression is one of the key methods within supervised learning.

What is Supervised Learning?

Supervised learning is a method in machine learning where the model learns by being shown examples that have both input data and the correct answers (called labels).

- Think of it like teaching a child to recognize objects by showing them a picture of a cat and telling them it's a cat. The goal is for the model to predict th
- right
  answer for new, unseen data based on what it learned from the labeled examples.

Linear Regression

Linear regression is one of the simplest algorithms in machine learning, and understanding Math behind it establishes a solid foundation.

Mathematics Behind Linear Regression:

General Equation: Linear regression models the relationship between inputs and an outcome using the equation of a straight line. The simplest form of the equation is:

$$y=\beta 0 +\beta 1\, x1 +\beta 2\, x2 +\ldots+\beta n\, xn +\varepsilon$$

Here,

- y (like house price) is what we are trying to predict.
- x1,x2,...xn represent the input features (like square footage, lot size, etc.)
- β1,β2,...βn are the coefficients that tell us how much each feature affects the outcome.

Mathematics Behind Linear Regression

House Price Prediction Example:

For our house price prediction, the equation could look like this:

$$\text{House Price} = \beta_0 + \beta_1(\text{Square Footage}) + \beta_2(\text{Lot Size}) + \beta_3(\text{Number of Rooms}) + \beta_4(\text{City/Cost of Living}) + \varepsilon$$

Each coefficient (like β1 for square footage) shows how much the price changes with an increase in that feature.

For example, for every 100 square feet added, the price might increase by a certain amount, depending on the value of β1.

Linear Algebra Representation:

We also represent the data in matrix form. This allows us to handle large datasets and multiple features more efficiently. The vector notation is simply a compact way of writing the same equation.

Matrix Form of Linear Regression

The general linear regression equation is:

$$y=\beta 0+\beta 1x1+\beta 2x2+\ldots +\beta nxn+\varepsilon$$

This can be written in matrix form as:

$$y=X\beta+\varepsilon$$

Where:

- y is the vector of the target variable (house pric
- X is the matrix of feature values (square footag size, number of rooms, etc.).
- β is the vector of coefficients.
- ε is the vector of error terms.

Matrix Form of Linear Regression

The general linear regression equation is:

$$y=\beta 0+\beta 1x1+\beta 2x2+\ldots +\beta nxn+\varepsilon$$

This can be written in matrix form as:

$$y=X\beta+\varepsilon$$

Where:

- y is the vector of the target variable (house pric
- X is the matrix of feature values (square footag size, number of rooms, etc.).
- β is the vector of coefficients.
- ε is the vector of error terms.

Matrix Form of Linear Regression

Example:

Let's say we have data for 3 houses and 4 features: square footage, lot size, number of rooms, and cost of living (city). We can represent this in matrix form.

$$\mathbf{X} = \begin{bmatrix} 1 & \text{sq\_ft}_1 & \text{lot\_size}_1 & \text{rooms}_1 & \text{cost\_living}_1 \\ 1 & \text{sq\_ft}_2 & \text{lot\_size}_2 & \text{rooms}_2 & \text{cost\_living}_2 \\ 1 & \text{sq\_ft}_3 & \text{lot\_size}_3 & \text{rooms}_3 & \text{cost\_living}_3 \end{bmatrix}$$

The "1" in the first column corresponds to the intercept β0

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix}$$

Now, the equation becomes:

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon$$

Where:

$$\mathbf{y} = \begin{bmatrix} \text{price}_1 \\ \text{price}_2 \\ \text{price}_3 \end{bmatrix}$$

This representation allows you to solve for the vector β (the coefficients) in a more efficient and scalable way when you have many features and data points.

Finding the Right Coefficients

Ordinary Least Squares (OLS):

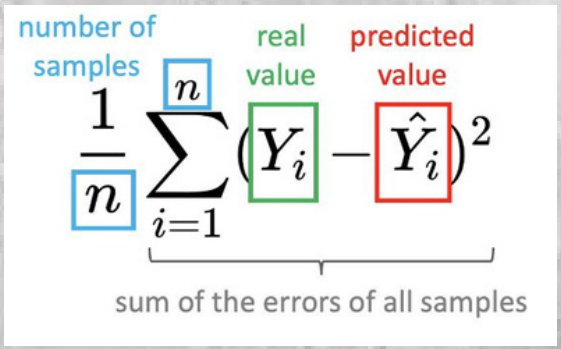
In linear regression, we use a method called Ordinary Least Squares (OLS) to find the best coefficients (β1, β2,...βn) that minimize the error between the predicted and actual values. OLS tries to draw a line through the data that fits it as closely as possible.

What is a Loss Function?

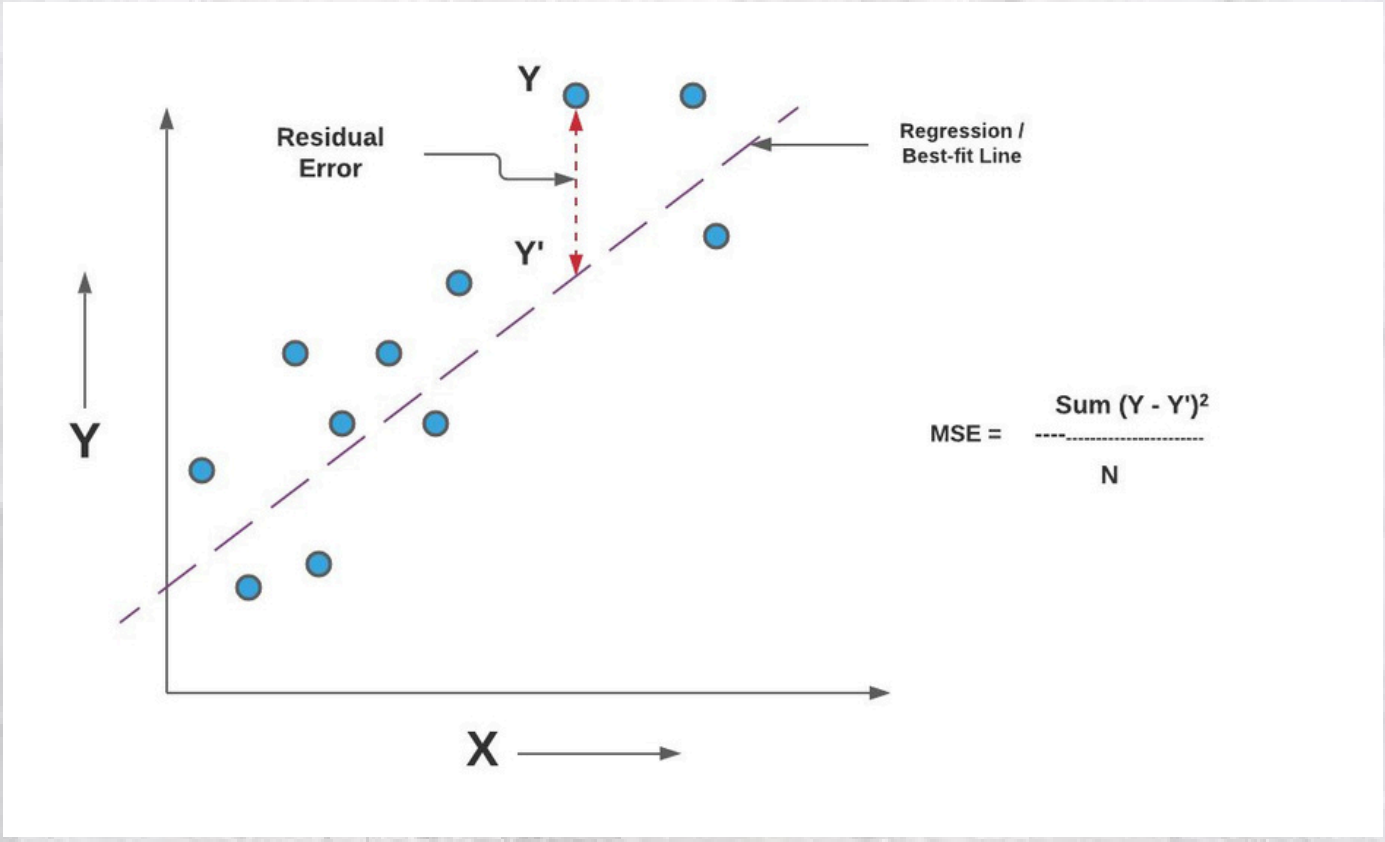
A loss function is a measure of how far off the model's predictions are from the actual outcomes. In the case of linear regression, we use Mean Squared Error (MSE) as the loss function. It calculates the average of the squared differences between the predicted values and the actual values.

Mean Squared Error (MSE):

The MSE tells us how much our model's predictions deviate from the actual outcomes. Squaring the errors makes sure that large errors are penalized more. The equation is:



Mean Squared Error (MSE)



House Price Example:

In our house price prediction example, if the actual price of a house is $500,000 but our model predicts $450,000, the error is $50,000. The MSE gives us a way to calculate the average of these errors across all the houses in our dataset.

Optimization Techniques

Concept of Optimization: Optimization is the process of adjusting the coefficients (like β1,β2,...βn) so that the loss function (MSE) is minimized. The goal is to find the best line (or hyperplane in multiple dimensions) that fits the data.

Gradient Descent: Gradient Descent is a popular method for optimization. It works by taking small steps in the direction that reduces the loss function. Think of it like walking downhill to find the lowest point in a valley.

How Gradient Descent Works: Gradient Descent updates the coefficients using the formula:

$$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} MSE$$

Where α\alphaα is the learning rate, which controls how big each step is. Too big, and you might overshoot the minimum. Too small, and it will take a long time to get there.

House Price Example: In predicting house prices, gradient descent adjusts the coefficients for square footage, lot size, etc., with each iteration to reduce the prediction error (MSE) until the model finds the best-fitting line.

Assumptions of Linear Regression

Linearity of Relationships: The relationship between the features and the target variable should be linear. This means that as one feature changes, the target variable should change proportionally.

Independence of Errors: The errors (or residuals) should not be correlated with each other. If they are, it means there might be some pattern in the errors that the model is missing.

Homoscedasticity: The variance of the errors should be constant across all levels of the input features. If the spread of errors increases or decreases for different values of the features, the model might not be well-fitted.

No Multicollinearity: The input features should not be too highly correlated with each other. If two features are very similar (e.g., square footage and lot size), it's hard to tell which one is actually affecting the outcome, making the model unstable.

Normal Distribution of Errors: The errors should follow a normal distribution, which helps ensure that the model's predictions are unbiased and accurate.

Advanced Topics

- Overfitting and Underfitting:
  - Overfitting: When the model is too complex and captures not just the true relationship but also the noise in the data. This means it works well on the training data but fails to generalize to new data.
  - Underfitting: When the model is too simple and doesn't capture the true relationship, leading to poor performance even on the training data.

- Regularization (Lasso and Ridge): Regularization techniques like Lasso (L1) and Ridge (L2) add a penalty to the loss function to prevent overfitting by reducing the size of the coefficients.

Augustine Joseph

Was this Helpful?

 Save it

 Follow Me

github.com/augustine-aj