Everything You Need to Know About Linear Regression

Tanvi Penumudy · Follow

Published in Analytics Vidhya

9 min read · Jan 8, 2021

__Linear Regression__ is one of the most widely used Artificial Intelligence algorithms in real-life Machine Learning problems — thanks to its simplicity, interpretability and speed! We shall now understand what's behind the working of this algorithm in the next few minutes!

Source: Google Images

What is Linear Regression?

Linear regression attempts to model the relationship between two variables by fitting a **linear** equation to observed data. One variable is considered to be an *explanatory variable* or an *independent variable*, and the other is considered to be a *dependent variable — Source: **Yale.edu***

Applications and Approaches of Linear Regression

Business Applications

Linear regression is used for a wide array of business prediction problems:

1. *Stock prediction*
2. *Predict future prices/costs*
3. *Predict future revenue*
4. *Comparing performance of new products*

Benefits of Linear Regression

1. *Ease*
2. *Interpretability*
3. *Scalability*
4. *Deploys and Performs well on Online Settings*

Machine Learning approaches to Linear Regression

1. *Simple and Multiple Linear Regression*
2. *Polynomial Regression*
3. *Ridge Regression and Lasso Regression (upgrades to Linear Regression)*
4. *Decision Trees Regression*
5. *Support Vector Regression (SVR)*

Linear Regression in Layman's Terms

> *You can think of linear regression as the answer to the question **"How can I use X to predict Y?"**, where X is some information that you have and Y is some information that you want to know.*

Let's look at this following classic example — Source

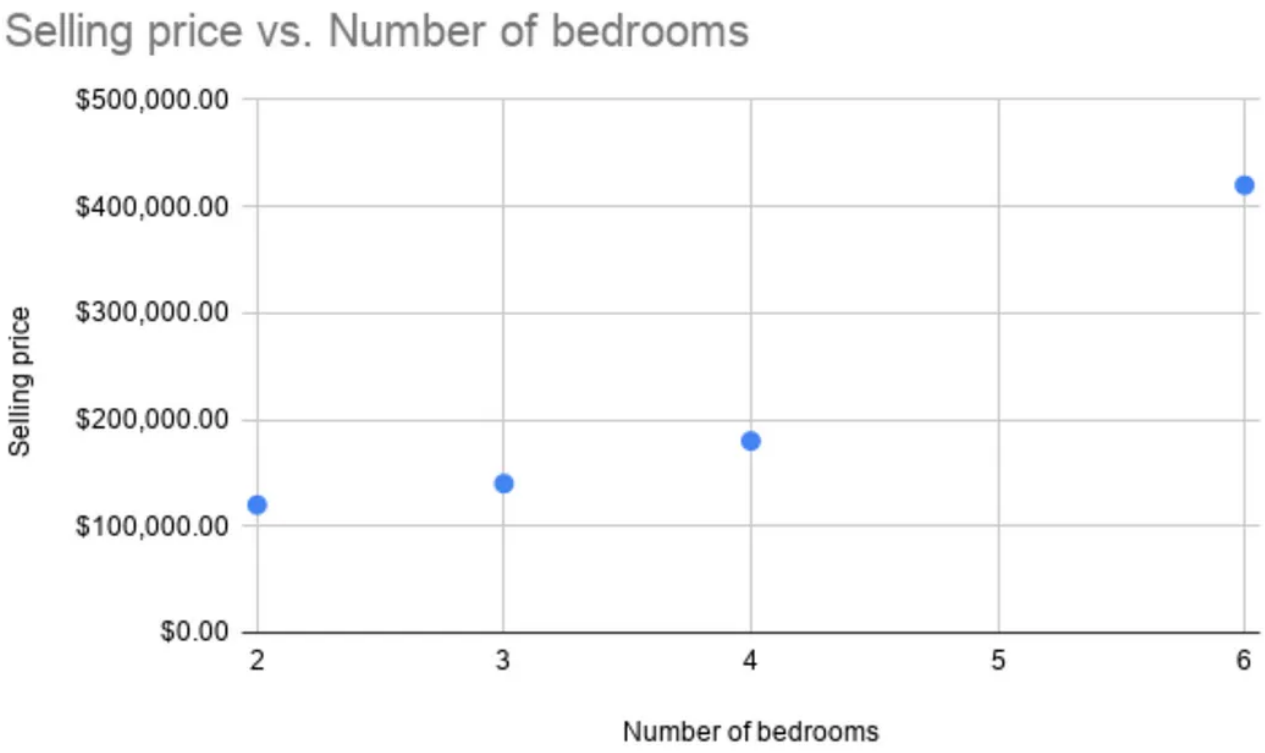
You might want to sell your house and you have information regarding the *number of bedrooms* ***(X)*** in that house and here, the catch is to find the *price of the house* ***(Y)***.

> *Linear regression creates an equation in which you input your given numbers (X) and it outputs the target variable that you want to find out (Y).*

In this case, we would use a dataset containing historic records of house purchases in the form of ***(“number of bedrooms”, “selling price”)***:

Number of bedrooms	Selling price
2	$120,000
3	$140,000
4	$180,000
6	$420,000

> *Let’s now visualize the same data*



Number of Bedrooms on **X-axis** and **Selling Price** on **Y-axis**

> *Looking at the scatter plot, it seems that there is a trend: the more bedrooms that a house has, the higher its selling price (which is not surprising, to be honest).*

Now, let's say that we trained a linear regression model to get an equation in the form:

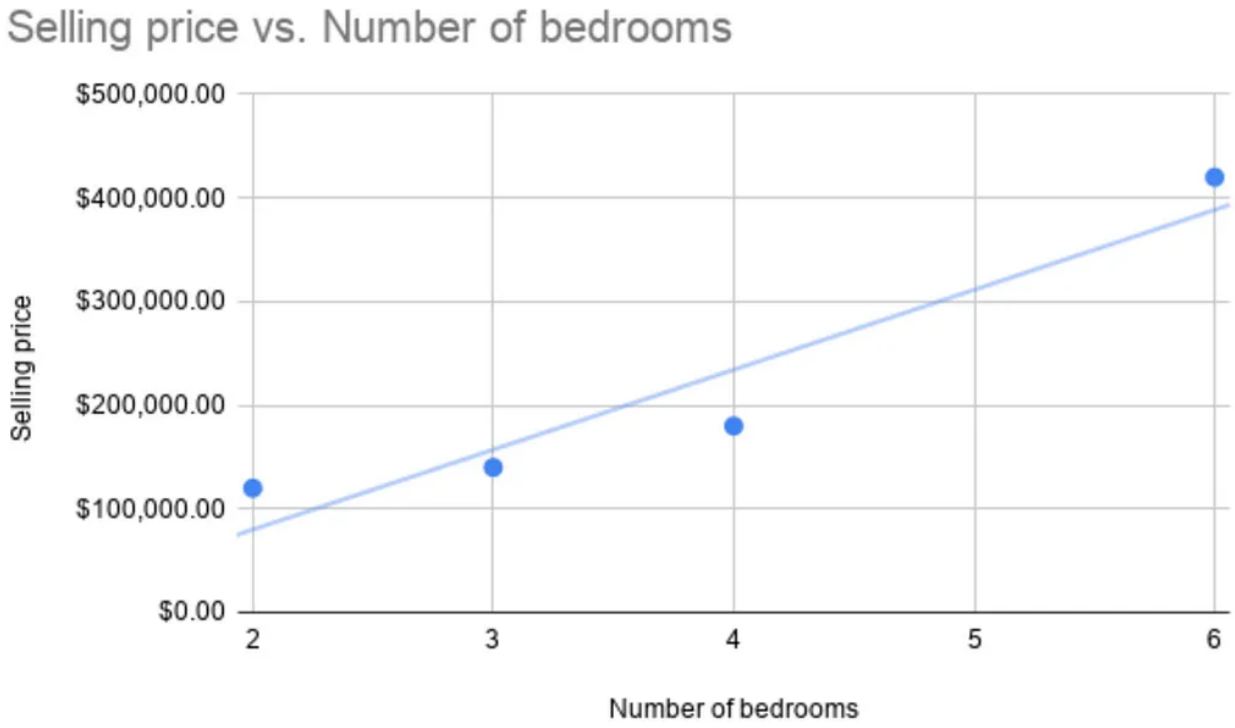
*Selling price = 77,143 * (Number of bedrooms) — 74,286*

> *The equation acts as a prediction. If you input the number of bedrooms, you get the predicted value for the price at which the house is sold.*

For the specific example above:

*Your selling price = 77,143 * 2 bedrooms — 74,286 = 80,000*

In other words, you could sell your 2-bedroom house for approximately $80,000. But linear regression does more than just that. We can also visualize it graphically to see what the price would be for houses with a different number of bedrooms —



> *This is because linear regression tries to find a straight line that* ***best fits the data****. Linear regression is not limited to real-estate problems: it can also be applied to a variety of business use cases.*

Mathematical Interpretation

Simple Linear Regression

Linear regression is such a useful and established algorithm, that it is both a statistical model and a machine learning model. Here, we will focus mainly on the machine learning side, but we will also draw some parallels to statistics in order to paint a complete picture.

Once trained, the model takes the form of a linear regression equation of this type:

$$y = w_o + w_1 x$$

Terms in this equation —

- ***y*** is the output variable. It is also called the *target variable* in machine learning, or the *dependent variable* in statistical modeling. It represents the continuous value that we are trying to predict.
- ***x*** is the input variable. In machine learning, x is referred to as the *feature,* while in statistics, it is called the *independent variable*. It represents the information given to us at any given time.
- ***w0*** is the *bias term* or y-axis intercept.
- ***w1*** is the *regression coefficient* or scale factor. In classical statistics, it is the equivalent of the slope on the best-fit straight line that is produced after the linear regression model has been fitted.
- ***wi*** are called *weights* in general.

> *The goal of the regression analysis (modeling) is to find the values for the unknown parameters of the equation; that is, to find the values for the weights w0 and w1*

Multiple Linear Regression

Both simple and multiple linear regressions assume that there is a linear relationship between the input variable(s) and the output target variable.

The main difference is the number of independent variables that they take as inputs. Simple linear regression just takes a single feature, while multiple linear

regression takes multiple x values. The above formula can be rewritten for a model with n-input variables as:

$$y = w_0 + w_1x_1 + w_2x_2 + ...+ w_nx_n$$

> *Where xi is the i-th feature with its own wi weight.*

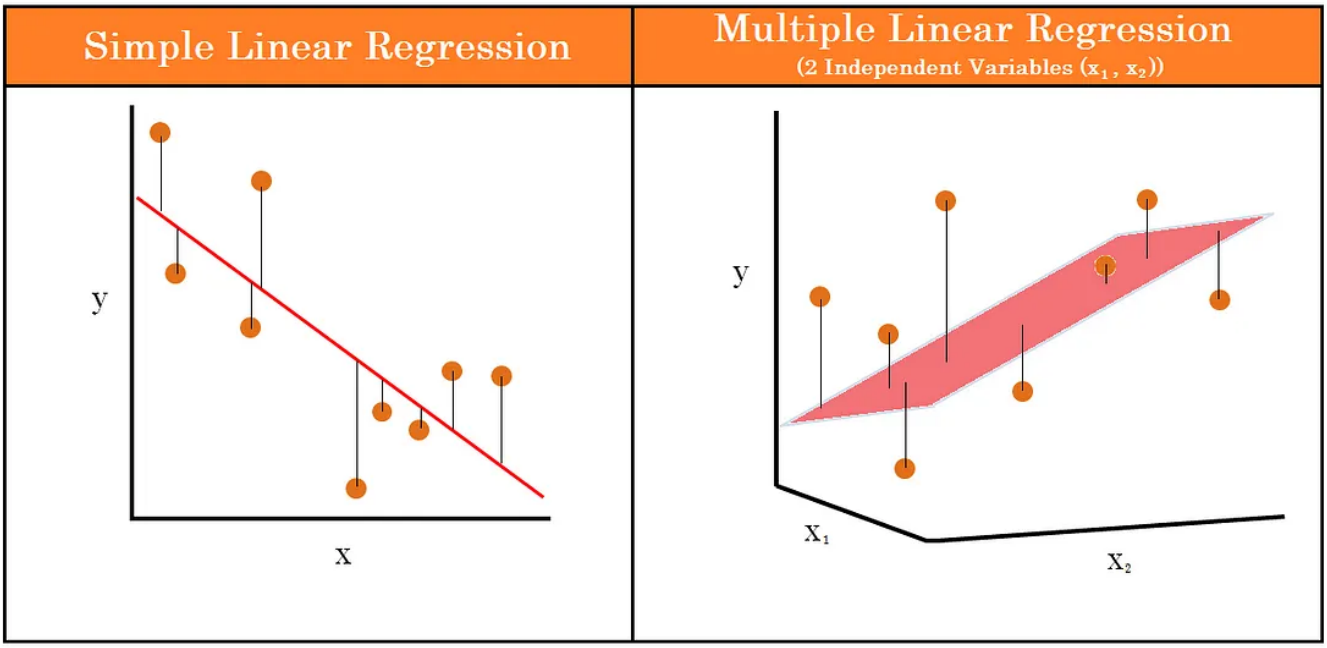


Image Source: Keboola

> *The simple linear regression model can be represented graphically as a best-fit line between the data points, while the multiple linear regression model can be represented as a* ***plane (in 2-dimensions)*** *or a* ***hyperplane (in higher dimensions).***

Despite their differences, both the simple and multiple regression models are linear models — they adopt the form of a *linear* equation. This is called the linear assumption.

> *Quite simply, it means that we assume that the type of relationship between the set of independent variables and independent features is linear.*

Training an LR Model

We train the linear regression algorithm with a method named ***Ordinary Least Squares*** — ***OLS***(or just Least Squares). The goal of training is to find the weights wi in the linear equation ***y = wo + w1x.***

1. Random weight initialization. In practice, w0 and w1 are unknown at the beginning. The goal of the procedure is to find the appropriate values for these model parameters. To start the process, we set the values of the weights at random or just initialize with 0.

> *There are other mathematical justifications such as **Xavier Initialization**, etc.*

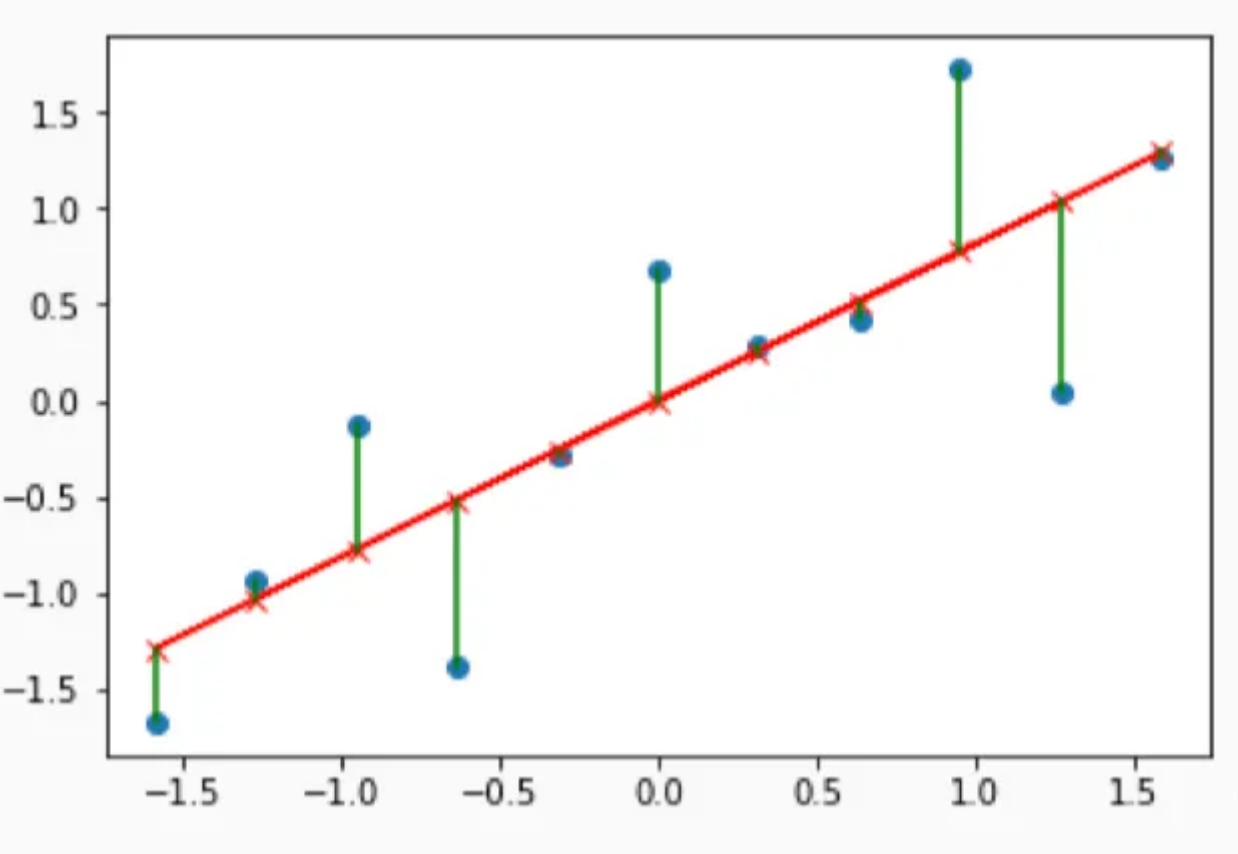
2. Input the initialized weights into the linear equation and generate a prediction for each observation point.

3. Calculate the ***Residual Sum of Squares*** *(RSS). Residuals, or error terms*, are the difference between each ***actual output*** and the ***predicted output***.

> *They are a point-by-point estimate of how well our regression function predicts outputs in comparison to true values. We obtain residuals by calculating* actual values — predicted values *for each observation.*

We ***square the residuals*** *for each observation point and sum the residuals to reach our RSS.*

The basis here is that a ***lower RSS*** means that our line of ***best fit*** comes closer to each data point and the vice versa.



The closer the actual values are (blue points) to the regression line (red line), the better.

4. Model parameter selection to minimize RSS. Machine learning approaches find the best parameters for the linear model by defining a *cost function* and *minimizing* it via ***gradient descent.***

> *By doing so, we obtain the best possible values for the weights.*

Cost Function

In the case of linear regression, the cost function is the same as the residual sum of errors. The algorithm solves the minimization problem and is achieved using *Gradient Descent.*

Gradient Descent

Gradient descent is a method of changing weights based on the loss function for each data point. We calculate the sum of squared errors at each input-output data point. We take a *partial derivative* of the *weight* and *bias* to get the *slope of the cost function* at each point.

> *Based on the slope, gradient descent updates the values for the set of weights and the bias and re-iterates the training loop over new values (moving a step closer to the desired goal).*

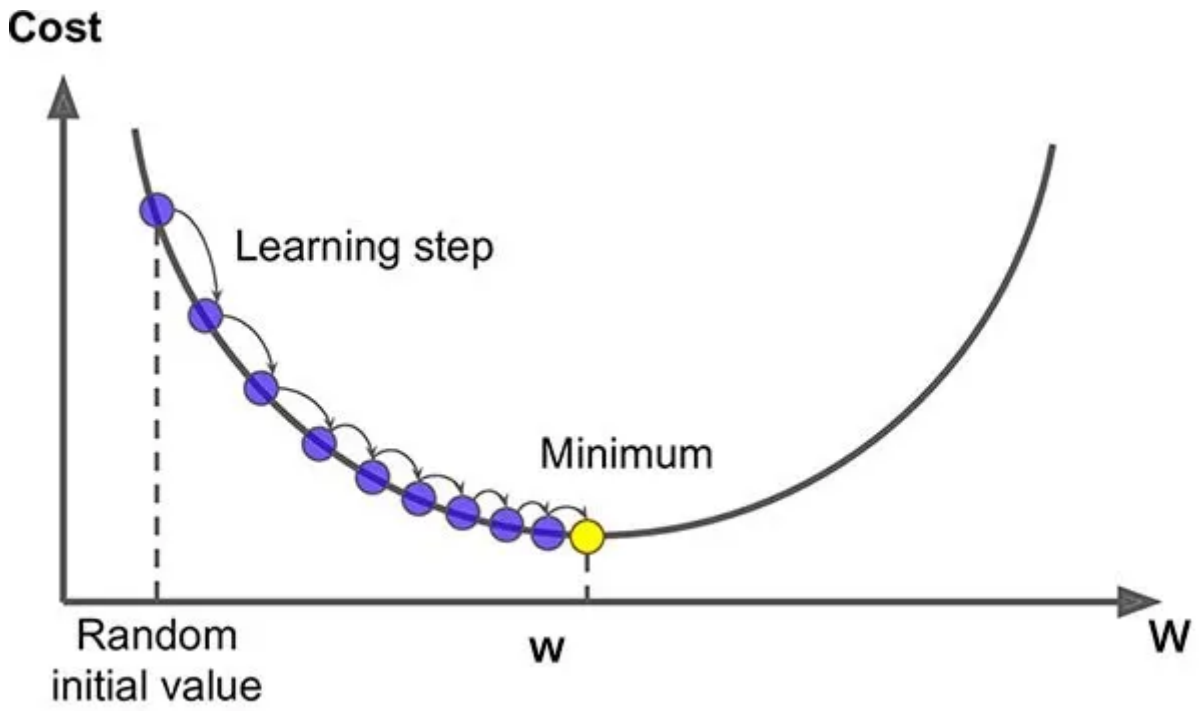


Image Source: Morioh — Gradient Descent Illustration

This iterative approach is repeated until a minimum error is reached, and gradient descent cannot minimize the cost function any further, it depends on various hyperparameters in which one of the crucial ones is the ***Learning Rate***.

> *The **learning rate** refers to how much the parameters are changed at each iteration. If the learning rate is too high, the model fails to converge and jumps from good to bad cost optimizations. If the learning rate is too low, the model will take too long to converge to the minimum error.*

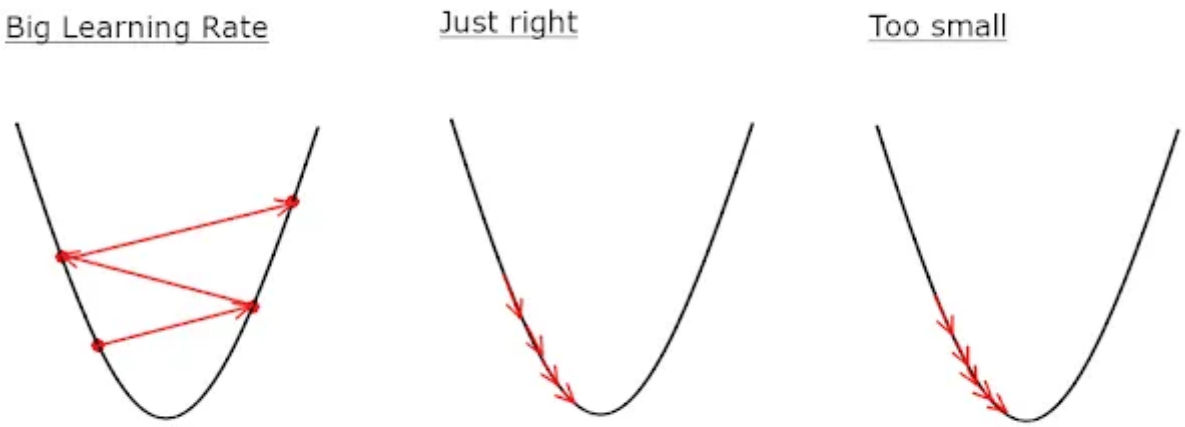


Image Source: Kaggle — Illustration of Learning Rate

Evaluating our Model

How do we evaluate the accuracy of our model?

> *First of all, you need to make sure that you train the model on the training dataset and build evaluation metrics on the test set to avoid overfitting. Afterwards, you can check several evaluation metrics to determine how well your model performed.*

There are various metrics to evaluate the goodness of fit:

Mean Squared Error (MSE)

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE is computed as RSS divided by the total number of data points, i.e. the total number of observations or examples in our given dataset. MSE tells us what the average RSS is per data point.

Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

RMSE takes the MSE value and applies a square root over it. RMSE can be directly used to interpret the 'average error' that our prediction model makes.

R2 or R-squared or R2 Score

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \mu)^2}$$

R-squared is a measure of how much variance in the dependent variable that our linear function accounts for.

> *Once we have trained and evaluated our model, we improve it to make more accurate predictions.*

Improving our Model

There are multiple methods to improve your linear regression model.

Data Preprocessing

The biggest improvement in your modeling will result from properly ***cleaning your data.***

Make sure to:

1. ***Remove outliers:*** Outliers in the quantitative response y skew the slope of the line disproportionately. Remove them to have a better-fitted line.

2. ***Remove multicollinearity:*** Create a *correlation matrix* for all of your features to check which pairs of features suffer from high correlation. Remove these features to keep just one.

3. ***Assert normal distribution:*** The model assumes that the independent variables follow a ***Gaussian distribution***. Transform your variables with log transform if they are not normally distributed.

4. ***Assert linear assumption:*** If your independent variables do not have a linear relationship with your predictor variable, log transform them to reshape polynomial relationships into linear.

> *For more on* ***Probability Distributions*** *refer —* *https://medium.com/datadriveninvestor/mathematics-for-machine-learning-part-5-8df72392ec10*

Feature Scaling

Features can come in different orders of magnitude. Features of different scales convert slower (or not at all) with gradient descent.

Normalize and Standardize your features to speed up and improve model training.

Regularization

Regularization *(not usually used for Simple Regression)* can be thought of as a ***feature selection method.***

> *Whereby features with lower contributions to the goodness of fit are removed and/or diminished in their effects, while the important features are emphasized.*

There are two regularization techniques that are frequently used in linear regression settings:

1. ***Lasso L1 Regression*** — uses a penalty term to remove predictor variables, which have low contributions to overall model performance

2. ***Ridge L2 Regression*** — uses a penalty term to lower the influence of predictor variables (but does not remove features)

Kaggle Challenges for Beginners

1. *Predict the salary based on years of work experience*