**ML Algorithms**

1. **Liner Regression:** *It is supervised machine learning algo.* 
   1. **Simple Linear regression**
      1. *It has only single input and single output or label column*
   2. **Multiple Linear regression**
      1. *If data has more than one input column*
   3. **Polynomial Linear regression**
      1. *If your data is not linear than you will use this algo*
   4. **Regularization**

**How to proceed for solving ML Problems?**

1. Plot your data.
2. Separate your Input and output/label columns
3. Do train train-test split
4. Train the model on training set.
5. Predict the with test set.

In Regression y = mx+b . where m is weightage of x (how much x depends on y) and b is minimum value if mx will zero.

**Mathematical intuition of Linear Regression:**

**Y = mx + b**

How to find the values of m and c:

1. **Closed form**(used when dimensions are low)
   1. OLS (Ordinary least square)
2. **Non-Closed form**(used when dimensions are high)
   1. Gradient descent (*used due to low time complexity*)

**OLS (Direct formula):**

For simple linear regression b =

m = error function =

where d = distance from best fit line. and d =

So we can rewrite the error function as =

Where

Then the error function will be

**We want to minimize the error for best fitting the line**

Now from this point we have to do some calculus partial derivation

Partial derivation of y, m = 0 and b = -1

By dividing both side with -2 we get

Now we write it as =

Now

So we can write the whole equation as = b =

Where n is the number of rows in the table

Now we have value of b we can now get the value (m) with the help derivation and the equation will be = m =

**For multiple linear regression**

Formula =

Where n = number of rows and m = number of columns

**Gradient Descent**

*Gradient descent is a first-order iterative optimization algorithm for finding a local differentiable function. The idea is to take repeated steps in the opposite direction of the gradient or approx. gradient of the function at the current point, because this is the direction of steepest descent.*

Note: It is just an optimization technique which gives you the minima of function you gave to it.

Used in:

1. Linear Regression
2. Logistic Regression
3. TSNE
4. It is back bone of Deep Learning

Mathematical formulation:

Steps. Start with a random value of b and m

Let epoch = 100 and ƞ(learning rate) = 0.01

we want to minimize the loss here we know the values of y and x, the loss function depends on the m and b, let assume for understanding we know the m and loss function depends only on b

We will do derivation w.r.t (b)

We will get the b\_slope =

We will do derivation w.r.t (m)

We will get the m\_slope =

For x in epoch:

bnew = bold  - ƞ\*slope(b)

mnew = mold - ƞ\*slope(m)

**Regression Metrics**

1. MAE (*mean absolute error*)
   1. y^ = prediction

(It’s not differentiable at zero due to modulus, Robust to outliers)

1. MSE (*mean squared error*)

(It’s differentiable at zero due to square, not Robust to outliers)

1. RMSE (*root over MSE*)

(not Robust to outliers)

1. R2 Score ( *not affected by data context works on every type of data*)
2. Adjusted R2 SME