a) if our features follow: Normal / Gaussian Distribution → model will get trained well!

if it doesn't follow: Normal / Gaussian Distribution, will try to do feature transformation to convert the data into Normal / Gaussian Distribution, if possible.

b) Standardization: Scaling the data → z-score, $\mu=0,\ \sigma=1$

use whenever there's Gradient Descent and find the global minima value, as soon as possible!

Helps to optimize the model & use to increase the training time of the model.

c) EQUATION:

y = mx+c

y = β0+β1x

hϴ(x) = ϴ0+ϴ1x   i.e PREDICTED VALUE

d) BEST FIT LINE: distance should be minimal between the predicted and real value!

i) will do it by applying the cost function

COST FUNCTION = $J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

ii) minimize the cost function to get the best fit line:

Method: GRADIENT DESCENT, will get the ϴ1 values but which one is right among them?

Will get that by a term known as CONVERGENCE ALGORITHM

Later the best pt will get is known as GLOBAL MINIMA

e) PERFORMANCE MATRIX: helps to measure: How good our model is?

i) R^2: bigger the better

ii) Adjusted R^2: helps to remove independent variable which is least useful

iii) MAE

iv) RMSE

f) SPLIT DATA: Training & Test Set; Common split ratio: **80% training, 20% testing** (or **70%-30%**).

g) CROSS – VALIDATION: K-Fold Cross-Validation!

The dataset is divided into k subsets. The model is trained k times, each time using a different fold as the test set and the remaining k−1 folds as the training set.

h) After finally training & testing our data we get to know if there's imbalance in the data, if yes it has **OVERFITTING** (Train: Good, Test: Poor; Low Bias, High Variance) or

UNDERFITTING (Train: Poor, Test: Good; High Bias, Low Variance)!

To solve OVERFITTING, will do **REGULARIZATION** (Ridge & Lasso)!

To solve UNDERFITTING, will do the following:

i) **Reduce Regularization**
ii) **Use a more complex model:** Polynomial Regression, Decision Trees, Random Forests, or Gradient Boosting Machines, AdaBoost, Neural Networks. These methods combine several weak learners to create a more powerful model that can capture complex relationships.
iii) **Increase the number of features or use feature engineering**:
Create new features, use feature transformations: Apply transformations like log transformations, polynomial features, or scaling to improve the model's ability to capture relationships.
iv) Train for more epochs (in case of neural networks) or increase the number of iterations in gradient-based algorithms like gradient descent. Sometimes the model hasn't had enough time to fully learn from the data, leading to underfitting.
v) **Add more data**
vi) **Remove noise or outliers**
vii) Try models that can handle nonlinear relationships between the features and the target, such as: Support Vector Machines (**SVM**) with nonlinear kernels.
viii) **Hyperparameter tuning:** Sometimes underfitting is due to suboptimal hyperparameters. Use techniques like Grid Search or Random Search to tune key parameters like learning rate, number of trees, maximum depth of trees, etc. in decision trees or ensemble models.

i) **Solve Overfitting:**

1) REGULARIZATION:

i) **Ridge Regression** (L2 regularization):

α value increases the coefficient of the magnitude decreases/shrinks/scale downs almost to 0

- ➔ Prevent overfitting!

ii) **Lasso Regression** (L1 regularization):

α value increases the coefficient of the magnitude decreases/shrinks/scale downs to exactly 0

- ➔ Prevent Overfitting & achieves Feature Selection!

j) Solve Underfitting:

1) REDUCE REGULARIZATION:

i) If we're using regularization techniques (e.g., Lasso or Ridge regression) and experiencing underfitting, it might be because our regularization strength is too high.

ii) Regularization shrinks the coefficients, and too much regularization can make the model too simple.

iii) Reduce the regularization strength by lowering the λ parameter. This allows the model to use more of the available features and coefficients, which can improve performance.

k) Common Problems & Solutions:

i) **Multicollinearity:**

Occurs when independent variables are highly correlated, leading to instability in coefficient estimation.

Solution: Remove highly correlated features **(VIF)** or use **Principal Component Analysis (PCA)**.

ii) **Homoscedasticity**

Assumption that the variance of errors is constant across all levels of the independent variable(s).

Solution: If violated, try transforming the dependent variable (e.g., log transformation).

l) **Model Evaluation on Test Data:**

Once the model has been trained, evaluate it using metrics such as **MSE**, **R^2**, **RMSE**, etc. This helps assess how well the model is generalizing to unseen data.