Data Preprocessing.

→ ① Handle inappropriate data. (pandas) ← both numeric / non-numeric supported

→ ② Deal with missing values. (pandas, sklearn) ← col^ns must be strictly numeric

→ ③ Deal with Categorical data. (pandas, sklearn)
- pandas: get dummies, concat
- sklearn: LabelEncoder, OneHotEncoder

→ ④ Deal with numerical data.
- Feature scaling

* Dealing with Categorical data.

Example →

eid	ecity	esal
1	Mumbai	1000
2	Bangalore	2000
3	Mumbai	1500
4	Chennai	1800

Categorical data doesn't hold any mathematical weightage.

need a strategy that can convert given col^n into numerical col^n without adding any mathematical weightage.

Algo to deal with Categorical data:

① Get the unique value of the categorical column and store the same in the list.

['Mumbai', 'Bangalore', 'Chennai']

② Sort the list in ascending order.

['Bangalore', 'Chennai', 'Mumbai']
0 1 2

③ Replace the column values with the respective index values from the list. (created on step2)

Original dataset

eid	ecity	esal
1	Mumbai	1000
2	Bangalor	2000
3	Mumbai	1500
4	Chennai	1800

⇒

eid	ecity	esal
1	2	1000
2	0	2000
3	2	1500
4	1	1800

LABEL ENCODING

(Problem - We introduced mathematical weightage)

④ Scale different categorical columns and substitute the data accordingly (to remove mathematical weightage)

* DUMMY VARIABLES

eid	ecity	esal
1	2	1000
2	0	2000
3	2	1500
4	1	1800

⇒

0	1	2	eid	esal
0	0	1	1	1000
1	0	0	2	2000
0	0	1	3	1500
0	1	0	4	1800

ONE HOT ENCODING

* To remove mathematical impact from labelled columns.

Dealing / Tuning Numerical Data.

Example:

age	salary
27	80000
26	85000
18	10000
12	35000

Given dataset,

range age : 10 → 30

salary : 10000 → 85000

Every ML algo expects your data to follow a standard scale / standard range without impacting the mathematical weightage of the same

To achieve the above,

Feature Scaling: (applicable only for feature column)

FEATURE ENGINEERING

→ ① Rescale the feature in a limited range which is decided by Data Scientist.

→ PROD PREFERRED ② * Standardize the data such that the mean is zero and standard deviation is 1.

→ ③ Normalize the data. (subject to algo performance)

- ⓐ L1 normalization
- ⓑ L2 normalization

Feature Scaling is MANDATORY ONLY FOR DIMENSIONALITY REDUCTION (PCA, LDA)

Technique 1 → Rescale the feature in a limited range which is decided by Data Scientist.

MinMax Scaler ⇒ user can define the range of data.

$$x_i' = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

where,

x → feature column / vector

x_i' → individual element calculated

x_i → individual feature

Standard Scaler:

$$x_i' = \frac{x_i - \bar{x}}{\sigma} \quad \ldots \text{ Z-score statistics}$$

where

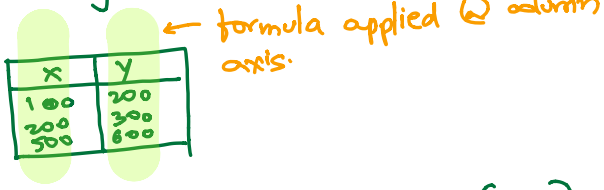
x_i' → individual element calc.

$\bar{x}$ → sample mean of feature col^n.

σ → standard deviation of feature col^n

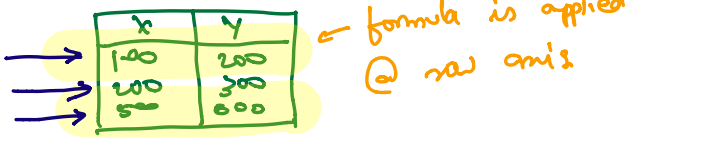
Feature Scaling :- Rescaling values such that the mathematical weightage remains the same.

Feature Scaling is applied column wise



Normalization :- Rescaling values in the range of (0, 1)

Normalization is applied row wise.



Normalization is ideally used to decrease error in Regression algo.

optimizer / Catalyst

- Gradient Descent
- Stochastic GD
- Batch GD

} not used in ML widely

extensively used in Deep Learning / Artificial Neural Networks

Formula:

1. Averaging Scale
2. RangeBound Scale
3. L1 normalization → Regression (Ridge Regression)
4. L2 normalization → Regression (Lasso Regression)

Supervised Learning

Features → input
Label → output

Regression

features : any
label : numeric.

Classification

Features: any
Label: categorical.

① Linear Regression
- Simple linear regression → one feature, one label
- multiple linear regression → 'n' features, one label
- Polynomial linear regression → one feature, one label.

↑ Polynomial features of one feature

x^n ← ↑ one.

② Decision Tree Regression

③ Random Forest Regression

④ Support Vector Regression

⑤ XG boost (extreme gradient boosting)

Linear Regression :-

Algebra → Slope-intercept form!

$$y = mx + c$$

where

$m \rightarrow$ slope

$c \rightarrow$ intercept

$x \rightarrow$ feature

$y \rightarrow$ label.

Rewrite the above eqn as;

$$y = b_0 + b_1 x_1$$

where, $b_0 \rightarrow$ intercept

$b_1 \rightarrow$ co-efficient of x_1 (slope of x_1)

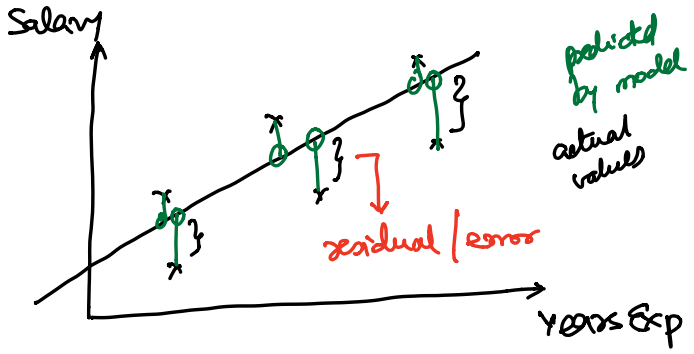
Equation of Simple linear regression

$$\text{Salary} = b_0 + b_1 (\text{YearsExperience})$$

Equation of multiple linear regression

$$\text{profit} = b_0 + b_1 (\text{R\&D spend}) + b_2 (\text{adm}) + b_3 (\text{mark})$$

— APPLIED MACHINE LEARNING —



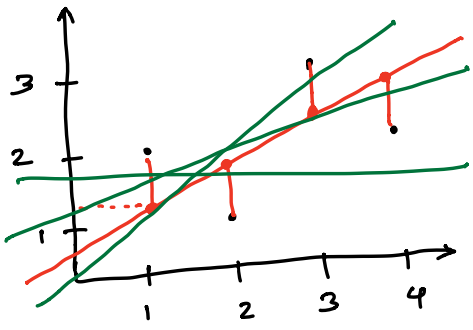
model that can predict the salary of the employee based on his/her experience

Best fit line → the line (model) that has less error!

$\sum$ residual is minimum!

Calc. errors:

① Mean Absolute Error:



x	y	predicted $\bar{y}$	error
1	2	1.2	0.8
2	1	1.8	0.8
3	3	2.3	0.7
4	2	2.8	0.8
			3.1

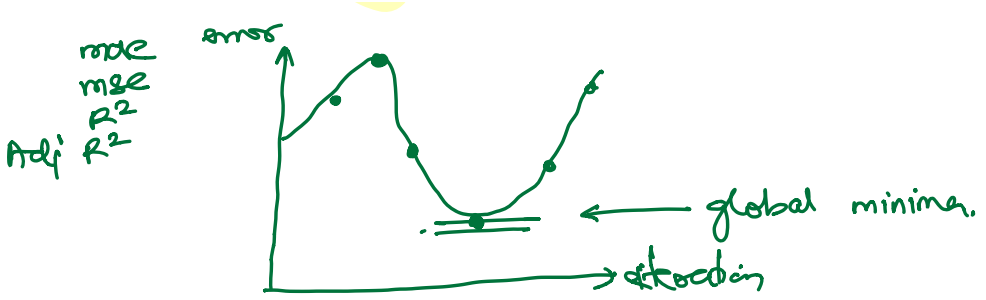
$$\frac{\sum_{i=1}^{n} |y - \bar{y}|}{n} = MAE = \frac{3.1}{4} = 0.775$$

error / loss

Goal → to code and select the line that is minimal loss.

Gradient Descent ←

iterations	error
1	0.775
2	0.88
3	0.62
4 →	0.33



Random sampling :

① Sequential sampling ← Small data but not recommended

② Random sampling ← recommended.

x	y
1	8
2	7
3	6
4	5
5	4
6	3
7	2
8	9
9	9
10	10

Sequential Sampling

train test split ratio → 80-20

training dataset →

x	y
1	8
2	7
3	6
4	5
5	4
6	3
7	2
8	1

testing dataset

x	y
9	9
10	10

seed = 1

→ mod function = $x \% 1 = 1$

(many algo)

training dataset

x	y
1	8
3	6
7	2
10	10
⋮	⋮

x	y
5	4
2	7

change of seed impact sample.