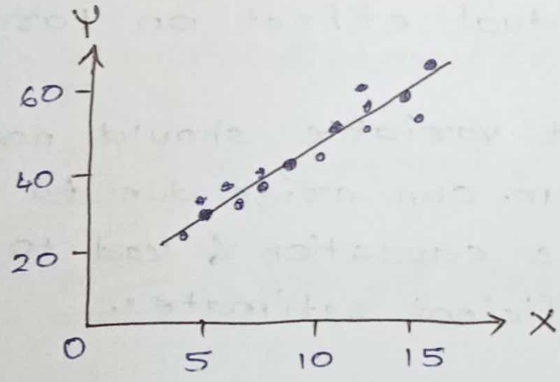
* Simple Linear Regression *

- **Supervised M/c Learning Algorithms :-** Algo learns from labelled data.
  - **Labelled data** - Dataset whose respective target value is already known.
  - **Supervised Learning** has 2 types -
    1. **Classification** - Predict class of dataset based on independent i/p variable.
       - Class is categorical / discrete values - like Yes / No, 0/1 etc.
    2. **Regression** - Predict continuous o/p variable.
       - Ex. House Price Prediction.

- **Linear Regression** -



Used to model the relationship between a dependent variable & set of independent variables.
1. Simple Linear Regr^n
2. Multiple Linear Regr^n

- **X** : Dependent / Target variable.
- **Y** : Independent / Feature Variables.
- **Regression Line** : Best-fit line of the model, by which we can predict '**Y**' value for new value of '**X**'.

Assumptions of Linear Regression:-

↳ Assumptions about data & its relationships.
↳ Violations can affect validity & reliability of regression results.

1. **Linearity** – Relation betⁿ dependend & independent variable is linear. Changes in both are constant.

2. **Independence of Errors** – Errors (residuals) assumed independent of each other.

3. **Homoscedasticity** – Assume variance of residuals is constant across all levels of independent variables.

4. **Normality of Errors** – Errors normally distributed.

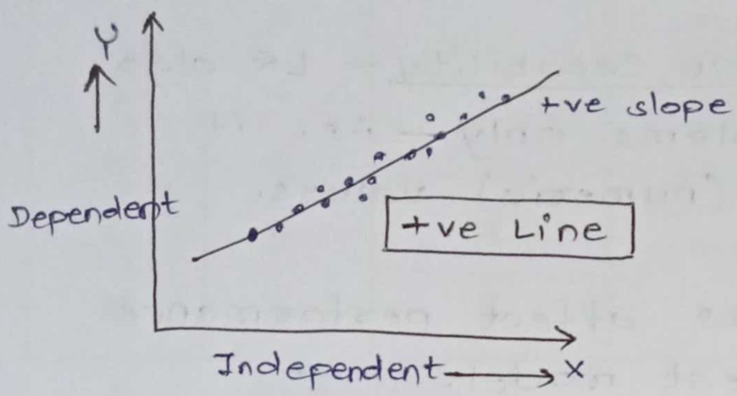
5. **No or Little Multicollinearity** – 2/more independent variables should not highly correlated. Difficult interpret each var individual effect on target.

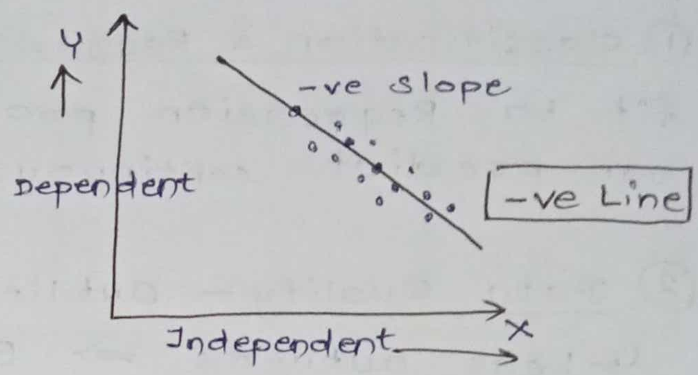
6. **No Endogeneity** – Indepent variable should not correlate with error term. Can arise due to omitted bias / simultaneous causation & lead to biased & inconsistent coefficient estimates.

7. **No Autocorrelation** – Auto corelation (Model residuals correlated with each other). ~~observations~~ Imp in time series data, observaⁿ dependent on previous.

8. **No Perfect Collinearity** – Perfect collinearity (one independent var perfectly predicted linearly by other independent variable).
   ↳ Leads to rank-deficient matrix, making it impossible to estimate unique regression coefficients.

Linear Regression:





1) Simple Linear Regression:

Model relationship between 1 dependent & 1 independent variable.

↳ This relationship can be represented as a linear equation.

- **Eqⁿ of Simple Linear Regression** –

$$\boxed{Y = \alpha_0 + \alpha_1 X + \epsilon}$$

Y – Dependent (Target) variable.

X – Predictor / Feature / Independent variable.

α_0 – Y-intercept, value of y when $x = 0$.

α_1 – Slope of line (Regression coefficient of x).

ϵ – Represents error term.

The equation is similar to eqⁿ of Line:

$$\boxed{Y = \underbrace{m}_{\text{Slope}} x + \underbrace{c}_{\text{Y-intercept}}}$$

- ## Applications of Linear Regression:

① Classification & Regression Capability – LR algo fit to Regression problems only – i.e. it can predict continuous (numeric) values.

② Data Quality – Outliers affect performance
↳ Less outliers ⇒ Best model.
↳ So, data should not contain outliers.

③ Computational Complexity – LR algo is computationally less expensive thal classification algo.

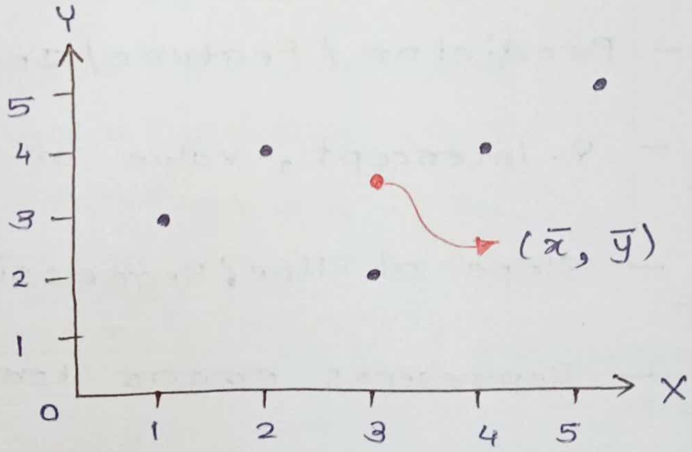
④ Comprehensive & Transparent – Easy to understand & explain. Represented using simple math expression.

- ## Working of Linear Regression:

- Data Set –

X	Y
1	3
2	4
3	2
4	4
5	5

- Plot the graph –



$$\bar{x} = \frac{1+2+3+4+5}{3} = 3$$

$$\bar{y} = \frac{3+4+2+4+5}{3} = 3.6$$

- Eqⁿ to Find slope –

$$m = \frac{\Sigma (x-\bar{x})(y-\bar{y})}{\Sigma (x-\bar{x})^2}$$

x	y	$(x-\bar{x})$	$(y-\bar{y})$	$(x-\bar{x})(y-\bar{y})$	$(x-\bar{x})^2$
1	3	-2	-0.6	1.2	4
2	4	-1	0.4	-0.4	1
3	2	0	-1.6	0	0
4	4	1	0.4	0.4	1
5	5	2	1.4	2.8	4
3	3.6			4.0	10

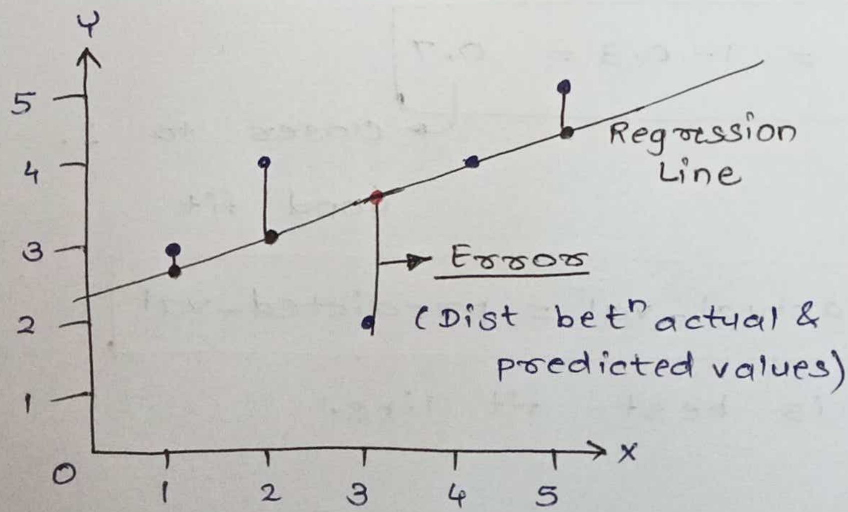
$$\therefore m = \frac{4}{10} = 0.4$$

$$c = \bar{y} - m\bar{x} = 3.6 - 0.4 * 3 = 2.4$$

For $m = 0.4$ & $c = 2.4$,
Predict y for $x = \{1, 2, 3, 4, 5\}$

x	y_predicted
1	$0.4 * 1 + 2.4 = 2.8$
2	$0.4 * 2 + 2.4 = 3.2$
3	$0.4 * 3 + 2.4 = 3.6$
4	$0.4 * 4 + 2.4 = 4.0$
5	$0.4 * 5 + 2.4 = 4.4$

- Plot final Graph -



- **Minimize Error:**
  Iteratively find m, predict y & find error.
  m with min error ⇒ Regrn Line

Find Goodness of fit :-

- **R-Square** – statistical measure of how close is data to fitted Regression Line.
  - Also known as coefficient of determination / coefficient of multiple determination.

$$\boxed{\begin{array}{c}\text{Dist (actual - mean)} \\ \text{v/s} \\ \text{Dist (predicted - mean)}\end{array}}$$

Predicted

$$\boxed{R^2 = 1 - \frac{\Sigma (y_p - \bar{y})}{\Sigma (y - \bar{y})}}$$

Mean

Actual

x	y	$(y-\bar{y})$	$(y-\bar{y})^2$	y_p	$(y_p-\bar{y})$	$(y_p-\bar{y})^2$
1	3	-0.6	0.36	2.8	-0.8	0.64
2	4	0.4	0.16	3.2	-0.4	0.16
3	2	-1.6	2.56	3.6	0	0
4	4	0.4	0.16	4.0	0.4	0.16
5	5	1.4	1.96	4.4	0.8	0.64
			5.2			1.6

$$\boxed{R^2 = 1 - \frac{1.6}{5.2} = 1 - 0.3 = 0.7}$$

closer to 1.

Good fit

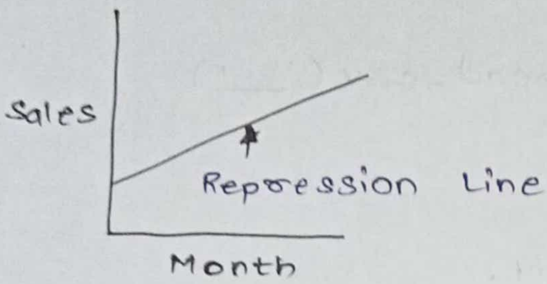
$$\boxed{\text{If } R^2 = 1 \Rightarrow \text{actual-val} = \text{predicted-val}}$$

This is best fit line

- Examples of Linear Regression :-

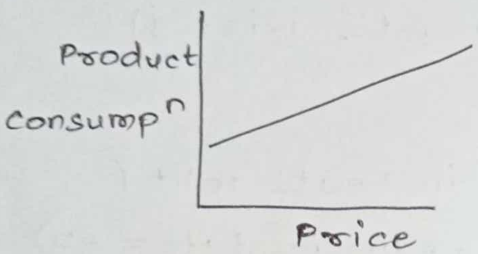
① Evaluate Trends & Sales estimates —

Can estimate sales in future.



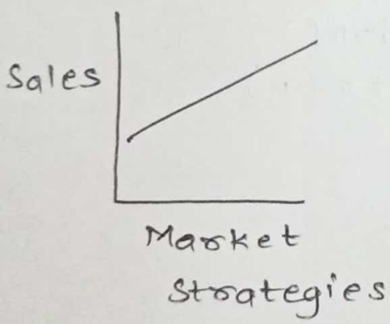
② Analyze Impact of Price change -

How price change affect the consumption of product.



③ Risk Assessment in Financial services —

How risk management affect business finance.



• Simple Linear Regression Implementation :-

① Import Libraries – pd, np, plt.

↓

② Import Data set – df = pd.read_csv('___')

↓

③ EDA –
- Null value treatment.
- Remove duplicates
- Handle categorical data.

↓

from sklearn.model_selection ⇒ train_test_split

④ Split Dataset – X = df.drop('target', axis = 1)

Y = df['target']

```
x_train, x_test, y_train, y_test = train_test_split(
                X, Y, test_size = 0.2, random_state = 42)
```

↓

from sklearn.linear_model ⇒ LinearRegression

⑤ Training Model –

```
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

↓

⑥ Result Prediction –

```
y_pred = regressor.predict(x_test)
```

↓

from sklearn.metrics ⇒ r2_score, mean_squared_error

⑦ Model Evaluation – Find model's goodness of fit.

```
r2 = r2_score(y_test, y_pred)

sq_err = mean_squared_error(y_test, y_pred)
```