* Multiple Linear Regression *

⊙ Multiple Linear Regression (MLR):-

Models Linear relationship bet^n single dependent continuous var & more than 1 independent variable.

↳ simple LR extension, takes more than one predictor (independent) var to predict 1 target var.

- Key points about MLR -

① Dependent var (Y) must be continuous, but the Predictor / Independent var (X) may be continuous / categorical variable.

② Each feature (Independent) variable forms Linear Relationship with dependent variable.

③ MLR fits the regression line through multi-dimensional space of data-points.

- Equation of MLR -

$$Y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \ldots\ldots + \alpha_n x_n + \epsilon$$

Y - Dependent (target) variable.

x_i - n number of independent variables.

α_i - Corresponding Regression coefficients of n Independent variables.

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

ϵ - Represent error term.

- **Assumptions of MLR –**

① Linear relationship bet^n each Predictor & target variable.

② Regression residuals must be normally distributed.

③ MLR assumes little / no collinearity (correlation bet^n independent variables) in data.

- **Finding Equation of Line from Implementation –**

① **Train the model –**

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

② **Find Regression Coefficient & Y-intercept –**

```
coeff = regressor.coef_      → Return list of all coefficients.
inter = regressor.intercept_ → Return single value.
```

③ **Equation of Regression Line –**

~~y = coeff[0] + co~~

$$y = inter + x_1 * coeff[0] + x_2 * coeff[1] + \ldots \ldots + x_n * coeff[n]$$

Multiple Linear Regression Implementation :-

① Import Libraries – pd, np, plt.

↓

② Import Dataset – df = pd.read_csv('__')

↓

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

↓

```
from sklearn.model_selection import 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)
```