# **Regression Models**

**Definition:**

Regression models are mathematical tools used in machine learning to understand the relationship between input features and a continuous or binary output variable. They help predict outcomes based on input features.

**Linear Regression:**

*Formula:*

In simple linear regression, there is only one independent variable, while in multiple linear regression, there are multiple independent variables. The goal of linear regression is to find the best-fitting line (or hyperplane in higher dimensions) that minimizes the difference between the observed values and the values predicted by the model.

Y = *β*0 + *β*1 \* x + *ϵ*

Where:

* *y* is the dependent variable (the variable we want to predict),
* *x* is the independent variable (the variable used to make predictions),
* *β*0​ is the intercept (the value of *y* when *x*=0),
* *β*1​ is the slope (the change in *y* for a one-unit change in *x*),
* *ϵ* represents the error term (the difference between the observed and predicted values).

**Example:**

Y=β0 +β1 X+ϵ

β0 = 30

β1 = 5

This means that our model can be written as:

Exam Score=30+5×Hours Studied+*ϵ*

So, if a student studies for 6 hours, we can predict their exam score as:

Exam Score=30+5×6=30+30=60

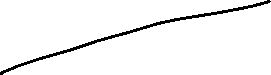
**Pseudocode:**

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

model.fit(X\_train, y\_train)

predictions = model.predict(X\_test)



**Logistic Regression:**

*Formula:*

Logistic regression is used for binary classification tasks. It models the probability of occurrence of an event using the logistic function:

P(Y=1|X) = 1 / (1 + e^-z)

Where:

- P(Y=1|X) is the probability of the positive class,

- X represents the input features,

- z is the linear combination of the input features and their corresponding weights.

**Description:**

Logistic regression models the relationship between the dependent binary variable and independent variables by estimating probabilities. It's commonly used when the dependent variable is binary (e.g., yes/no, true/false).

**Example**:

Predicting whether an email is spam (1) or not spam (0) based on features like the length of the email, presence of certain keywords, and sender's address.

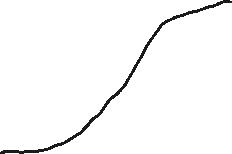
**Pseudocode**:

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression()

model.fit(X\_train, y\_train) # X\_train contains feature values, y\_train contains target values

predictions = model.predict(X\_test) # X\_test contains feature values for prediction



**Difference between Linear Regression and Logistic Regression:**

1. Output Type:

- Linear Regression: Continuous.

- Logistic Regression: Binary.

2. Function Used:

- Linear Regression: Linear.

- Logistic Regression: Sigmoid.

3. Target Variable:

- Linear Regression: Continuous.

- Logistic Regression: Binary.

4. Use Cases:

- Linear Regression: Predicting continuous values.

- Logistic Regression: Binary classification.

5. Evaluation Metrics:

- Linear Regression: MSE, RMSE, R-squared.

- Logistic Regression: Accuracy, precision, recall, F1-score.

6. Assumptions:

- Linear Regression: Assumes linear relationship between features and target.

- Logistic Regression: Assumes linear relationship between log odds and features.

**Gradient Descent Method:**

**Description:**

Gradient descent is an optimization algorithm used to minimize a function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient.

**Algorithm:**

Gradient descent updates the parameters (weights) of the model in the opposite direction of the gradient of the cost function. The general update rule for each parameter \( \theta\_j \) is:

Oi = Oj – a\*((dJ(O)/(dOj))

Where:

- a (alpha) is the learning rate,

- J(O) (theta) is the cost function,

- ((dJ(O)/(dOj)) is the partial derivative of the cost function with respect to Oj.

**Optimization:**

Both linear and logistic regression models can be optimized using gradient descent to find the optimal parameters that minimize the cost function.