**Regression Algorithms**

Regression algorithms are used for predicting **continuous values**.

**1.1 Linear Regression**

Linear Regression models the relationship between an **independent variable (X)** and a **dependent variable (Y)** using a straight-line equation:



where:

* Y = Target variable (dependent)
* X = Feature (independent)
* b0 = Intercept
* b1 ​ = Slope (coefficient)
* ϵ = Error term (difference between actual and predicted values)

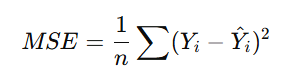
**Example:**

Predicting house prices based on square footage.

* **Input (X):** Size of the house (sq ft)
* **Output (Y):** Price of the house
* **Equation:** Price=50,000+300×SizePrice = 50,000 + 300 \times \text{Size}Price=50,000+300×Size

**Key Concepts:**

* **Cost Function (Mean Squared Error - MSE)**:



* **Gradient Descent**: Optimizes parameters to minimize error.
* **Assumptions**: Linearity, independence of errors, homoscedasticity.

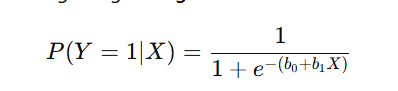
**2. Classification Algorithms**

Classification algorithms are used for predicting **categorical values** (e.g., spam vs. not spam).

**2.1 Logistic Regression**

Despite its name, **Logistic Regression** is a **classification algorithm** used for binary classification.

Instead of modeling a straight line like Linear Regression, Logistic Regression models the probability of an event occurring using the **sigmoid function**:

​ 

**Example:**

Predicting whether an email is spam (1) or not spam (0).

* **Input (X):** Number of spammy words in an email.
* **Output (Y):** Spam (1) or Not Spam (0).

**Key Points:**

* Outputs a probability between **0 and 1**.
* If **P(Y=1) > 0.5**, classify as **1**; otherwise, classify as **0**.
* Uses **Maximum Likelihood Estimation (MLE)** instead of MSE.

**2.2 K-Nearest Neighbors (KNN)**

A **non-parametric classification algorithm** that assigns a class to a new data point based on the majority class among its kkk nearest neighbors.

**How it works:**

1. Choose a value of kkk (e.g., 3, 5, 7).
2. Find the **k nearest neighbors** based on distance (e.g., Euclidean distance).
3. Assign the most common class among the kkk neighbors.

**Example:**

Classifying a fruit based on weight and texture.

* If k=3k = 3k=3 and the nearest neighbors are **(Apple, Apple, Orange)** → classify as **Apple**.
* If k=5k = 5k=5 and the nearest neighbors are **(Apple, Orange, Orange, Apple, Orange)** → classify as **Orange**.

**Key Points:**

* Works well for **small datasets**.
* Sensitive to **feature scaling** (use normalization).
* The choice of kkk affects accuracy.

**3. Model Evaluation Metrics**

To evaluate classification models, we use **confusion matrix-based metrics**.

**3.1 Confusion Matrix**

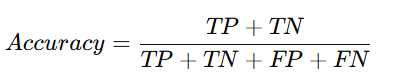
A **confusion matrix** is a table that summarizes classification predictions:

|  |  |  |
| --- | --- | --- |
| Actual / Predicted | Predicted Positive (1) | Predicted Negative (0) |
| Actual Positive (1) | True Positive (TP) | False Negative (FN) |
| Actual Negative (0) | False Positive (FP) | True Negative (TN) |

* **True Positive (TP)**: Model correctly predicts **positive**.
* **True Negative (TN)**: Model correctly predicts **negative**.
* **False Positive (FP) (Type I Error)**: Model incorrectly predicts **positive**.
* **False Negative (FN) (Type II Error)**: Model incorrectly predicts **negative**.

**3.2 Accuracy**

Measures how many predictions were correct.

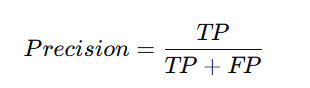


**Example:** If 90 out of 100 predictions are correct, accuracy = **90%**.

⚠️ **Limitation:** Accuracy can be misleading in imbalanced datasets (e.g., predicting rare diseases).

**3.3 Precision (Positive Predictive Value)**

Measures how many of the predicted positives were actually positive.

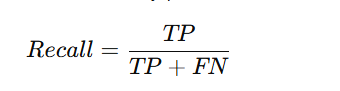


**Example:** If 100 patients are predicted as having a disease, but only 80 actually have it, precision = **80%**.

⚠️ **Use Precision when False Positives are costly** (e.g., spam detection, medical tests).

**3.4 Recall (Sensitivity, True Positive Rate)**

Measures how many actual positives were correctly predicted.



**Example:** If 100 people actually have a disease and the model detects 70, recall = **70%**.

⚠️ **Use Recall when False Negatives are costly** (e.g., cancer detection, fraud detection).

**3.5 F1-Score**

Balances Precision and Recall.

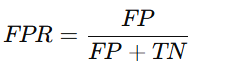
F1=2×

* **F1-Score is high when both Precision and Recall are high**.
* **Useful for imbalanced datasets**.

**Example:** If Precision = 0.8 and Recall = 0.7,

**3.6 ROC Curve & AUC (Area Under Curve)**

ROC Curve plots **True Positive Rate (Recall)** vs **False Positive Rate (FPR)**:



* **A perfect model**: AUC = **1.0** (Top-left corner).
* **Random guessing**: AUC = **0.5** (Diagonal line).

**Example:** If AUC = **0.90**, the model is **90% likely** to rank a positive instance higher than a negative one.

**Summary Table**

