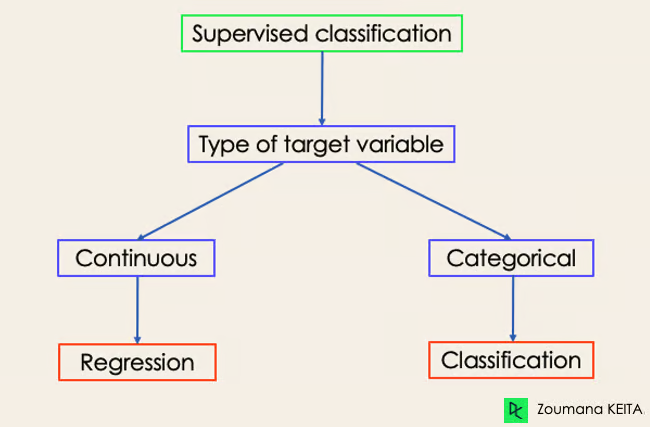
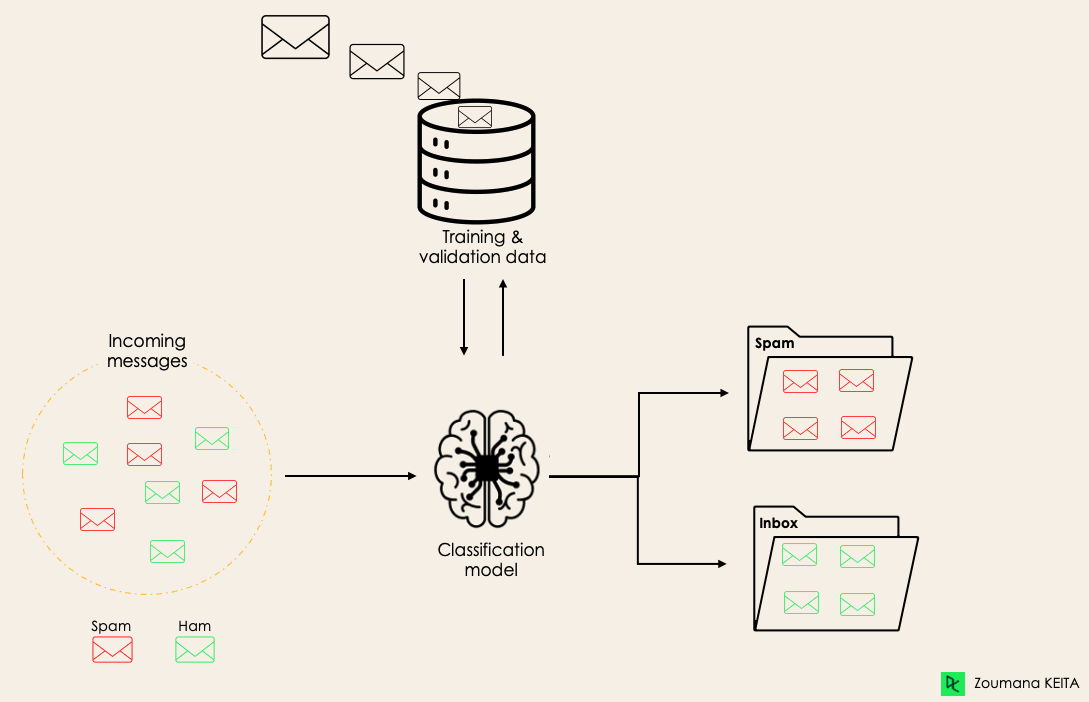
**RESOURCES a**

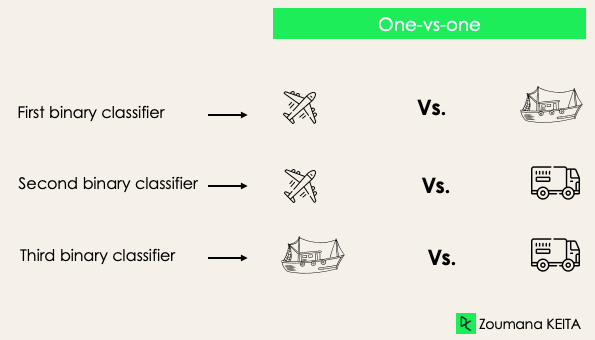
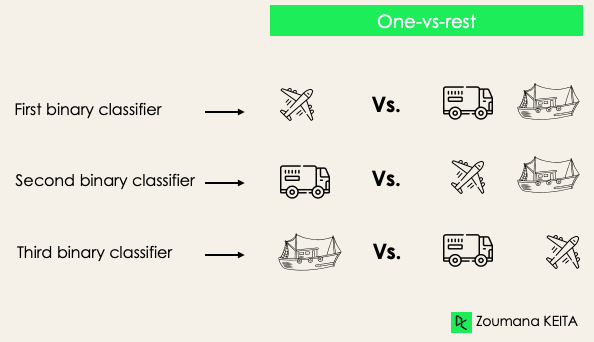
*Classification Models*

* Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data.
* In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data.
* There are two types of learners in machine learning classification: lazy and eager learners.
  + Eager learners are machine learning algorithms that first build a model from the training dataset before making any prediction on future datasets.
  + They spend more time during the training process because of their eagerness to have a better generalization during the training from learning the weights, but they require less time to make predictions.
    - Logistic Regression.
    - Support Vector Machine.
    - Decision Trees.
    - Artificial Neural Networks.
  + Lazy learners or instance-based learners, on the other hand, do not create any model immediately from the training data.
  + They just memorize the training data, and each time there is a need to make a prediction, they search for the nearest neighbor from the whole training data, which makes them very slow during prediction.
    - K-Nearest Neighbor.
    - Case-based reasoning.

*Binary Transformation Approaches*

* OvO - One-versus-one
  + this strategy trains as many classifiers as there are pairs of labels.
  + If we have a 3-class classification, we will have three pairs of labels, thus three classifiers.
  + In general, for N labels, we will have classifiers.
  + Each classifier is trained on a single binary dataset, and the final class is predicted by a majority vote between all the classifiers.
  + One-vs-one approach works best for SVM and other kernel-based algorithms.

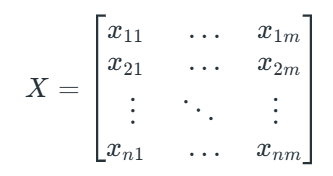
* OvA – One-versus-all
  + at this stage, we start by considering each label as an independent label and consider the rest combined as only one label.
  + With 3-classes, we will have three classifiers.
  + In general, for N labels, we will have N binary classifiers.

*Logistic Regression*

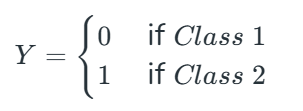
* Logistic regression is used for binary classification where we use sigmoid function, that takes input as independent variables and produces a probability value between 0 and 1.
* The sigmoid function is a mathematical function used to map the predicted values to probabilities.
* It maps any real value into another value within a range of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the “S” form.
* The S-form curve is called the Sigmoid function or the logistic function.
* In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Working for Logistic Regression

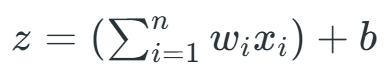
* Independent input features:

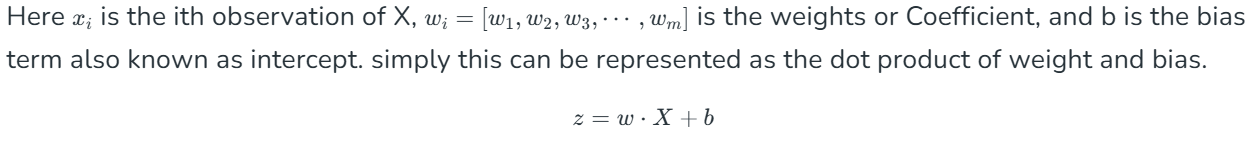


* Dependent variable:

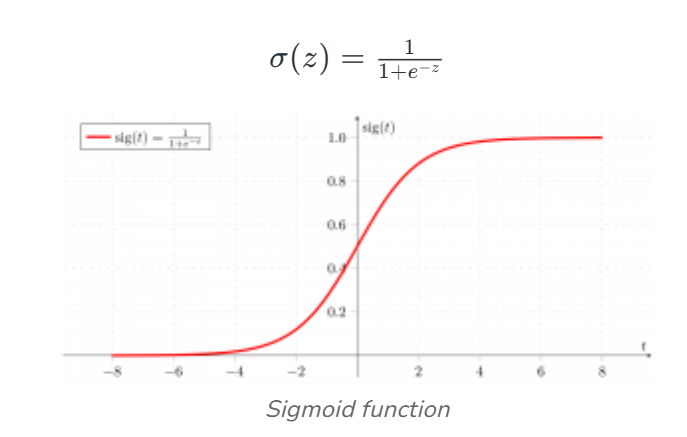


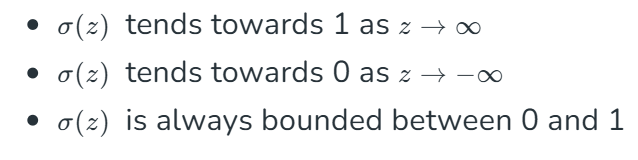
* Function:

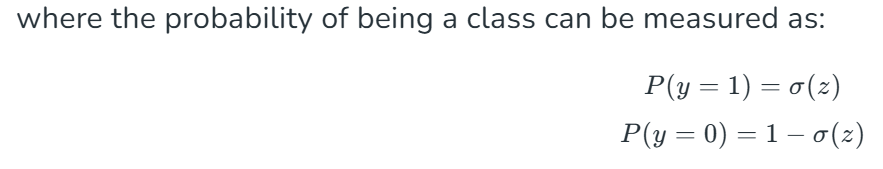




* Sigmoid:





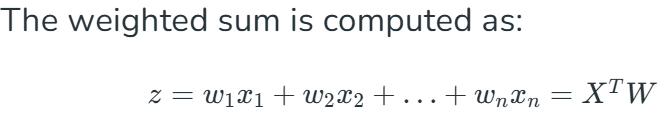
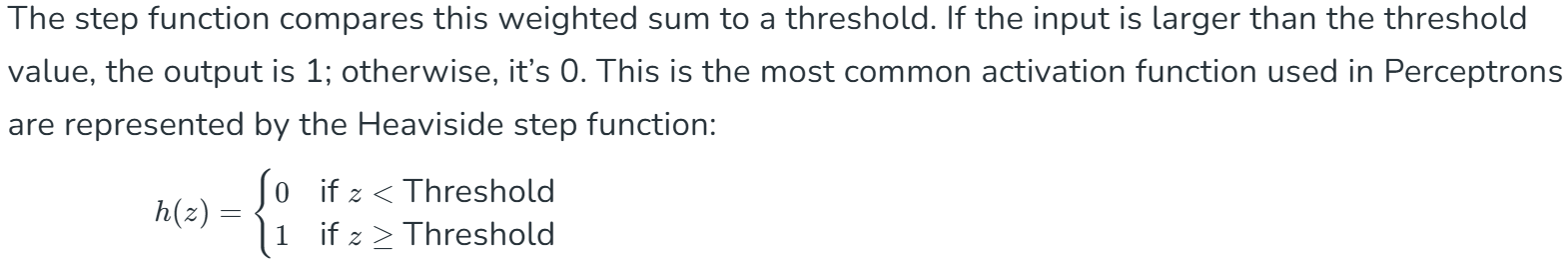


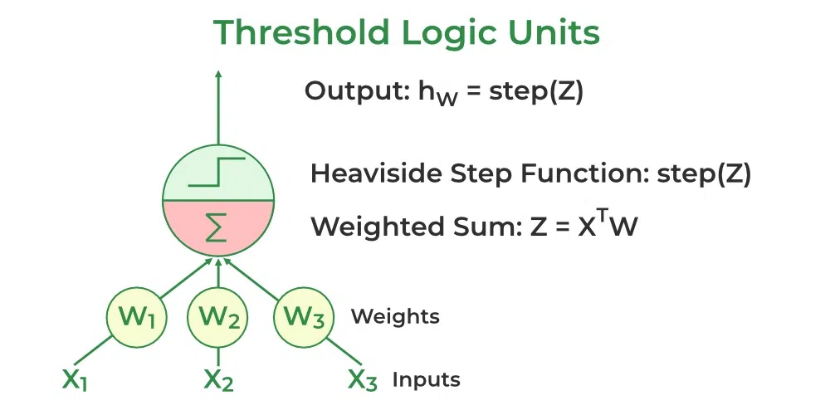
Equation: [Link](https://www.geeksforgeeks.org/understanding-logistic-regression/)

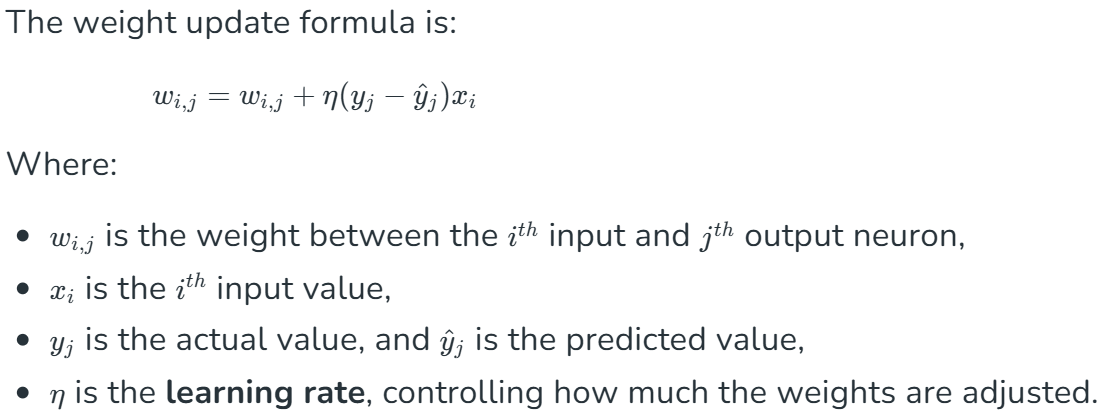
*Perceptron Algorithm*

* Perceptron is a type of neural network that performs binary classification that maps input features to an output decision, usually classifying data into one of two categories, such as 0 or 1.
* Perceptron consists of a single layer of input nodes that are fully connected to a layer of output nodes. It is particularly good at learning linearly separable patterns. It utilizes a variation of artificial neurons called Threshold Logic Units (TLU).
* Types of Perceptron:
  + Single-Layer Perceptron is a type of perceptron is limited to learning linearly separable patterns. It is effective for tasks where the data can be divided into distinct categories through a straight line. While powerful in its simplicity, it struggles with more complex problems where the relationship between inputs and outputs is non-linear.
  + Multi-Layer Perceptron possess enhanced processing capabilities as they consist of two or more layers, adept at handling more complex patterns and relationships within the data.

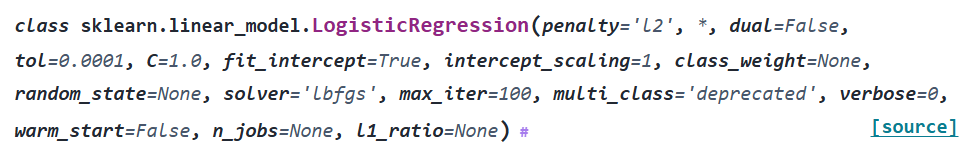
Working of a Perceptron

* A weight is assigned to each input node of a perceptron, indicating the importance of that input in determining the output.
* The Perceptron’s output is calculated as a weighted sum of the inputs, which is then passed through an activation function to decide whether the Perceptron will fire.
* 
* 



* During training, the Perceptron’s weights are adjusted to minimize the difference between the predicted output and the actual output.
* This is achieved using supervised learning algorithms like the delta rule or the Perceptron learning rule.
* 

Parameters:



**1. penalty**

* **Explanation**: Determines the type of regularization applied to prevent overfitting. Options include:
  + 'l1': Lasso regularization, encourages sparsity in the model (some coefficients become zero).
  + 'l2': Ridge regularization, penalizes large coefficients to make them smaller.
  + 'elasticnet': A mix of L1 and L2.
  + 'none': No regularization.

**2. dual**

* **Explanation**: Whether to use a dual formulation for optimization. It's only applicable for 'l2' penalty with a linear solver and when there are more features than samples (n\_features > n\_samples).

**3. tol**

* **Explanation**: The tolerance for the stopping criteria. Lower values mean the algorithm runs longer for a more precise solution.

**4. C**

* **Explanation**: Inverse of regularization strength. Smaller values mean stronger regularization. For example:
  + C = 1 is default.
  + C = 0.1 means stronger regularization.
  + C = 10 means weaker regularization.

**5. fit\_intercept**

* **Explanation**: Whether to include an intercept (constant term) in the model.
  + True: Adds an intercept.
  + False: Assumes the data is already centered and does not add an intercept.

**6. solver**

* **Explanation**: The algorithm used to optimize the logistic regression. Options include:
  + 'lbfgs': A robust solver suitable for small datasets and supports L2 regularization.
  + 'liblinear': A solver for smaller datasets, supports both L1 and L2 penalties.
  + 'sag' and 'saga': Solvers for large datasets, faster with sparse data.
  + 'newton-cg': Handles L2 regularization and is efficient for small to medium datasets.

**7. max\_iter**

* **Explanation**: Maximum number of iterations the solver will take to converge. If the algorithm doesn’t converge, increasing this value might help.

**8. multi\_class**

* **Explanation**: Determines how to handle multi-class classification problems:
  + 'ovr' (One-vs-Rest): Default, trains separate binary classifiers for each class.
  + 'multinomial': Uses a softmax function to handle all classes together.
  + 'auto': Chooses the best strategy based on the solver.

**9. class\_weight**

* **Explanation**: Balances the weights for different classes. Useful when dealing with imbalanced data:
  + None: No balancing, all classes are equally weighted.
  + 'balanced': Automatically adjusts weights inversely proportional to class frequencies.

**10. random\_state**

* **Explanation**: Controls the randomness of the algorithm for reproducibility. Set this to a specific integer to get consistent results.

**11. verbose**

* **Explanation**: Controls how much information is printed during training.
  + 0: No information.
  + Higher values (e.g., 1): Print more detailed output.

**12. n\_jobs**

* **Explanation**: Number of CPU cores to use for computation. Use -1 to utilize all available cores for faster training.

**13. l1\_ratio**

* **Explanation**: Used when penalty='elasticnet'. It controls the balance between L1 and L2 regularization.
  + 0: Equivalent to L2.
  + 1: Equivalent to L1.
  + Values in between (e.g., 0.5): A mix of L1 and L2.

**L1 Regularization (Lasso)**

* **Definition**: Adds the **absolute value** of the coefficients as a penalty term to the loss function.

where wi​ are the model's coefficients and λ is a regularization parameter.

* **Key Features**:
  + Encourages sparsity: Many coefficients are driven to exactly zero, effectively selecting only the most important features.
  + Useful for feature selection in high-dimensional datasets.

**L2 Regularization (Ridge)**

* **Definition**: Adds the **squared value** of the coefficients as a penalty term to the loss function.
* **Key Features**:
  + Penalizes large coefficients but does not force them to zero. Instead, it shrinks them toward smaller values.
  + Helps reduce the model's sensitivity to small variations in the data, improving generalization.

**Elastic Net**

Elastic Net combines L1 and L2 regularization. It balances sparsity and smoothness:

This is controlled by the l1\_ratio parameter in logistic regression with the elasticnet penalty.

*Metrics to Evaluate ML Classification Models*

