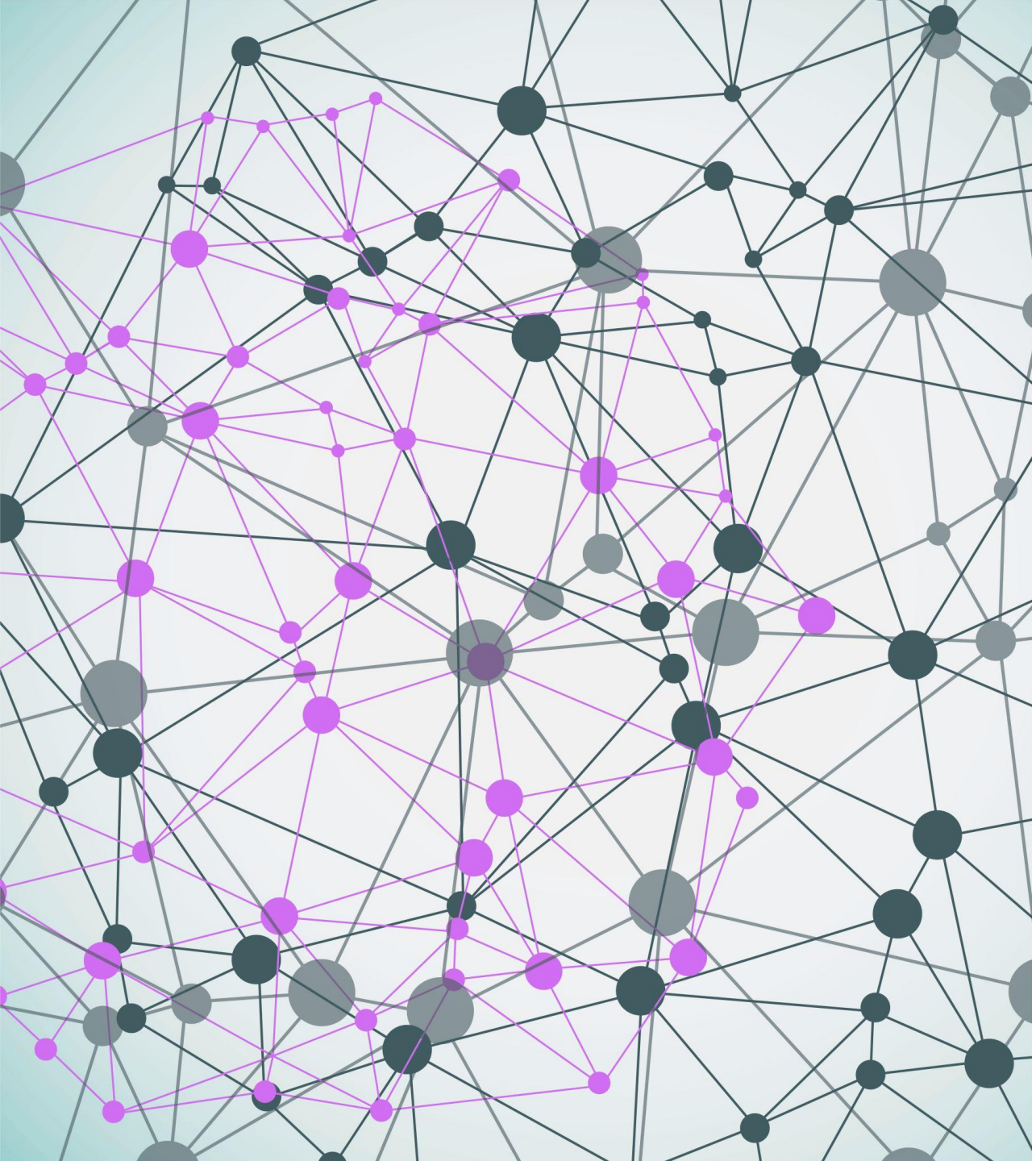


PH 3022
Machine Learning and Neural Computation

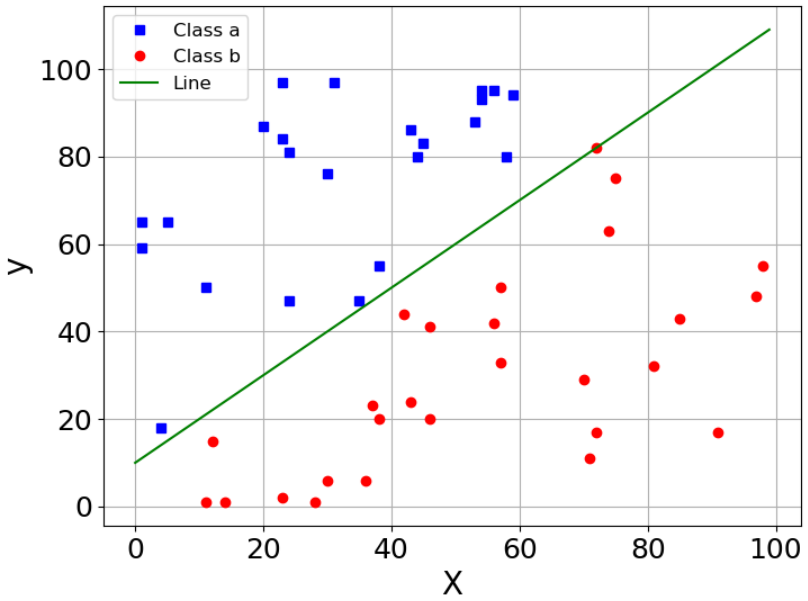
Logistic Regression

Department of Physics – University of Colombo

Binary Classifiers

- Binary classifiers are machine learning models designed to distinguish between two classes or categories.
- These models are trained to assign instances to one of two possible outcomes, typically labeled as positive (1) or negative (0).
- The term "positive" typically refers to the class that is of interest or considered as the positive outcome.
- Example: A classifier classifies spam and non-spam emails.
- We can assume non-spam emails as positive and spam emails as negative samples.

Logistic Regression



Logistic regression is a classification model, not a regression model, despite its name.

It is a straightforward and efficient method for **binary** and **linear** classification problems.

Known for its simplicity, logistic regression performs well, particularly with linearly separable classes.

Linearly separable classes: This refers to a graph where a ~~straight line~~ divides the two data classes. Linear Function

Logistic Function

- Logistic regression employs a logistic function, also known as the **sigmoid function**, to map predictions and their probabilities.
- The sigmoid function transforms any real value into a range between 0 and 1, creating an S-shaped curve.
- Assume that a dataset contains n number of features (x). We can write weighted sum of all features as

$$z = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_N x_N$$

- Here, w_i values are set of weights. $w_0$coefficient is known as the bias or intercept term.
- We can write the sigmoid function as

$$h(z) = \frac{1}{1 + e^{-z}}$$

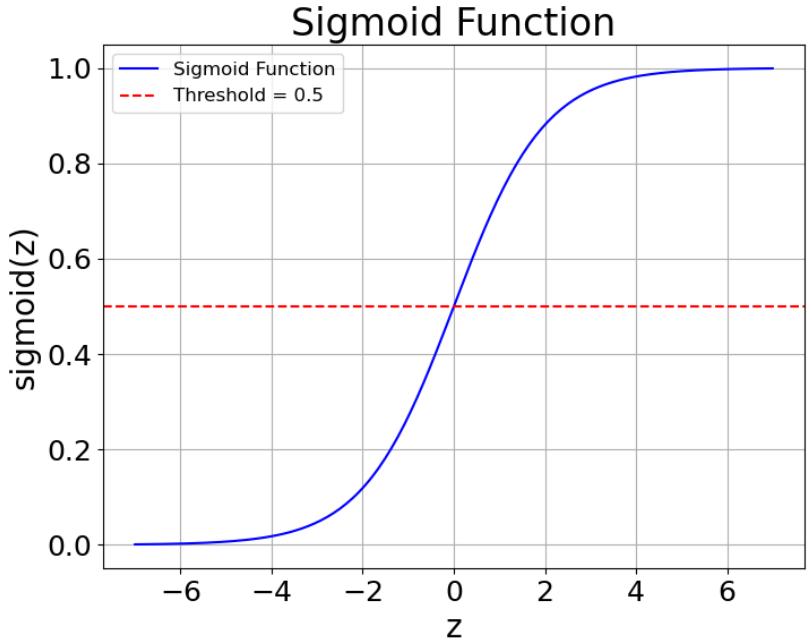
$h(z)$ represents the probability that the dependent variable (target) is 1.

Logistic Function

x_1	x_2	x_3	...	x_N	Class
0.1	2	5.1	...	8	0
0.6	6	4.7	...	7	1
0.4	4	5.6	...	5	1
0.6	1	4.9	...	3	0
...	....	...	...	...	...

$$z = w_0 + w_1x_1 + w_2x_2 + \cdots + w_Nx_N$$

$$h(z) = \text{sigmoid}(z) = \frac{1}{1 + e^{-z}}$$



If $\text{sigmoid}(z) \geq 0.5 \rightarrow \text{class} = 1$

If $\text{sigmoid}(z) < 0.5 \rightarrow \text{class} = 0$

Cross-entropy Loss Function

$$L(h) = -y_i \log\big(h(x_i)\big) - (1 - y_i) \log(1 - h(x_i))$$

- y_i is the true class label (0 or 1) of the i-th instance.
- $h(x_i)$ is the predicted probability that the i-th instance belongs to class 1.
- m is the number of data instances.

- Cross-entropy loss function quantifies the dissimilarity between the predicted probabilities and the true labels.

Cross-entropy Cost Function

$$J(b) = -\frac{1}{m}\Sigma_{i=1}^{m}\left[y_i \log\left(h(z_i)\right) + (1 - y_i)\log(1 - h(z_i))\right]$$

- The cross-entropy cost function is used to refer to an average of the loss functions over an entire training data.
- It penalizes the model more when it makes confident wrong predictions and less when it is uncertain or correct.
- The summation over all instances ($\sum_i^m$) is an average over the entire dataset, where m is the number of instances.

Cross-entropy Cost Function

Cost Function

$$J(b) = -\frac{1}{m}\Sigma_{i=1}^{m}\left[y_i \log\left(h(x_i)\right) + \left(1 - y_i\right)\log\left(1 - h(x_i)\right)\right]$$

Consider a single instance

Loss Function

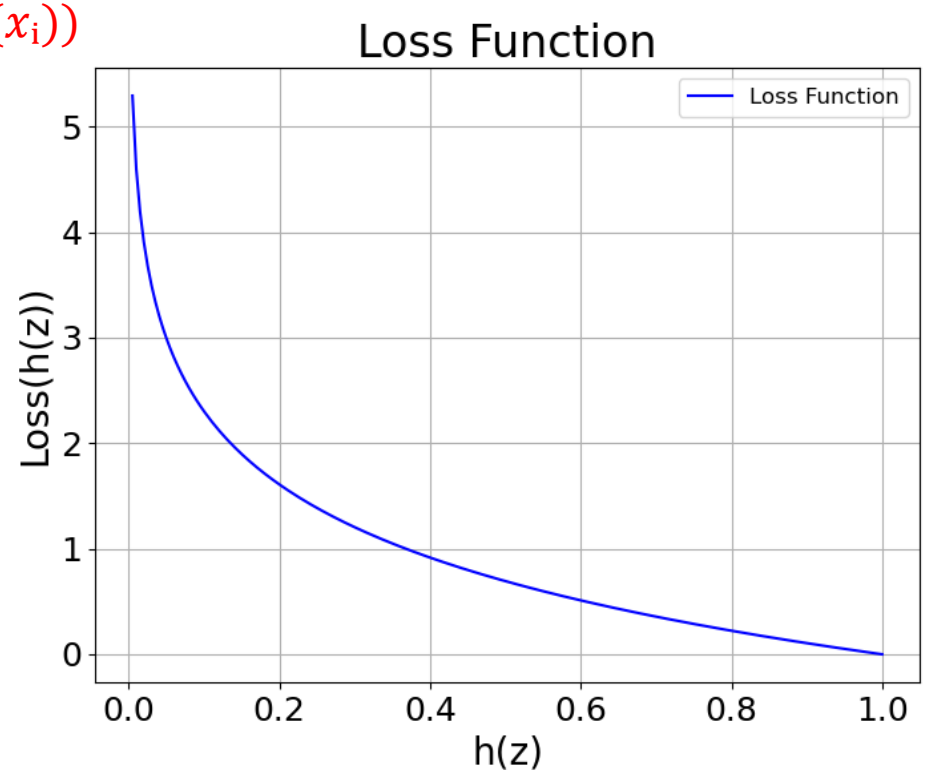
$$L(h) = -y_i \log\left(h(x_i)\right) - \left(1 - y_i\right)\log\left(1 - h(x_i)\right)$$

Example: If true class is 1 ($y_i = 1$)

$$L(h) = -\log(h(z_i))$$

If the model predicts that the class is 0 (low $h(z_i)$), the Loss is high.
That means **wrong predictions provide high loss**.

If the model predicts that the class is 1 (high $h(z_i)$), the Loss is low.
That means **correct predictions provide low loss**.



Cross-entropy Cost Function

$$J(b) = -\frac{1}{m}\Sigma_{i=1}^{m}\left[y_i \log\left(h(x_i)\right) + \left(1 - y_i\right)\log\left(1 - h(x_i)\right)\right]$$

Consider a single instance

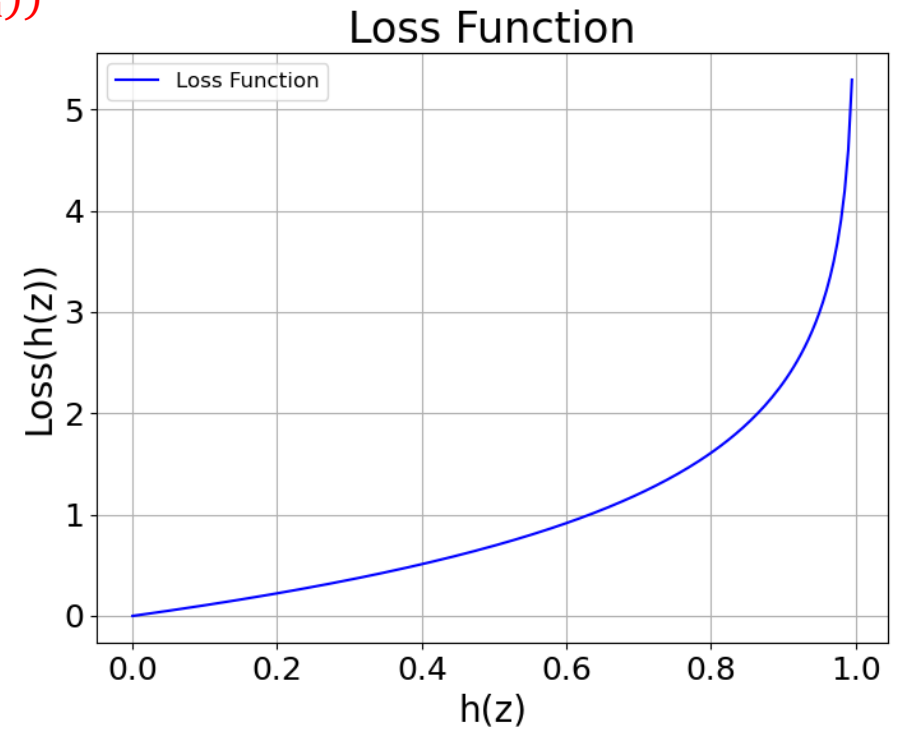
$$L(h) = -y_i \log\left(h(x_i)\right) - \left(1 - y_i\right)\log\left(1 - h(x_i)\right)$$

Example: If true class is 0 ($y_i = 0$)

$$L(h) = -\log(1 - h(z_i))$$

If the model predicts that the class is 1 (~~low~~ high $h(z_i)$), the Loss is high.
That means **wrong predictions provide high loss**.

If the model predicts that the class is 0 (~~high~~ Low $h(z_i)$), the Loss is low.
That means **correct predictions provide low loss**.



Optimization Algorithms

$$z = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_N x_N$$

$$h(z) = \frac{1}{1 + e^{-z}}$$

$$J(b) = -\frac{1}{m} \Sigma_{i=1}^{m} [y_i \log(h(z_i)) + (1 - y_i) \log(1 - h(z_i))]$$

- Initialize weights (w_i) randomly
- Loop until convergence :
  - Compute gradient $\frac{\partial J}{\partial w_i}$
  - Update weights $w_i^{\text{new}} = w_i - \eta \frac{\partial J}{\partial w_i}$
- Return weights

In scikit-learn package, several optimization algorithms are available.

Regularization

One of the primary purposes of regularization is to prevent overfitting. Regularization techniques penalize overly complex models, discouraging them from fitting the noise and outliers in the training data. This helps ensure that the model generalizes well to new, unseen data.

In linear regression, there are three main regularization techniques.

1. Lasso Regression (L1)
2. Ridge Regression (L2)
3. Elastic Regression

Lasso Regression (L1)

$$J = -\frac{1}{m}\Sigma_{i=1}^{m}\left[y_i \log\left(h(z_i)\right) + \left(1 - y_i\right)\log\left(1 - h(z_i)\right)\right] + \frac{\lambda}{2m}\sum_{i=1}^{N}|w_i|$$

- The Lasso regression model minimizes the cost function J by choosing the appropriate w values.
- The regularization parameter αcontrols the strength of regularization.
- When $\lambda = 0$, the model will serve as linear regression without regulation.
- A larger α results in stronger regularization, which tends to shrink the coefficients more aggressively.
- L1 regularization tends to produce sparse models by driving some coefficients to exactly zero. This can be useful for feature selection.

Reference: https://ml-explained.com/blog/logistic-regression-explained
https://www.youtube.com/watch?v=o57T25nh13g

Ridge Regression (L2)

In L2 regularization, the regularization term is the sum of the absolute values of the coefficients:

$$J = -\frac{1}{m}\Sigma_{i=1}^{m}\ \left[y_i \log\left(h(z_i)\right)\ +\ \left(1\ -\ y_i\right)\log\left(1\ -\ h(z_i)\right)\right] + \frac{\lambda}{2m}\sum_{i=1}^{N} w_i^2$$

N: Number of samples

y_i: Actual target values

y_i^{pred}: Predicted target value

w: Coeficients in the linear equation

α: Regulation Parameter. This is the **hyperparameter** need to be tuned

Reference: https://ml-explained.com/blog/logistic-regression-explained
https://www.youtube.com/watch?v=o57T25nh13g

Elastic Regression

$$J = -\frac{1}{m}\Sigma_{i=1}^{m}\left[y_i \log\left(h(z_i)\right) + \left(1 - y_i\right)\log\left(1 - h(z_i)\right)\right] + \frac{\lambda}{2m}\left[\gamma \overbrace{\sum_{i=1}^{N}|w_i|}^{\text{L1}} + (1-\gamma)\overbrace{\sum_{i=1}^{N} w_i^2}^{\text{L2}}\right]$$

- Elastic Net regularization is a hybrid of L1 (Lasso) and L2 (Ridge) regularization techniques.
- It combines both L1 and L2 regularization terms in the linear regression cost function.
- The elastic net regularization term is a linear combination of the L1 and L2 regularization terms, controlled by two parameters: λ and γ.

Confusion Matrix

A confusion matrix is a table used in machine learning to evaluate the performance of a classification algorithm.

It shows the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made by the model.

True Positive (TP):
Instances that are actually positive and are correctly predicted as positive by the model.

True Negative (TN):
Instances that are actually negative and are correctly predicted as negative by the model.

False Positive (FP):
Instances that are actually negative but are incorrectly predicted as positive by the model.

False Negative (FN):
Instances that are actually positive but are incorrectly predicted as negative by the model (Type II error).

Confusion Matrix

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

True Positive (TP):
Instances that are actually positive and are correctly predicted as positive by the model.

True Negative (TN):
Instances that are actually negative and are correctly predicted as negative by the model.

False Positive (FP):
Instances that are actually negative but are incorrectly predicted as positive by the model.

False Negative (FN):
Instances that are actually positive but are incorrectly predicted as negative by the model (Type II error).

Confusion Matrix

The term "positive" typically refers to the class that is of interest or considered as the positive outcome.

Example: A classifier classifies spam and non-spam emails.

We can assume non-spam emails as positive and spam emails as negative samples.

Assume that we have 130 data emails in the dataset.

Which contains 70 non-spam samples and 55 spam samples.

We obtain the following Confusion Matrix using a machine learning model.

	Actual Positive	Actual Negative
Predicted Positive	60	5
Predicted Negative	10	50

TP: 60 **FP**: 5

FN: 10 **TN**: 50

Confusion Matrix

Using Scikit-learn package

```
from sklearn.metrics import confusion_matrix
y_true = [0, 1, 1, 1, 0]
y_pred = [0, 0, 1, 0, 1]
confusion_matrix(y_true, y_pred)
```

```
array([[1, 1],
       [2, 1]], dtype=int64)
```

```
tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
(tn, fp, fn, tp)
```

```
(1, 1, 2, 1)
```

Metrics Derived from a Confusion Matrix

- **Accuracy**: Proportion of correctly classified instances.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + FN + TN)}$$

- **Precision:** Proportion of instances predicted as positive that are actually positive.

$$\text{Precision} = \frac{TP}{(TP + FP)}$$

- **Recall:** Proportion of actual positive instances that are correctly predicted as positive.

$$\text{Precision} = \frac{TP}{(TP + FN)}$$

- **F1 Score:** Harmonic mean of precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Classification Report

- A classification report is a summary of the performance metrics for a classification model, often generated using the results of a confusion matrix.
- It provides a detailed breakdown of key metrics, including precision, recall, F1 score, and support for each class in a multi-class classification problem.

```
from sklearn.metrics import classification_report
y_true = [0, 1, 1, 1, 0]
y_pred = [0, 0, 1, 0, 1]
target_names = ['Negative', 'Positive']
print(classification_report(y_true, y_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

    Negative       0.33      0.50      0.40         2
    Positive       0.50      0.33      0.40         3

    accuracy                           0.40         5
   macro avg       0.42      0.42      0.40         5
weighted avg       0.43      0.40      0.40         5
```

Multiclass Classification

- Multiclass classification is a classification technique that handles scenarios where the target variable can belong to multiple classes.
- In this approach, each new sample or data point is assigned exclusively to one class among the available options.

There are four classes (0, 1, 2, 3)

feature_0	feature_1	feature_2	feature_3	feature_4	target
-0.105073	1.803544	-0.793379	0.533485	-1.900431	1
-2.187566	-2.734266	3.944383	0.338252	-0.778070	2
-2.542504	-0.187366	1.998287	0.398449	2.817549	3
1.643421	-1.039507	-0.959467	1.545257	0.888577	2
1.471156	0.879819	-1.770159	-0.605393	-0.999241	1
1.134209	0.860432	-1.660172	-0.144288	0.309758	3
-1.667988	1.358628	0.462571	-1.012273	1.014721	3
-2.611169	-0.522331	2.810236	0.672943	-0.441410	0
-0.158851	-0.221209	0.424128	0.522562	-0.815427	0
1.149881	-2.173389	0.252713	0.014244	0.841695	0
-1.805485	1.386684	0.559444	0.620748	1.079556	3

One-vs-Rest

One-vs-Rest transforms a multi-class classification problem into multiple binary classification tasks, creating a separate binary model for each class.

However, this strategy may not be well-suited for large datasets due to the creation of an individual model for each class, leading to scalability challenges.

If there are N classes in the Target Variable, then N binary classifiers would be created.

Assume we have four classes: 0, 1, 2, and 3.

Then we need 4 binary classifiers.

Model	Positive Class	Negative Class
Model 1	0	[1,2,3]
Model 2	1	[0,2,3]
Model 3	2	[0,1,3]
Model 4	3	[0,1,2]

One-vs-Rest

- The prediction is based on the model that expresses the highest confidence.
- For this method to work, each model must provide predictions in the form of class membership probabilities or scores resembling probabilities.
- The argmax operation is then applied to these scores, selecting the class index associated with the highest score, to make the final class prediction.

Model	Positive Class	Negative Class
Model 1	0	[1,2,3]
Model 2	1	[0,2,3]
Model 3	2	[0,1,3]
Model 4	3	[0,1,2]

One-vs-One

One-vs-One transforms a multi-class classification problem into multiple binary classification tasks, where each class competes against every other class.

Compared to One-vs-Rest Classification, One-vs-One generates a larger number of models.

if number of classes in the Target Variable is N, then the number of binary classifiers created would be $N \times \frac{N-1}{2}$.

During prediction, each classifier votes for a class, and the class with the most votes wins.

Assume we have four classes: 0, 1, 2, and 3.

Then we need 6 binary classifiers.

Model	Positive Class	Negative Class
Model 1	0	1
Model 2	0	2
Model 3	0	3
Model 4	1	2
Model 5	1	3
Model 6	2	3