Logistic Regression

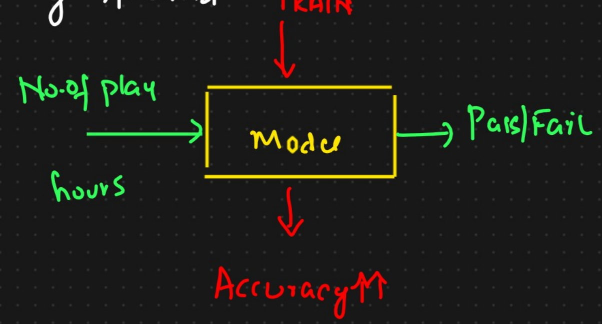
To solve classification problem
- → Binary classification ←
- → Multiclass classification

Dataset

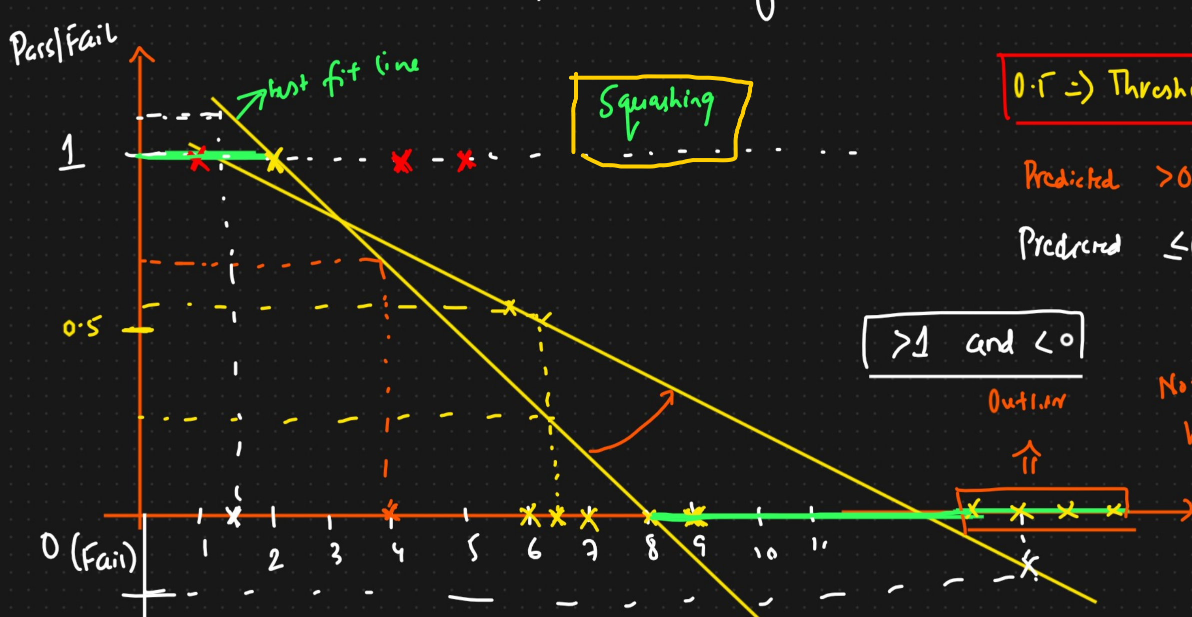
Independent feature ↓ | Dependent or o/p feature ↓

No.of play hours	Pass/Fail {y}	
9	Fail	0
8	Fail	0
7	Fail	0
6	Fail	0
5	Pass	1
4	Pass	1
1	Pass	1
2	Fail	0

$\hat{y}$ → predicted



Can we solve this classification problem using Regression?



0.5 ⇒ Threshold value

Predicted >0.5 ⇒ Pass

Predicted ≤0.5 → Fail

Why we cannot use Linear Regression for classification

① Best fit line changes because of outliers → prediction goes wrong

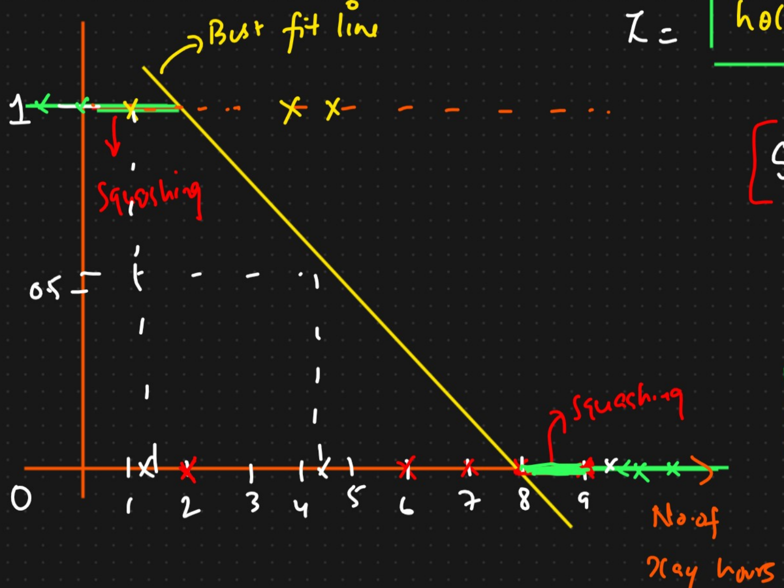
② The outcome comes >1 and <0 also

To solve this problem we use: Logistic Regression

$\Downarrow$

$\boxed{0 \text{ to } 1}$ ⇒ Squashing Technique

How Logistic Regression Solves Classification Problem



$z = \boxed{h_\theta(x) = \theta_0 + \theta_1 x_1}$ → Best fit line

$\Downarrow$

[Sigmoid Activation function]

$\Downarrow$

0 to 1

$$\boxed{\sigma = \frac{1}{1+e^{-z}}} \Rightarrow 0 \text{ to } 1$$

$$h_\theta(x) = \sigma(\overset{\Uparrow z}{\theta_0 + \theta_1 x_1})$$

$$\boxed{h_\theta(x) = \frac{1}{1+e^{-z}}} \Rightarrow z = \theta_0 + \theta_1 x_1 \Rightarrow \boxed{h_\theta(x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1)}}}$$

↳ Logistic Regression

Linear Regression Cost function

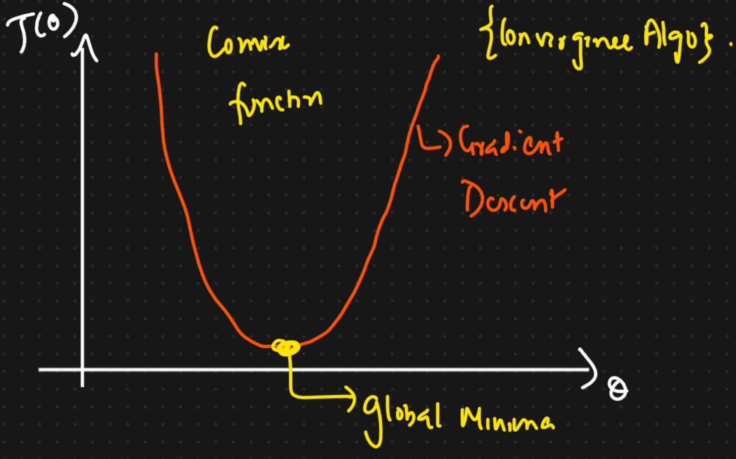
$$J(\theta_0, \theta_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x)_i)^2$$

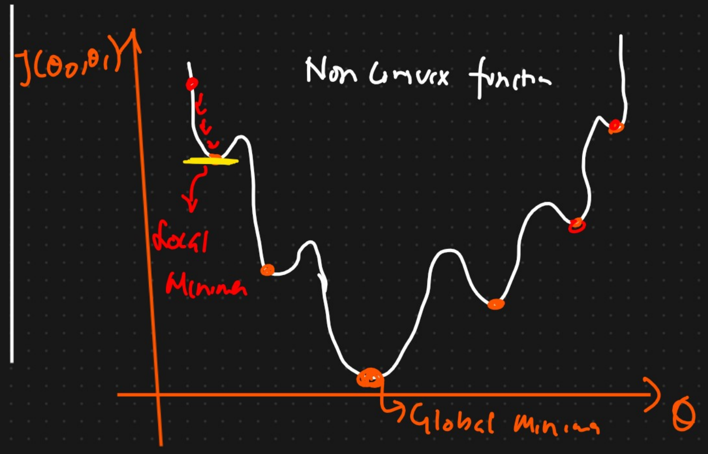
$$h_\theta(x) = \theta_0 + \theta_1 x_1$$

Logistic Regression Cost function

$$J(\theta_0, \theta_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x)_i)^2$$

$$h_\theta(x) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1)}}$$





Log Loss

$$J(\theta_0,\theta_1) = \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$

$$h_\theta(x) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1)}}$$

$$\boxed{J(\theta_0,\theta_1) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))} \Rightarrow \text{Convex function}$$

if $y=1$

$$= -\log(h_\theta(x)) \Rightarrow y=1$$

if $y=0 \Rightarrow -\log(1-h_\theta(x))$

Final Aim :

Minimize Cost function $J(\theta_0,\theta_1)$ by changing θ_0 & θ_1

Convergence Algorithm

Repeat $\quad j=0,1$

$\{$

$$\theta_j : \theta_j - \alpha \frac{\partial}{\partial \theta} J(\theta, \theta_1)$$

$\}$

Logistic regression is a statistical method used for binary classification problems, where the target variable has only two possible outcomes. It's called "logistic" because it models the probability of the default class using the logistic function.

In logistic regression, the dependent variable is binary, meaning it takes only two values, usually coded as 0 and 1, where 1 typically represents the presence of some phenomenon (e.g., "yes", "success", "positive outcome"), and 0 represents the absence (e.g., "no", "failure", "negative outcome").

The logistic regression model calculates the probability that a given input belongs to one of the two classes. It does this by fitting a logistic curve to the data. The logistic function (also called the sigmoid function) maps any real-valued number into a value between 0 and 1, which can be interpreted as a probability.

Logistic regression is a statistical method used for binary classification, predicting the probability of an outcome with two possible values. It models the relationship between a dependent binary variable and one or more independent variables using the logistic function, which outputs probabilities between 0 and 1. The model estimates the log-odds of the probability of the default class occurring. Coefficients are determined using maximum likelihood estimation. Logistic regression is widely used in fields like medicine, finance, and marketing for tasks such as disease prediction, credit risk assessment, and customer churn analysis.

Logistic regression is called "regression" because it was originally developed within the context of regression analysis. Here's a detailed explanation:

Historical Context
Origin: Logistic regression was developed as an extension of linear regression, which is used for predicting a continuous outcome. The term "regression" was retained because the method builds on the principles of linear regression.
Technical Explanation
Regression Model: In logistic regression, we are still modeling a relationship between a dependent variable and one or more independent variables. The difference is in the nature of the dependent variable:

Linear Regression: Predicts a continuous outcome.
Logistic Regression: Predicts a categorical outcome, typically binary (0 or 1).

Logistic Regression With Regularization Parameters

Cost function

$$\boxed{J(\theta_0,\theta_1) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))}$$

$$h_\theta(x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1)}}$$

$$J(\theta_0,\theta_1) = \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)). & \text{if } y=0 \end{cases}$$

Reduce Overfitting

$$J(\theta_0,\theta_1) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x)) + L2 \text{ Regularization}$$

feature selection

$$J(\theta,\theta_1) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x)) + L1 \text{ Regularization}$$

$$J(\theta,\theta_1) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x)) + L2 \text{ Reg} + L1 \text{ Reg.}$$

L2 Regularization ⇒ Reduce Overfitting

$$J(\theta,\theta_1) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x)) + \lambda\sum_{i=1}^{n}(\text{slope})^2$$

L1 Regularization ⇒ Feature Selection

λ ⇒ Hyperparameter

$$J(\theta, \theta_1) = -y \log(h_\theta(x)) - (1-y)\log(1-h_\theta(x)) + \lambda \sum_{i=1}^{n} |slope|$$

ElasticNet

$$J(\theta, \theta_1) = -y \log(h_\theta(x)) - (1-y)\log(1-h_\theta(x)) + \lambda_1 \sum_{i=1}^{n} (slope)^2 + \lambda_2 \sum_{i=1}^{n} |slope|$$

C & λ Relationship

$$\boxed{C = \frac{1}{\lambda}} \Rightarrow \boxed{\lambda = \frac{1}{C}}$$

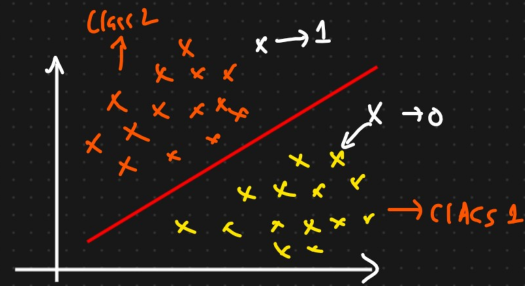
A confusion matrix is a performance measurement tool used in machine learning and statistics to evaluate the accuracy of a classification algorithm. It provides a summary of prediction results on a classification problem, showing the count of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Here's a detailed explanation of each component:

True Positive (TP): The number of instances that are correctly classified as positive.
False Positive (FP): The number of instances that are incorrectly classified as positive (Type I error).
True Negative (TN): The number of instances that are correctly classified as negative.
False Negative (FN): The number of instances that are incorrectly classified as negative (Type II error).

Performance Metrics, Accuracy, Precision, Recall And F-Beta

Topics to be covered

1. Confusion Matrix ✓
2. Accuracy ✓
3. Precision ✓
4. Recall ✓
5. F-Beta Score



Dataset

x_1	x_2	y (Actual)	$\hat{y}$ (Predicted) ← Model prediction	
–	–	0	1	→ Wrong Prediction
–	–	1	1	→ Correct Prediction
–	–	0	0	→ " "
–	–	1	1	→ " "
–	–	1	1	→ " "
–	–	0	1	→ Wrong Prediction
–	–	1	0	

1. Confusion Matrix

Actual values → y

Predicted value → $\hat{y}$	1	0
1	3	2
0	1	1

Predicted \ Actual	1	0
1	TP	FP
0	FN	TN

True is for correct prediction of model
positive is for 1 predicted value

$$\text{Model Acc.} = \frac{TP + TN}{TP + FP + FN + TN}$$

1.1 Accuracy

$$\frac{\text{correctly predicted instances}}{\text{total instances in the dataset}} = \frac{3+1}{3+2+1+1} = \frac{4}{7} = 0.571 \approx 57.1\%$$

In logistic regression, accuracy is one of the metrics used to evaluate the performance of the model. It represents the proportion of correctly classified instances out of the total instances in the dataset.

However, accuracy alone may not always provide a complete picture of model performance, especially in scenarios where classes are imbalanced. For example, if one class dominates the dataset, a naive classifier that always predicts that class could achieve high accuracy without actually learning anything useful.

(1) DATASET → Imbalanced dataset

⇓

1000 datapoints → 900 → 1, 100 → 0 ⇒ Imbalanced dataset

Dumb Model → o/p → 1 ⇒ $\boxed{\text{Accuracy} = 90\%}$

Imbalanced Dataset

In this scenario we cannot use Accuracy performance

② Precision $= \boxed{\dfrac{TP}{TP+FP}}$ } Out of all the actual values how many are correctly predicted

total no of +ve prediction

	1	0
1	TP	FP
0	FN	TN

FP → Important

FP ↓↓

In logistic regression, precision is a metric used to evaluate the quality of the positive predictions made by the model. It is defined as the ratio of true positives to the total number of positive predictions made by the model

Precision is important when the cost of false positives is high.
False poositive should be minimum

③ Recall $= \boxed{\dfrac{TP}{TP+FN}}$

⇒ FN ↓↓

⇒ Out of all the predicted values how many are correctly predicted with actual values

	1	0
1	TP	FP
0	FN	TN

In logistic regression, recall, also known as sensitivity or true positive rate, is a metric used to evaluate the ability of the model to correctly identify positive instances from all actual positive instances in the dataset. It measures the proportion of true positive instances that were correctly identified by the model.

Usecase 1 ⇒ Spam classification

Recall is particularly important when the cost of false negatives is high. For example, in medical diagnosis, missing a positive case could have serious consequences, so high recall is desired to minimize such errors.

Mail $\xrightarrow{\text{predict}}$ → Spam
→ Not a Spam

	1	0
1	TP	FP
0	FN	TN

Mail → Spam
Model → Spam } Good Scenario

FP ↓↓↓

0 ⇐ Mail → Not a Spam
1 ⇐ Model → Spam } ⇐ FP is Important
Blunder

1 ⇐ Mail → Spam
0 ⇐ Model → Not a Spam } ⇒ FN

⇓

PRECISION PERFORMANCE METRICS.

Usecase 2 ⇒ $\boxed{\text{FN is Important}}$

To predict whether a person has diabetes or not

	Diabetes	No Diabetes → Actual
Diabetes	TP	FP
No Diabetes	FN↓↓	TN

① Actual → Diabetes
Model → Diabetes } Good

② Actual → Diabetes
Model → No. Diabetes } Blunder ⇒ FN↓↓↓ ⇒ Important

③ Actual → No Diabetes
Model → Diabetes } FP ⇒ wrong prediction

④ Actual → No Diabetes
Model → No Diabetes } Correct

⇓

$\boxed{\text{RECALL}}$

Assignment: ① Tomorrow the stock will crash or not

Reduce FP↓↓ or FN↓↓

Both Important FP and FN

④ F-Beta Score $= (1+\beta^2)\dfrac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$

① If FP & FN are both important means accuracy and precision equally important

$\beta = 1$

Harmonic Mean

$$F1\ \text{Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

② If FP is more important than FN

$\beta = 0.5$

$$F\ 0.5\ \text{Score} = (1+0.25)\frac{P*R}{P+R}$$

③ If FN >> FP

$\beta = 2$

$$\boxed{F2\ \text{Score} = (1+4)\frac{P*R}{P+R}}$$

The F-beta score is a generalization of the F1 score that allows you to adjust the balance between precision and recall. It is particularly useful when you want to place more emphasis on either precision or recall, depending on the specific requirements of your application.

=1, F beta the score is the same as the F1 score. When
>1, recall is given more weight. When
<1, precision is given more weight.

CROSS VALIDATION AND ITS TYPES

1000 datapoints

DATASET

{DATA leakage}

70% → TRAINING

30% → Test → Model performance or Accuracy

Model Train ⟵ TRAINING

80% → TRAIN

20% → Validation → Model Hyperparameter Tunning

→ CROSS VALIDATION

random-state = 100 ✓

random-state = 42 ✓

85%, 70%, 90%, 95%, 96%

[70% ⟷ 96%]

Exp 1	TRAIN	VALIDATE	Acc 1
Exp 2	TRAIN	VALIDATE	Acc 2
Exp 3	TRAIN	VALIDATE	Acc 3

CROSS VALIDATION CV = 5

[Acc 1, Acc 2, Acc 3]

[80%, 85% 78%]

⇓

Final Acc = Mean of this Accuracy

Types of Cross Validation

① Leave One Out Cross Validation (LOO CV)

TRAINING DATA → 500 Records

Exp 1	1	499
	1	TRAINING → Acc 1

Validation DATA

Exp 2 | Tr | 1 | Training → Acc 2

Validation

⋮

Exp 500 | TRAINING | 1 → Acc 500

TRAINING → Model TRAIN

VALIDATION → Model Predict

⇒ Mean of All these Accuracies

Disadvantage

1. Time Complexity is huge for training Big datasets
2. Model Overfit → TRAINING Acc ↑↑
   New data → Validation Acc ↓↓

2. Leave p out CV

p = 10, 20, 30, 40 → Hyperparameter

⇓

$\boxed{CV=5}$

3. K Fold Cross Validation

K = 5 n = 500

Validation size $= \frac{500}{5} = 100$

500

Exp 1 | | → Acc 1

Validation Data TRAINING

Exp 2 | TRAIN | Validation | TRAIN → Acc 2

⋮

Average of all the Accuracy.

Exp 5 | TRAIN | Validation | → Acc 5

f1 f2 y

4

0, 0, 0, 0, 1, 1, 1, 0, 1

④ Stratified K Fold CV

{Imbalanced DATASET}.

{50's 1, 50's 0}

K=5 n=500 Validation = 100

No. of 0's & 1's is Equal

Exp 1 | 100 | TRAINING DATA |

Validation

n=500 100

{350 → 1, 150 → 0} {50's 0, 50's 1}

Exp 2 | | 100 | |

validation

Steps of k-Fold Cross-Validation
Shuffle the dataset randomly.
Split the dataset into k equally sized folds.
For each fold:
Train the model using k-1 folds.
Validate the model on the remaining fold.
Compute the performance metric (e.g., accuracy, F1 score) for each of the k runs.
Average the performance metrics to get the final estimate.

⑤ Time Series CV

Amazon review Sentiment Analysis

Product A

Stratified k-Fold Cross-Validation: Ensures that each fold has approximately the same percentage of samples of each target class as the original dataset. This is useful for imbalanced datasets.

JAN → DEC

TRAINING | VALIDATION

Day 1 Day 2 Day 3 Day 4 | - - - - - Day N

Time Series Application

Cross-validation is a technique used to evaluate the performance of a machine learning model by partitioning the data into subsets, training the model on some subsets, and testing it on the remaining subsets. This process helps ensure that the model generalizes well to unseen data and is not overfitting or underfitting. There are several types of cross-validation, with k-fold cross-validation being one of the most commonly used methods.

k-Fold Cross-Validation
In k-fold cross-validation, the dataset is divided into k equally sized folds. The model is trained and validated k times, each time using a different fold as the validation set and the remaining k-1 folds as the training set. The performance metric is averaged over the k runs to provide a more robust estimate of the model's performance.

Hyperparameter Tuning

↳ Finding the best parameters while training the model

① GridSearchCV

② RandomizedSearchCV

1) GridSearchCV [Grid Search + CROSS VALIDATION] [CV=5]

Logistic Regression

penalty{'l1', 'l2', 'elasticnet', None},

solver{'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga'}

⇓

CV ⇒ K fold CV

K=5

model = LogisticRegression(elasticnet, newton-cg)

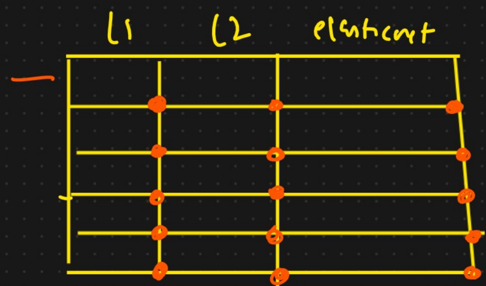
⇓

model.predict(new_data)

TRAIN AND VALIDATION = 5

⇓

Average Accuracy ⇒



Increase the Model Performance or Accuracy

Disadvantage

① Time Complexity increases for Training the Model

② RandomizedSearchCV ←

$\boxed{n\_iter=10}$ + $\boxed{CV=5}$

10 different Combination + CV=5

Advantage

(1) Time Complexity Decrease

Hyperparameter tuning is the process of finding the best set of hyperparameters for a machine learning model. Hyperparameters are parameters that are not learned from the data but are set before the training process, such as the learning rate in neural networks, the number of trees in a random forest, or the C parameter in SVMs. Tuning these parameters is crucial for optimizing the performance of the model.

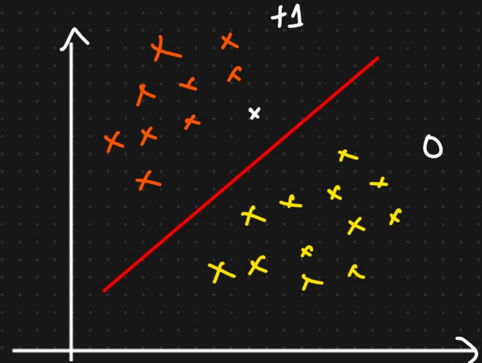
Common Methods for Hyperparameter Tuning
Grid Search: This method involves specifying a grid of hyperparameter values and exhaustively searching through all possible combinations to find the best one.

Random Search: Instead of searching all possible combinations, random search samples a fixed number of hyperparameter combinations from the specified grid.

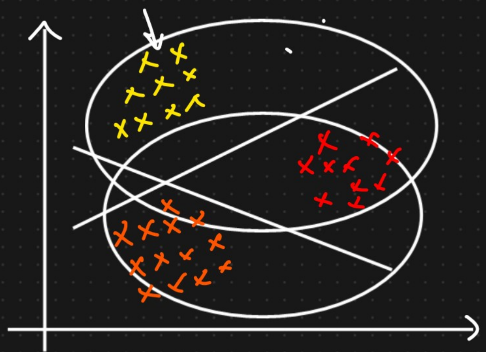
Logistic Regression For Multiclass Classification

① One Versus Rest

② Multinomial



f1	f2	f3	O/p
—	—	—	1
—	-	—	0
—	—	—	2
—	—	—	2
—	—	—	0
—	—	—	1



OvR

$\boxed{M_1}$ ⇓ Binary Classification

$\boxed{M_2}$ ⇓ Logistic Binary Classification

M_3 ⇓ Binary Classification

f1	f2	f3	O/p	O1	O2	O3
—	—	—	1	0	1	0
—	-	—	0	1	0	0
—	—	—	2	0	0	1
—	—	—	2	0	0	1
—	—	—	0	1	0	0
—	—	—	1	0	1	0

	I/p feature	O/p
$M_1 \Rightarrow$	f_1, f_2, f_3	$O_1 \rightarrow$ Binary classification
$M_2 \Rightarrow$	f_1, f_2, f_3	$O_2 \rightarrow$ Binary classification
$M_3 \Rightarrow$	$f_1\ f_2\ f_3$	$O_3 \rightarrow$ " "

$$\text{New Data point} = \begin{array}{ccc} M_1 & M_2 & M_3 \end{array} \left[0.20,\ 0.35,\ \underline{0.45} \right] \Rightarrow 1$$

⇓ ⇓

$\boxed{03} \leftarrow$ O/p Multinomial probabilities.

Logistic regression can be extended to handle multiclass classification problems, where there are more than two classes to predict. The two main approaches for using logistic regression in multiclass classification are One-vs-Rest (OvR) and Softmax Regression (also known as Multinomial Logistic Regression).

One-vs-Rest (OvR)
In the One-vs-Rest approach, a separate binary classifier is trained for each class. Each classifier predicts whether a sample belongs to its class or not. During prediction, the class with the highest confidence score is selected.

Softmax Regression (Multinomial Logistic Regression)
Softmax regression generalizes logistic regression to multiple classes by using a softmax function to predict the probabilities of each class. The class with the highest probability is chosen as the prediction.