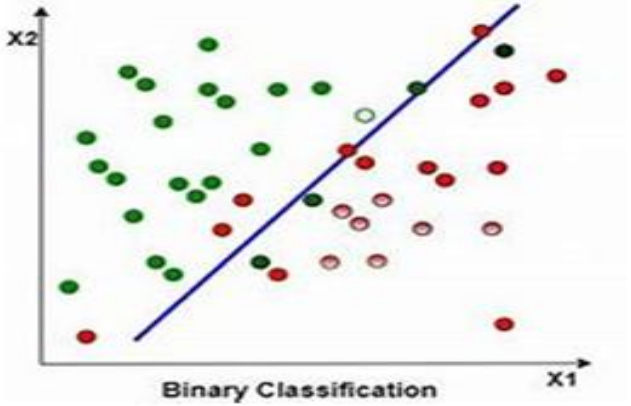
Logistic Regression

Logistic regression (Classification Algorithm)

- It is a predictive analysis algorithm and based on the concept of probability.
- Measures the relationship between the independent variables (features) and Discrete dependent variable by estimating probabilities using logistic function.
- It used to assign observations to a discrete set of classes. Also, it is called Binomial Logistic Regression.
- E.g.
  - Email spam or not spam,
  - Online transactions Fraud or not Fraud,
  - Tumor Malignant or Benign.



- Logistic regression transforms its output using **sigmoid (logistic)** function to return a probability value.
- Logistic regression limits the hypothesis function between 0 to 1 i.e., h(x) / $y' \epsilon$ [0,1]

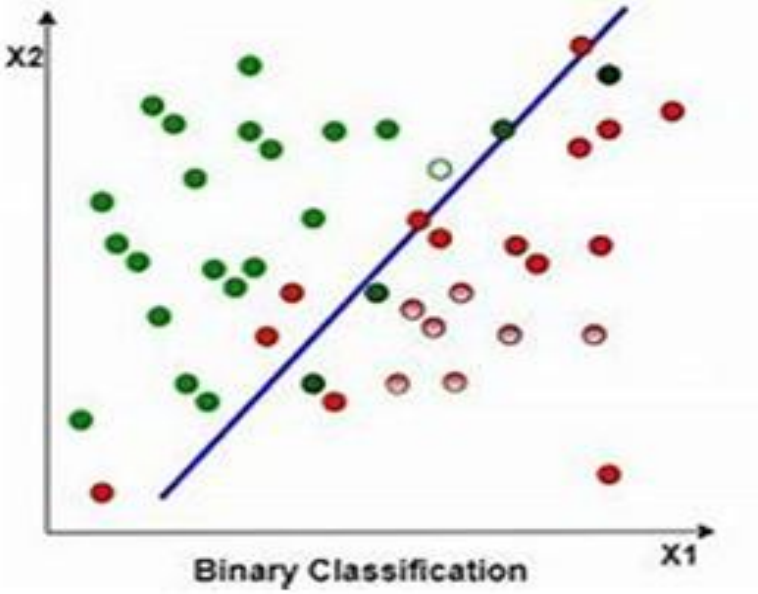
$$0 \leq h_\theta(x) \leq 1$$

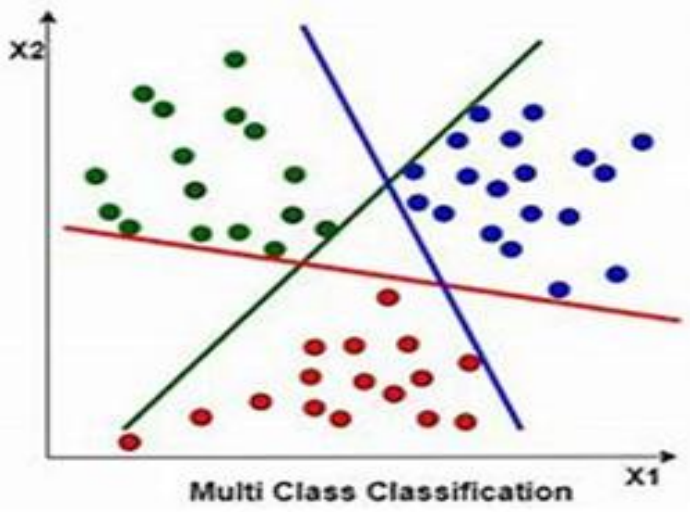
Types of classification using Logistic Regression

- Binary classification(e.g., Tumor Malignant or Benign)
- Multi-Class classification(e.g., Cats, dogs or Sheep's)

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity i.e., independent variables must be independent of each other.



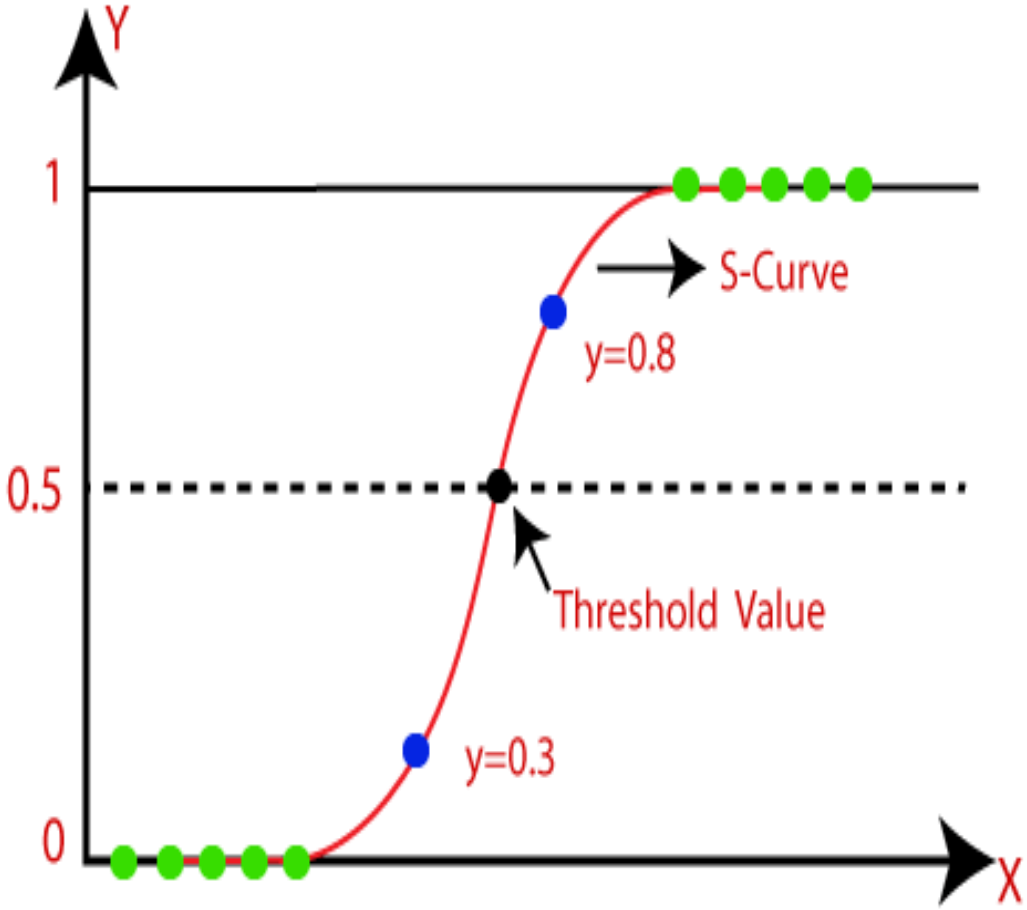


Sigmoid Function

- Maps the predicted values to probabilities.
- Sigmoid Function maps any real value into another value between 0 and 1.
- Hence it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or logistic function.

$$\sigma(z) = p = \frac{1}{1+e^{-z}} = \frac{e^{z}}{e^{z}+1}.$$

- $z = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots\ldots + \theta_n x_n$
- x_i is independent variable $i = 0,1,2, \ldots, n$



Threshold classifier output y' *or* $h_\theta(x)$ at 0.5:

If $\boldsymbol{h_\theta(x) \geq 0.5}$, predict "**y = 1**"

If $\boldsymbol{h_\theta(x) < 0.5}$, predict "**y = 0**"

Cost / Error/ Loss function

1. Log Loss is the negative average of the log of corrected predicted probabilities (P) for each instance

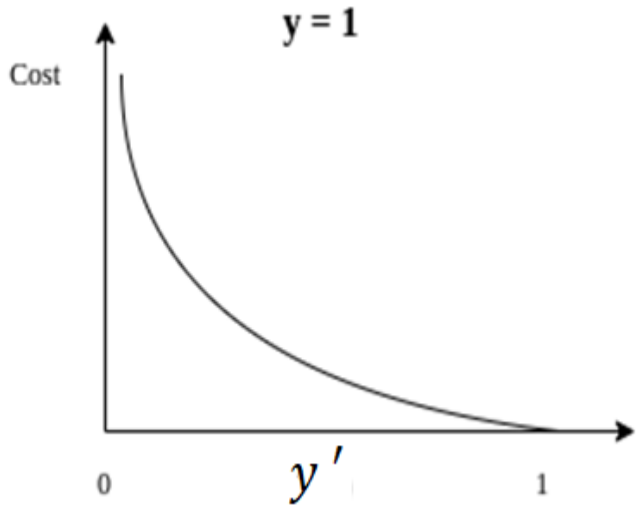
$$log\ loss = -1/N \sum_{i=1}^{N} (log\ (P\,i\,))$$

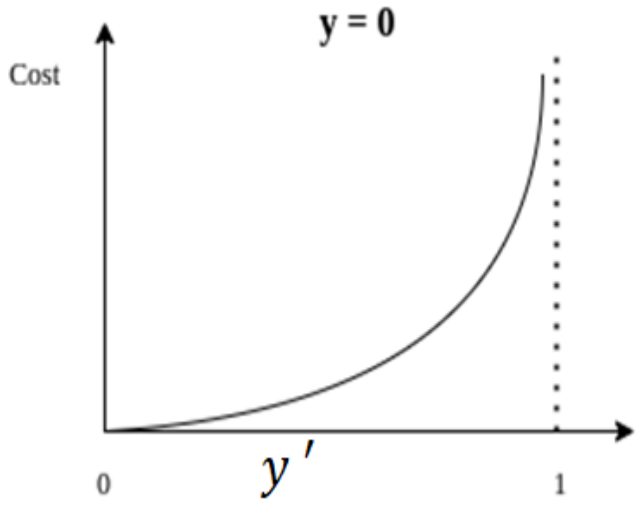
2. Binary cross entropy is used as cost function.

- Cost Function $J\,(\theta) = -(\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\ log\ y'^{(i)} + (1 - y^{(i)})\ log(1 - y'^{(i)})))$

$y^{(i)}$ - Ground truths or Actual output, $y'^{(i)}$ - Prediction output, m - No. of data points or samples

- When the actual output y = 1, the cost is 0 for y' = 1.
- Similarly, if y = 0, the cost is 0 for y' = 0.
- So, the output can either be {0} or {1}



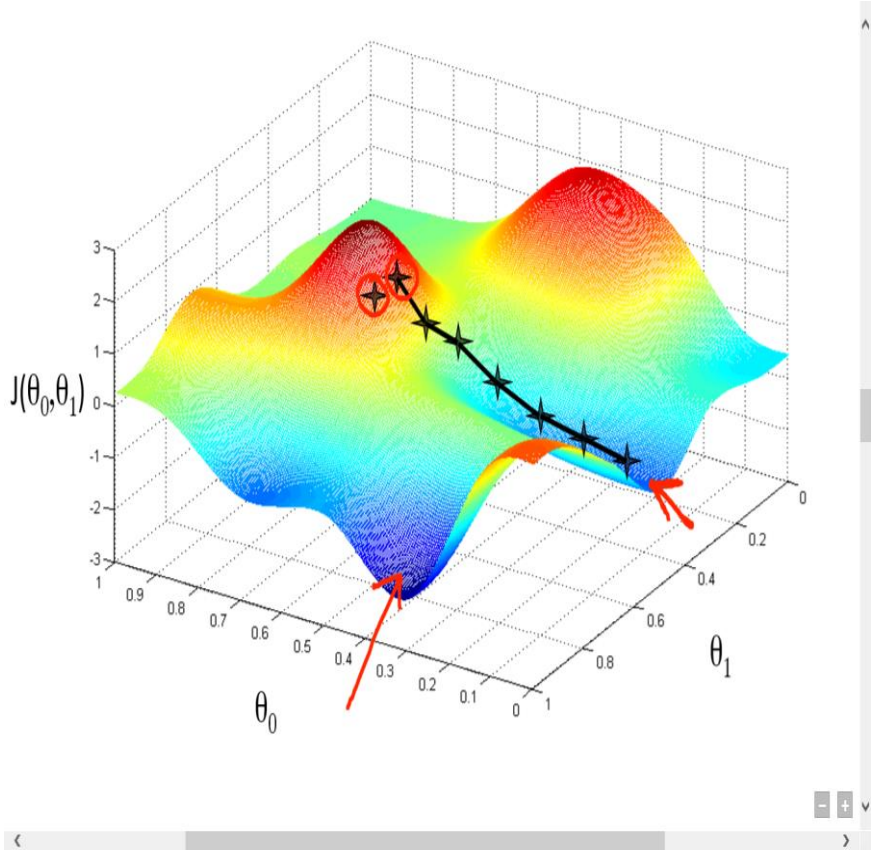


Gradient Descent

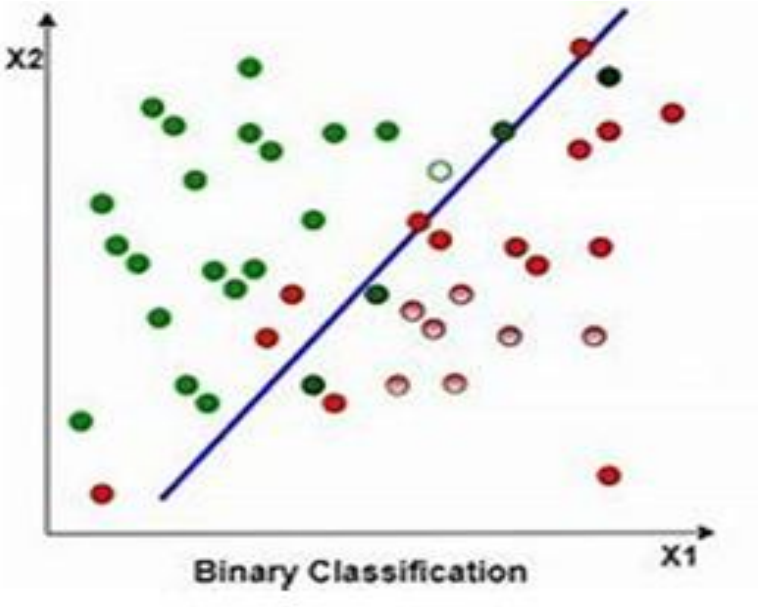
- The objective of training a machine learning model is to minimize the loss or error between ground truths and predictions by changing the trainable parameters.
- Gradient is the extension of derivative in multi-dimensional space, tells the direction along which the loss or error is optimally minimized.
- Gradient is defined as the maximum rate of change.

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- θ_j-Training parameter
- α – Learning rate (Magnitude of the next step)
- $J(\theta)$ – Error / Cost function
- $\frac{\partial}{\partial \theta_j}$ - Direction of the next step



Practical Steps in Logistic Regression:



To implement the Logistic Regression using Python,

- **Data Pre-processing**
- **Fitting Logistic Regression to the Training set**

  LogisticRegression(solver = 'liblinear', multi class = 'ovr')

  multi_class − {'ovr', 'multinomial', 'auto'}, default = 'ovr' − option for a binary problem.

  - Solver is algorithm - {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default='lbfgs'
    - For small datasets, 'liblinear' is a good and 'sag' and 'saga' are faster for large dataset.
    - For multiclass problems - newton-cg', 'sag', 'saga' and 'lbfgs'
    - 'liblinear' is limited to one-versus-rest schemes.
- **Predicting the test result**
- **Test accuracy of the result(Creation of Confusion matrix)**
- **Visualizing the test set result.**

Regularization in Logistic Regression

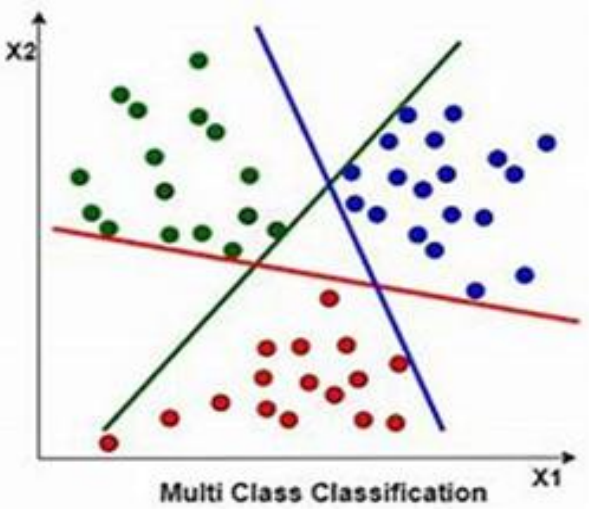
- Overfitting occurs when model tries to cover all the data points, or more than the required data points present in the dataset.
- Due to this, the model learns noise and inaccurate values in the dataset. It leads to the overfitting with low bias and high variance.
- Regularization needs to avoid the overfitting.
- It keeps all the features in the dataset but reduce magnitude/values of parameters θ.
- Each feature contributes to prediction.
- Logistic regression model uses two strategies to overcome the overfitting:
    - L_1 regularization (**Losso Regression)**.
    - L_2 regularization (**Ridge Regression)**.
    - Early stopping, i.e., limiting the number of training steps or the learning rate.

Regularization in Logistic Regression

- Add L_1 regularization term to the cost function
  - $J(\theta) = -(\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\, log\, y'^{(i)} + (1-y^{(i)})\, log(1-y'^{(i)}))) + \lambda \sum_{j=0}^{n}|\theta_j|$
- Add L_2 regularization term to the cost function
  - $J(\theta) = -(\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\, log\, y'^{(i)} + (1-y^{(i)})\, log(1-y'^{(i)}))) + \lambda \sum_{j=0}^{n}\theta_j^2$

Multinomial Logistic Regression for Multi-class classification

- It predicts a multinomial probability (i.e., more than two classes) for each input example.
  - Split the multi-class classification problem into multiple binary classification problems
  - Then apply a standard logistic regression model on each subproblem. It includes one-vs-rest and one-vs-one wrapper models.
  - **An alternate approach** is directly predicting the probability of given input example that belongs to each known class label.



Changes in binomial logistic regression to multinomial logistic regression:

- Apply cross-entropy loss
- Change the output from a single probability value to one probability for each class label.

model = LogisticRegression(multi_class='multinomial', solver='lbfgs')

- Cost Function $J(\theta) = -(\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\ log\ y'^{(i)} + (1-y^{(i)})\ log(1-y'^{(i)})))$

Gradient Descent to update the parameters

- $\theta_j = \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$

VIT-AP UNIVERSITY

References

1. Tom M. Mitchell, Machine Learning, McGraw Hill , 2017.

2. EthemAlpaydin, Introduction to Machine Learning (Adaptive Computation and Machine Learning), The MIT Press, 2017.

3. Wikipedia