

## 1.3. Logistic Regression

---

### What is Classification Problem?

Classification is a supervised learning problem wherein the target variable is categorical unlike regression where the target variable is continuous. Predicting a qualitative (categorical) response for an observation can be referred to as classifying that observation, since it involves assigning the observation to a category, or class and the process for predicting such responses is known as classification. Classification can be binary i.e. only two possible values of the target variable or multi-class i.e. more than two categories.

Often the methods used for classification first predict the probability of each of the categories of a categorical variable, as the basis for making the classification.

There are many possible classification techniques, or classifiers such as logistic regression, K-nearest neighbours (KNN), decision trees and random forests.

Classification techniques are an essential part of machine learning and data mining applications. Classification problems occur often, perhaps even more so than regression problems. Approximately 70% of problems in Data Science are classification problems.

#### Classification Problem use case examples

Domain	Question
Telecom	Is a customer likely to leave the network? (churn prediction)
Retail	Is he a prospective customer?, that is, likelihood of purchase vs. non-purchase?
Insurance	To issue insurance, should a customer be sent for a medical checkup?
Insurance	Will the customer renew the insurance?
Banking	Will a customer default on the loan amount?
Banking	Should a customer be given a loan?
Manufacturing	Will the equipment fail?
Health Care	Is the patient infected with a disease?
Health Care	What type of disease does a patient have?
Entertainment	What is the genre of music?

Just as in the regression setting, in the classification setting we have a set of training observations  $(x_1, y_1), \dots, (x_n, y_n)$  that we can use to build a classifier.

As already discussed, classification is about predicting a categorical output from inputs of arbitrary types (numerical and/or categorical). Since the output is qualitative, it can only take values from a finite set.

We generally use  $K$  to denote the number of elements in the set of possible output values. The set of possible output values can, for instance, be  $\{\text{false}, \text{true}\}$  ( $K = 2$ ) or  $\{\text{Underweight}, \text{Healthy}, \text{Overweight}, \text{Obese}\}$  ( $K = 4$ ). We will refer to these elements as classes or labels. The number of classes  $K$  is assumed to be known throughout supervised machine learning.

To prepare for a concise mathematical notation, we generically use integers  $1, 2, \dots, K$  to denote the output classes. The integer labeling of the classes is arbitrary, and we use it only for notational convenience. The use of integers does not mean there is any inherent ordering of the classes.

When there are only  $K = 2$  classes, we have the important special case of binary classification. In binary classification, we often use the labels 0 and 1 (instead of 1 and 2). Occasionally, we will also use the terms positive (class  $K = 1$ ) and negative (class  $K = 0$ ) as well.

Classification amounts to predicting the output from the input. In statistical approach, we understand classification as the problem of predicting class probabilities

$$P(y | X)$$

where  $y$  is the output ( $1, 2, \dots$  or  $K$ ) and  $X$  is the input and  $P(y | X)$  describes the probability for the output  $y$  (a class label) given that we know the input  $X$ .

## Logistic Regression

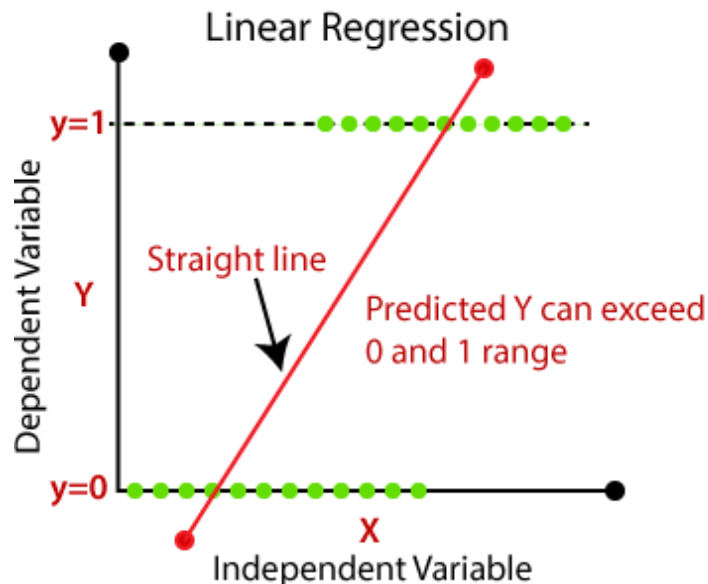
Logistic Regression is one of the simplest and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. Its basic fundamental concepts are also constructive in deep learning.

Logistic regression is a statistical model that in its basic form uses a *logistic function* to model a binary dependent variable. Mathematically, a binary logistic model has a dependent variable with two possible values, such as Yes/No which is represented by a target variable or label, where the two values are labeled "0" and "1".

**WARNING:** *Despite its name, logistic regression is a method for classification, not regression! The (confusing) name is due only to historical reasons. Thus, Logistic Regression is all about predicting binary variables, not predicting continuous variables.*

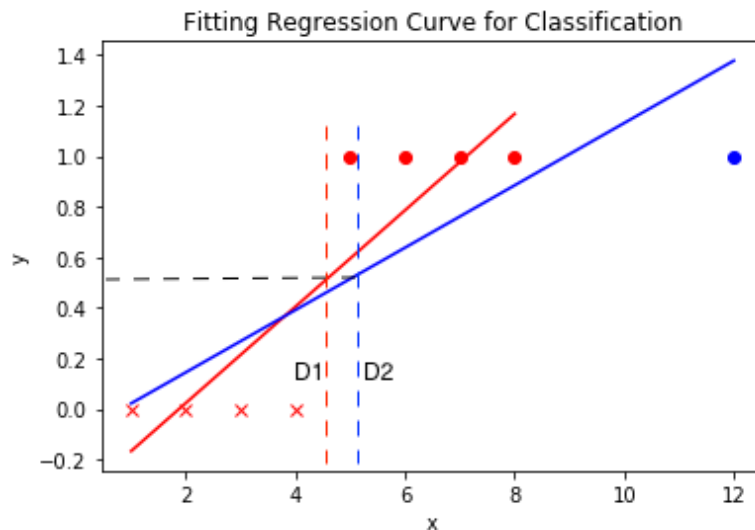
## Why Linear Regression will not work?

The most basic step would be to try and fit regression curve to see if one can achieve classification using the same approach as in Linear Regression. Below is a plot attempting the same.



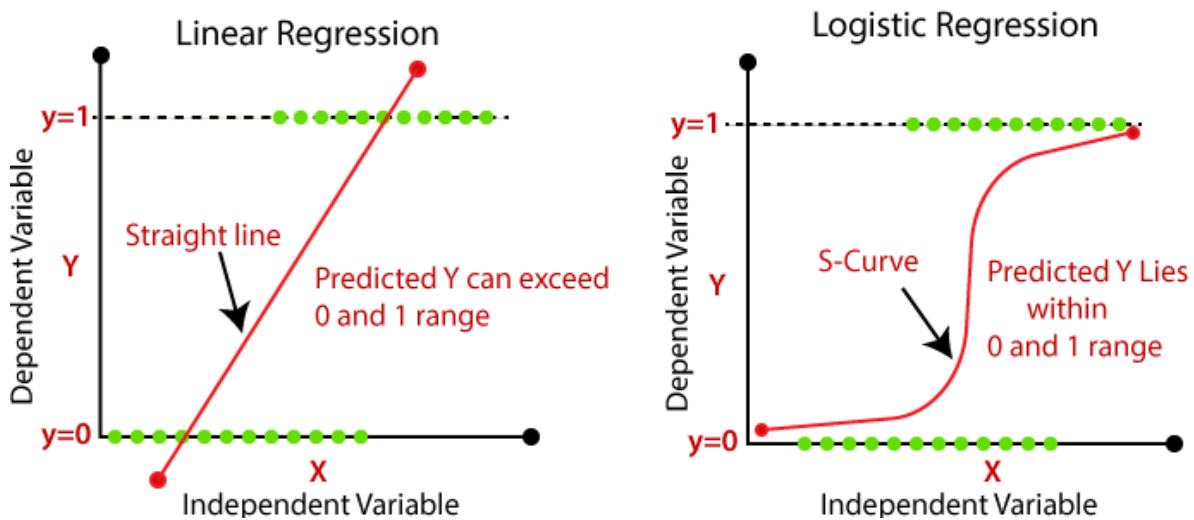
Linear regression model can generate the predicted probability as any number ranging from negative to positive infinity, whereas probability of an outcome can only lie between  $0 < P(x) < 1$ .

Also, Linear regression has a considerable effect on outliers.



It can be seen below that the attempt to achieve classification using regression curve and thresholding will not always yield conclusive results. In first case say only red data points are in the dataset, then fitting the curve and setting the threshold at 0.5 would work, but say an outlier is present like the blue data point then the same decision boundary D1 would shift to D2 if the threshold is kept constant and the boundary would not be perfect.

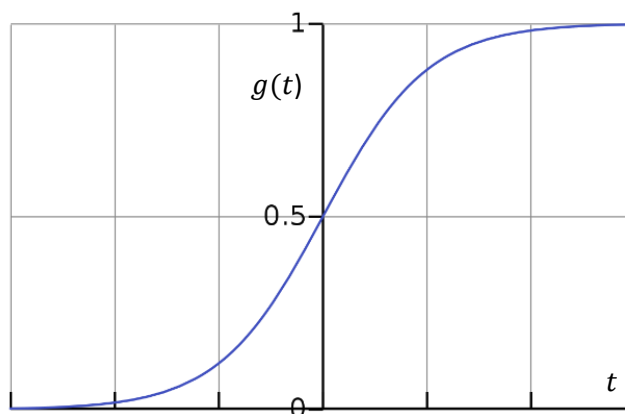
As a result of such constraints, we cannot directly apply linear regression because it won't be a good fit. Instead, we can transform our linear regression to a logistic regression curve. And hence, we need to apply the logistic function (also called the 'inverse logit' or 'sigmoid function').



## The Logistic Regression model

Instead of fitting a straight line (or hyperplane), the logistic regression model uses the *logistic function* to squeeze the output of a linear equation between 0 and 1. The logistic function is defined as:

$$g(t) = \frac{1}{1 + e^{-t}} = \frac{e^t}{e^t + 1}$$



The step from linear regression to logistic regression is kind of straightforward. In the linear regression model, we have modelled the relationship between outcome and features with a linear equation:

$$\hat{y} = b_0 + b_1x$$

In logistic regression, the response variable describes the probability that the outcome is the positive case i.e.,  $y = 1$ . The probability that the output is 1 given its input can be represented as:

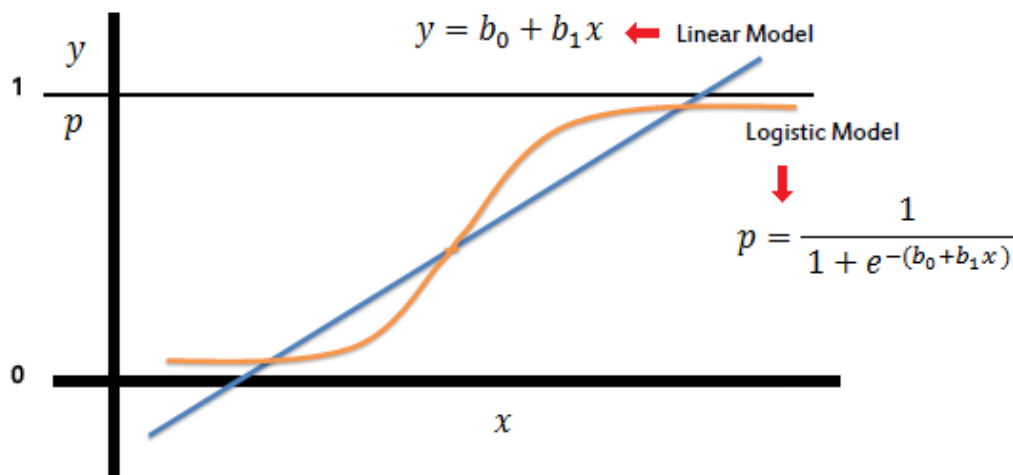
$$P(y = 1 | x)$$

If the response variable is equal to or exceeds a discrimination threshold, the positive class is predicted; otherwise, the negative class is predicted. The response variable is modeled as a function of a linear combination of the explanatory variables using the logistic function.

For classification, we prefer probabilities between 0 and 1, so we wrap the right side of the linear regression equation into the logistic function. This forces the output to assume only values between 0 and 1.

$$p = P(y = 1 | x) = \frac{1}{1 + e^{-(b_0 + b_1x)}}$$

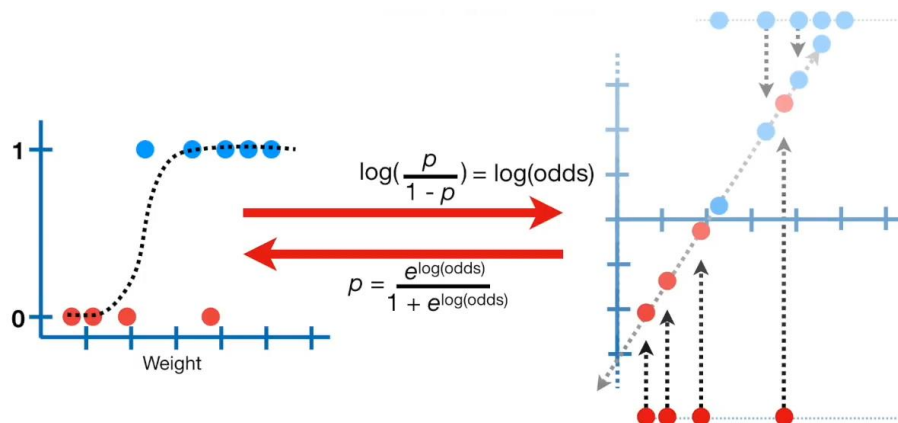
We can see that logistic regression uses independent variables in the same way as linear regression but passes them through a sigmoid activation so that the outputs are bound between 0 and 1. Rather than modeling the response  $Y$  directly, logistic regression models the probability that  $Y$  belongs to a particular category.



## Interpretation of coefficients

The interpretation of the coefficients in logistic regression differs from the interpretation of the coefficients in linear regression, since the outcome in logistic regression is a probability between 0 and 1. The coefficients do not influence the probability linearly any longer.

The linear equation is transformed by the logistic function to a probability. Therefore, we need to reformulate the equation for the interpretation so that only the linear term is on the right side of the formula.

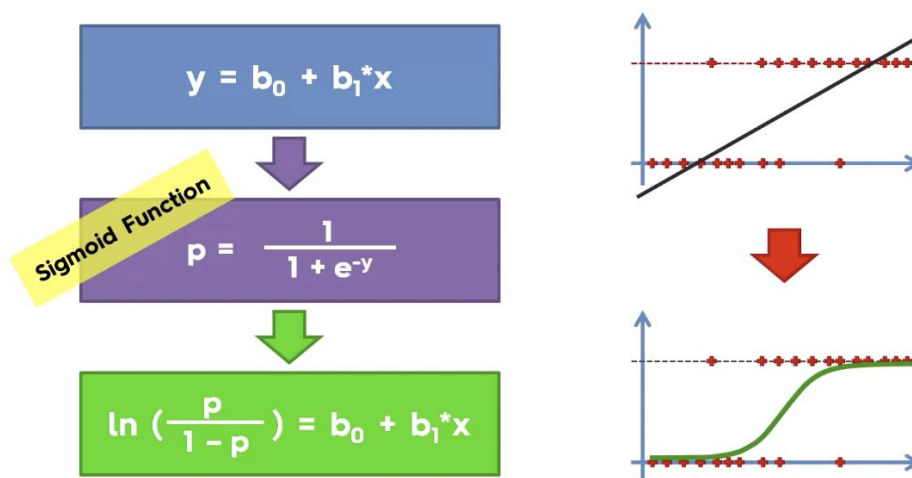


The logit function is the inverse of the logistic function. It links the probability back to a linear combination of the explanatory variables:

$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1x$$

**The term  $\frac{p}{1-p}$  is called as odds.** Thus,

$$\ln(odds) = b_0 + b_1x$$



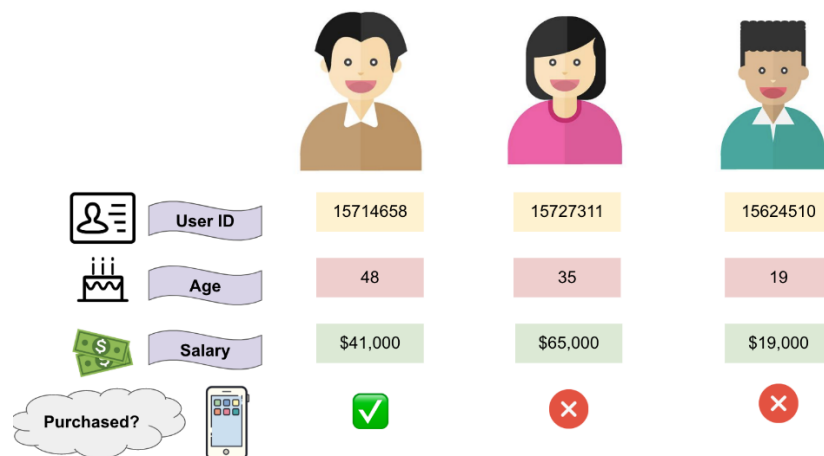
We can see from the above figure that the output of the linear regression is passed through a sigmoid function that can map any real value between 0 and 1.











If the curve goes to positive infinity,  $y$  predicted will become 1, and if the curve goes to negative infinity,  $y$  predicted will become 0. If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO.

### Example:

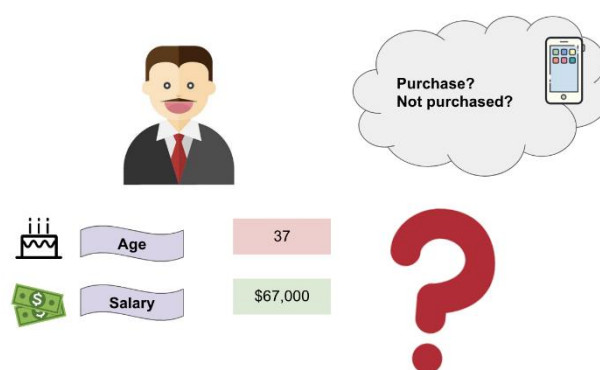
Imagine that you are a store manager at a smartphone store, increasing the sales revenue is your goal this month. Therefore, you need to know who the potential customers are in order to maximize the sale amount.

The client information you have is including Estimated Salary, Gender, Age, and Customer ID.



			
 User ID	15714658	15727311	15624510
 Age	48	35	19
 Salary	\$41,000	\$65,000	\$19,000
 Purchased?			

If now we have a new potential client who is 37 years old and earns \$67,000, can we predict whether he will purchase an iPhone or not (Purchase?/ Not purchase?)



See Jupyter Notebook 'iphone\_purchase\_pred.ipynb' for logistic regression implementation for this problem.

## Model Evaluation of a Binary Classification

A variety of metrics exist to evaluate the performance of binary classifiers against true labels. The most common metrics are accuracy, precision, recall, F1 measure, and ROC AUC score. All of these measures depend on the concepts of true positives, true negatives, false positives, and false negatives.

- ✓ Positive and negative refer to the classes.
- ✓ True and false denote whether the predicted class is the same as the true class.

### Confusion Matrix

A confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a classifier. It is used for describing the performance of a classification model.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	<b>TP</b>	<b>FP</b>
	Negative (0)	<b>FN</b>	<b>TN</b>

The columns represent the actual (true) class or label. The rows represent the prediction. Each individual box represents the number of one of the following:

- **True Positive (TP):** The model correctly predicts the outcome as positive.
- **True Negative (TN):** The model correctly predicts the outcome as negative.
- **False Positive (FP):** The model incorrectly predicted the outcome as positive, but the actual result is negative. It is also known as Type-I error.
- **False Negative (FN):** The model incorrectly predicted the outcome as negative, but the actual result is positive. It is also known as Type-II error.

\*\*\* Ideally a good model should have high TN and TP and less of Type I & II errors.



### True Positive (TP)

- ✓ The predicted value matches the actual value
- ✓ The actual value was positive and the model predicted a positive value

### True Negative (TN)

- ✓ The predicted value matches the actual value
- ✓ The actual value was negative and the model predicted a negative value

### False Positive (FP) – Type 1 error

- ✓ The predicted value was falsely predicted
- ✓ The actual value was negative but the model predicted a positive value
- ✓ Also known as the Type 1 error

### False Negative (FN) – Type 2 error

- ✓ The predicted value was falsely predicted
- ✓ The actual value was positive but the model predicted a negative value
- ✓ Also known as the Type 2 error

**Example:** Imagine that we created a machine learning model that predicts whether a patient has cancer or not. The table on the left shows twelve predictions that the model made as well as the actual result of each patient. With our paired-data, you can then fill out the confusion matrix

Predicted Class	Actual Class
No Cancer	No Cancer
Has Cancer	Has Cancer
No Cancer	No Cancer
No Cancer	No Cancer
No Cancer	Has Cancer
Has Cancer	Has Cancer
No Cancer	No Cancer
Has Cancer	No Cancer
No Cancer	No Cancer
No Cancer	No Cancer
Has Cancer	No Cancer
No Cancer	No Cancer



		Actual Class	
		Has Cancer	No Cancer
Predicted Class	Has Cancer	2	2
	No Cancer	1	7

**Example:** For our smartphone purchase classifier example, a true positive prediction is when the classifier correctly predicts that client will purchase the iphone. A true negative prediction is when the classifier correctly predicts that the client will not purchase the iphone.

A prediction that a client will buy the smartphone when in actual data he/she doesn't is a false positive prediction, and a prediction incorrectly classified as not purchased when the client actually does is false negative prediction.

## Computing Accuracy, Recall, Precision, and Other Metrics

Based on the confusion matrix, you can calculate the following metrics:

- **Accuracy:** This is defined as the sum of all correct predictions divided by the total number of predictions, or mathematically:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

		<i>Accuracy</i> <i>Actual</i>	
		0	1
<i>Prediction</i>	0	TN	FN
	1	FP	TP

While accuracy measures the overall correctness of the classifier, it does not distinguish between false positive errors and false negative errors. Some applications may be more sensitive to false negatives than false positives, or vice versa. Furthermore, accuracy is not an informative metric if the proportions of the classes are skewed in the population.

This metric is easy to understand. After all, if the model correctly predicts 99 out of 100 samples, the accuracy is 0.99, which would be very impressive in the real world. But consider the following situation: Imagine that you're trying to predict the failure of equipment based on the sample data. Out of 1,000 samples, only three are defective. If you use a dumb algorithm that always returns negative (meaning no failure) for all results, then the accuracy is 997/1000, which is 0.997. This is very impressive, but does this mean it's a good algorithm? No. If there are 500 defective items in the dataset of 1,000 items, then the accuracy metric immediately indicates the flaw of the algorithm. In short, accuracy works best with evenly distributed data points, but it works really badly for a skewed dataset.

For these reasons, classifiers are often evaluated using two additional measures called precision and recall.

- **Precision:** This metric is defined to be

$$\frac{TP}{(TP + FP)}$$

This metric is concerned with number of correct positive predictions.

		<i>Precision</i> <i>Actual</i>	
		0	1
<i>Prediction</i>	0	TN	FN
	1	FP	TP

You can think of precision as “of those predicted to be positive, how many were actually predicted correctly?”

- **Recall** (also known as True Positive Rate (TPR)): This metric is defined to be

$$\frac{TP}{(TP + FN)}$$

This metric is concerned with the number of correctly predicted positive events. You can think of recall as “of those positive events, how many were predicted correctly?”

		<i>Recall</i> <i>Actual</i>	
		0	1
<i>Prediction</i>	0	TN	FN
	1	FP	TP

Sometimes called sensitivity in medical domains, recall is the fraction of the truly positive instances that the classifier recognizes. A recall score of one indicates that the classifier did not make any false negative predictions.

- **F1 Score:** This metric is defined to be

$$2 * \frac{(\text{precision} * \text{recall})}{(\text{precision} + \text{recall})}.$$

It is a good way to summarize the evaluation of the algorithm in a single number. It can have a maximum score of 1 (perfect precision and recall) and a minimum of 0. Overall, it is a measure of the preciseness and robustness of your model.

The F1 measure penalizes classifiers with imbalanced precision and recall scores, like the trivial classifier that always predicts the positive class. A model with perfect precision and recall scores will achieve an F1 score of one. A model with a perfect precision score and a recall score of zero will achieve an F1 score of zero.

#### Classification Performance Metrics

Metric	Description	Formula
Accuracy	what % of predictions were correct?	$(TP+TN)/(TP+TN+FP+FN)$
Misclassification Rate	what % of prediction is wrong?	$(FP+FN)/(TP+TN+FP+FN)$
True Positive Rate OR Sensitivity OR Recall (completeness)	what % of positive cases did model catch?	$TP/(FN+TP)$
False Positive Rate	what % of 'No' were predicted as 'Yes'?	$FP/(FP+TN)$
Specificity	what % of 'No' were predicted as 'No'?	$TN/(TN+FP)$
Precision (exactness)	what % of positive predictions were correct?	$TP/(TP+FP)$
F1 score	Weighted average of precision and recall	$2*((\text{precision} * \text{recall}) / (\text{precision} + \text{recall}))$

## Receiver Operating Characteristic (ROC) Curve

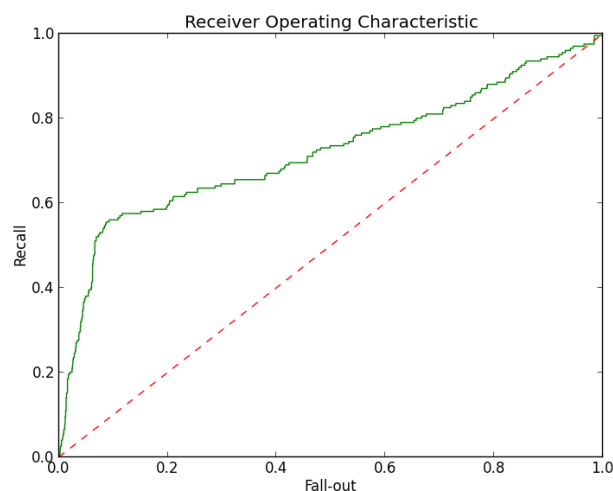
A Receiver Operating Characteristic, or ROC curve, visualizes a classifier's performance.

The ROC curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values.

ROC curves plot the classifier's recall against its False Positive Rate (or Fall-Out). FPR is the number of false positives divided by the total number of negatives. It is calculated using the following formula:

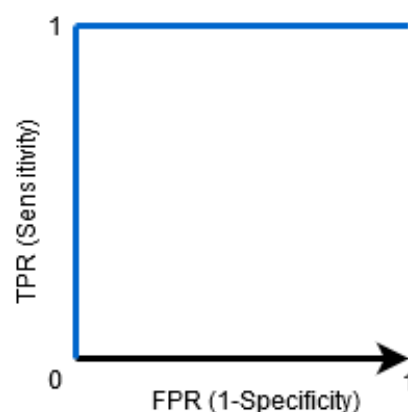
$$FPR = \frac{FP}{TN + FP}$$

FPR corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points. In other words, the higher FPR, the more negative data points you'll misclassify.



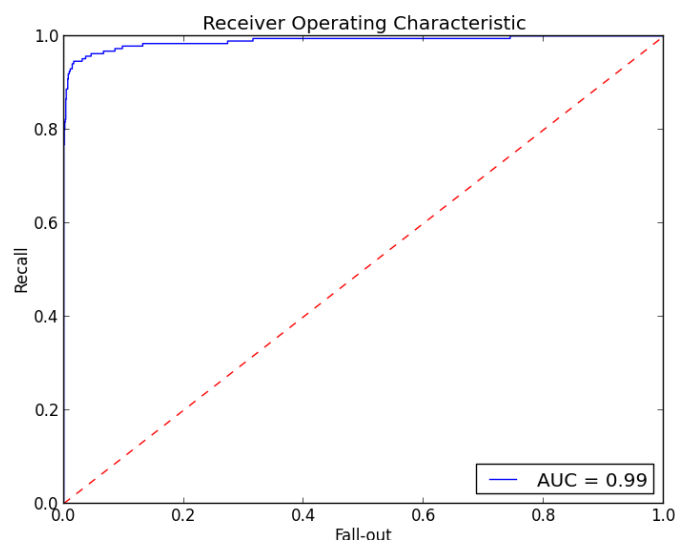
**AUC** is the area under the ROC curve; it reduces the ROC curve to a single value, which represents the expected performance of the classifier. The AUC is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

***The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.***



When  $AUC = 1$ , then the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.

When  $AUC=0.5$ , then the classifier is not able to distinguish between Positive and Negative class points. Meaning either the classifier is predicting random class or constant class for all the data points.



So, the higher the AUC value for a classifier, the better its ability to distinguish between positive and negative classes.

## Extra Readings

- The probability that an event will occur is the fraction of times you expect to see that event in many trials. If the probability of an event occurring is  $p$ , then the probability of the event not occurring is  $1 - p$ . Probabilities always range between 0 and 1.
- The odds are defined as the probability that the event will occur divided by the probability that the event will not occur. Unlike probability, the odds are not constrained to lie between 0 and 1 but can take any value from zero to infinity.

If the probability of Success is  $p$ , then the odds of that event is:

$$odds = \frac{p}{1 - p}$$

*Example:* If the probability of success ( $P$ ) is 0.60 (60%), then the probability of failure ( $1 - P$ ) is  $1 - 0.60 = 0.40$  i.e., 40%.

Then the odds are  $0.60 / (1 - 0.60) = 0.60 / 0.40 = 1.5$ .

- To transform the model from linear regression to logistic regression we use logistic function.

In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables.

The natural logarithm of the odds ratio is equivalent to a linear function of the independent variables. The antilog of the logit function allows us to find the estimated regression equation.

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

$$\frac{p}{1-p} = e^{\beta_0 + \beta_1 x}$$

$$p = e^{\beta_0 + \beta_1 x} (1 - p)$$

$$p = e^{\beta_0 + \beta_1 x} - e^{\beta_0 + \beta_1 x} * p$$

$$p + e^{\beta_0 + \beta_1 x} * p = e^{\beta_0 + \beta_1 x}$$

$$p(1 + e^{\beta_0 + \beta_1 x}) = e^{\beta_0 + \beta_1 x}$$

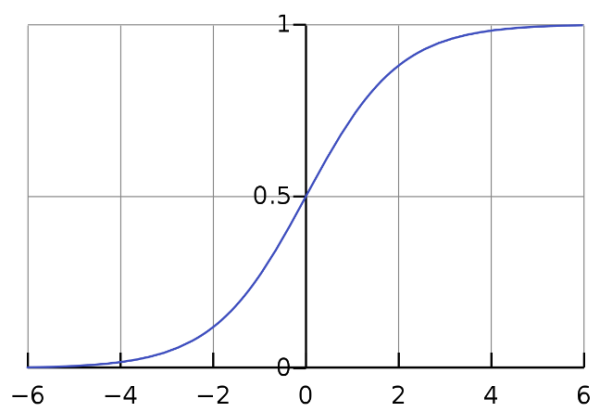
$$p = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

$$\hat{p} = \frac{e^{b_0 + b_1 x}}{1 + e^{b_0 + b_1 x}}$$

- **Sigmoid Function:**

A sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve. A common example of a sigmoid function is the logistic function defined by the formula:

$$f(t) = \frac{1}{1 + e^{-t}} = \frac{e^t}{e^t + 1}$$



The logistic sigmoid function is invertible, and its inverse is the logit function.

$$\ln\left(\frac{p}{1-p}\right) \text{ is the } \text{logit}(p)$$

Remember,  $\log_e x = \ln x$

$$\text{logit}^{-1}(t) = \frac{1}{1 + e^{-t}} = \frac{e^t}{1 + e^t}$$

### Logit Function:

When you apply the natural logarithm function to the odds, you get the logit function. The logit function is the logarithm of the odds

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

### • Multiple Logistic Regression

We now consider the problem of predicting a binary response using multiple predictors. By analogy with the extension from simple to multiple linear regression, we can generalize as follows:

For binary classification ( $y$  is either 0 or 1), learning a classifier amounts to learning a model of  $P(y = 1 | X)$  and  $P(y = 0 | X)$ .

From the previous lessons, we have the linear regression model. If we slightly change the notation

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p = \beta^T X$$

The input to this function is  $X$ , and the output, here denoted by  $z$ . Note that we have skipped the noise  $\varepsilon$ . In classification we are interested in  $P(y = 1|x)$ , which however is a function of  $X$  which takes values only within the interval  $[0, 1]$ .

The key idea underlying logistic regression is thus to 'squeeze' the output from linear regression  $z$  into the interval  $[0,1]$  by using the logistic function:

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

Since the logistic function is limited to take values between 0 and 1, we obtain altogether a function from  $X$  to  $[0, 1]$ , which we can use as a model for  $P(y = 1 | x)$ ,

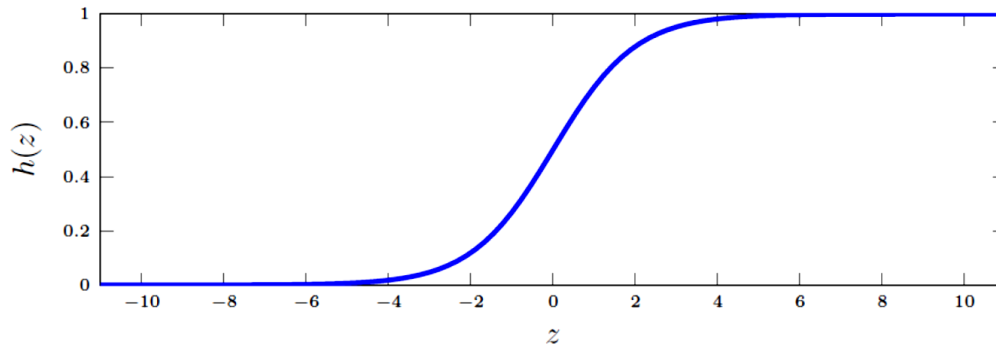
$$P(y = 1 | X) = \frac{1}{1 + e^{\beta^T X}}$$

Note that this implicitly also gives us that,

$$P(y = 0 | X) = 1 - \frac{1}{1 + e^{\beta^T X}} = \frac{e^{\beta^T X}}{1 + e^{\beta^T X}}$$



We now have a model for  $P(y = 1 | X)$  and  $P(y = 0 | X)$ , which contains unknown parameters  $\beta$  that can be learned from training data. That is, we have constructed a binary classifier, which is called as logistic regression.



**Figure 3.1:** The logistic function  $h(z) = \frac{e^z}{1+e^z}$ .

- **Decision Boundary:**

Decision boundary helps to differentiate probabilities into positive class and negative class.

Consider a linear regression given by  $h(x) = y = \beta^T x$

The hypothesis for logistic regression is then given by,  $h(x) = g(\beta^T x) = \frac{1}{1 + e^{-\beta^T x}}$

Where  $g(t) = \frac{1}{1 + e^{-t}}$  and is called sigmoid function or logistic function.

In logistic regression, the value of  $h(x)$  or  $y$  is interpreted as the probability the input  $x$  belongs to class  $y = 1$ . i.e.

$$h(x) = P(y = 1 | x)$$

The fundamental properties of probability hold here, i.e.,

$$P(y = 1 | x) + P(y = 0 | x) = 1$$

If 0.5 is chosen as the threshold for the binary classification, i.e.

$$\hat{y} = 1, \quad \text{if } h(x) \geq 0.5$$

$$\hat{y} = 0, \quad \text{if } h(x) < 0.5$$

From the plot of sigmoid function, it is seen that

$$g(t) \geq 0.5, \quad \text{if } t \geq 0$$

$$g(t) < 0.5, \quad \text{if } t < 0$$

Using above two statements we can write,

$$\hat{y} = 1, \quad \text{if } \boldsymbol{\beta}^T \mathbf{x} \geq 0$$

$$\hat{y} = 0, \quad \text{if } \boldsymbol{\beta}^T \mathbf{x} < 0$$

Suppose the training data is as show in the plot above where dots and Xs are the two different classes. Let the  $h(x)$  and the optimal value of  $\boldsymbol{\beta}$  be given by,

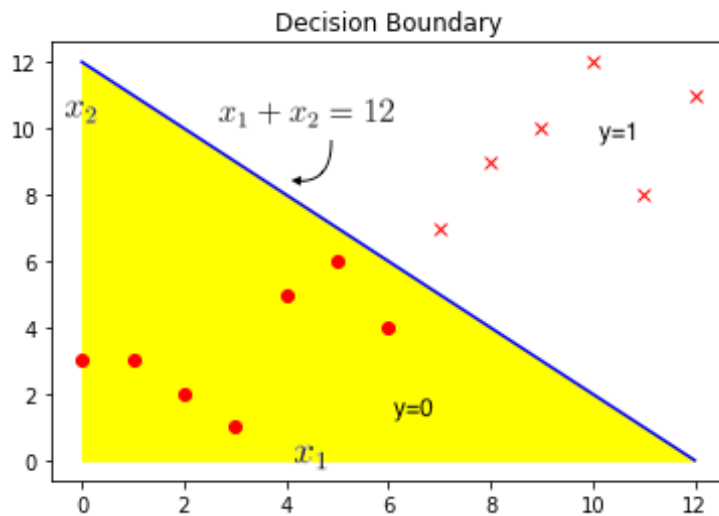
$$h(x) = g(\beta_0 + \beta_1 x_1 + \beta_2 x_2)$$

$$\boldsymbol{\beta} = \begin{bmatrix} -12 \\ 1 \\ 1 \end{bmatrix}$$

Using these values,

$$\hat{y} = 1, \quad \text{if } -12 + x_1 + x_2 \geq 0 \text{ or } x_1 + x_2 \geq 12$$

$$\hat{y} = 0, \quad \text{if } -12 + x_1 + x_2 < 0 \text{ or } x_1 + x_2 < 12$$



If the line  $x_1 + x_2 = 12$  is plotted as shown in the plot above then the region below i.e. the yellow region is where  $x_1 + x_2 < 12$  and predicted 0 and similarly the white region above the line is where  $x_1 + x_2 \geq 12$  and hence predicted 1.

The line here is called the **decision boundary**. As the name suggests this line separates the region with prediction 0 from region with prediction 1.

- **Deciding Threshold Value:**

1. *Low Precision/High Recall:* In applications where we want to reduce the number of false negatives without necessarily reducing the number false positives, we choose a decision value which has a low value of Precision or high value of Recall. For example, in a cancer diagnosis application, we do not want any affected patient to be classified as not affected without giving much heed to if the patient is being wrongfully diagnosed with cancer. This is because, the absence of cancer can be detected by further medical diseases but the presence of the disease cannot be detected in an already rejected candidate.

2. *High Precision/Low Recall:* In applications where we want to reduce the number of false positives without necessarily reducing the number false negatives, we choose a decision value which has a high value of Precision or low value of Recall. For example, if we are classifying customers whether they will react positively or negatively to a personalized advertisement, we want to be absolutely sure that the customer will react positively to the advertisement because otherwise, a negative reaction can cause a loss potential sales from the customer.

---