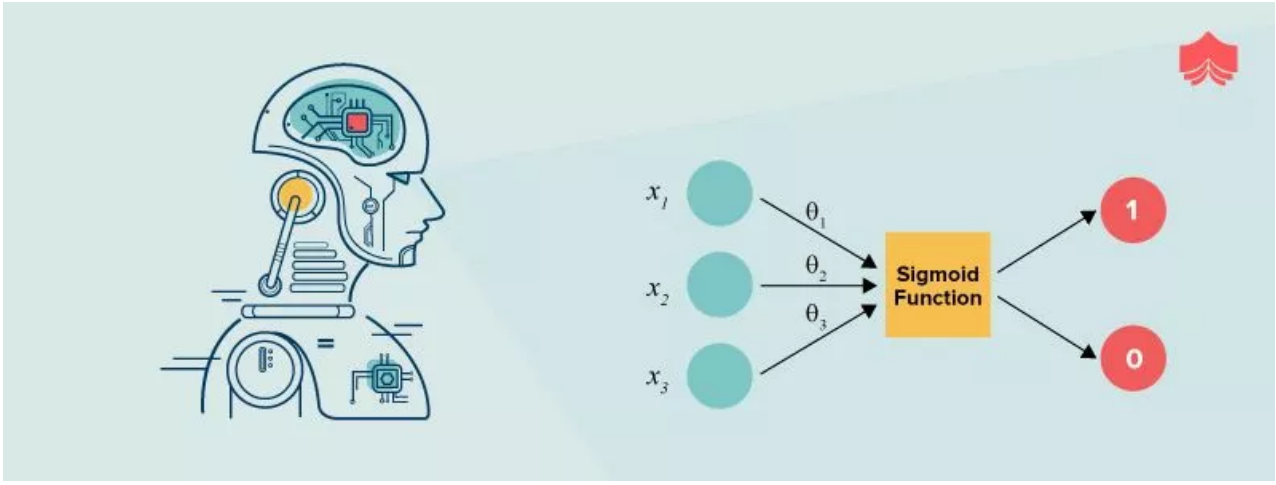
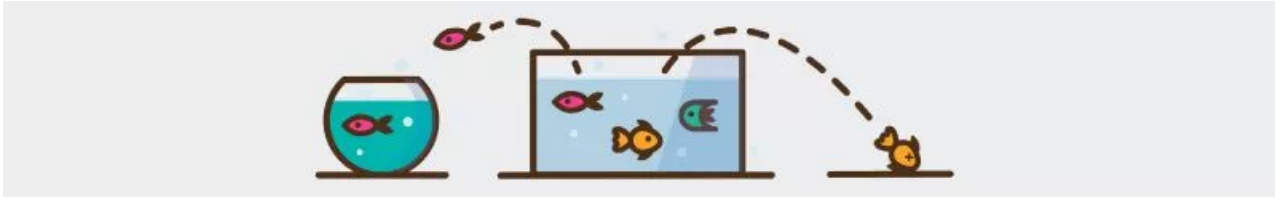
What is Logistic Regression?

knowledgehut.com/blog/data-science/logistic-regression-for-machine-learning



Every machine learning algorithm performs best under a given set of conditions. To ensure good performance, we must know which algorithm to use depending on the problem at hand. You cannot just use one particular algorithm for all problems. For example: Linear regression algorithm cannot be applied on a categorical dependent variable. This is where Logistic Regression comes in.



Logistic Regression is a popular statistical model used for binary classification, that is for predictions of the type *this or that*, *yes or no*, *A or B*, etc. Logistic regression can, however, be used for multiclass classification, but here we will focus on its simplest application. It is one of the most frequently used machine learning algorithms for binary classifications that translates the input to 0 or 1. For example,

- 0: negative class
- 1: positive class

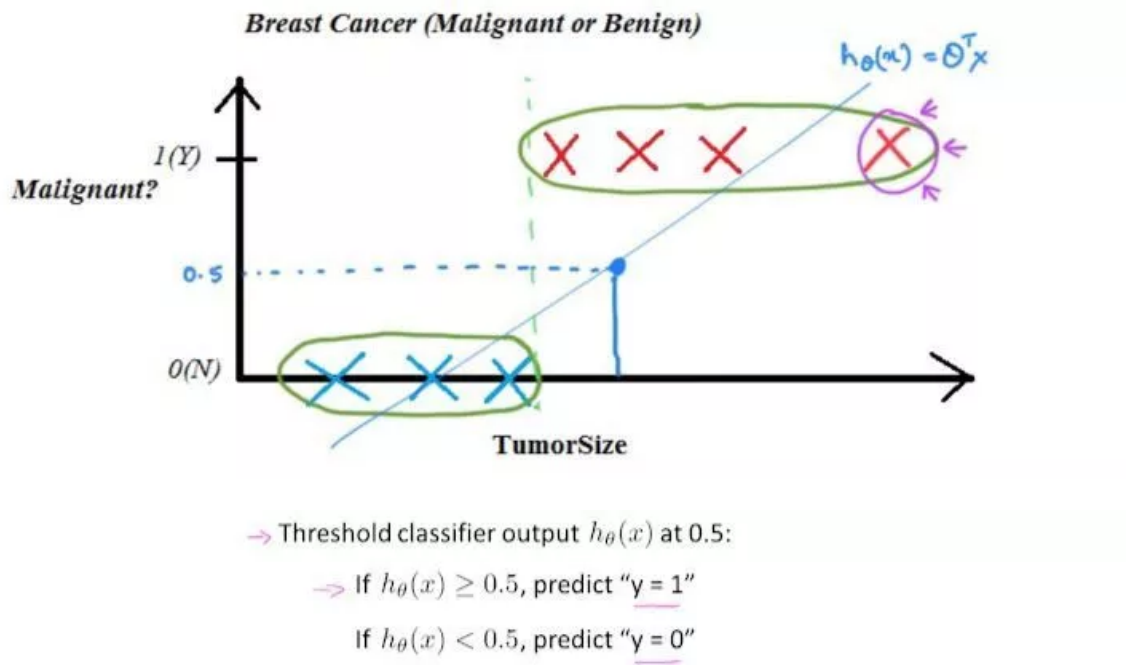
Some examples of classification are mentioned below:

- Email: spam / not spam
- Online transactions: fraudulent / not fraudulent
- Tumor: malignant / not malignant

Let us look at the issues we encounter in Linear Regression.

Issue 1 of Linear Regression

As you can see on the graph mentioned below, the prediction would leave out malignant tumors as the gradient becomes less steep with an additional data point on the extreme right.



Issue 2 of Linear Regression

- Hypothesis can be larger than 1 or smaller than zero
- Hence, we have to use logistic regression

Logistic Regression is the appropriate regression analysis to conduct when the dependent variable has a binary solution. Similar to all other types of regression systems, Logistic Regression is also a type of predictive regression system. Logistic regression is used to evaluate the relationship between one dependent binary variable and one or more independent variables. It gives discrete outputs ranging between 0 and 1.

A simple example of Logistic Regression is: Does calorie intake, weather, and age have any influence on the risk of having a heart attack? The question can have a discrete answer, either "yes" or "no".

Logistic Regression Hypothesis

The logistic regression classifier can be derived by analogy to the ***linear regression hypothesis*** *which is*:

$$h_\theta(\mathbf{x}) = \theta^\top \mathbf{x}$$

Linear regression hypothesis

However, the logistic regression hypothesis *generalizes* from the linear regression hypothesis in that it uses the ***logistic function***:

$$h_\theta(\mathbf{x}) = g(\theta^\top \mathbf{x})$$
$$g(z) = \frac{1}{1+e^{-z}}$$

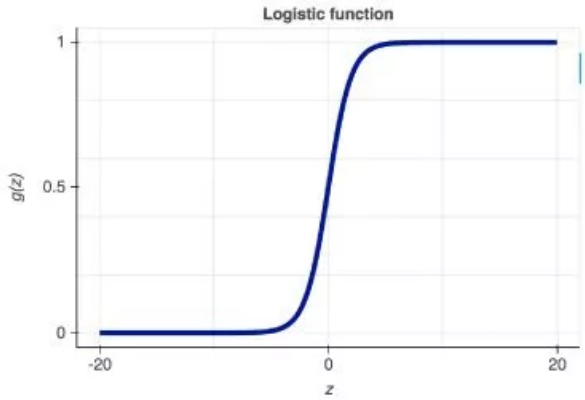
The result is the logistic regression hypothesis:

$$h_\theta(\mathbf{x}) = \frac{1}{1+e^{-\theta^\top \mathbf{x}}}$$

Logistic regression hypothesis

The function **g(z)** is the **logistic function**, also known as the *sigmoid function*.

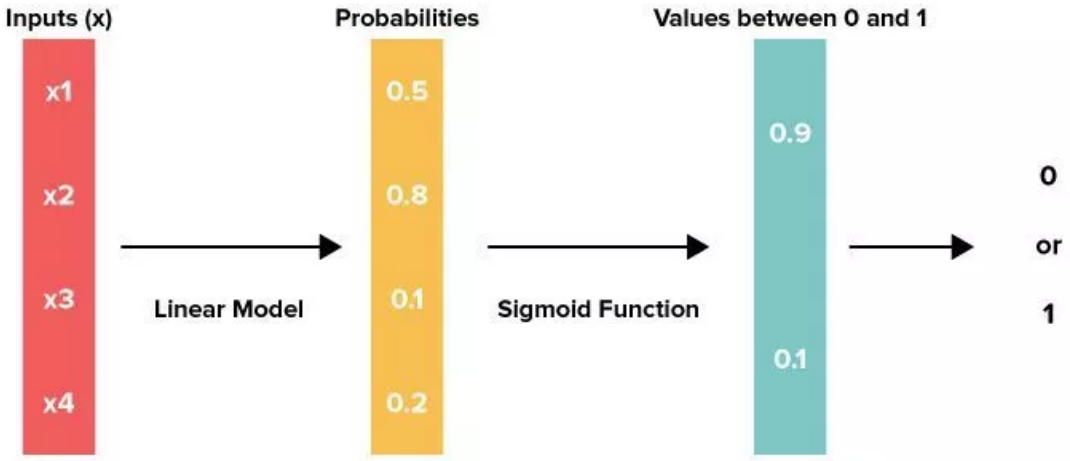
The logistic function has asymptotes at 0 and 1, and it crosses the y-axis at 0.5.



How Logistic Regression works?

Logistic Regression uses a more complex cost function than Linear Regression, this cost function is called the **'Sigmoid function'** or also known as the 'logistic function' instead of a linear function.

The hypothesis of logistic regression tends to limit the cost function between 0 and 1. Therefore linear functions fail to represent it as it can have a value greater than 1 or less than 0 which is not possible as per the hypothesis of logistic regression.



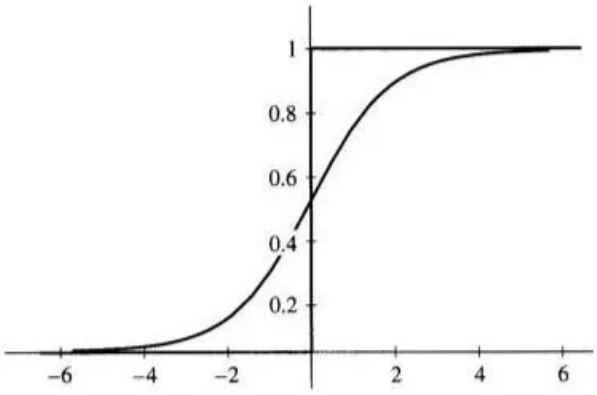
Sigmoid function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.

Formula:

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

Where,

f(x) = output between 0 and 1 (probability estimate)
x = input to the function
e = base of natural log

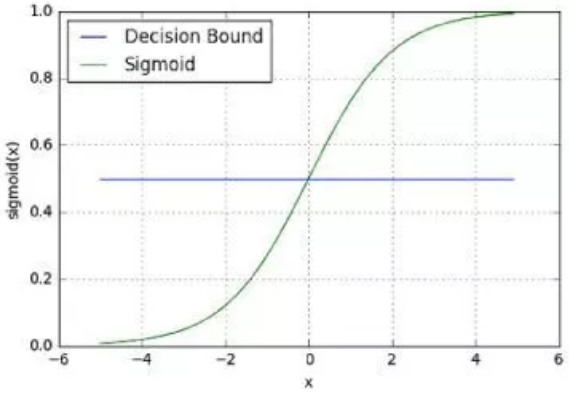


Decision Boundary

The prediction function returns a probability score between 0 and 1. If you want to map the discrete class (true/false, yes/no), you will have to select a threshold value above which you will be classifying values into class 1 and below the threshold value into class 2.

$p \geq 0.5, class = 1$
$p < 0.5, class = 0$

For example, suppose the threshold value is 0.5 and your prediction function returns 0.7, it will be classified as positive. If your predicted value is 0.2, which is less than the threshold value, it will be classified as negative. For logistic regression with multiple classes we could select the class with the highest predicted probability.



Our aim should be to maximize the likelihood that a random data point gets classified correctly, which is called Maximum Likelihood Estimation. Maximum Likelihood Estimation is a general approach to estimating parameters in statistical models. The likelihood can be maximized using an optimization algorithm. Newton's Method is one such algorithm which can be used to find maximum (or minimum) of many different functions, including the likelihood function. Other than Newton's Method, you can also use Gradient Descent.

Cost Function

We have covered Cost Function earlier in the blog on Linear Regression. In brief, a cost function is created for optimization purpose so that we can minimize it and create a model with minimum error.

Cost function for Logistic Regression are:

- $\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \quad \text{if } y = 1$
- $\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \quad \text{if } y = 0$

The above functions can be written together as:

$$J(\theta) = -\frac{1}{m}\sum\left[y^{(i)}\log(h\theta(x(i))) + \left(1 - y^{(i)}\right)\log(1 - h\theta(x(i)))\right]$$

Gradient Descent

After finding out the cost function for Logistic Regression, our job should be to minimize it i.e. min J(θ). The cost function can be reduced by using Gradient Descent.

The general form of gradient descent:

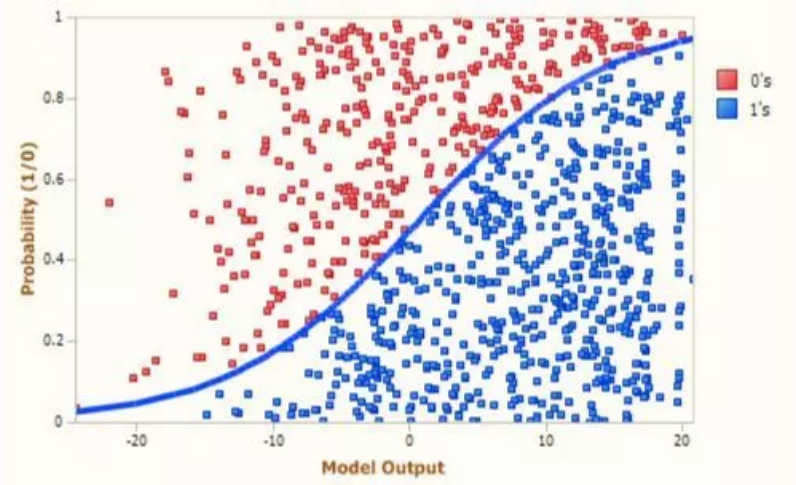
$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$$

The derivative part can be solved using calculus so the equation comes to:

$$\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

When to use Logistic Regression?

Logistic Regression is used when the input needs to be separated into "two regions" by a linear boundary. The data points are separated using a linear line as shown:



Based on the number of categories, Logistic regression can be classified as:

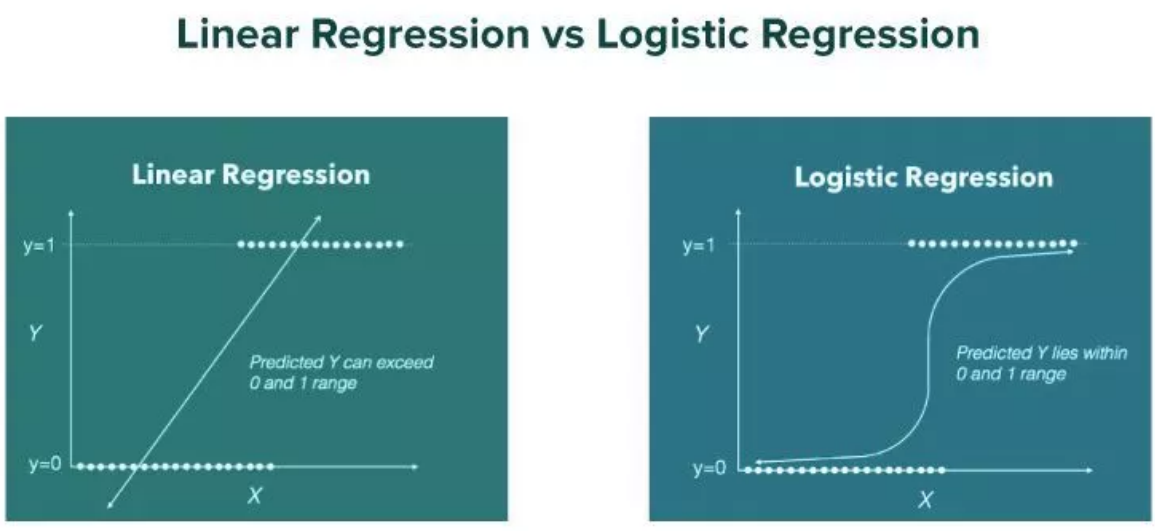
1. **binomial:** target variable can have only 2 possible types: “0” or “1” which may represent “win” vs “loss”, “pass” vs “fail”, “dead” vs “alive”, etc.
2. **multinomial:** target variable can have 3 or more possible types which are not ordered(i.e. types have no quantitative significance) like “disease A” vs “disease B” vs “disease C”.
3. **ordinal:** it deals with target variables with ordered categories. For example, a test score can be categorized as:“very poor”, “poor”, “good”, “very good”. Here, each category can be given a score like 0, 1, 2, 3.

Let us explore the simplest form of Logistic Regression, i.e Binomial Logistic Regression. It can be used while solving a classification problem, i.e. when the y-variable takes on only two values. Such a variable is said to be a “binary” or “dichotomous” variable. “Dichotomous” basically means two categories such as yes/no, defective/non-defective, success/failure, and so on. “Binary” refers to the 0's and 1’s.

Linear vs Logistic Regression

	Linear Regression	Logistic Regression
Outcome	In linear regression, the outcome (dependent variable) is continuous. It can have any one of an infinite number of possible values.	In logistic regression, the outcome (dependent variable) has only a limited number of possible values.
The dependent variable	Linear regression is used when your response variable is continuous. For instance, weight, height, number of hours, etc.	Logistic regression is used when the response variable is categorical in nature. For instance, yes/no, true/false, red/green/blue, 1st/2nd/3rd/4th, etc.

	Linear Regression	Logistic Regression
The independent variable	In Linear Regression, the independent variables can be correlated with each other.	In logistic Regression, the independent variables should not be correlated with each other. (no multi-collinearity)
Equation	Linear regression gives an equation which is of the form Y = mX + C, means equation with degree 1.	Logistic regression gives an equation which is of the form Y = eX + e-X.
Coefficient interpretation	In linear regression, the coefficient interpretation of independent variables are quite straightforward (i.e. holding all other variables constant, with a unit increase in this variable, the dependent variable is expected to increase/decrease by xxx).	In logistic regression, depends on the family (binomial, Poisson, etc.) and link (log, logit, inverse-log, etc.) you use, the interpretation is different.
Error minimization technique	Linear regression uses ordinary least squares method to minimise the errors and arrive at a best possible fit, while logistic regression uses maximum likelihood method to arrive at the solution.	Logistic regression is just the opposite. Using the logistic loss function causes large errors to be penalized to an asymptotic constant.



How is OLS different from MLE?

Linear regression is estimated using Ordinary Least Squares (OLS) while logistic regression is estimated using Maximum Likelihood Estimation (MLE) approach.

Ordinary Least Squares (OLS) also called the linear least squares is a method to approximately determine the unknown parameters of a linear regression model. Ordinary least squares is obtained by minimizing the total squared vertical distances between the observed responses within the dataset and the responses predicted by the linear approximation(represented by the line of best fit or regression line). The resulting estimator can be represented using a simple formula.

For example, let's say you have a set of equations which consist of several equations with unknown parameters. The ordinary least squares method may be used because this is the most standard approach in finding the approximate solution to your overly determined systems. In other words, it is your overall solution in minimizing the sum of the squares of errors in your equation. Data that best fits the ordinary least squares minimizes the sum of squared residuals. Residual is the difference between an observed value and the predicted value provided by a model.

Maximum likelihood estimation, or MLE, is a method used in estimating the parameters of a statistical model, and for fitting a statistical model to data. If you want to find the height measurement of every basketball player in a specific location, maximum likelihood estimation can be used. If you could not afford to measure all of the basketball players' heights, the maximum likelihood estimation can come in very handy. Using the maximum likelihood estimation, you can estimate the mean and variance of the height of your subjects. The MLE would set the mean and variance as parameters in determining the specific parametric values in a given model.

To sum it up, the maximum likelihood estimation covers a set of parameters which can be used for predicting the data needed in a normal distribution. A given, fixed set of data and its probability model would likely produce the predicted data. The MLE would give us a unified approach when it comes to the estimation. But in some cases, we cannot use the maximum likelihood estimation because of recognized errors or the problem actually doesn't even exist in reality.

Building Logistic Regression Model

To build a logistic regression model we can use statsmodel and the inbuilt logistic regression function present in the sklearn library.

```
# Importing Packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

# Reading German Credit Data
raw_data = pd.read_csv("/content/German_Credit_data.csv")
raw_data.head()
```

Building Logistic Regression Base Model after data preparation:

```
import statsmodels.api as sm
#Build Logit Model
logit = sm.Logit(y_train,x_train)

# fit the model
model1 = logit.fit()

# Printing Logistic Regression model results
model1.summary2()

Optimization terminated successfully.
Current function value: 0.480402
Iterations 6

Model:                  Logit                            Pseudo R-squared:  0.197

Dependent Variable:     Creditability                    AIC:
712.5629
Date:                   2019-09-19 09:55                 BIC:
803.5845
No. Observations:       700                              Log-Likelihood:   -336.28
Df Model:               19                               LL-Null:          -418.79
Df Residuals:           680                              LLR p-value:
2.6772e-25
Converged:              1.0000                           Scale:             1.0000
No. Iterations:         6.0000
```

We will calculate the model accuracy on the test dataset using 'score' function.

```
# Checking the accuracy with test data
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,predicted_df['Predicted_Class']))

0.74
```

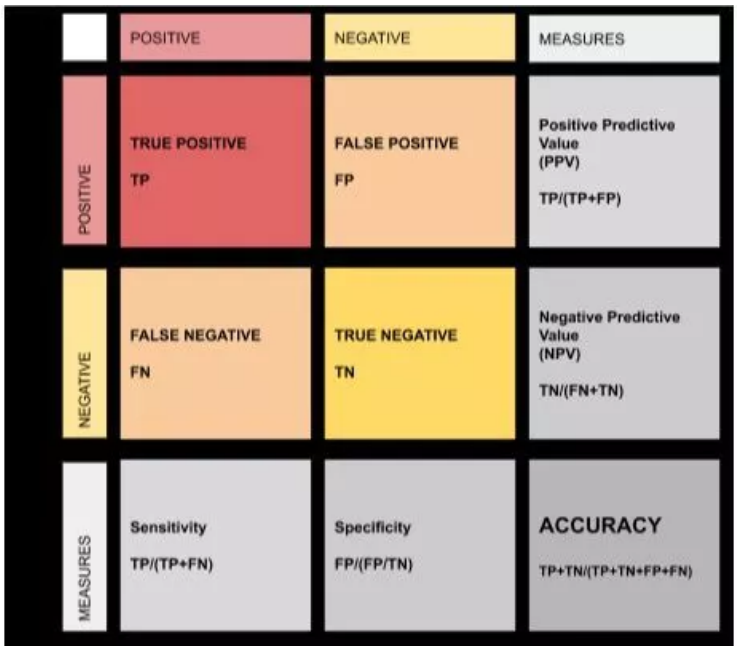
We can see the accuracy of 74%.

Model Evaluation

Model evaluation metrics are used to find out the goodness of the fit between model and data, to compare the different models, in the context of model selection, and to predict how predictions are expected to be accurate.

What is a Confusion Matrix?

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions.



Confusion Matrix gives insight not only into the errors being made by your classifier but more importantly the types of errors that are being made. It is this breakdown that overcomes the limitation of using classification accuracy alone.

How to Calculate a Confusion Matrix

Below is the process for calculating a confusion Matrix:

1. You need a test dataset or a validation dataset with expected outcome values.
2. Make a prediction for each row in your test dataset.
3. From the expected outcomes and predictions count:

- The number of correct predictions for each class.
- The number of incorrect predictions for each class, organized by the class that was predicted.

These numbers are then organized into a table or a matrix as follows:

- **Expected down the side:** Each row of the matrix corresponds to a predicted class.
- **Predicted across the top:** Each column of the matrix corresponds to an actual class.

The counts of correct and incorrect classification are then filled into the table.
The total number of correct predictions for a class goes into the expected row for that class value and the predicted column for that class value.

In the same way, the total number of incorrect predictions for a class goes into the expected row for that class value and the predicted column for that class value.

2-Class Confusion Matrix Case Study

Let us consider we have a two-class classification problem of predicting whether a photograph contains a man or a woman. We have a test dataset of 10 records with expected outcomes and a set of predictions from our classification algorithm.

Expected	Predicted
Man	Woman
Man	Man
Woman	Woman
Man	Man
Woman	Man
Woman	Woman
Woman	Woman
Man	Man
Man	Woman
Woman	Woman

Let's start off and calculate the classification accuracy for this set of predictions.

Suppose the algorithm made 7 of the 10 predictions correct with an accuracy of 70%, then:

```
accuracy = total correct predictions / total predictions made * 100
accuracy = 7/10*100
```

But what are the types of errors made?
We can determine that by turning our results into a ***confusion matrix***:
First, we must calculate the number of correct predictions for each class.

- men classified as men: 3

- women classified as women: 4

Now, we can calculate the number of incorrect predictions for each class, organized by the predicted value:

- men classified as women: 2
- woman classified as men: 1

We can now arrange these values into the 2-class confusion matrix:

	men	women
men	3	1
women	2	4

From the above table we learn that:

- The total actual men in the dataset is the sum of the values on the men column.
- The total actual women in the dataset is the sum of values in the women's column.
- The correct values are organized in a diagonal line from top left to bottom-right of the matrix.
- More errors were made by predicting men as women than predicting women as men.

Two-Class Problems Are Special

In a two-class problem, we are often looking to discriminate between observations with a specific outcome, from normal observations. Such as a disease state or event from no-disease state or no-event. In this way, we can assign the event row as “positive” and the no-event row as “negative“. We can then assign the event column of predictions as “true” and the no-event as “false“.

This gives us:

- “true positive” for correctly predicted event values.
- “false positive” for incorrectly predicted event values.
- “true negative” for correctly predicted no-event values.
- “false negative” for incorrectly predicted no-event values.

We can summarize this in the confusion matrix as follows:

	event	no-event
men	3	1
women	2	4

This can help in calculating more advanced classification metrics such as precision, recall, specificity and sensitivity of our classifier.

```
Sensitivity/ recall= 7/ (7+5)= 0.583
Specificity= 3/ (3+5)= 0.375
Precision= 7/ (7+3)= 0.7
```

The code mentioned below shows the implementation of confusion matrix in Python with respect to the example used earlier:

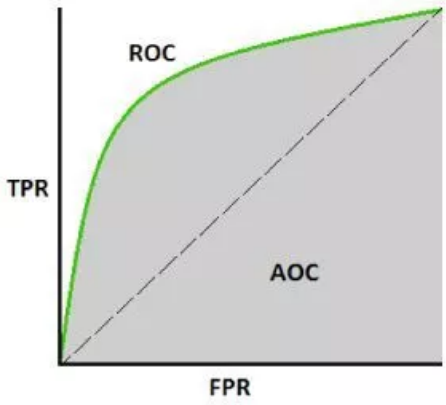
```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test,
predicted_df['Predicted_Class']).ravel()
confusion_matrix

array([ 37,  63,  15, 185])
```

The results from the confusion matrix are telling us that 37 and 185 are the number of correct predictions. 63 and 15 are the number of incorrect predictions.

Receiver Operating Characteristic (ROC)

The **receiver operating characteristic** (ROC), or the ROC curve, is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity or the sensitivity index d', known as "d-prime" in signal detection and biomedical informatics, or recall in machine learning. The false-positive rate is also known as the fall-out and can be calculated as (1 - specificity). The ROC curve is thus the sensitivity as a function of fall-out.



There are a number of methods of evaluating whether a logistic model is a good model. One such way is sensitivity and specificity. Sensitivity and specificity are statistical measures of the performance of a binary classification test, also known in statistics as classification function:

Sensitivity / Recall (also known as the true positive rate, or the recall) measures the proportion of actual positives which are correctly identified as such (e.g., the percentage of sick people who are correctly identified as having the condition), and is complementary to the false negative rate. It shows how good a test is at detecting the positives. A test can cheat and maximize this by always returning “positive”.

```
Sensitivity= true positives/ (true positive + false negative)
```

Specificity (also called the true negative rate) measures the proportion of negatives which are correctly identified as such (e.g., the percentage of healthy people who are correctly identified as not having the condition), and is complementary to the false positive rate. It shows how good a test is at avoiding false alarms. A test can cheat and maximize this by always returning “negative”.

```
Specificity= true negatives/ (true negative + false positives)
```

Precision is used as a measure to calculate the success of predicted values to the values which were supposed to be successful. Precision is used with recall, the percent of all relevant documents that is returned by the search. The two measures are sometimes used together in the F1 Score (or f-measure) to provide a single measurement for a system. It shows how many of the positively classified were relevant. A test can cheat and maximize this by only returning positive on one result it’s most confident in.

```
Precision= true positives/ (true positive + true negative)
```

The precision-recall curve shows the trade-off between precision and recall for different threshold. The decision for the value of the threshold value is majorly affected by the values of precision and recall. Ideally, we want both precision and recall to be 1, but this seldom is the case. In case of a Precision-Recall tradeoff we use the following arguments to decide upon the threshold:-

1. **Low Precision/High Recall:** In applications where we want to reduce the number of false negatives without necessarily reducing the number of 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 of 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 personalised 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 of potential sales from the customer.

The code mentioned below shows the implementation in Python with respect to the example used earlier:

```
from sklearn.metrics import classification_report

print(classification_report(y_test, predicted_df['Predicted_Class']))
```

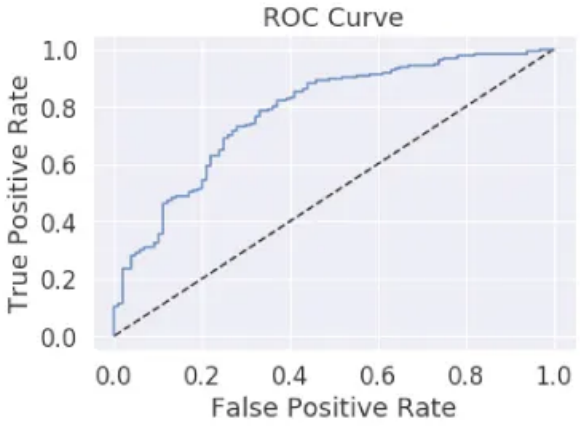
	precision	recall	f1-score	support
0	0.71	0.37	0.49	100
1	0.75	0.93	0.83	200
accuracy			0.74	300
macro avg	0.73	0.65	0.66	300
weighted avg	0.73	0.74	0.71	300

The f1-score tells you the accuracy of the classifier in classifying the data points in that particular class compared to all other classes. It is calculated by taking the harmonic mean of precision and recall. The support is the number of samples of the true response that lies in that class.

```
y_pred_prob = model1.predict(x_test)

from sklearn.metrics import roc_curve
# Generate ROC curve values: fpr, tpr, thresholds
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

# Plot ROC curve
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```




```
# AUCfrom sklearn.metrics import roc_auc_score
roc_auc_score(y_test,predicted_df['Predicted_Class'])

0.6475
```

Area Under the Curve is 0.6475

Hosmer Lemeshow Goodness-of-Fit

- It measures the association between actual events and predicted probability.
- How well our model fits depends on the difference between the model and the observed data. One approach for binary data is to implement a Hosmer Lemeshow goodness of fit test

- In HL test, the null hypothesis states, the model fits the data well. Model appears to fit well if we have no significant difference between the model and the observed data (i.e. the p-value > 0.05, so not rejecting the Ho)
- Or in other words, if the test is NOT statistically significant, that indicates the model is a good fit.
- As with all measures of model fit, use this as just one piece of information in deciding how well this model fits. It doesn't work well in very large or very small data sets, but is often useful nonetheless.

$$G^2_{HL} = \sum_{j=1}^{n} \{[(O_j-E_j)^2]/[E_j(1-E_j/n_j)]\} \sim X_s^2$$

- X^2 = chi squared.
- n_j = number of observations in the group.
- O_j = number of observed cases in the j th group.
- O_j = number of expected cases in the j th group.

Gini Coefficient

- The Gini coefficient is sometimes used in classification problems.
- Gini coefficient can be straight away derived from the AUC ROC number. Gini is nothing but the ratio between area between the ROC curve and the diagonal line & the area of the above triangle. Following is the formulae used :

```
Gini=2*AUC–1
```

Gini above 60% is a good model.

Akaike Information Criterion and Bayesian Information Criterion

AIC and BIC values are like adjusted R-squared values in linear regression.

- `AIC= -2ln(SSE)+ 2k`
- `BIC = n*ln(SSE/n) + k*ln(n)`

Pros and Cons of Logistic Regression

Many of the pros and cons of the linear regression model also apply to the logistic regression model. Although Logistic regression is used widely by many people for solving various types of problems, it fails to hold up its performance due to its various limitations and also other predictive models provide better predictive results.

Pros

- The logistic regression model not only acts as a classification model, but also gives you probabilities. This is a big advantage over other models where they can only provide the final classification. Knowing that an instance has a 99% probability for a class compared to 51% makes a big difference. Logistic Regression performs well when the dataset is linearly separable.
- Logistic Regression not only gives a measure of how relevant a predictor (coefficient size) is, but also its direction of association (positive or negative). We see that Logistic regression is easier to implement, interpret and very efficient to train.

Cons

- Logistic regression can suffer from complete separation. If there is a feature that would perfectly separate the two classes, the logistic regression model can no longer be trained. This is because the weight for that feature would not converge, because the optimal weight would be infinite. This is really a bit unfortunate, because such a feature is really very useful. But you do not need machine learning if you have a simple rule that separates both classes. The problem of complete separation can be solved by introducing penalization of the weights or defining a prior probability distribution of weights.
- Logistic regression is less prone to overfitting but it can overfit in high dimensional datasets and in that case, regularization techniques should be considered to avoid over-fitting in such scenarios.

In this article we have seen what Logistic Regression is, how it works, when we should use it, comparison of Logistic and Linear Regression, the difference between the approach and usage of two estimation techniques: Maximum Likelihood Estimation and Ordinary Least Square Method, evaluation of model using Confusion Matrix and the advantages and disadvantages of Logistic Regression. We have also covered some basics of sigmoid function, cost function and gradient descent.

If you are inspired by the opportunities provided by machine learning, enrol in our Data Science and Machine Learning Courses for more lucrative career options in this landscape.