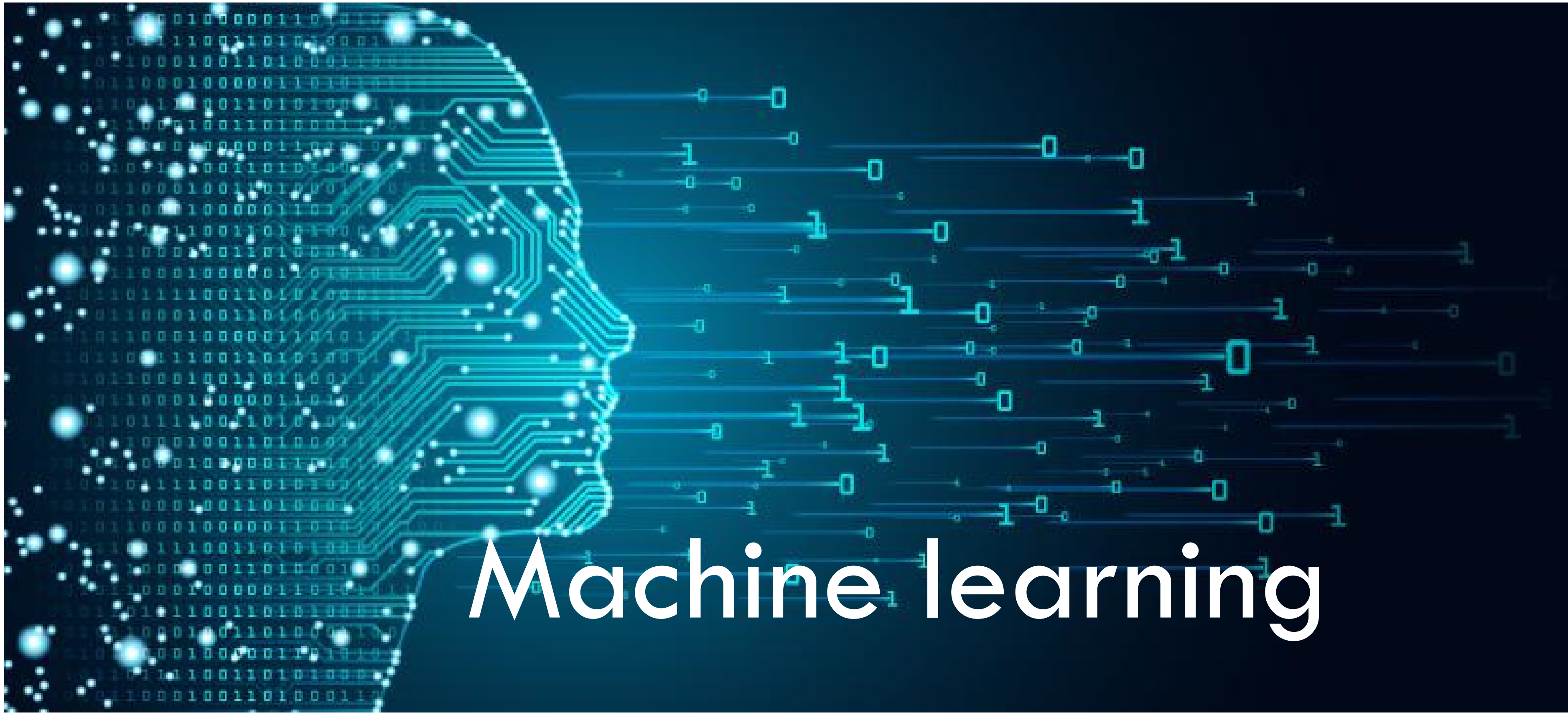
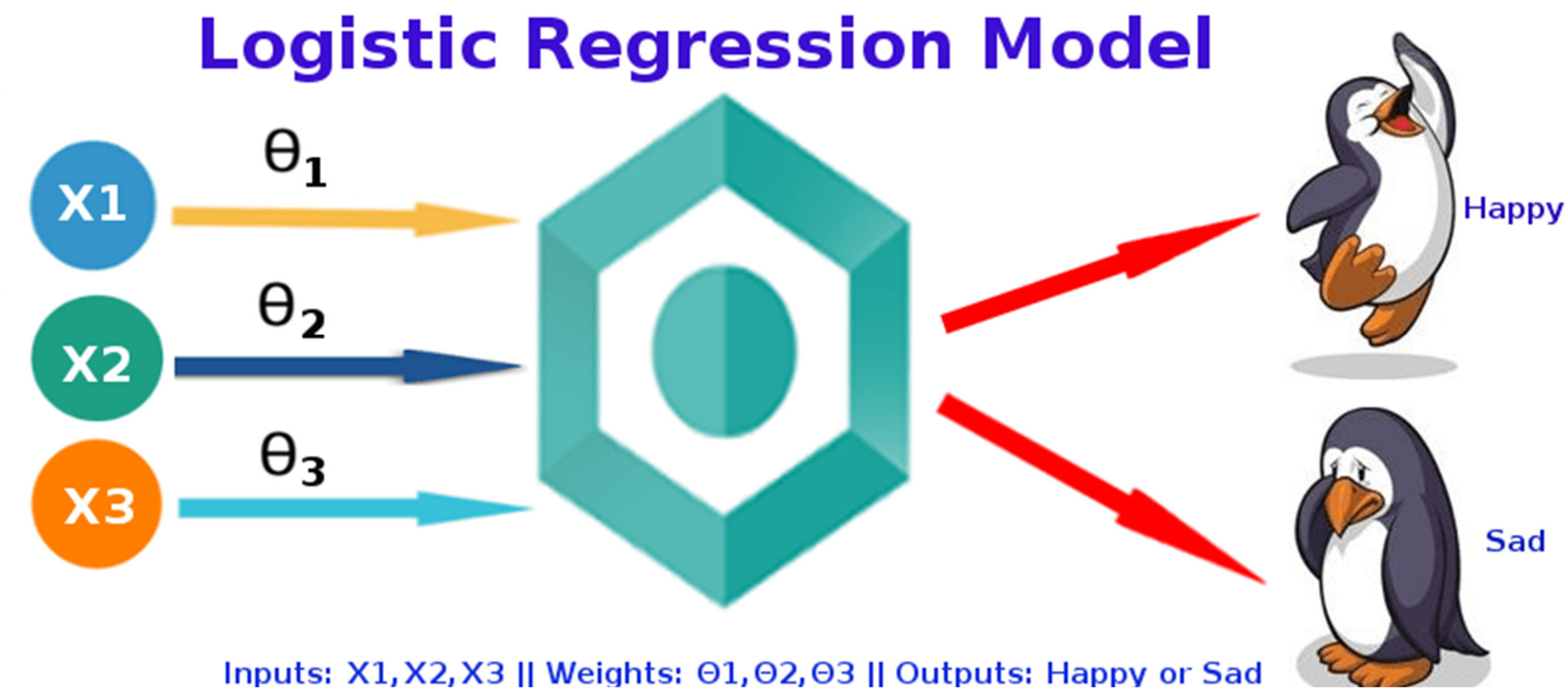


Machine learning

Algorithms - Logistic Regression

Pr. BEN LAHMAR EL Habib

Introduction

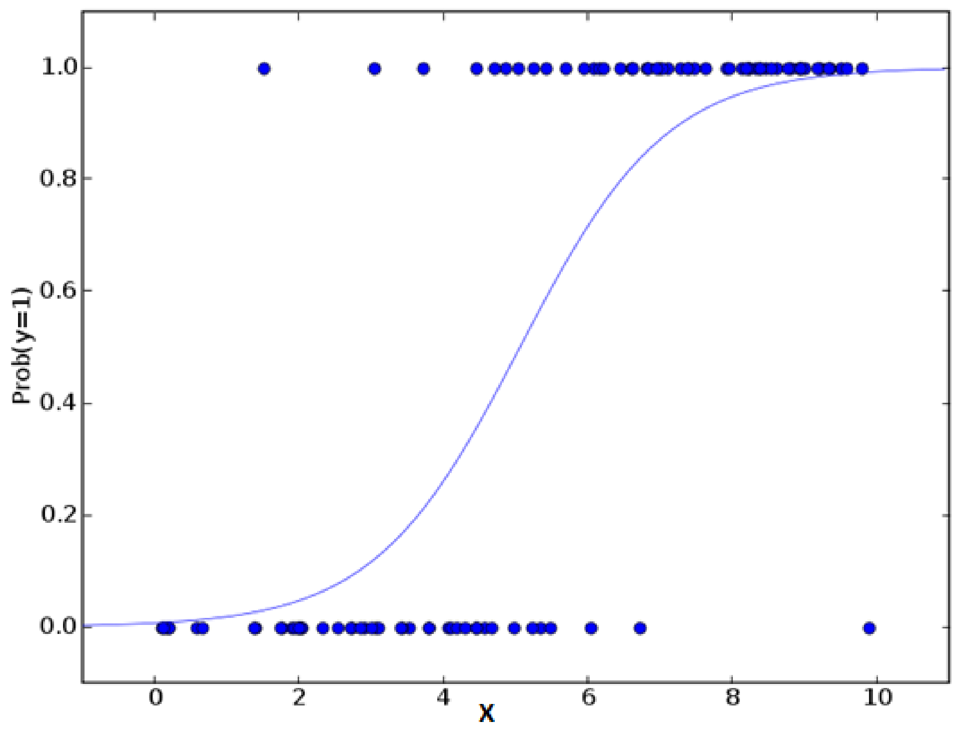


logistic regression

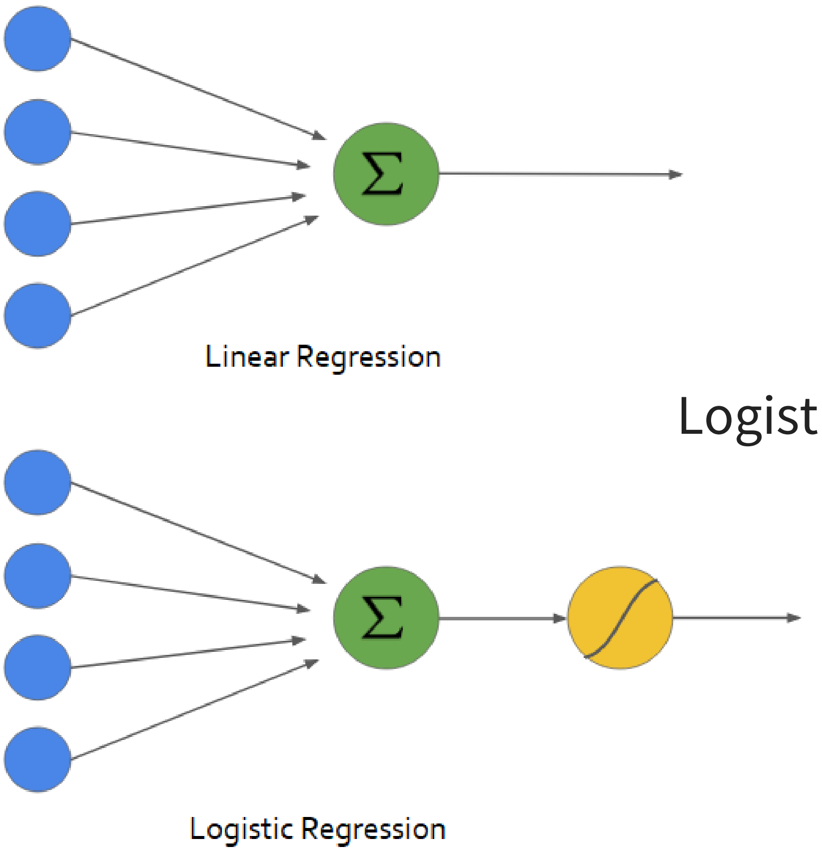
- In a lot of ways, linear regression and logistic regression are similar. But, the biggest difference lies in what they are used for.
- Linear regression algorithms are used to predict/forecast values but logistic regression is used for classification tasks.

Logistic Regression

- It is a classification not a regression algorithm.
- It is used to estimate discrete values (Binary values like 0/1, yes/no, true/false) based on given set of independent variable(s).



LiR vs LoR



Logistic regression is a linear classifier

Logistic Regression

- Logistic Regression is used when the dependent variable(target) is categorical.
- For example,
  - To predict whether an email is spam (1) or (0)
  - Whether the tumor is malignant (1) or not (0)
  - whether a website is fraudulent (1) or not (0)

Logistic regression

- Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity.

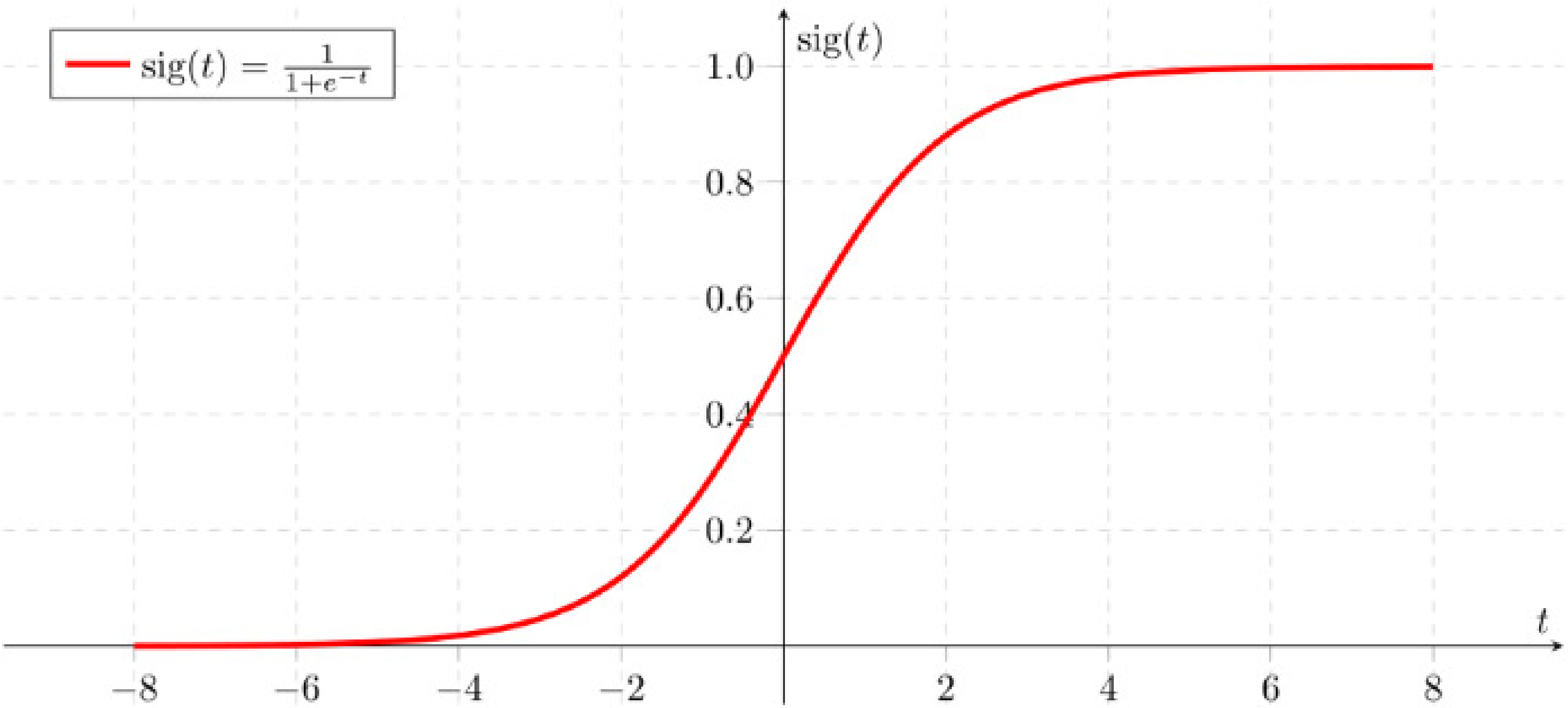
Logistic regression

- The predicted value can be anywhere between negative infinity to positive infinity. We need the output of the algorithm to be class variable, i.e 0-no, 1-yes.
- Therefore, we are squashing the output of the linear equation into a range of [0,1]. To squash the predicted value between 0 and 1, we use the sigmoid function.

Model

- Output = 0 or 1
- Hypothesis => $Z = \Theta X + B$
- $h_{\Theta}(x)$ = sigmoid (Z)
  - If ‘Z’ goes to infinity, Y(predicted) will become 1 and if ‘Z’ goes to negative infinity, Y(predicted) will become 0.

Sigmoid Activation Function



Logistic regression is a linear classifier

Sigmoid function

$$z = \theta_0 + \theta_1 \cdot x_1 + \theta \cdot x_2 + \cdots \qquad g(x) = \frac{1}{1+e^{-x}}$$

Linear Equation and Sigmoid Function

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

Squashed output-h

Mathematically this can be written as

$h_{\Theta}(x) = P\ (Y=1|X;\ theta)$

Probability that Y=1 given X which is parameterized by ‘theta’.

$P\ (Y=1|X;\ theta) + P\ (Y=0|X;\ theta) = 1$

$P\ (Y=0|X;\ theta) = 1 - P\ (Y=1|X;\ theta)$

Types of Logistic Regression

- 1. Binary Logistic Regression
  - The categorical response has only two 2 possible outcomes. Example: Spam or Not
- 2. Multinomial Logistic Regression
  - Three or more categories without ordering. Example: Predicting which food is preferred more (Veg, Non-Veg, Vegan)
- 3. Ordinal Logistic Regression
  - Three or more categories with ordering. Example: Movie rating from 1 to 5

Cost Function

- Since we are trying to predict class values, we cannot use the same cost function used in linear regression algorithm. Therefore, we use a logarithmic loss function to calculate the cost for misclassifying.

Cost Function

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

- The above cost function can be rewritten as below since calculating gradients from the above equation is difficult.

$$-\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Gradients

$$J = \frac{-1}{m} \cdot [\sum_{i=1}^{m} y_i \cdot \log h_i + (1 - y_i) \cdot \log 1 - h_i]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{-1}{m} \cdot [\sum_{i=1}^{m} \frac{y_i}{h_i} \cdot h_i^2 \cdot x_n \cdot \frac{1 - h_i}{h_i} + \frac{1 - y_i}{1 - h_i} \cdot -h_i^2 \cdot x_n \cdot \frac{1 - h_i}{h_i}]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{-1}{m} \cdot [\sum_{i=1}^{m} x_n \cdot (1 - h_i) \cdot y_i - x_n \cdot h_i \cdot (1 - y_i)]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{1}{m} \cdot x_i \cdot [\sum_{i=1}^{m} h_i - y_i]$$

Classification Performance

- Binary classification has four possible types of results:
  - **True negatives:** correctly predicted negatives (zeros)
  - **True positives:** correctly predicted positives (ones)
  - **False negatives:** incorrectly predicted negatives (zeros)
  - **False positives:** incorrectly predicted positives (ones)

Indicators of binary classifiers

- The most straightforward indicator of **classification accuracy** is the ratio of the number of correct predictions to the total number of predictions (or observations). Other indicators of binary classifiers include the following:
  - **The positive predictive value** is the ratio of the number of true positives to the sum of the numbers of true and false positives.
  - **The negative predictive value** is the ratio of the number of true negatives to the sum of the numbers of true and false negatives.
  - **The sensitivity** (also known as recall or true positive rate) is the ratio of the number of true positives to the number of actual positives.
  - **The specificity** (or true negative rate) is the ratio of the number of true negatives to the number of actual negatives.

Ref :Wikipedia

Positive predictive value [edit]

The positive predictive value (PPV) is defined as

$$\mathrm{PPV} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false positives}} = \frac{\text{number of true positives}}{\text{number of positive calls}}$$

where a "true positive" is the event that the test makes a positive prediction, and the subject has a positive result under the gold standard, and a "false positive" is the event that the test makes a positive prediction, and the subject has a negative result under the gold standard. The ideal value of the PPV, with a perfect test, is 1 (100%), and the worst possible value would be zero.

In case-control studies the PPV has to be computed from sensitivity, specificity, but also including the prevalence:

$$\mathrm{PPV} = \frac{\text{sensitivity} \times \text{prevalence}}{\text{sensitivity} \times \text{prevalence} + (1 - \text{specificity}) \times (1 - \text{prevalence})}$$

The complement of the PPV is the false discovery rate (FDR):

$$\mathrm{FDR} = 1 - \mathrm{PPV} = \frac{\text{number of false positives}}{\text{number of true positives} + \text{number of false positives}} = \frac{\text{number of false positives}}{\text{number of positive calls}}$$

Negative predictive value [edit]

The negative predictive value is defined as:

$$\mathrm{NPV} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false negatives}} = \frac{\text{number of true negatives}}{\text{number of negative calls}}$$

where a "true negative" is the event that the test makes a negative prediction, and the subject has a negative result under the gold standard, and a "false negative" is the event that the test makes a negative prediction, and the subject has a positive result under the gold standard. The ideal value of the NPV, with a perfect test, is 1 (100%), and the worst possible value would be zero.

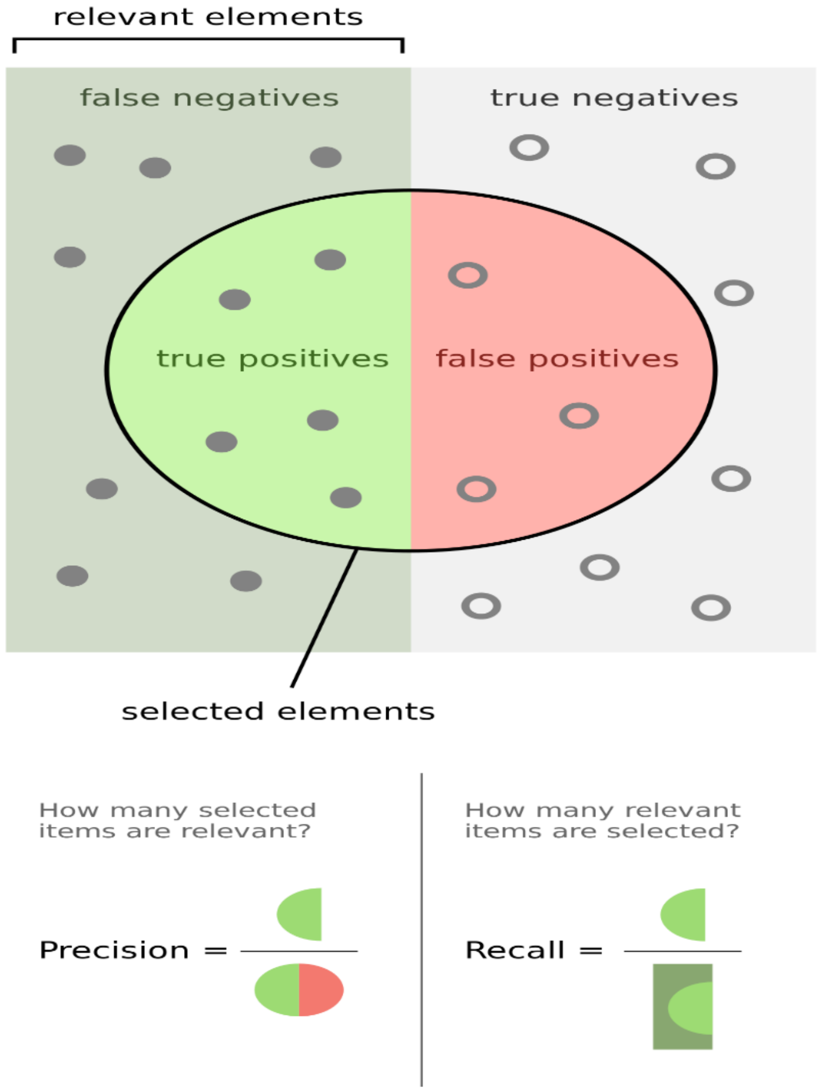
The NPV can also be computed from sensitivity, specificity, and prevalence:

$$\mathrm{NPV} = \frac{\text{specificity} \times (1 - \text{prevalence})}{(1 - \text{sensitivity}) \times \text{prevalence} + \text{specificity} \times (1 - \text{prevalence})}$$

The complement of the NPV is the **false omission rate** (FOR):

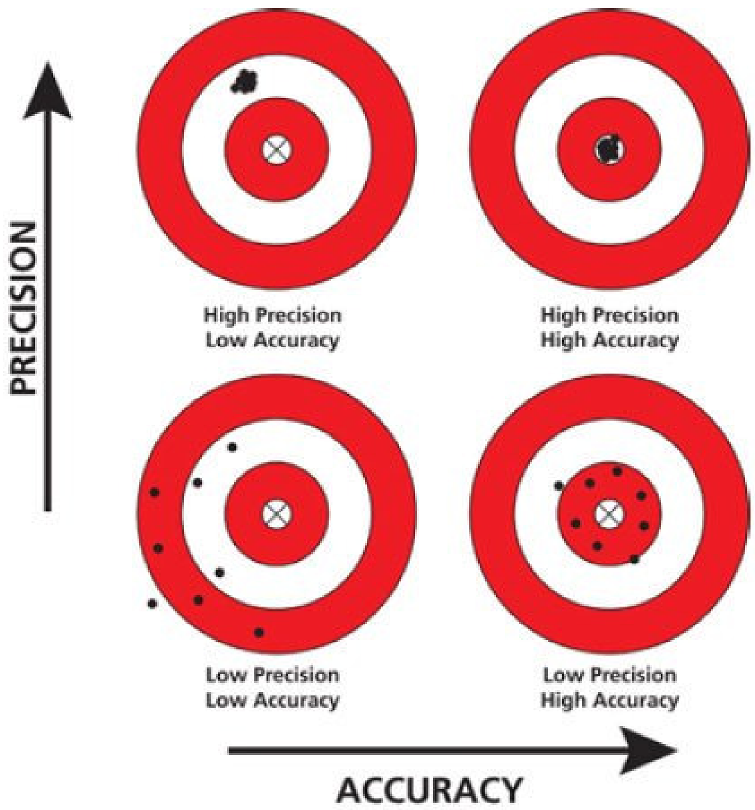
$$\mathrm{FOR} = 1 - \mathrm{NPV} = \frac{\text{number of false negatives}}{\text{number of true negatives} + \text{number of false negatives}} = \frac{\text{number of false negatives}}{\text{number of negative calls}}$$

https://en.wikipedia.org/wiki/Positive_and_negative_predictive_values#Positive_predictive_value



$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\textbf{\textit{F - measure}} = \frac{2*Recall*Precision}{Recall + Precision}$$



Implementation

Python code

```
#Import Library
from sklearn.linear_model import LogisticRegression
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_d
ataset
# Create logistic regression object
model = LogisticRegression()
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Equation coefficient and Intercept
print('Coefficient: \n', model.coef_)
print('Intercept: \n', model.intercept_)
#Predict Output
predicted= model.predict(x_test)
```

Implementation

R Code

```
x <- cbind(x_train,y_train)
# Train the model using the training sets and check score
logistic <- glm(y_train ~ ., data = x,family='binomial')
summary(logistic)
#Predict Output
predicted= predict(logistic,x_test)
```

Tutos

- Go to github.com/benlahmar