**What is a Classification Problem?**

In the exciting world of data science, a **classification problem** is like being a detective who tries to sort things into different categories. Imagine you have a basket of fruits, and your task is to classify each fruit as either an apple or a banana based on its characteristics. This is similar to what a classification problem is all about.

* **Definition**: A classification problem is a type of problem in data science where we aim to categorize or classify data points into predefined classes or categories. These classes could be things like "yes" or "no," "spam" or "not spam," "cat," or "dog." It's all about making decisions and assigning labels to data based on certain characteristics or features.
* **Examples**: Here are a few real-world examples to help you understand classification problems better
  1. **Email Spam Detection**: One of the most common classification problems is spam email detection. The goal is to classify incoming emails as either "spam" or "not spam" so that your inbox remains clutter-free.
  2. **Medical Diagnosis**: In the field of healthcare, doctors use classification algorithms to diagnose diseases. For instance, a system might classify medical images as "normal" or "cancerous."
  3. **Credit Card Fraud Detection**: Banks use classification to detect fraudulent credit card transactions. The system decides whether a transaction is "legitimate" or "fraudulent" based on patterns and behavior.
* **How it Works**: To tackle a classification problem, data scientists use machine learning algorithms that learn from historical data. These algorithms analyze the features of each data point and try to find patterns that distinguish one class from another. Once the algorithm has learned these patterns, it can make predictions for new, unseen data.

To understand algorithms such as Logistic Regression, we need to learn what a sigmoid function does.

**The Sigmoid Function**

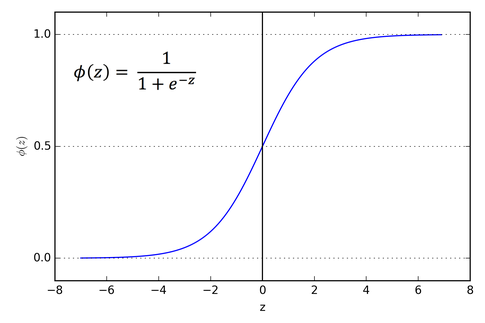
A sigmoid function is a mathematical function that maps any real-valued number to a value between 0 and 1. It is characterized by its S-shaped curve, which resembles the letter "S". Sigmoid functions are widely used in various fields, including mathematics, statistics, machine learning, and artificial neural networks.

One of the most common sigmoid functions is the logistic sigmoid function, often denoted as σ(x) or sigmoid(x).

The **logistic sigmoid** **function** is defined as follows:

***σ*(*x*)=1/(1+*e*-*x*​)**

* "x" is the input value, which can be any real number.
* "e" is the base of the natural logarithm (approximately 2.71828).

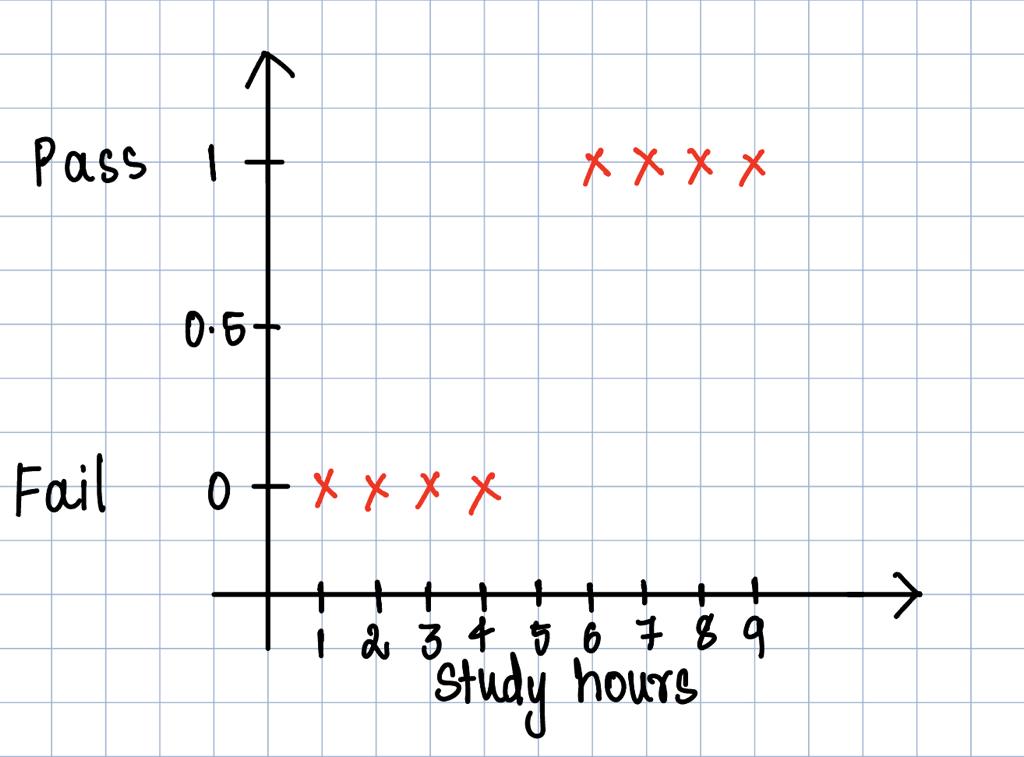
A graph of a function

Description automatically generated

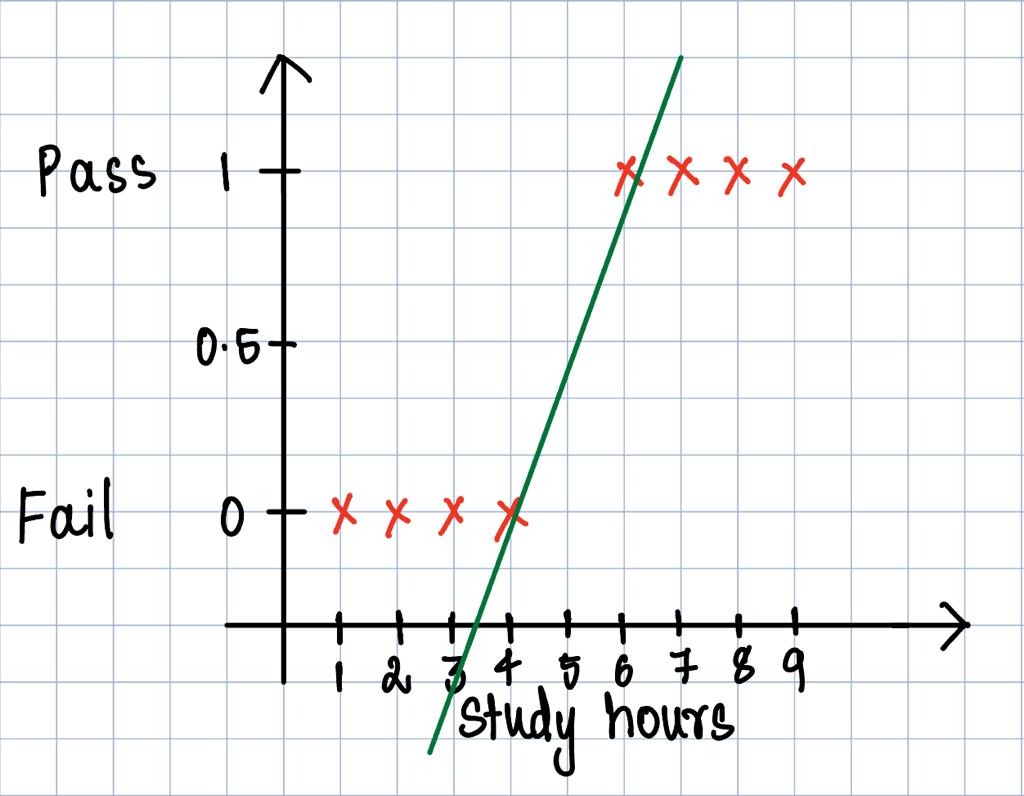
* The logistic sigmoid function takes any real number "x" as input and transforms it into a value between 0 and 1.
  + When "x" is very large (positive or negative), the sigmoid function approaches 1 or 0, respectively.
  + When "x" is close to zero, the sigmoid function outputs approximately 0.5.
* The logistic sigmoid function is commonly used in **binary classification** problems in machine learning, where it converts a real-valued input into a probability score between 0 and 1. For example, in logistic regression, the sigmoid function is used to model the probability of an event occurring based on input features.

**Solving a Classification Problem using Linear Regression**

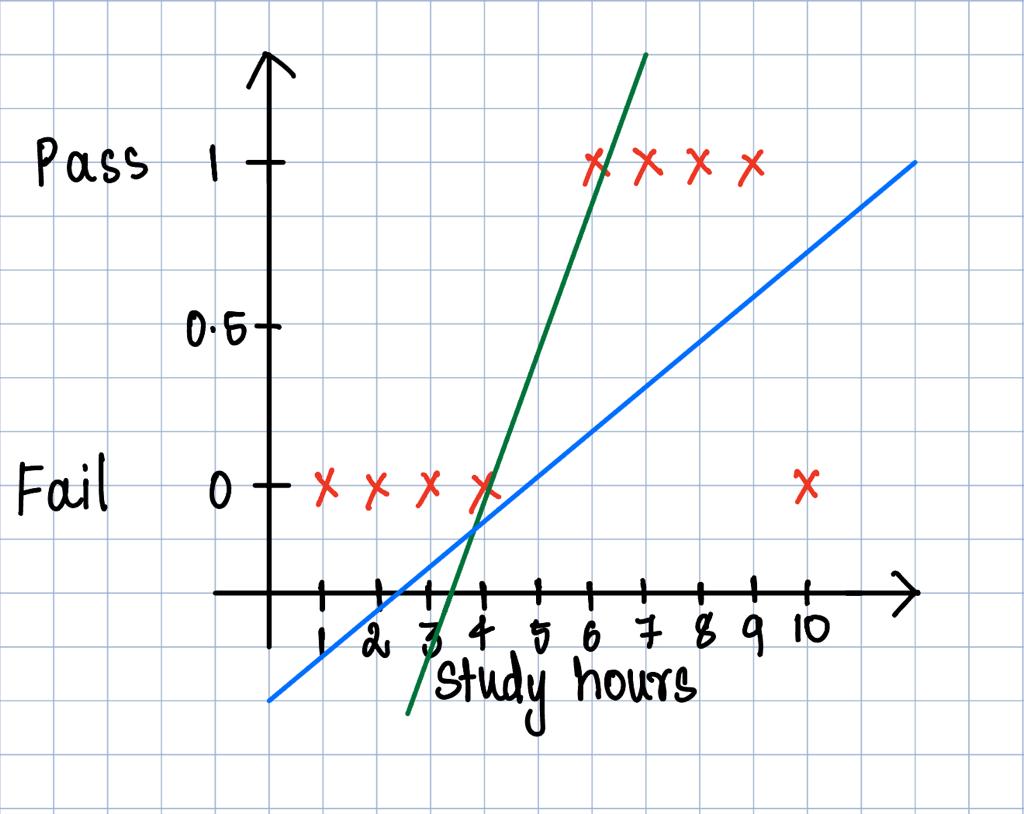
* Suppose there is a problem of predicting whether a particular student will pass or fail, and the given attribute of the student is number of study hours, the graph can be plotted as given below:



* In this graph 1 represents pass and 0 represents fail and all the data points are marked with red crosses.
* What the linear regression algorithm can do is plot a line where it approximately acts as a decision boundary, if h­θ(x) > 0.5, then the student passes and if h­θ(x) < 0.5 then the student fails:



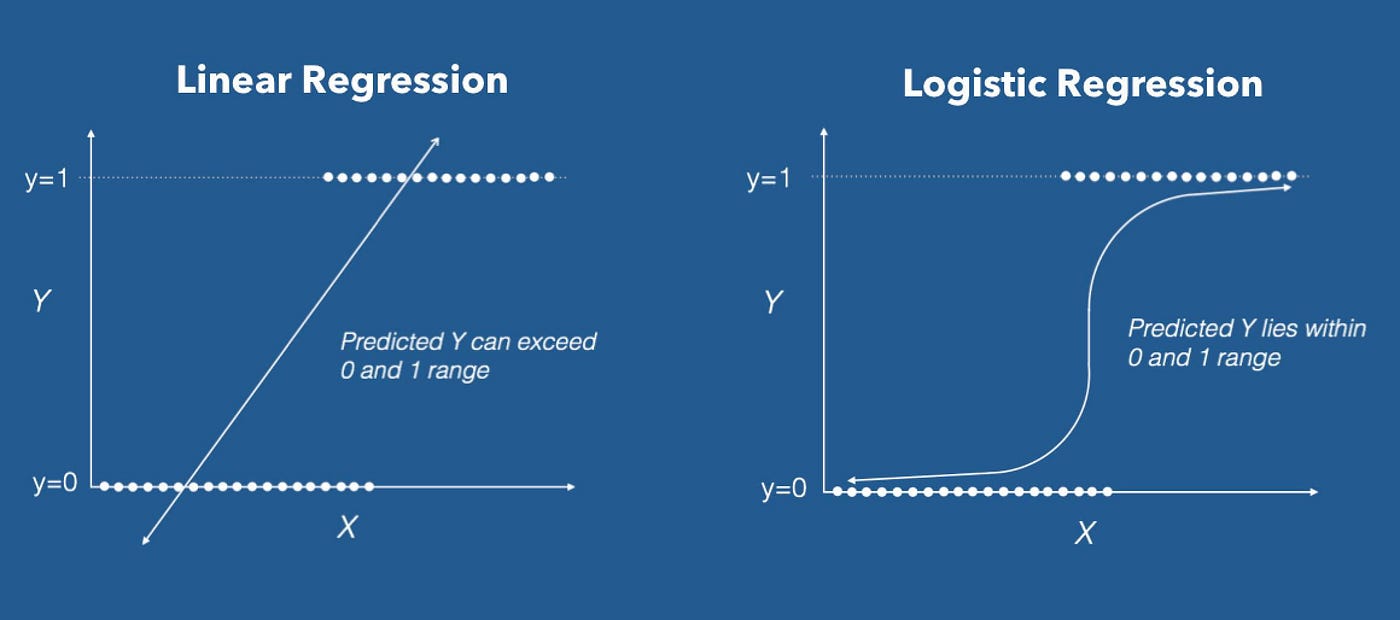
* As we can see that the Linear Regression line quite accurately divides the entire space of data points into two categories, but there arise two problems with this approach, i.e. if there is an existence of an outlier the slope of the line can shift to such an extent that the predictions made can be wrong.
* For an instance let’s assume we have another data point studying 10 hours but somehow still ends up failing, this data point alone will shift the line in a way that the slope will reduce and hence giving out wrong predictions.



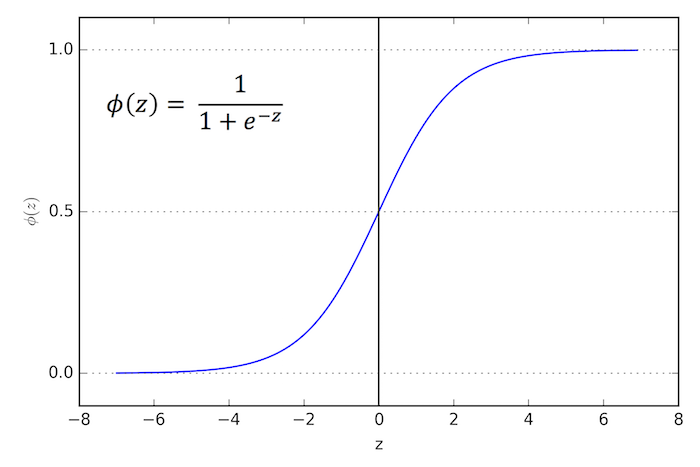
* The green line represents the line without the 10-hour datapoint while the blue line represents the line which takes the 10-hour datapoint into consideration and changes its slope as shown above thereby predicting wrong values.
* Another problem that we encounter is, due to it being a linear regression problem there can be inputs which can lead to predictions which are less that 0 and more that one.

**Logistic Regression**

* To tackle this problem of linear regression in solving a classification task, we use logistic regression where in we use the sigmoid function, which is explained above, this solves the problem by squashing the line between 0 and 1.



* Like in the case of linear regression the hypothesis line is given by **hθ(x) = θ0  + θ1x**, in the case of logistic regression it is given by **hθ(x) = g(θ0  + θ1x)**, where g represents the sigmoid function, it is also written as **hθ(x) = g(z).**



* Here we can see for all values of z (including less than 0 or greater than 1), our hypothesis is still in the range of 0 to 1.

So Logistic Regression can be represented by:

**if h­θ(z) > 0.5 🡪 h­θ(z) = 1**

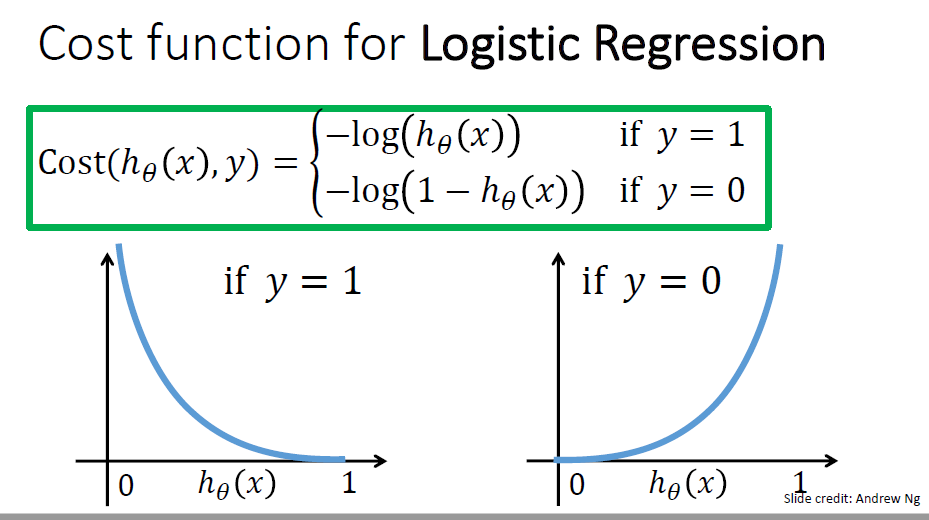
**if h­θ(z) < 0.5 🡪 h­θ(z) = 0**

**where h­θ(z) = 1/(1+e-z­) and z = θ0  + θ1x**

* To get the best fit line we updated the parameters **θ0** and **θ1** in **Linear Regression**,similarly we will do the **same for** **Logistic Regression** but for a different cost function.

**Gradient Descent in Logistic Regression**

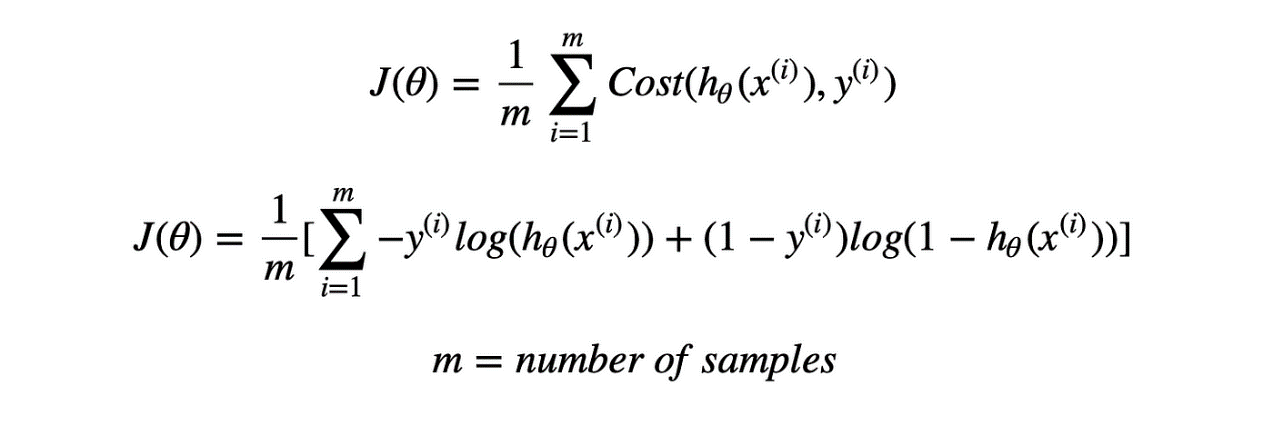
In logistic regression, we aim to find the best set of parameters (weights/**θ1** and bias/**θ0**) that allow us to make accurate predictions. To do this, we define a **cost function often called the "log loss" or "cross-entropy loss"** that measures the difference between our model's predictions and the actual outcomes.



**Explanation for the cost function:**

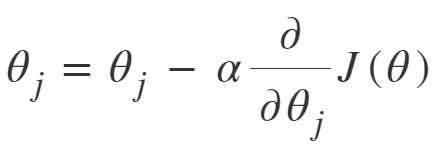
* **Note: h­θ(x) is predicted value, y is actual value. h­θ(x) ranges from 0 to 1 and y is either 0 or 1.**
* **When y = 1**,
  + **h­θ(x)** is 1, the error is almost zero,
  + **h­θ(x)** is 0 (wrong prediction) then the error is very large (extends to infinite).
* **When y = 0**,
  + **h­θ(x)** is 0, the error is almost zero,
  + **h­θ(x)** is 1 then the error is very large (extends to infinite).

The combined **Cost Function** is given by:

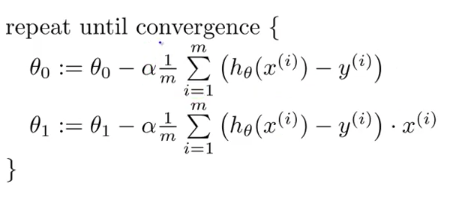


* m is the number of instances, yi represents the actual value of the instance of index i and h­θ(x)i represents the predicted value of the instance of index i.

Now the same **gradient descent algorithm** that was used in the case of Linear Regression is used and is shown below:



On Computing using the differentiation of cost function we get the exact same algorithm as in the case of linear regression’s gradient descent.



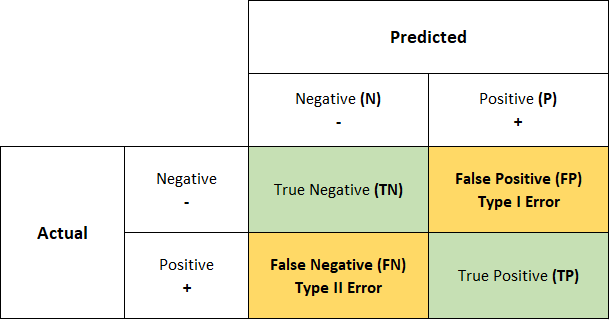
This algorithm upon reaching convergence finds the best pair of parameters.

**Performance Metrics for Logistic Regression**

**Confusion Matrix:** A confusion matrix is a tabular representation used to evaluate the performance of a classification model, such as logistic regression. It provides a more detailed view of how the model's predictions match the actual outcomes.

A confusion matrix typically consists of four components:

* **True Negatives (TN):** The number of instances correctly classified as the negative class.
* **False Positives (FP):** The number of instances incorrectly classified as the positive class (actual negative but predicted positive).
* **True Positives (TP):** The number of instances correctly classified as the positive class.
* **False Negatives (FN):** The number of instances incorrectly classified as the negative class (actual positive but predicted negative).



The confusion matrix is a valuable tool for understanding how well your logistic regression model is performing, especially when you want to analyze the trade-offs between different metrics (e.g., precision vs. recall) and make informed decisions about model performance and threshold selection.

**Types of Metrics, calculated using confusion matrix**

1. **Accuracy:** Accuracy measures the proportion of correctly classified instances out of the total number of instances. It's a straightforward metric for overall model performance.

Formula: Correct Predictions / Total Predictions = (TP + TN) / (TP + TN + FP + FN)

**Why not Accuracy?**

Accuracy is a commonly used metric to assess the overall performance of a classification model. It measures the proportion of correctly classified instances out of the total number of instances. While accuracy is intuitive and easy to understand, it may not always be the best metric, especially when dealing with imbalanced datasets as these datasets are dominated by one class.

**Example: Detecting Rare Diseases**

Imagine you are developing a machine learning model to detect a rare medical condition that affects only 1% of the population. You collect data from many individuals, and you build a classification model to identify whether a person has this rare disease (positive class) or not (negative class). Now imagine we have a total of 1000 instances in this dataset, which means we will only have 10 instances of a positive class.

1. **Scenario 1**, **A Naïve Model:** You train a naïve model that predicts "negative" (no disease) for every individual, regardless of their actual condition. In this case, the model will have high accuracy because it correctly classifies the majority (99%) of individuals as "negative." However, it completely fails to detect the rare disease, and its true positive count (TP) is zero.

(TP + TN) / (TP + TN + FP + FN) = **(0 + 990) / (0 + 990 + 0 + 10) = 990 / 1000 = 0.99**

1. **Scenario 2**, **A More Sophisticated Model**: You train a more sophisticated model that identifies some true positives (correctly identifies individuals with the disease) but also has a few false positives (incorrectly identifies some healthy individuals as having the disease). This model may correctly identify 10% of individuals with the disease while misclassifying 5% of healthy individuals as having the disease.

(TP + TN) / (TP + TN + FP + FN) = **(100 + 830) / (100 + 830 + 50 + 20) = 0.93**

**Accuracy Comparison:**

* In Scenario 1 (naïve model), the accuracy would be 99% because it correctly classifies the majority of individuals. However, it is practically useless for detecting rare diseases because it has a true positive count of zero (disease not identified).
* In Scenario 2 (more sophisticated model), the accuracy might be lower, perhaps around 93%, because it correctly identifies some individuals with the disease and has a few false positives. While it has lower accuracy, it is much more valuable for detecting rare diseases.

**Conclusion:** In the context of the rare disease example, we can see that accuracy can be misleading. The naïve model with high accuracy fails to serve its primary purpose, which is disease detection. The more sophisticated model, despite having a lower accuracy, provides much more meaningful information by identifying some true positives.

Accuracy is not the best metric in situations where:

* Class Imbalance: When one class significantly outweighs the other, as in the rare disease example, accuracy can be skewed because the model tends to predict the majority class.
* Different Costs of Errors: When false positives and false negatives have different consequences. For example, in medical diagnosis, a false negative (missing a disease) can be more severe than a false positive.

**Metrics according to use case**

1. **Precision**:

* Definition: Precision measures the proportion of **true positive** (TP) predictions among **all positive** (TP + FP) **predictions** made by the model.
* Precision = TP/(TP+FP)
* It focuses on minimizing false positives.
* Use Case: Precision is useful when minimizing false positives is crucial, and the cost of false positives is high.
* Problem Solved: Precision helps in scenarios where accuracy may be misleading due to an imbalance between classes. For example, in the **crime department**, false positives are more dangerous than false negatives since an innocent man could be charged with a crime they didn’t do.

1. **Recall** (Sensitivity): Measures the proportion of **true positive** (TP) predictions among **all** **actual positive** (TP + FN) **instances**.

* Recall = TP/(TP+FN)
* It focuses on minimizing false negatives.
* Use Case: Recall is important when minimizing false negatives is critical, and the cost of missing positives is high.
* Problem Solved: Recall is valuable when accuracy alone doesn't tell the whole story. It ensures that the model identifies as many positive instances as possible, even if it results in a few false positives. For instance, in **disease diagnosis**, a high recall ensures that most cases of the disease are correctly identified, while there might be false positives, false negatives can be very dangerous.

1. **F1-Score:**
   * Definition: The F1-Score is the harmonic mean of precision and recall. It provides a balanced measure that takes both false positives and false negatives into account.
   * F1-Score = (2 x Precision x Recall)/ (Precision + Recall)
   * Value can be 1 or 0. 1 implies that the model perfectly classifies predictions correctly and 0 implies that model is unable to classify any observation into the correct class.
   * Use Case: The F1-Score is suitable when you want to balance the trade-offs between precision and recall.
   * Problem Solved: In cases where precision and recall need to be considered simultaneously, the F1-Score provides a single metric that combines them. It ensures that the model performs well in both minimizing false positives and false negatives. For example, in information retrieval systems, you want to provide relevant search results (high recall) while avoiding irrelevant results (high precision), making the F1-Score a useful metric.