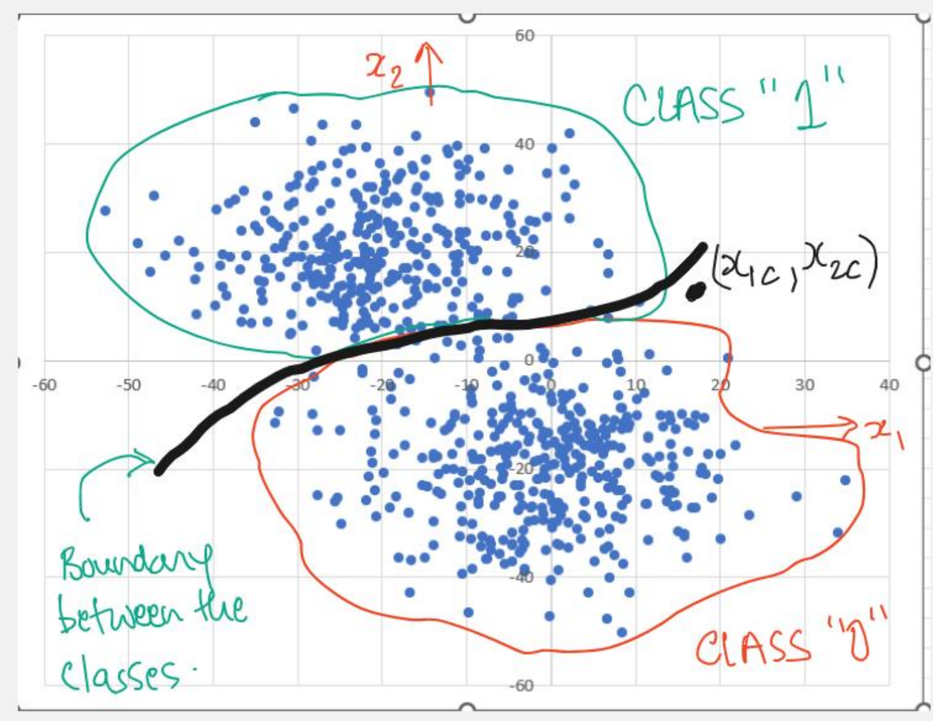
Logistic Regression

The Classification Problem



1.Given the observations' set $\{x_{1i}, x_{2i}\}$

2.And a **classification** or **label** associated with each observation - Class-**0** or Class-**1**

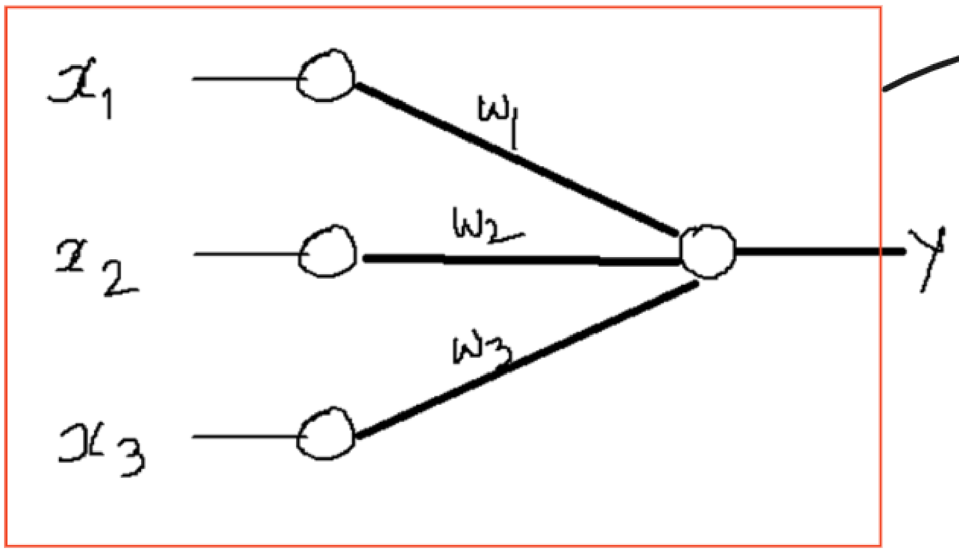
3.Create a model that will **learn** the boundary between the two classes such that, given any candidate observation $\{x_{1c}, x_{2c}\}$, it will be classified into one of these classes with high reliability

Logistic Regression is **one** method to solve this problem

Logistic Unit

- If the combination of inputs
  - $\{x_1, x_2, x_3, \ldots, x_n\}$
- Results in a response that is a “categorical variable”
  - With two possible states: 0 and 1
- Then, we have a unit that is known as the
  - Logistic Unit
- And we need a function that will
  - Trigger 0 or 1 as an output, based on the inputs
  - Such a function is known as an **Activation Function**

Logistic Unit and Logistic Regression



LOGISTIC UNIT

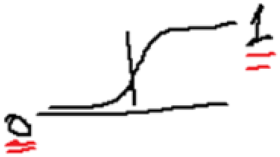
y has two possible states $\begin{cases} 0 \\ 1 \end{cases}$

Let $a = w_1x_1 + w_2x_2 + w_3x_3 + b$

We need a function that will convert 'a' into either 0 or 1

The SIGMOID function $\sigma(a) = \frac{1}{1+e^{-a}}$ has such a property

Its shape is

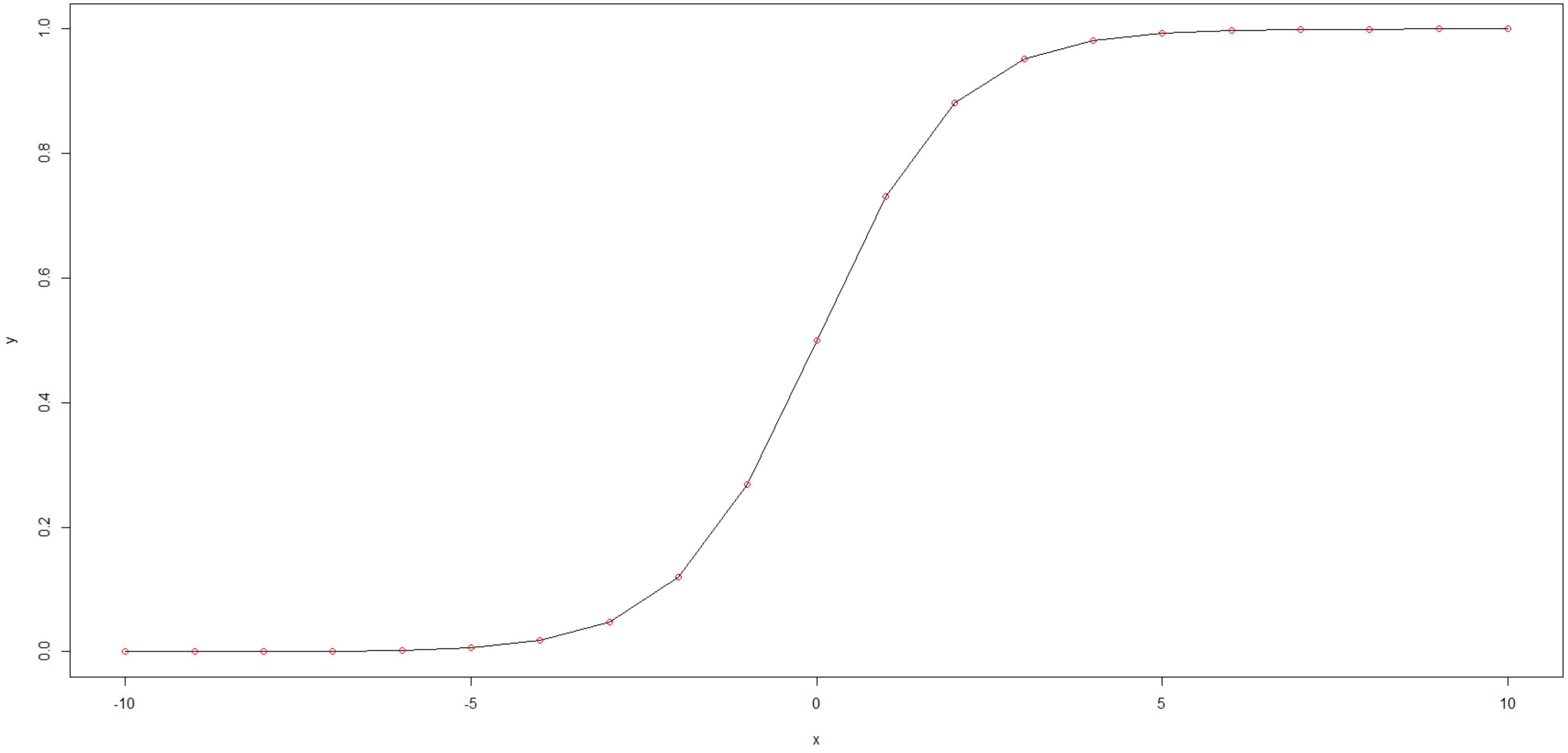


we can express $p(y|x) = \sigma(a) = \frac{1}{1+\frac{1}{e^a}}$

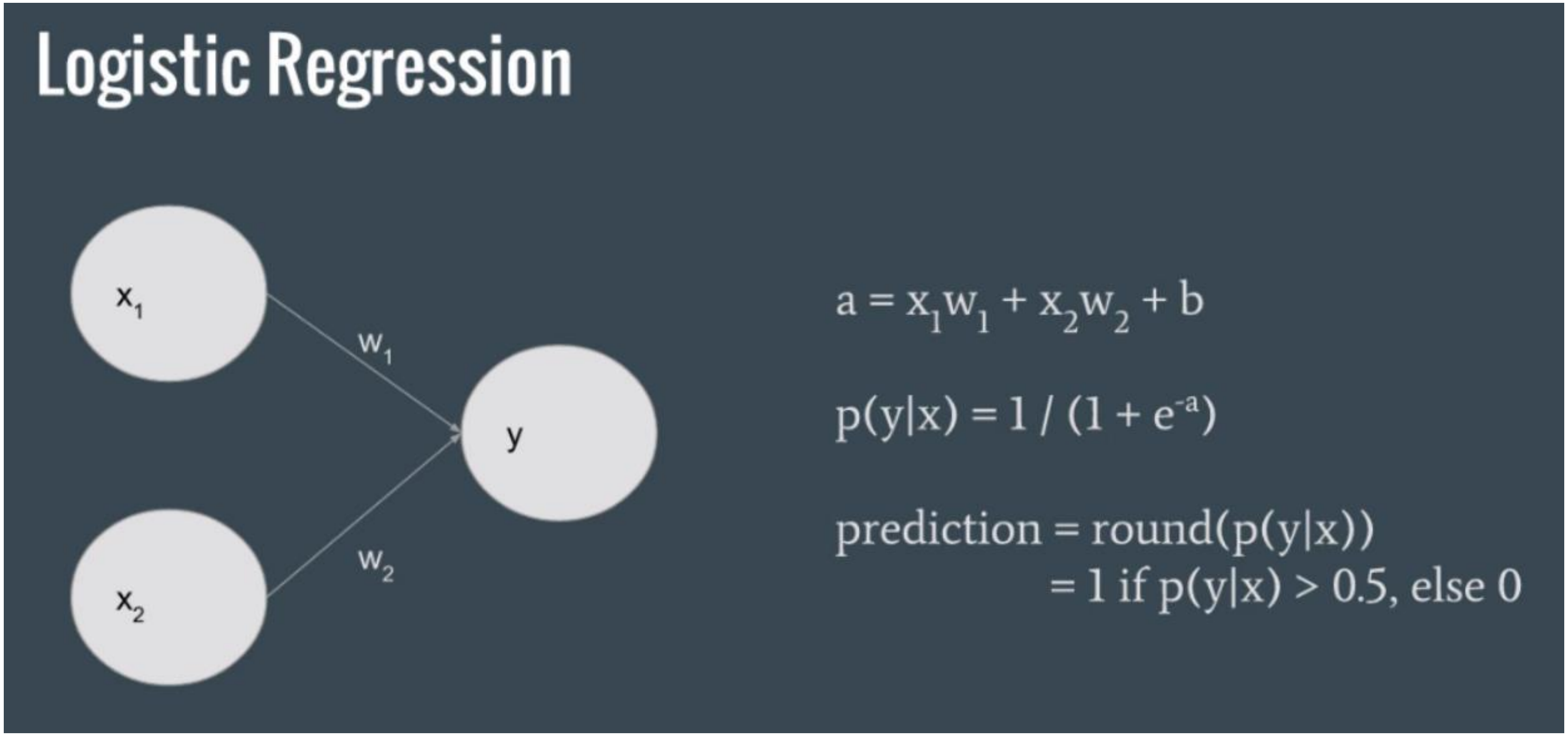
$\boxed{Y = \text{ROUND}(p(y/x))}$

SIGMOID

- $S(a) = 1/(1 + e^{-a})$



Logistic Regression simplified



Logistic Regression

$a = x_1w_1 + x_2w_2 + b$

$p(y|x) = 1 / (1 + e^{-a})$

$$\text{prediction} = \text{round}(p(y|x))$$
$$= 1 \text{ if } p(y|x) > 0.5, \text{ else } 0$$

Minimizing the error function (Logistic)

$$J = -\sum_{n=1}^{N} t_n log(y_n) + (1 - t_n)log(1 - y_n)$$

$$\frac{\partial J}{\partial w_i} = \sum_{n=1}^{N} \frac{\partial J}{\partial y_n}\frac{\partial y_n}{\partial a_n}\frac{\partial a_n}{\partial w_i}$$

$$a_n = w^T x_n$$

$$\boxed{\frac{\partial J}{\partial y_n} = - \quad t_n \frac{1}{y_n} + (1 - t_n)\frac{1}{1-y_n}(-1)}$$

$$y_n = \sigma(a_n) = \frac{1}{1+e^{-a_n}}$$

$$\frac{\partial y_n}{\partial a_n} = \frac{-1}{(1+e^{-a_n})^2}(e^{-a_n})(-1)$$

$$\boxed{\frac{\partial y_n}{\partial a_n} = \frac{e^{-a_n}}{(1+e^{-a_n})^2} = \frac{1}{1+e^{-a_n}}\frac{e^{-a_n}}{1+e^{-a_n}} = y_n(1 - y_n)}$$

$$a_n = w^T x_n$$

$$a_n = w_0 x_{n0} + w_1 x_{n1} + w_2 x_{n2} + \ldots$$

$$\boxed{\frac{\partial a_n}{\partial w_i} = x_{ni}}$$

$$\frac{\partial J}{\partial w_i} = -\sum_{n=1}^{N} \frac{t_n}{y_n} y_n(1 - y_n)x_{ni} - \frac{1-t_n}{1-y_n} y_n(1 - y_n)x_{ni}$$

$$\frac{\partial J}{\partial w_i} = -\sum_{n=1}^{N} t_n(1 - y_n)x_{ni} - (1 - t_n)y_n x_{ni}$$

$$\frac{\partial J}{\partial w_i} = -\sum_{n=1}^{N} [t_n - t_n y_n - y_n + t_n y_n]x_{ni}$$

$$\frac{\partial J}{\partial w_i} = \sum_{n=1}^{N} (y_n - t_n)x_{ni}$$

$$\frac{\partial J}{\partial w} = \sum_{n=1}^{N} (y_n - t_n)x_n$$

$$\boxed{\frac{\partial J}{\partial w} = X^T(Y - T)}$$

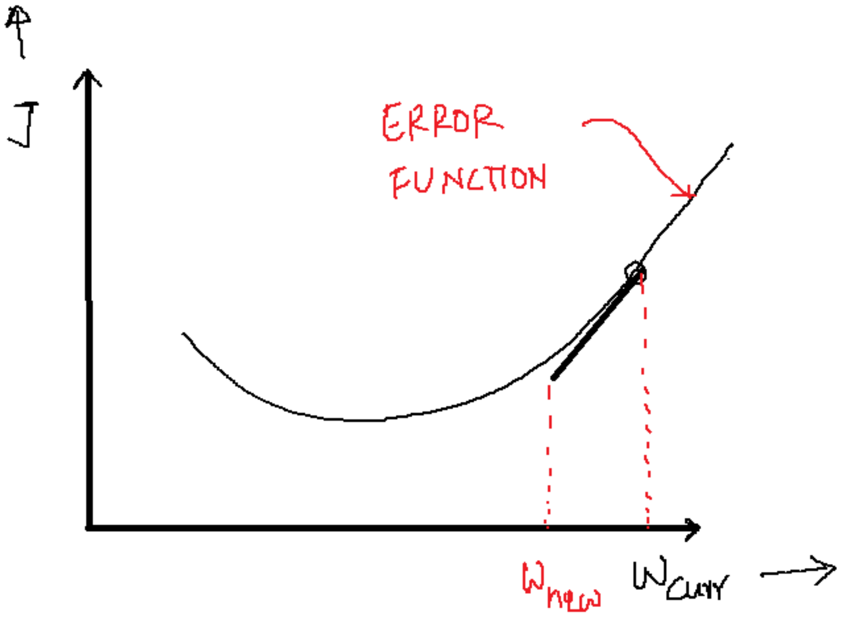
Calculating the weights w_i

- Now that we have an expression for minimizing the error function with respect to w_i
- We can begin the process of calculating the weights themselves
- This is an iterative procedure known as the "Gradient Descent Method" and it works as follows: (also refer next slide)
  1. Initialize **w** randomly
  2. Find out the predicted Y
  3. Find out gradient $\frac{\partial J}{\partial w} = X^T(Y - T)$
  4. Descend along the gradient to get new weights
  5. Repeat until termination criteria is reached

Basis of the Gradient Descent method

- The error function is given by:

$$J = -\sum_{i=1}^{N} t_i log(y_i) + (1 - t_i) log(1 - y_i)$$



- The gradient of this error $\frac{\partial J}{\partial w} = X^T(Y - T)$:

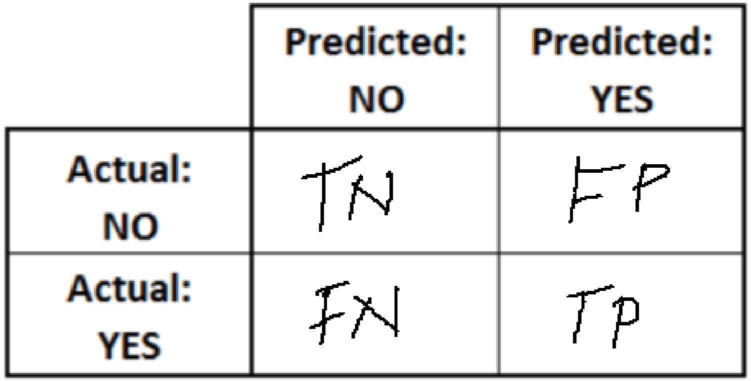
- In gradient descent, the weights are updated as:

$$w_{new} = w_{curr} - \eta \cdot \frac{dJ}{dw}$$

$$w \leftarrow w - \eta \nabla J$$

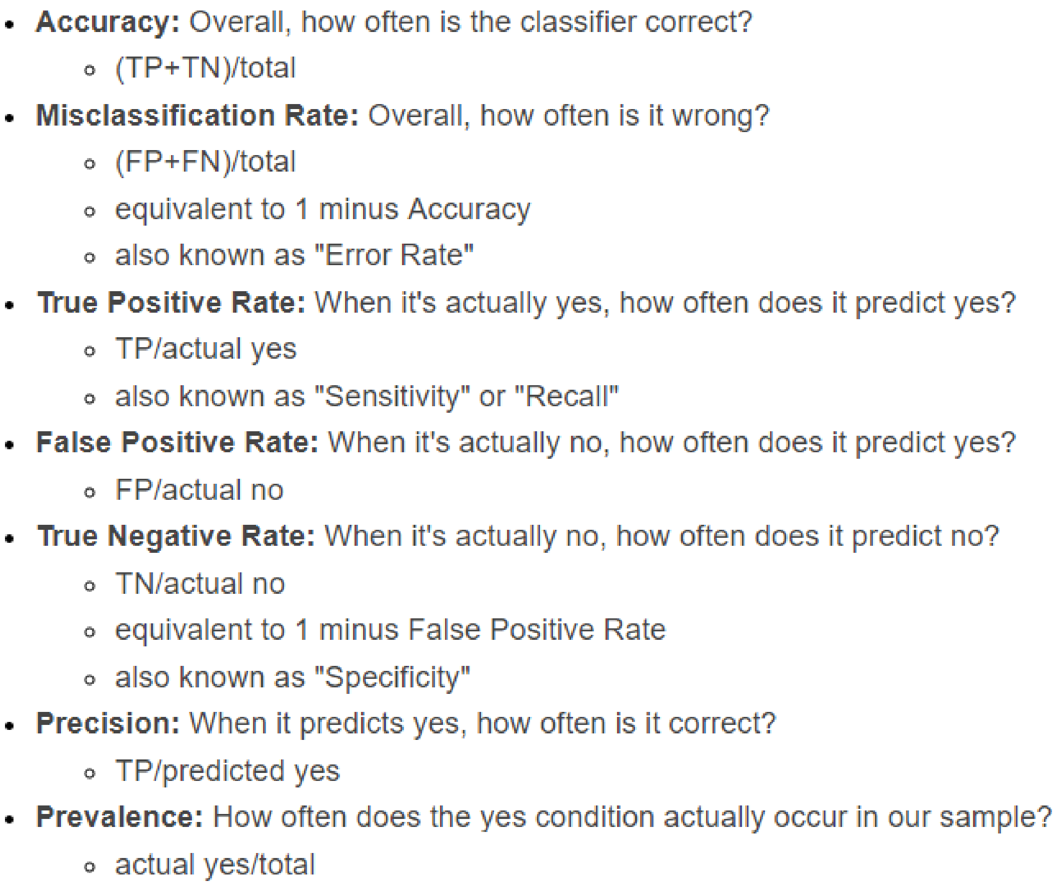
$$\eta \longrightarrow \text{LEARNING RATE}$$

Logistic Regression: Quality Metrics

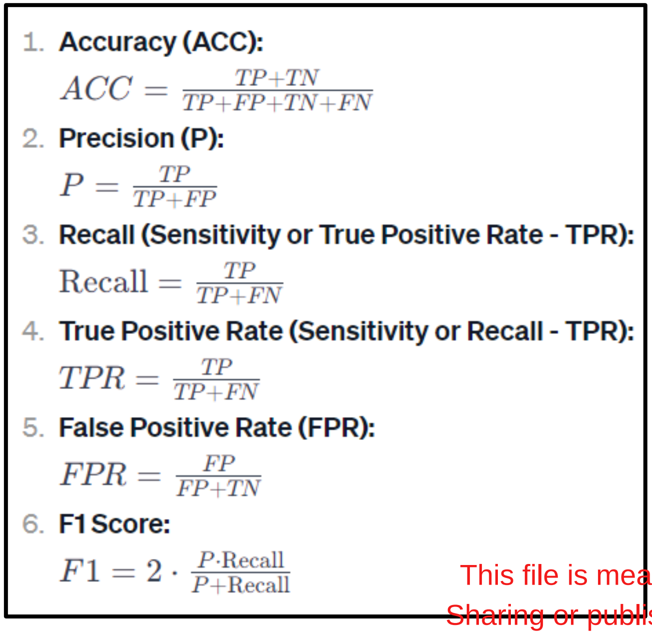


	Predicted: NO	Predicted: YES
Actual: NO	TN	FP
Actual: YES	FN	TP

CONFUSION MATRIX



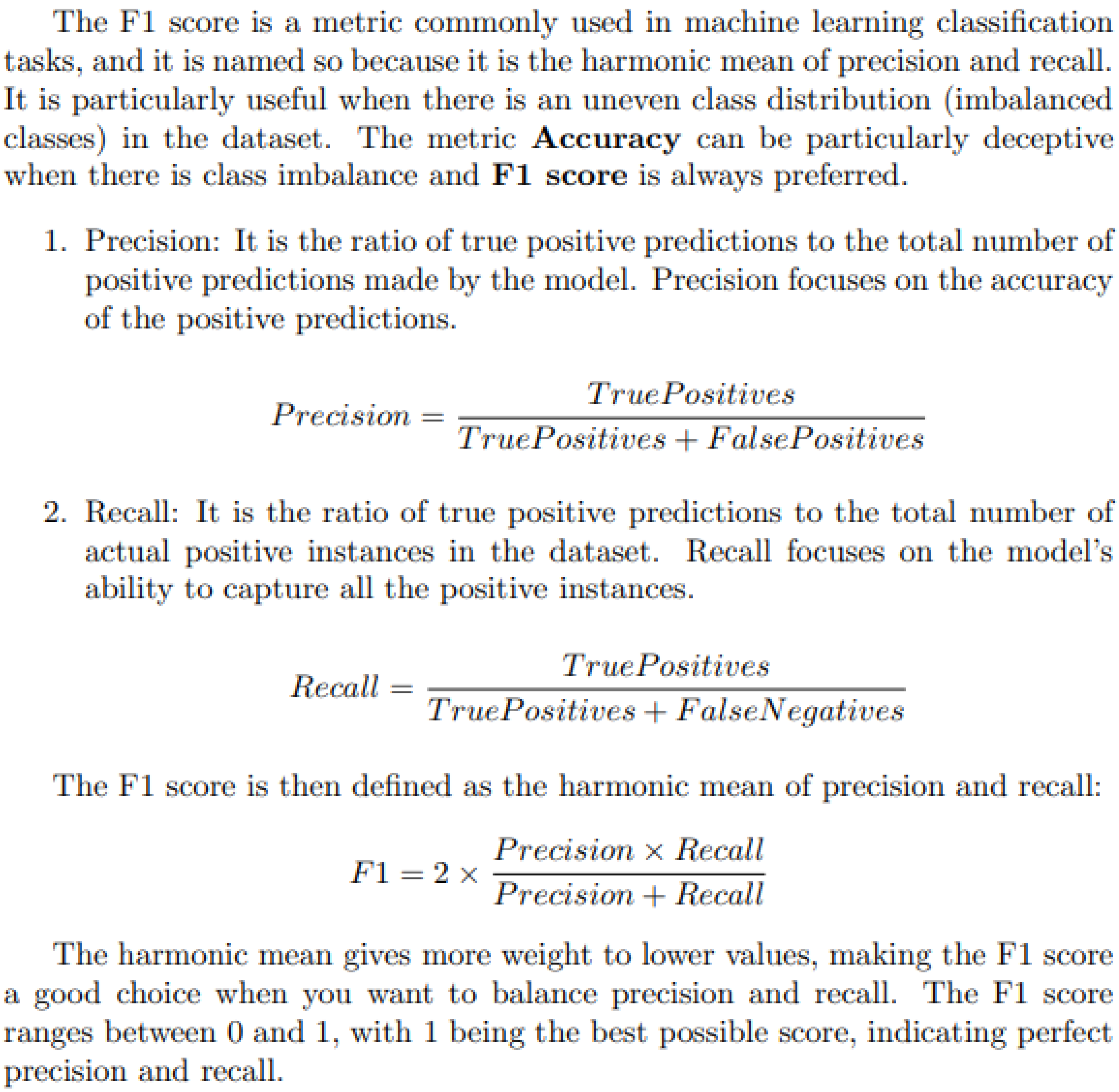
- **Accuracy:** Overall, how often is the classifier correct?
  - (TP+TN)/total
- **Misclassification Rate:** Overall, how often is it wrong?
  - (FP+FN)/total
  - equivalent to 1 minus Accuracy
  - also known as "Error Rate"
- **True Positive Rate:** When it's actually yes, how often does it predict yes?
  - TP/actual yes
  - also known as "Sensitivity" or "Recall"
- **False Positive Rate:** When it's actually no, how often does it predict yes?
  - FP/actual no
- **True Negative Rate:** When it's actually no, how often does it predict no?
  - TN/actual no
  - equivalent to 1 minus False Positive Rate
  - also known as "Specificity"
- **Precision:** When it predicts yes, how often is it correct?
  - TP/predicted yes
- **Prevalence:** How often does the yes condition actually occur in our sample?
  - actual yes/total



1. **Accuracy (ACC):**
   $$ACC = \frac{TP+TN}{TP+FP+TN+FN}$$
2. **Precision (P):**
   $$P = \frac{TP}{TP+FP}$$
3. **Recall (Sensitivity or True Positive Rate - TPR):**
   $$\text{Recall} = \frac{TP}{TP+FN}$$
4. **True Positive Rate (Sensitivity or Recall - TPR):**
   $$TPR = \frac{TP}{TP+FN}$$
5. **False Positive Rate (FPR):**
   $$FPR = \frac{FP}{FP+TN}$$
6. **F1 Score:**
   $$F1 = 2 \cdot \frac{P \cdot \text{Recall}}{P+\text{Recall}}$$

Before calculating the metrics by using the confusion matrix, check and understand the 'meaning' ascribed to the rows and columns. Are the ACTUAL values represented by the ROWS and PREDICTED values by the COLUMNS, or VICE VERSA

The F1 Score



The F1 score is a metric commonly used in machine learning classification tasks, and it is named so because it is the harmonic mean of precision and recall. It is particularly useful when there is an uneven class distribution (imbalanced classes) in the dataset. The metric **Accuracy** can be particularly deceptive when there is class imbalance and **F1 score** is always preferred.

1. Precision: It is the ratio of true positive predictions to the total number of positive predictions made by the model. Precision focuses on the accuracy of the positive predictions.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

2. Recall: It is the ratio of true positive predictions to the total number of actual positive instances in the dataset. Recall focuses on the model's ability to capture all the positive instances.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

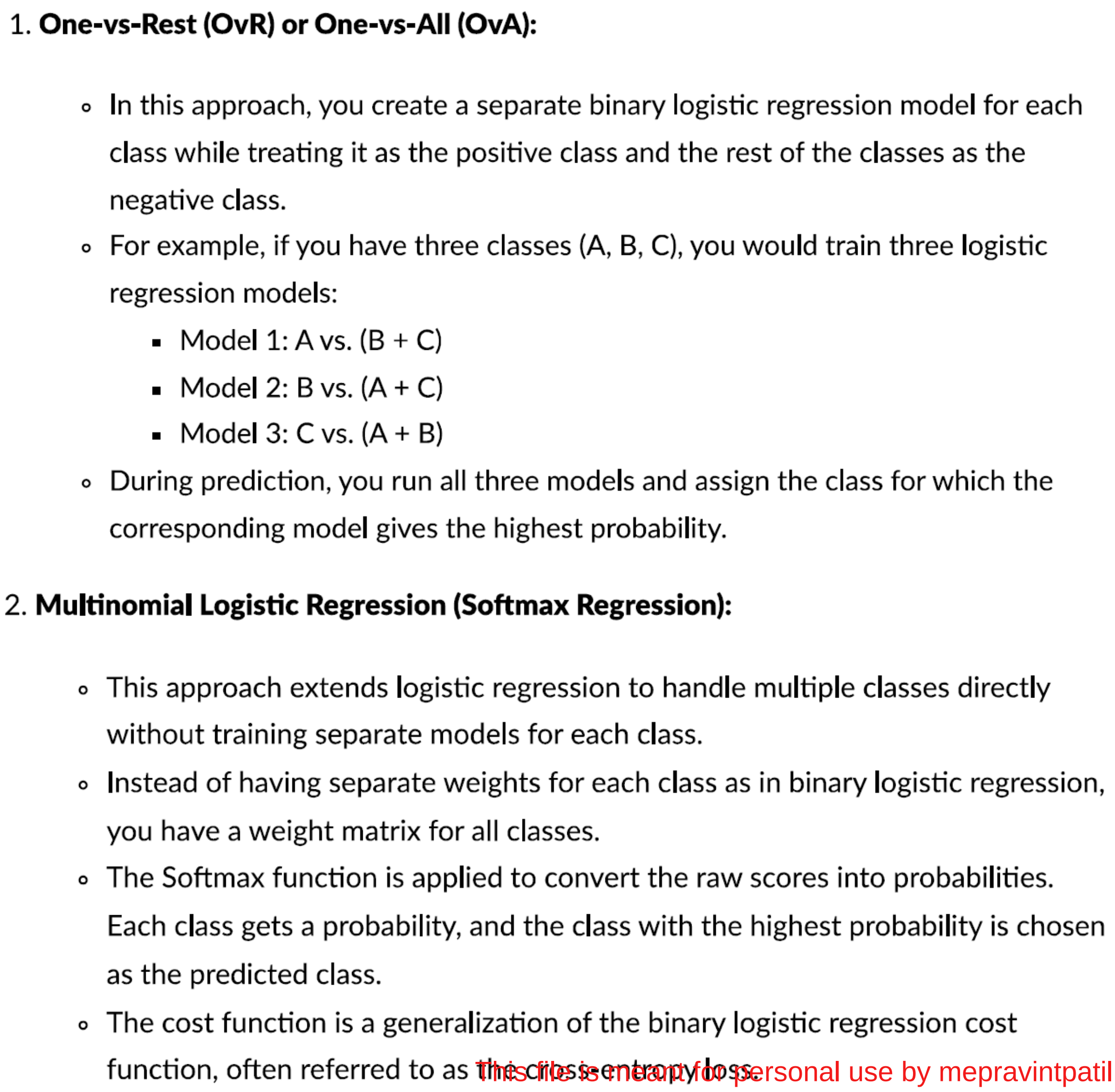
The F1 score is then defined as the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The harmonic mean gives more weight to lower values, making the F1 score a good choice when you want to balance precision and recall. The F1 score ranges between 0 and 1, with 1 being the best possible score, indicating perfect precision and recall.

Handling more than 2 classes …

The following techniques are used to handle multiple classes:

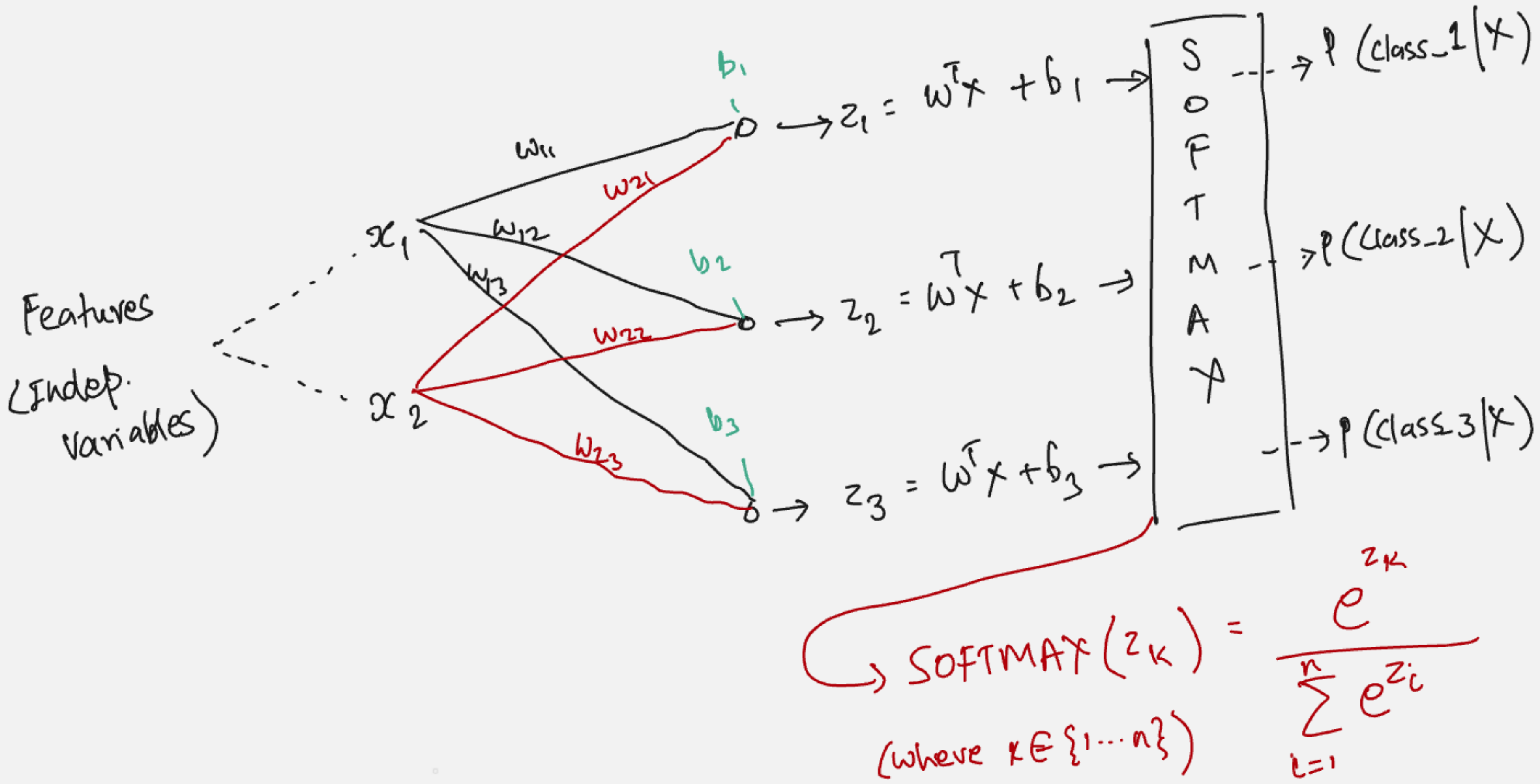


1. **One-vs-Rest (OvR) or One-vs-All (OvA):**
   - In this approach, you create a separate binary logistic regression model for each class while treating it as the positive class and the rest of the classes as the negative class.
   - For example, if you have three classes (A, B, C), you would train three logistic regression models:
     - Model 1: A vs. (B + C)
     - Model 2: B vs. (A + C)
     - Model 3: C vs. (A + B)
   - During prediction, you run all three models and assign the class for which the corresponding model gives the highest probability.
2. **Multinomial Logistic Regression (Softmax Regression):**
   - This approach extends logistic regression to handle multiple classes directly without training separate models for each class.
   - Instead of having separate weights for each class as in binary logistic regression, you have a weight matrix for all classes.
   - The Softmax function is applied to convert the raw scores into probabilities. Each class gets a probability, and the class with the highest probability is chosen as the predicted class.
   - The cost function is a generalization of the binary logistic regression cost function, often referred to as the cross-entropy loss.

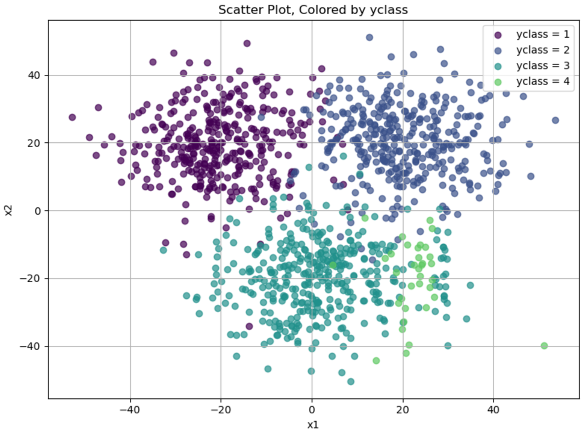
The choice between One-vs-Rest and Multinomial Logistic Regression depends on factors like dataset size, computational resources, and the nature of the problem.

Softmax regression is a more direct and computationally efficient approach for handling multiple classes when compared to One-vs-Rest.

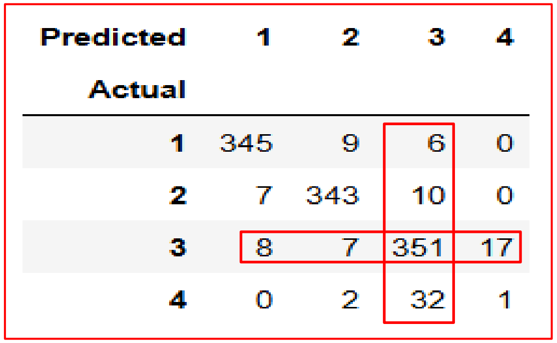
SOFTMAX: Used to create probabilities



Confusion Matrix: Multi-Class



Before calculating classification metrics by using the confusion matrix (CM), check and understand the 'meaning' ascribed to the rows and columns. Are the ACTUAL values represented by the ROWS and PREDICTED values by the COLUMNS, or VICE_VERSA



Predicted	1	2	3	4
Actual				
1	345	9	6	0
2	7	343	10	0
3	8	7	351	17
4	0	2	32	1

How to read the Confusion Matrix?

- In the CMs alongside, consider class '3'

Original Class	Observations	Predicted Class	Result
Class 3	8	Class 1	Incorrect (FN)
Class 3	7	Class 2	Incorrect (FN)
Class 3	351	Class 3	Correct (TP)
Class 3	17	Class 4	Incorrect (FN)
Class 1	6	Class 3	Incorrect (FP)
Class 2	10	Class 3	Incorrect (FP)
Class 3	351	Class 3	Correct (TP)
Class 4	32	Class 3	Incorrect (FP)

```
Overall Accuracy: 0.9138840070298769

Classification Report:
              precision    recall  f1-score   support

           1       0.96      0.96      0.96       360
           2       0.95      0.95      0.95       360
           3       0.88      0.92      0.90       383
           4       0.06      0.03      0.04        35

    accuracy                           0.91      1138
   macro avg       0.71      0.71      0.71      1138
weighted avg       0.90      0.91      0.91      1138
```

Precision _3 : $\frac{351}{(6+10+351+32)} = 0.88$

Recall _3 : $\frac{351}{(8+7+351+17)} = 0.916 \sim 0.92$

ROC Curves

Confusion Matrix can be created for various threshold values of classification probability

Default threshold value of probability is '0.5'

if $p < 0.5$ ⇒ classification is '0'

if $p \geq 0.5$ ⇒ classification is '1'

One measure of the quality of the model involves assessing its performance for various probability threshold values and asking the question:

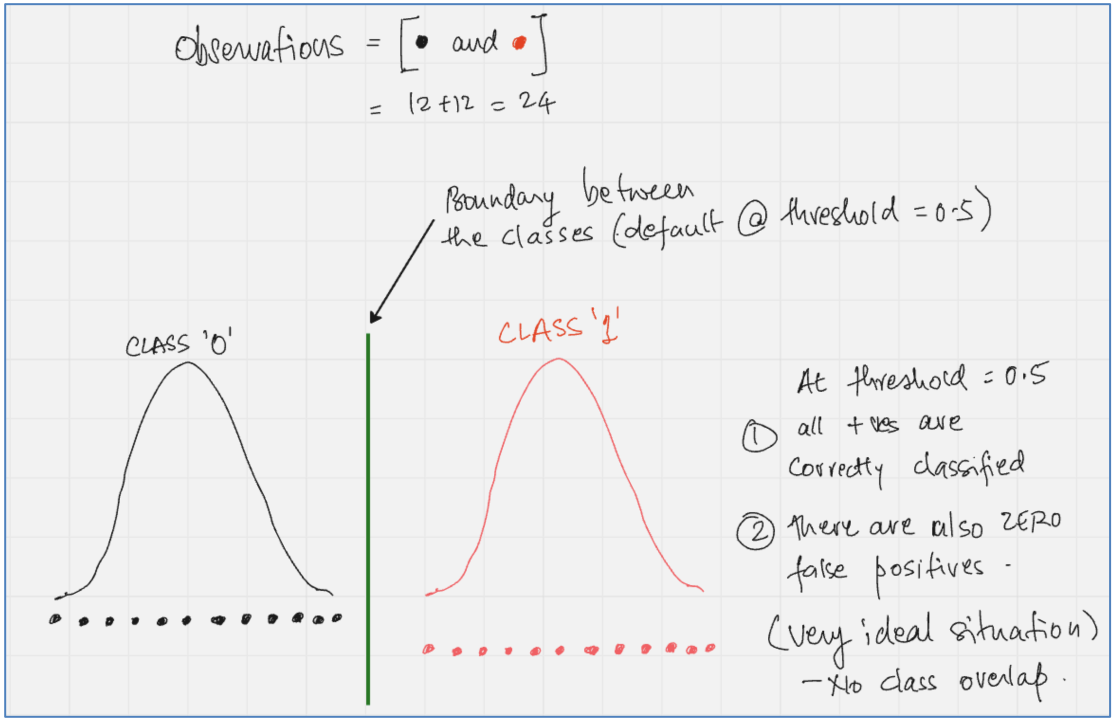
HOW RELIABLY WILL THE MODEL DETECT TRUE POSITIVES, BEFORE IT STARTS DETECTING FALSE POSITIVES

A good model should detect all the TRUE POSITIVES before it starts flagging the FALSE POSITIVES.

A plot of TPR v/s FPR is created to carry out this analysis. The resulting plot is known as the ROC characteristic of the model.

ROC characteristic of the model.
↳ RECEIVER OPERATING CHARACTERISTIC

ROC - Explanation



Observations = [● and ●]

= 12+12 = 24

Boundary between the classes (default @ threshold = 0.5)

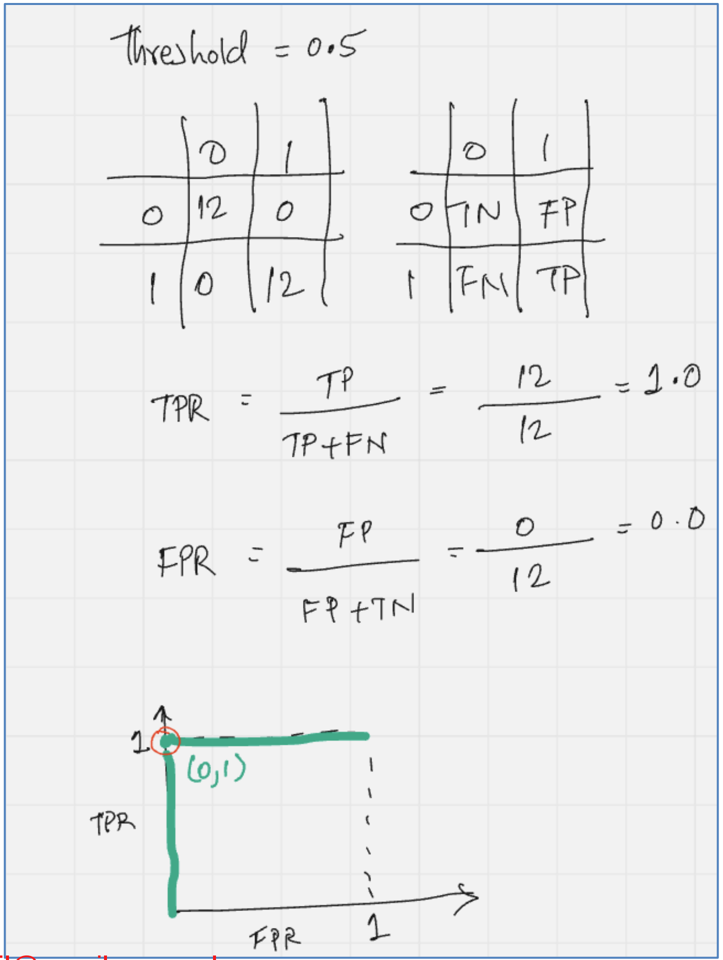
At threshold = 0.5

① all +ves are correctly classified

② there are also ZERO false positives.

(Very ideal situation)
– No class overlap.

An ideal situation with no overlaps between the classes



Threshold = 0.5

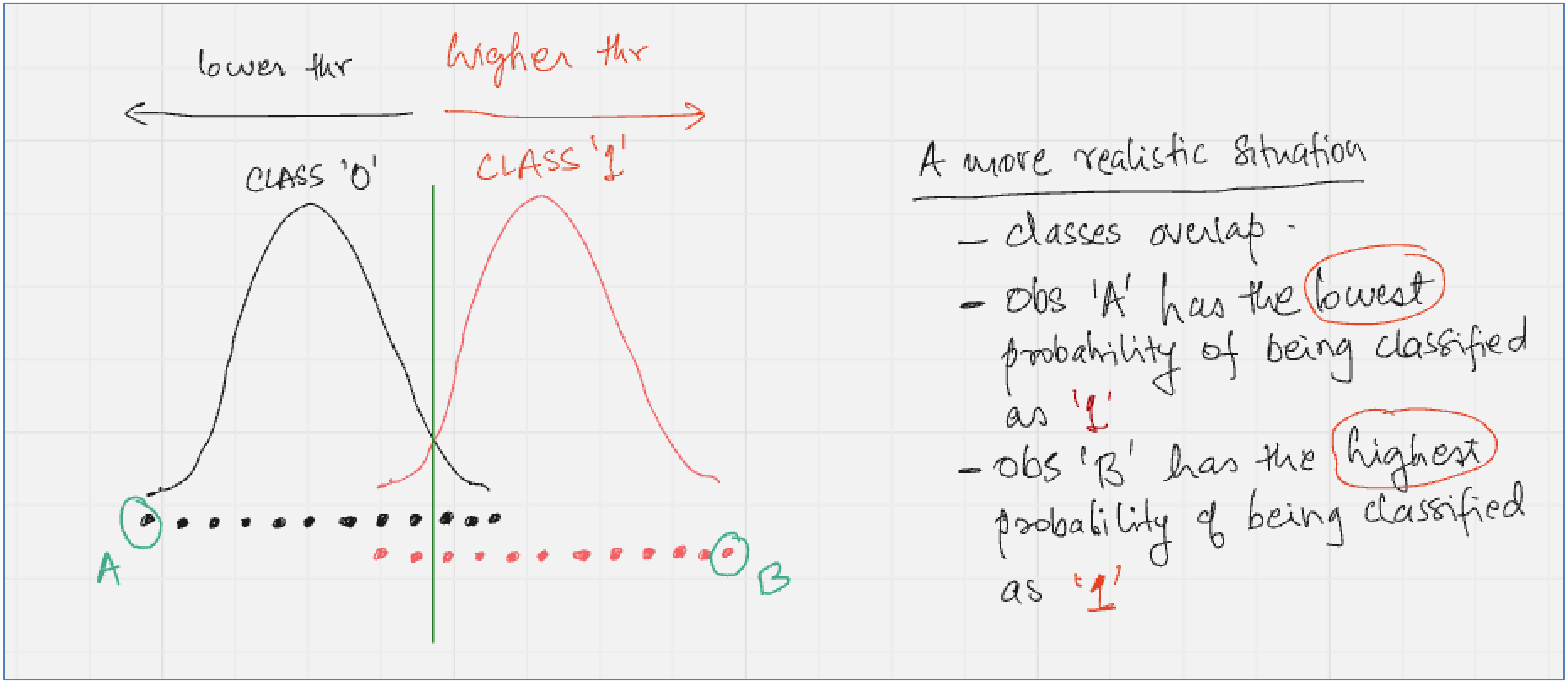
	0	1
0	12	0
1	0	12

	0	1
0	TN	FP
1	FN	TP

$$TPR = \frac{TP}{TP+FN} = \frac{12}{12} = 1.0$$

$$FPR = \frac{FP}{FP+TN} = \frac{0}{12} = 0.0$$

ROC - Explanation

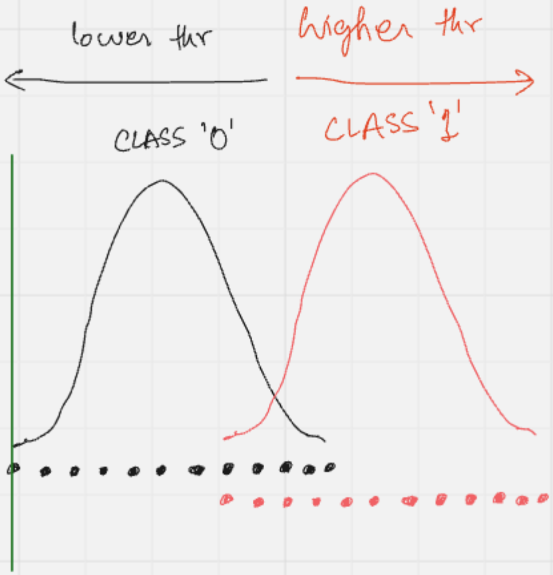


A more realistic situation

- classes overlap.
- obs 'A' has the lowest probability of being classified as '1'
- obs 'B' has the highest probability of being classified as '1'

If the threshold probability for classification is increased from 0.5, the classification boundary shifts to the right, and vice-versa

ROC - Explanation



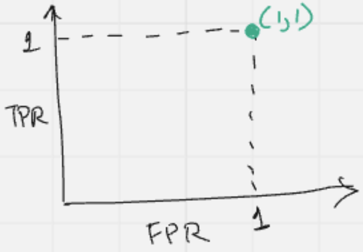
- Lets have the boundary shifted to the far left
- This corresponds to threshold = 0.0
- All points will get classified as '1'
- Confusion matrix

	0	1
0	0	12
1	0	12

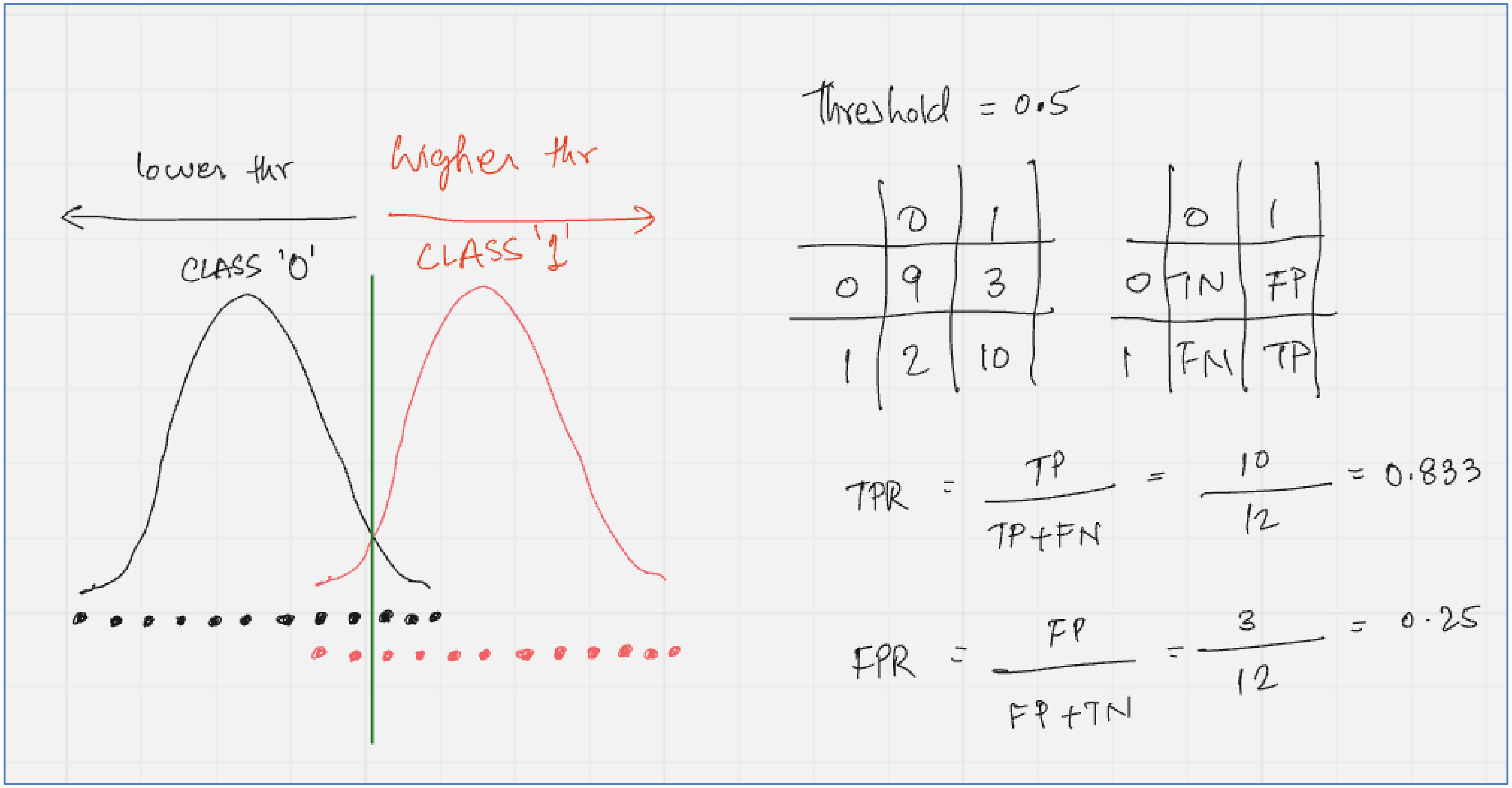
	0	1
0	TN	FP
1	FN	TP

$$TPR = \frac{TP}{TP + FN} = \frac{12}{12 + 0} = 1$$

$$FPR = \frac{FP}{FP + TN} = \frac{12}{12 + 0} = 1$$



ROC - Explanation



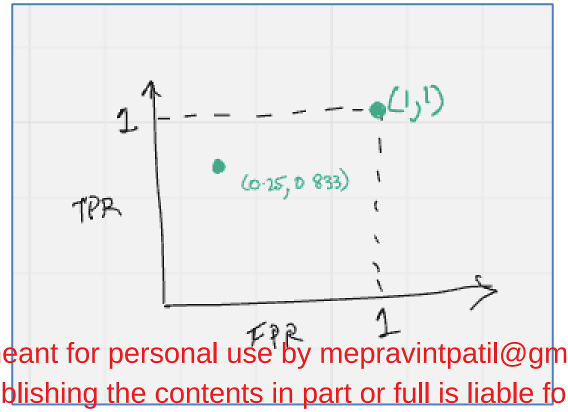
Threshold = 0.5

	0	1
0	9	3
1	2	10

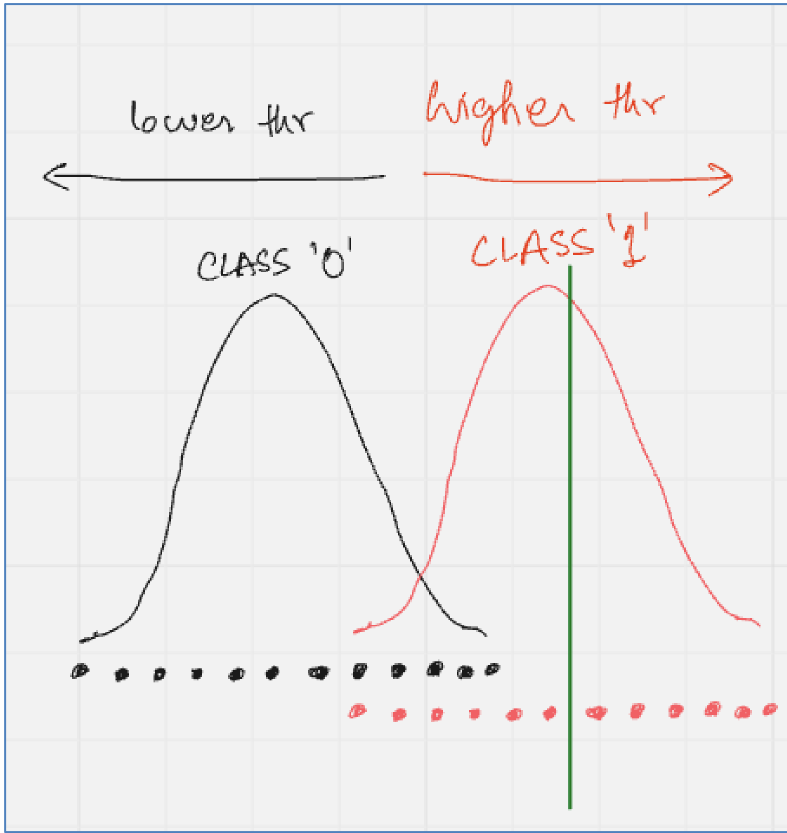
	0	1
0	TN	FP
1	FN	TP

$$TPR = \frac{TP}{TP+FN} = \frac{10}{12} = 0.833$$

$$FPR = \frac{FP}{FP+TN} = \frac{3}{12} = 0.25$$



ROC - Explanation



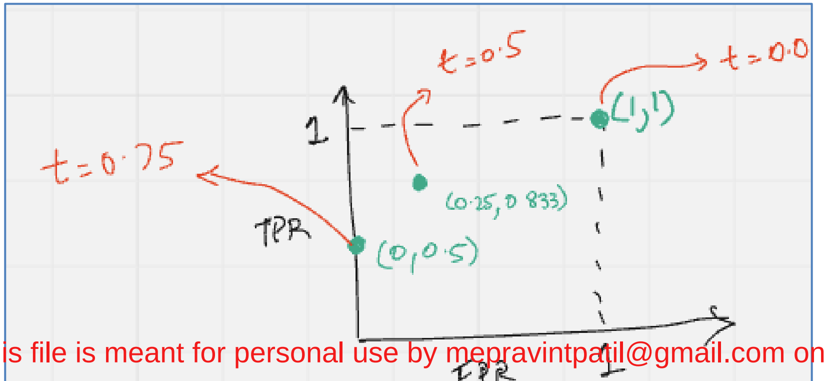
Threshold = 0.75

	0	1
0	12	0
1	6	6

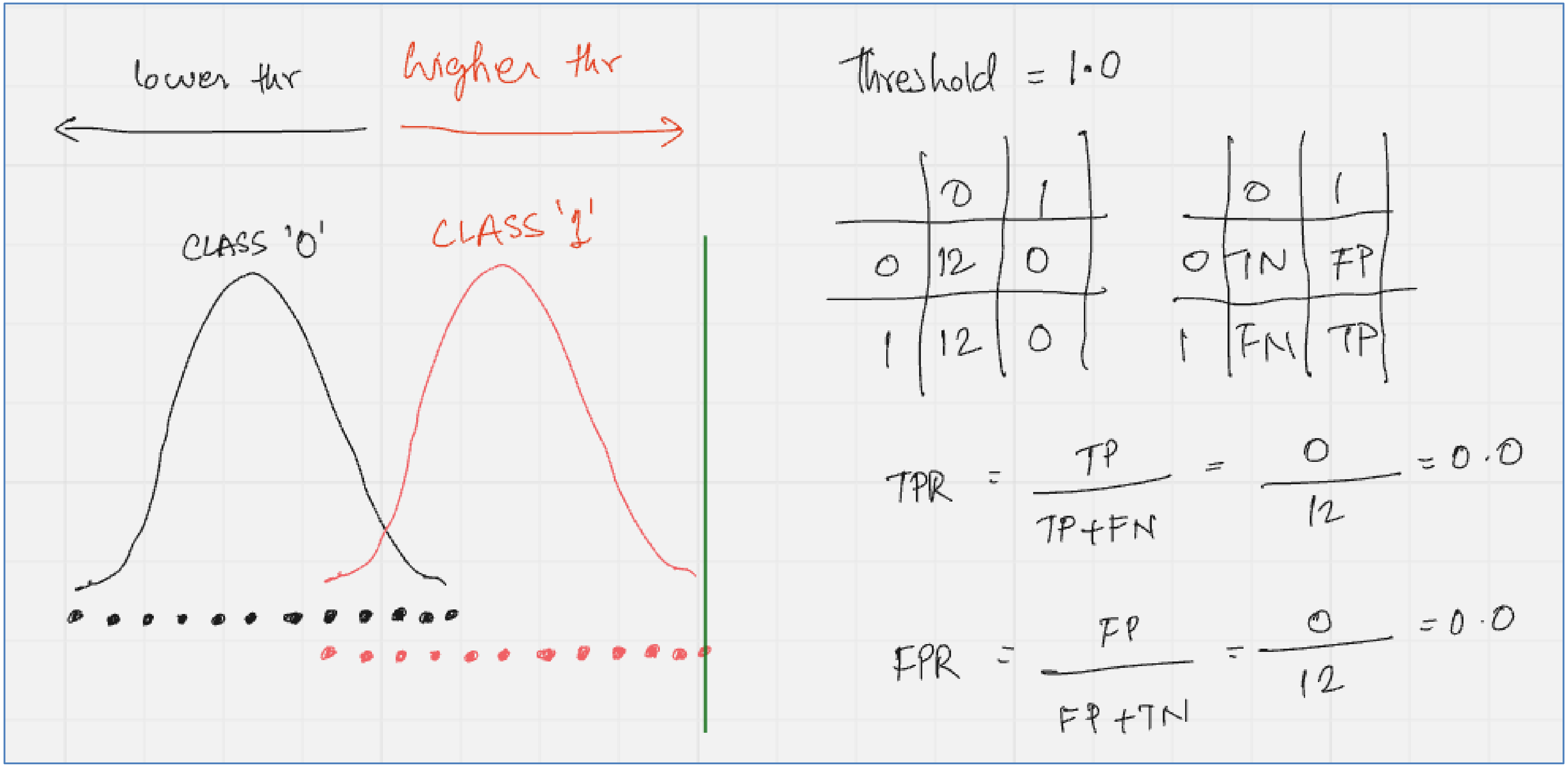
	0	1
0	TN	FP
1	FN	TP

$$TPR = \frac{TP}{TP+FN} = \frac{6}{12} = 0.5$$

$$FPR = \frac{FP}{FP+TN} = \frac{0}{12} = 0.0$$



ROC - Explanation



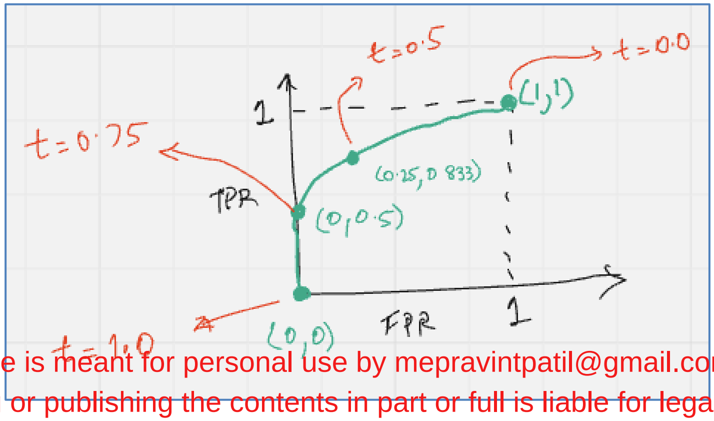
Threshold = 1.0

	0	1
0	12	0
1	12	0

	0	1
0	TN	FP
1	FN	TP

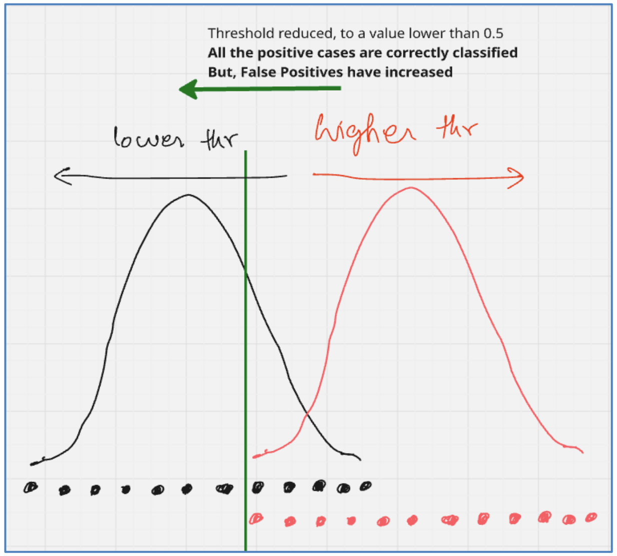
$$TPR = \frac{TP}{TP+FN} = \frac{0}{12} = 0.0$$

$$FPR = \frac{FP}{FP+TN} = \frac{0}{12} = 0.0$$



ROC Based Threshold Modification - Use Cases

Changing the classification threshold from the default 0.5 is a common and often crucial step in deploying a machine learning classification model. The “best” threshold is rarely 0.5 and depends heavily on the specific context, the costs associated with different types of errors, and the business objective.



Threshold reduced, to a value lower than 0.5

All the positive cases are correctly classified

But, False Positives have increased

CRITICAL ILLNESS PREDICTION

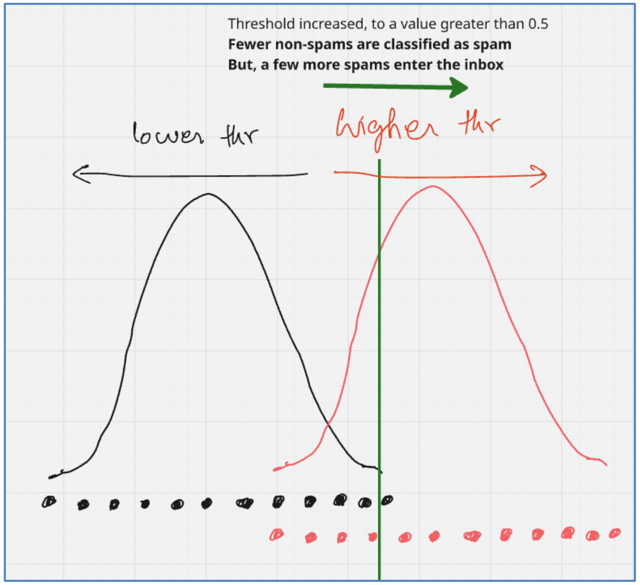
Situation: Imagine a model designed to detect a serious, life-threatening disease (e.g., cancer, a severe infection). A **false negative** (missing an actual disease case) could have catastrophic consequences, potentially leading to delayed treatment and increased mortality. A **false positive** (diagnosing a healthy person with the disease) is undesirable but less severe; it might lead to further tests, anxiety, and some inconvenience, but it’s generally reversible.

Threshold Adjustment: In this scenario, you would want to **lower the classification threshold (e.g., to 0.3 or 0.2)**.

Impact: Lowering the threshold makes the model more sensitive to positive cases, increasing the **True Positive Rate (Recall)**. This means it will catch more actual disease cases, even at the cost of increasing the **False Positive Rate**. The goal is to minimize false negatives, ensuring that very few sick patients are missed, even if it means some healthy patients get a false alarm.

ROC Based Threshold Modification - Use Cases

Changing the classification threshold from the default 0.5 is a common and often crucial step in deploying a machine learning classification model. The “best” threshold is rarely 0.5 and depends heavily on the specific context, the costs associated with different types of errors, and the business objective.



Threshold increased, to a value greater than 0.5

Fewer non-spams are classified as spam

But, a few more spams enter the inbox

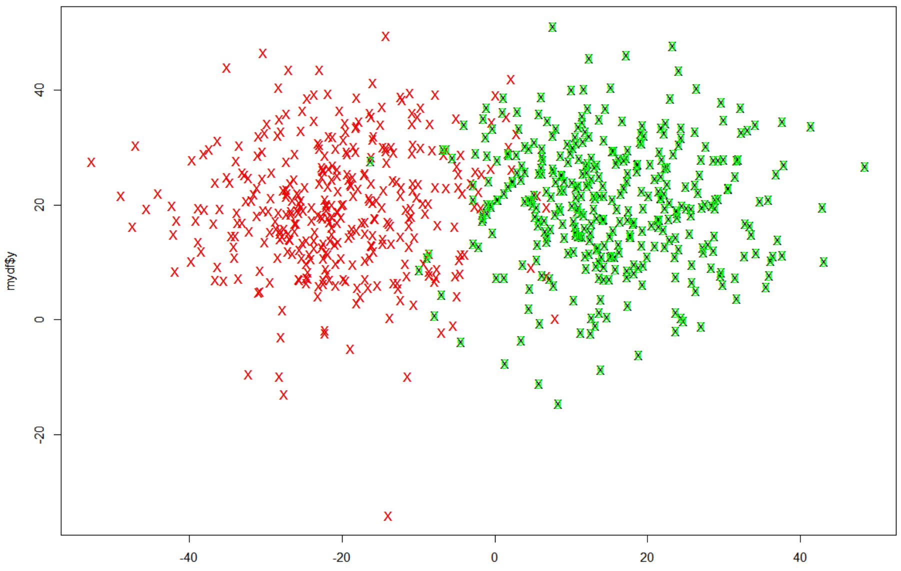
SPAM CLASSIFICATION

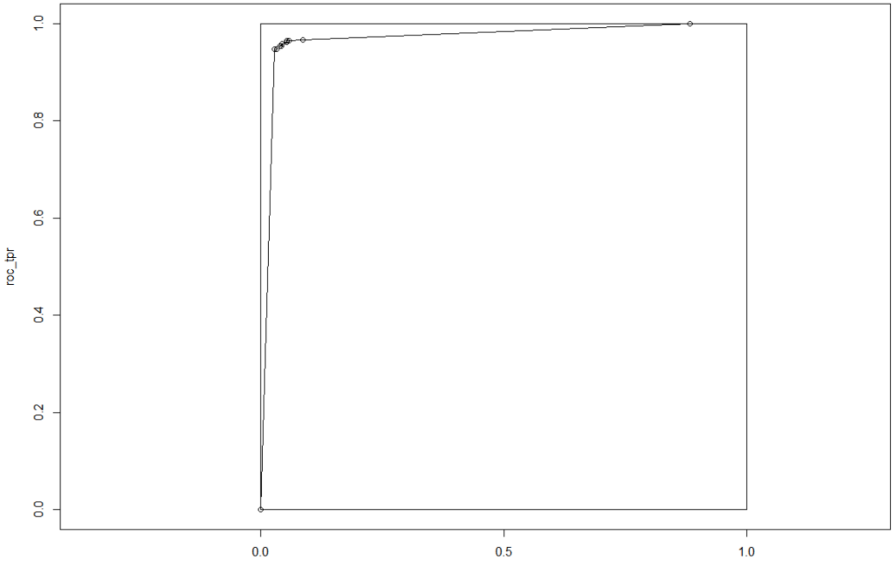
Situation: Consider an email spam filter. A **false positive** (classifying a legitimate email as spam and moving it to the junk folder) is highly undesirable. Users might miss important communications, leading to frustration and potential loss of critical information. A **false negative** (a spam email getting through to the inbox) is annoying but generally less disruptive than missing a legitimate email.

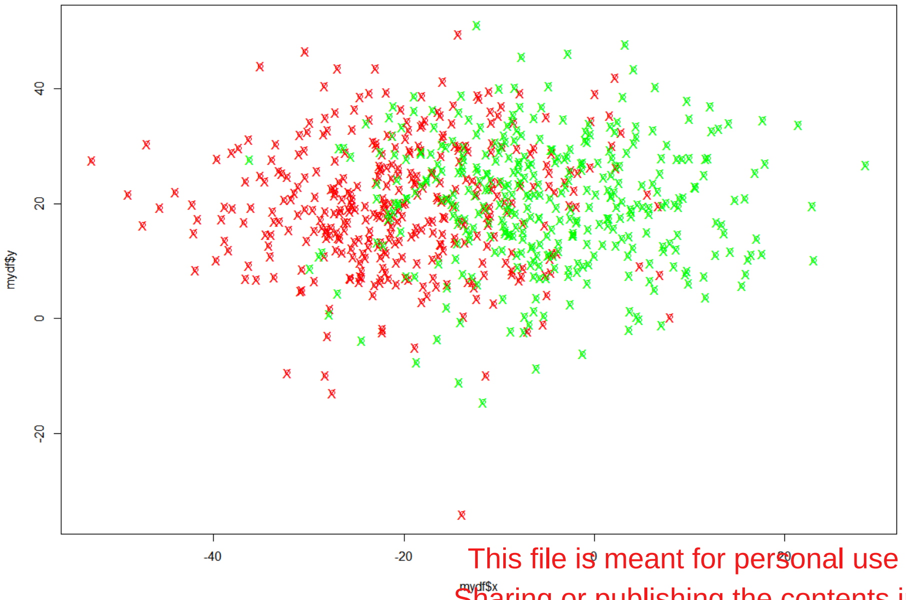
Threshold Adjustment: Here, you would want to **raise the classification threshold (e.g., to 0.7 or 0.8)**.

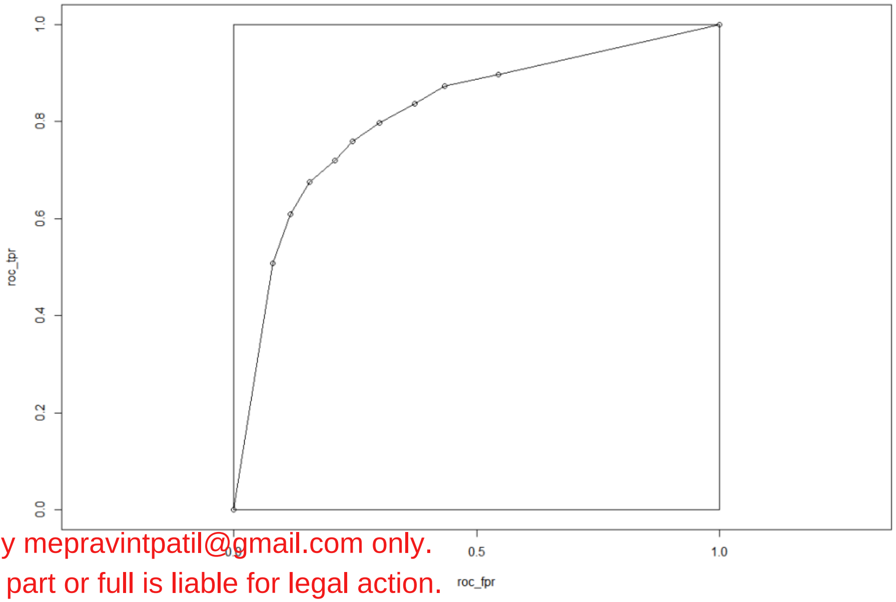
Impact: Raising the threshold makes the model more conservative in classifying emails as spam, increasing the **Precision**. This means that when the model *does* classify an email as spam, it’s very likely to be correct, thus minimizing the chances of legitimate emails being flagged as spam. This comes at the cost of allowing more spam to slip through (lower Recall).

Impact of Data on ROC Plots

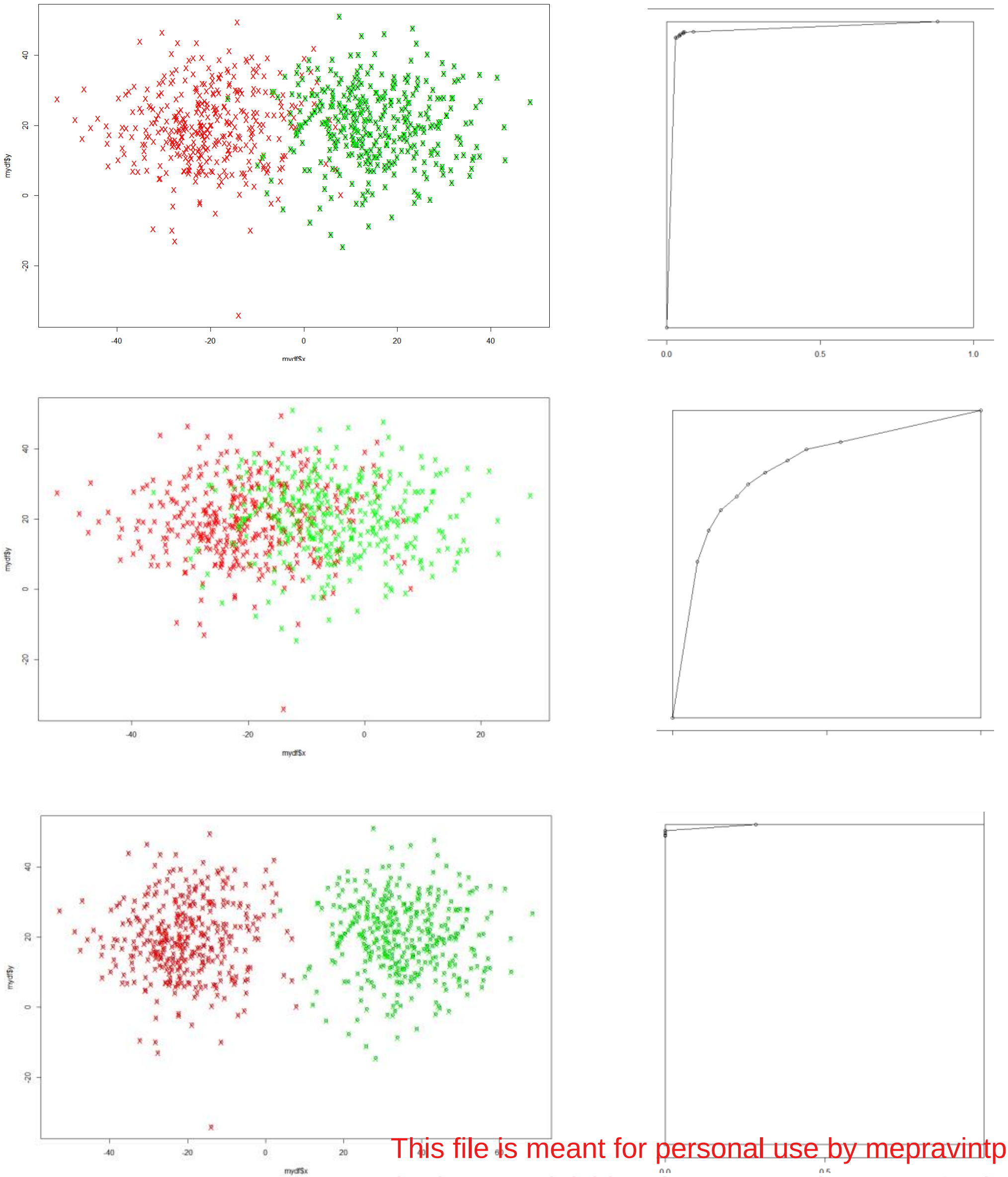








Impact of Data Characteristics on ROC Plots



It can be seen that clear class separation results in a sharper ROC curve

Quality of the classifier is indicated by the area under the ROC curve (known as AUC).

AUC should be as close to '1' as possible