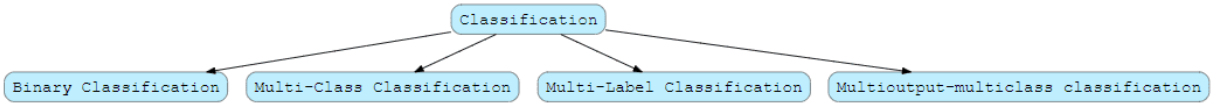
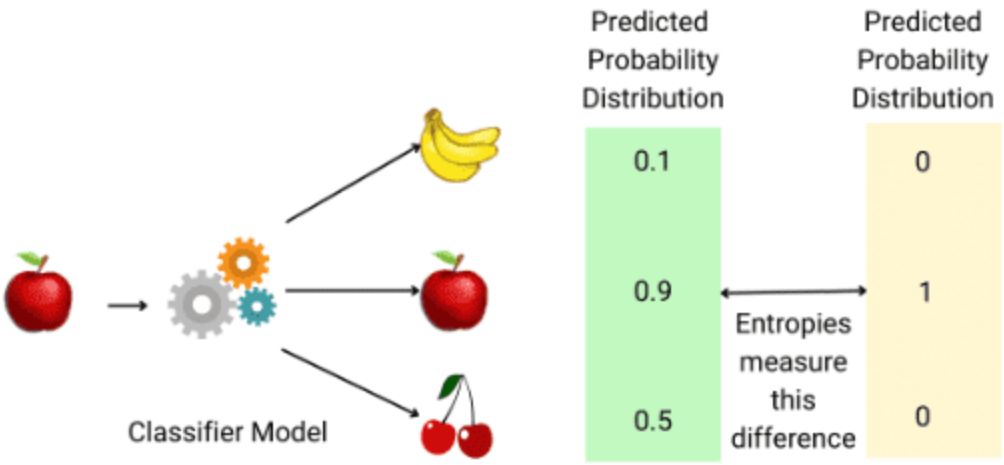
Types of Classification Problems



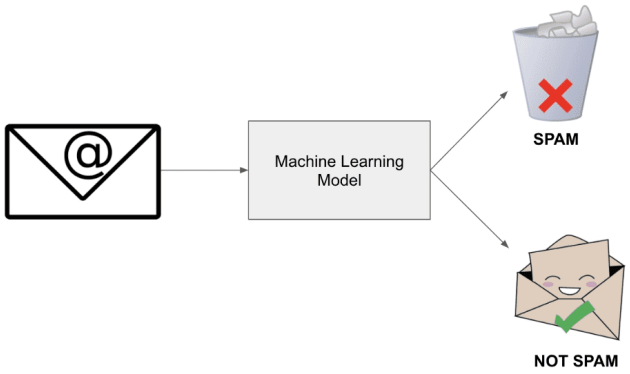
What is Classification?

- **Classification Problem**: A type of machine learning problem where the goal is to predict the category or class of given data points.
- **Output Variable**: In a classification problem, the output variable is a category or a discrete value.
- **Mapping Function**: The classification predictive modeling approximates a mapping function (f) from input variables (X) to discrete output variables (y).

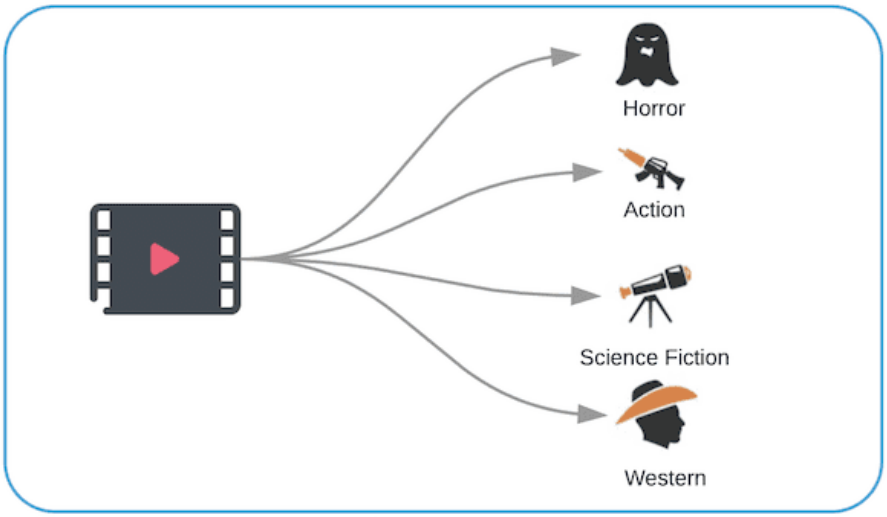


Classification problems in Machine Learning

- **Binary Classification:** This is a type of supervised learning where instances are labeled with two classes. For example, a medical test could be used to determine whether a patient has a particular disease (positive) or not (negative)

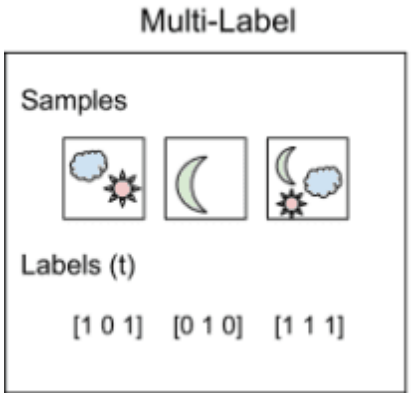


- **Multi-class Classification:** In this type of classification, instances are labeled with more than two classes. For instance, classifying a city's weather into one of the following classes: dry, hot, cold, or humid

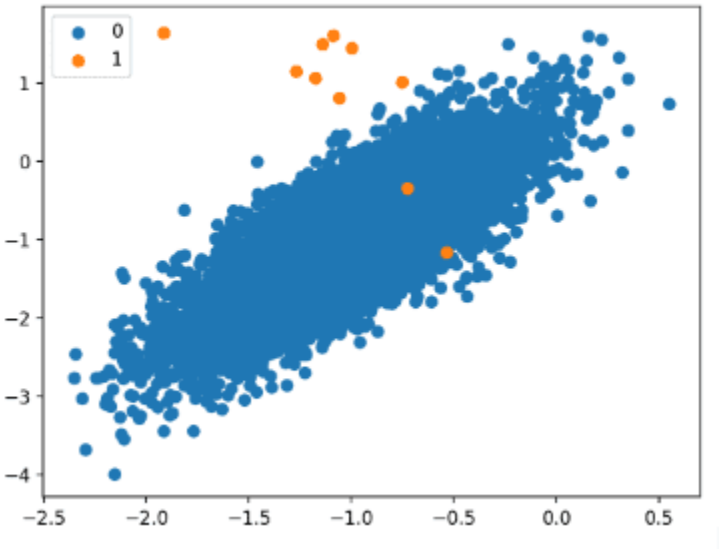


Multi-class classification

- **Multi-label Classification:** This involves predicting more than one label for each instance. For example, a picture might consist of multiple objects like a car, an airplane, and a train, and the task might be to predict all the labels of the image



- **Imbalanced Classification:** This refers to classification tasks where the number of instances in each class are unequally distributed. For example, in fraud detection, most of the transactions that happen daily are normal, and fraudulent transactions are rare.



Strategies for Multi-class and Multi-label Classification

- **One-vs-Rest (One-vs-All):** This strategy fits one binary classification model for each class vs. all other classes

  For example, if we have a multi-class classification problem with classes red, green, and blue, the binary classification can be categorized as

follows:

- Problem one: red vs. green/blue
- Problem two: blue vs. green/red
- Problem three: green vs. blue/red

This means that for each class, a binary classifier is trained against all the other classes. The classifier with the highest confidence in its prediction is typically chosen as the final output.

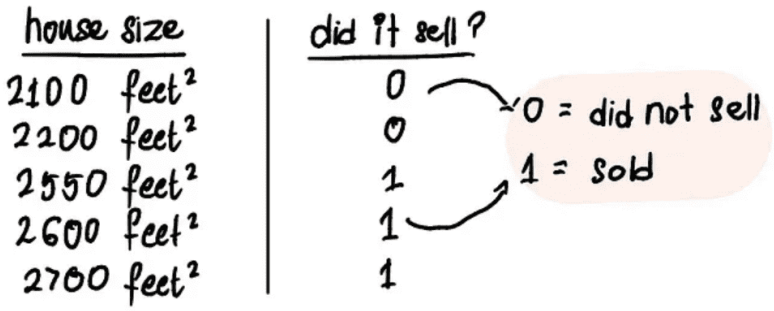
- **One-vs-One:** This strategy fits one binary classification model for each pair of classes

  For instance, given a multi-class classification problem with classes red, blue, and green, this could be divided into three binary classification datasets as follows:
  - Problem one: red vs. blue
  - Problem two: red vs. green
  - Problem three: blue vs. green

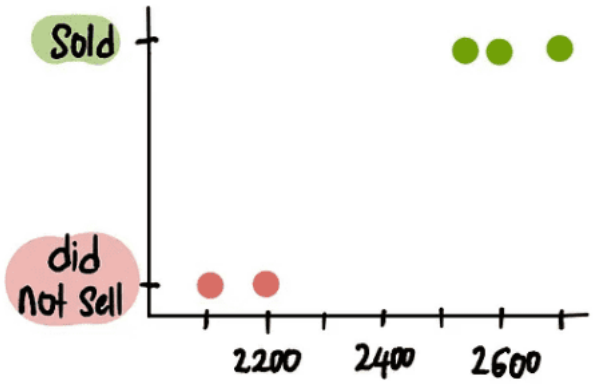
Each binary classification model may predict one class label and the model with the most predictions or votes is predicted by the one-vs-one strategy.

Logistic Regression

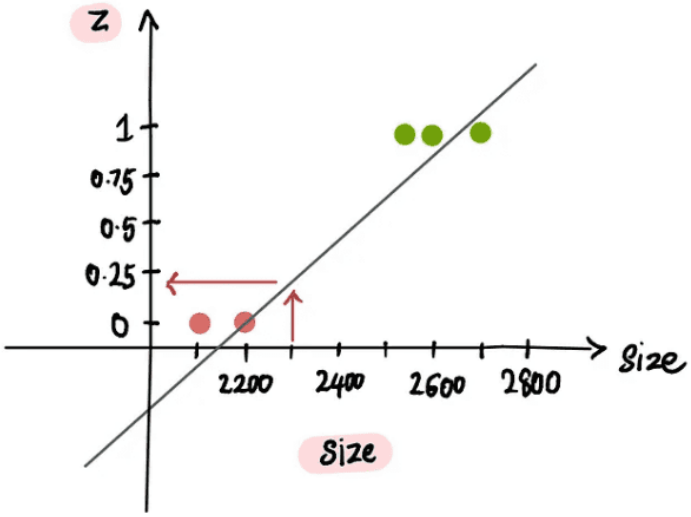
- Logistic regression allows us to estimate the probability of a given instance.
- This probability will range from 0 to 1.
- This probability will be capped by a decision boundary and associated with a discrete positive or negative class.



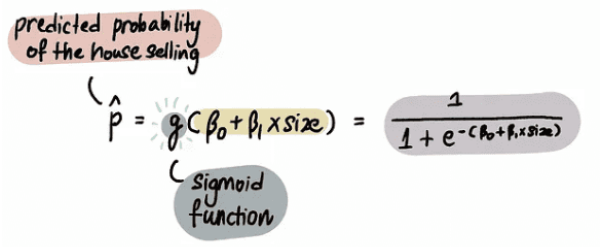
If this is plot ,



Now let plot as probabilities



So what is required to get this probability ?

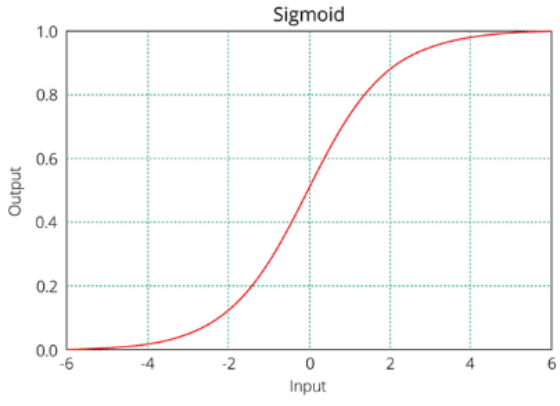


- **Odds** : Likelihood or probability of a favorable event happening
- **Log Odds** : Natural logarithm of the odds

Linear Regression Expression

y_pred=w0∗x0+w1∗x1+w2∗x2+w3∗x3+w4∗x4 … wn∗xn

1. $e^{\ln\left[\frac{p}{1-p}\right]} = e^{\beta_0 + \beta_1 \times size}$
2. $\frac{p}{1-p} = e^{\beta_0 + \beta_1 \times size}$
3. $p = e^{\beta_0 + \beta_1 \times size} - p \cdot e^{\beta_0 + \beta_1 \times size}$
4. $p\left[1 + e^{\beta_0 + \beta_1 \times size}\right] = e^{\beta_0 + \beta_1 \times size}$
5. $p = \frac{e^{\beta_0 + \beta_1 \times size}}{1 + e^{\beta_0 + \beta_1 \times size}}$
6. $p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \times size)}}$



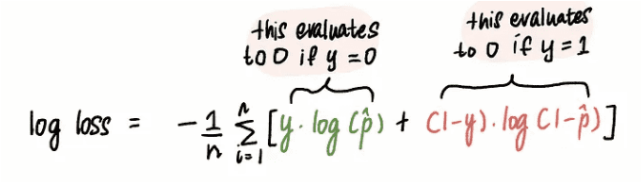
Sigmoid Function

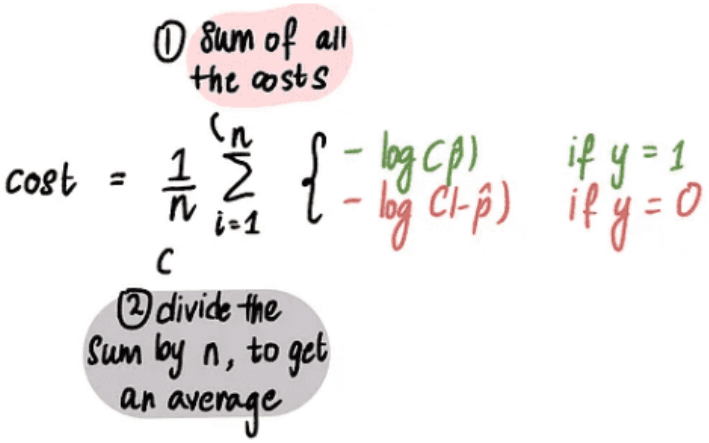
Inducing the Predictions

$ClassLabel = \{ 1 \quad g(y\hat{}) \geqslant 0.5 , \{ 0 \quad g(y\hat{}) < 0.5$

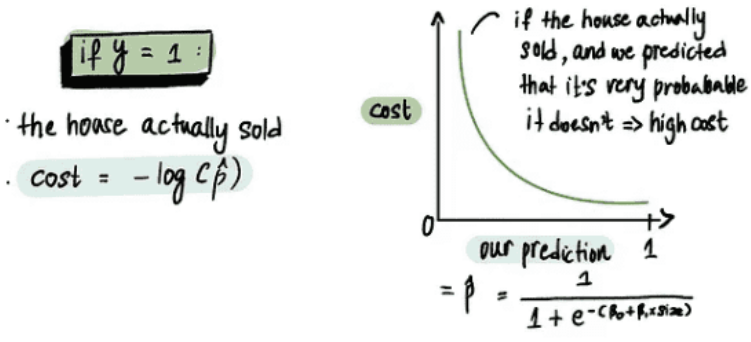
- Predict a class label of 1 if $g(y\hat{}) \geqslant 0.5$, and predict a class label of 0 if $g(y\hat{}) < 0.5$.
- We can also change the threshold of 0.5 to evaluate the model.

Negative Log Loss or Cross Entropy Error:

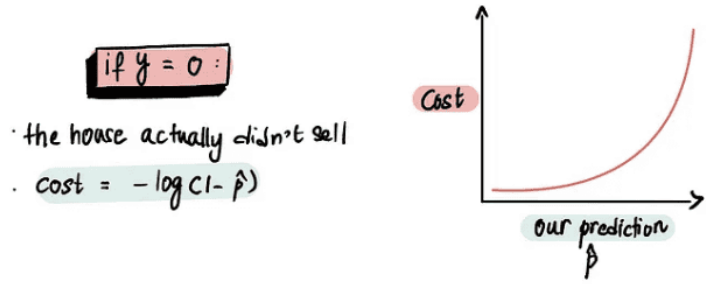




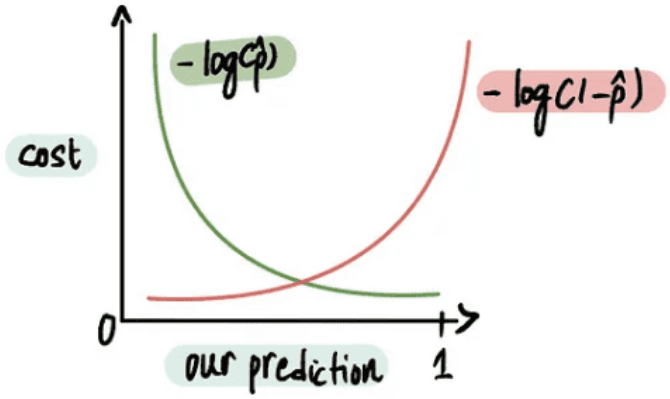
If `y = 1` and we predicted 1 (i.e., 100% probability it sold), there is no penalty. However, if we predicted 0 (i.e., 0% probability it didn't sell), then we get penalized heavily.



Similarly, if `y = 0` and we predicted a high probability of the house selling, we should be penalized heavily, and if we predicted a low probability of the house selling, there should be a lower penalty. The more off we are, the more it costs us.

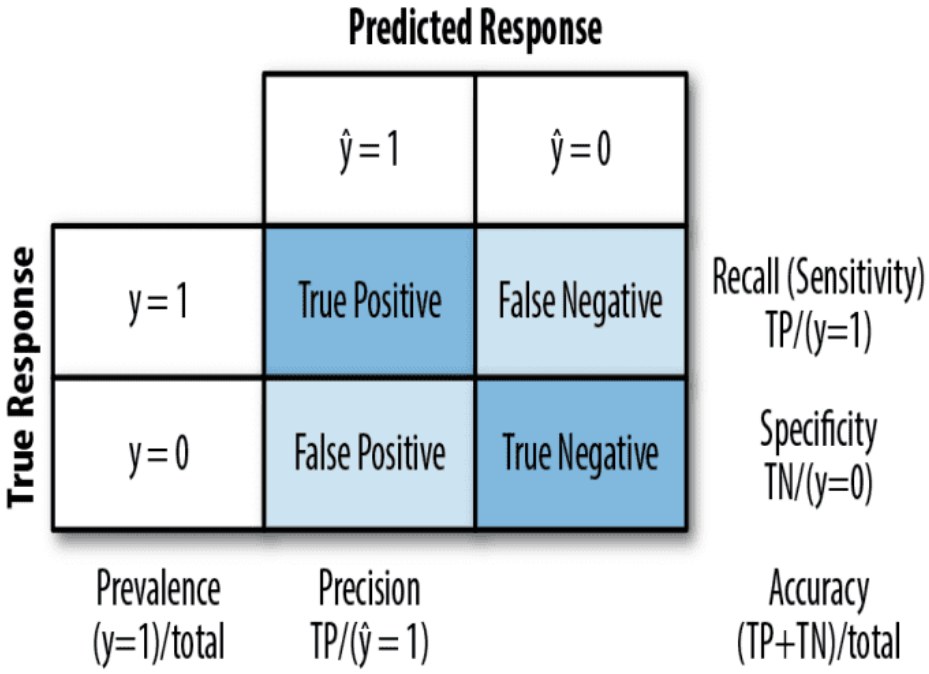


Combined graph of cost function:



How to Determine if the Model Is Good or Bad - *Evaluation Metrics*

- The model predicts a class as positive and it was positive (TP).
- The model predicts a class as positive and it was negative (FP).
- The model predicts a class as negative and it was negative (TN).
- The model predicts a class as negative and it was positive (FN).



- **Accuracy**: Measures the proportion of true results (both true positives and true negatives) among the total number of cases examined.
  - **Formula**: $\text{Accuracy} = TP+TN \,/\, FP+FN+TP+TN$
- **Precision**: Also known as the Positive Predictive Value (PPV), measures the proportion of true positives among all the positive results predicted by the model.
  - **Formula**: $\text{Precision} = TP \,/\, FP+TP$
- **Recall**: Also known as the True Positive Rate (TPR) or Sensitivity, measures the proportion of actual positives that are correctly identified as such.
  - **Formula**: $\text{Recall} = TP \,/\, FN + TP$
- **Specificity**: Also known as the True Negative Rate (TNR), measures the proportion of actual negatives that are correctly identified as such.
  - **Formula**: $\text{Specificity} = TN \,/\, FP+TN$
- **True Positive Rate (TPR)**, also known as Sensitivity or Recall, measures the proportion of actual positives that are correctly identified by the model.
  - **Formula**: $\text{TPR} = TP/\, FN+TP$
- **False Positive Rate (FPR)** measures the proportion of actual negatives that are incorrectly identified as positives by the model.

    - **Formula**: $\text{FPR} = FP / TN + FP$
- **True Negative Rate (TNR)**, also known as Specificity, measures the proportion of actual negatives that are correctly identified by the model.
    - **Formula**: $\text{TNR} = TN / FP + TN$
- **False Negative Rate (FNR)** measures the proportion of actual positives that are incorrectly identified as negatives by the model.
    - **Formula**: $\text{FNR} = TP / FN + FN$
- **F-Measure** F-measure is an attempt to combine both measures into a single value. Specifically, F-measure, also referred to as F_1, is computed as the weighted harmonic mean of precision and recall,
    - **F1 = 2 * (precision * recall) / (precision + recall)**

Scenario:

Imagine we have a dataset of 1000 emails, and our model has classified them as follows:

- True Positives (TP): 400 emails correctly identified as spam.
- False Positives (FP): 100 emails incorrectly identified as spam (they are actually not spam).
- True Negatives (TN): 450 emails correctly identified as not spam.
- False Negatives (FN): 50 emails incorrectly identified as not spam (they are actually spam).

Confusion Matrix:

	Predicted: Yes (Spam)	**Predicted: No (Not Spam)**
Actual: Yes (Spam)	TP = 400	FN = 50
Actual: No (Not Spam)	FP = 100	TN = 450

Calculations:

1. **Accuracy**:
    - $400+50+100+450400+450=1000850=0.85$ or 85%
    - Interpretation: The model correctly classified 85% of the emails.
2. **Precision** (Positive Predictive Value):
    - $400+100400=500400=0.8$ or 80%
    - Interpretation: When the model predicts an email is spam, it is correct 80% of the time.
3. **Recall** (True Positive Rate or Sensitivity):
    - $400+50400=450400=0.89$ or 89%
    - Interpretation: The model correctly identifies 89% of all actual spam emails.
4. **Specificity** (True Negative Rate):
    - $450+100450=550450=0.82$ or 82%
    - Interpretation: The model correctly identifies 82% of all actual non-spam emails.
5. **False Positive Rate**:
    - $100+450100=550100=0.18$ or 18%
    - Interpretation: 18% of the non-spam emails were incorrectly classified as spam.
6. **False Negative Rate**:
    - $400+5050=45050=0.11$ or 11%
    - Interpretation: 11% of the spam emails were missed by the model and incorrectly classified as not spam.

- F1 Score
    - Let's calculate the F1 Score:
    - $F1=2\times0.8x0.89 / 0.8+0.89$
    - Interpretation: 0.843. This indicates a strong balance between Precision and Recall, suggesting the model is relatively robust in terms of both correctly identifying spam emails and minimizing the number of non-spam emails incorrectly classified as spam

    -

Interpretations:

- **Accuracy** shows a general effectiveness of the model across both classes but doesn't distinguish between the types of errors.
- **Precision** indicates that when the model predicts an email as spam, it is fairly reliable, though there is still a 20% chance it could be wrong.
- **Recall** shows the model is quite good at catching spam emails, though it misses about 11% of them.
- **Specificity** tells us the model is also reasonably good at recognizing non-spam emails, but it mistakenly flags some as spam.
- **FPR** and **FNR** provide insight into the error rates for each class, indicating areas where the model's performance could be improved.

AUC-ROC

An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

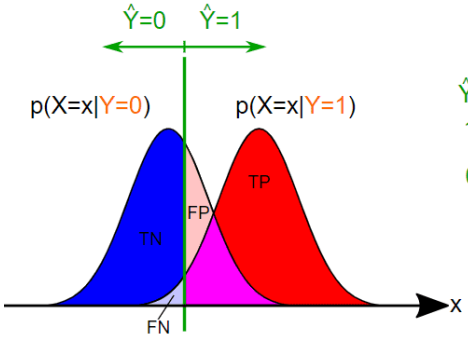
- True Positive Rate
- False Positive Rate

True Positive Rate (**TPR**) is a synonym for recall and is therefore defined as follows:

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

False Positive Rate (**FPR**) is defined as follows:

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



Ŷ \ Y	1	0
1	TP	FP
0	FN	TN