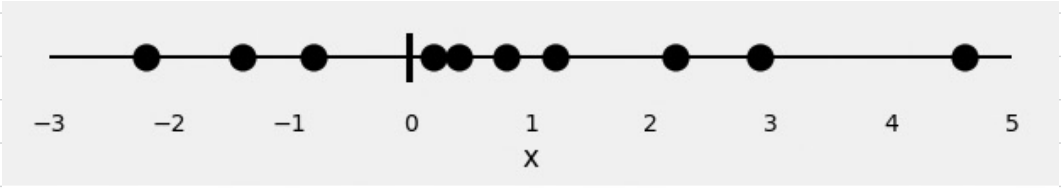
# Logistic Regression

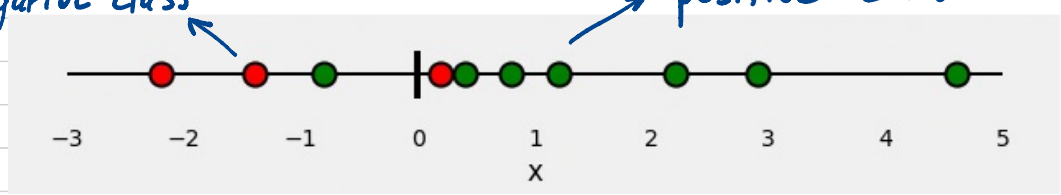
feature : x

x = [-2.2, -1.4, -0.8, 0.2, 0.4, 0.8, 1.2, 2.2, 2.9, 4.6] → 'medical' measurement (e.g. cholesrol)




The data
label : Red or Green
↳ no heart disease
↳ heart disease

'negative' class
'positive' class




Goal: Binary classification (supervised learning)

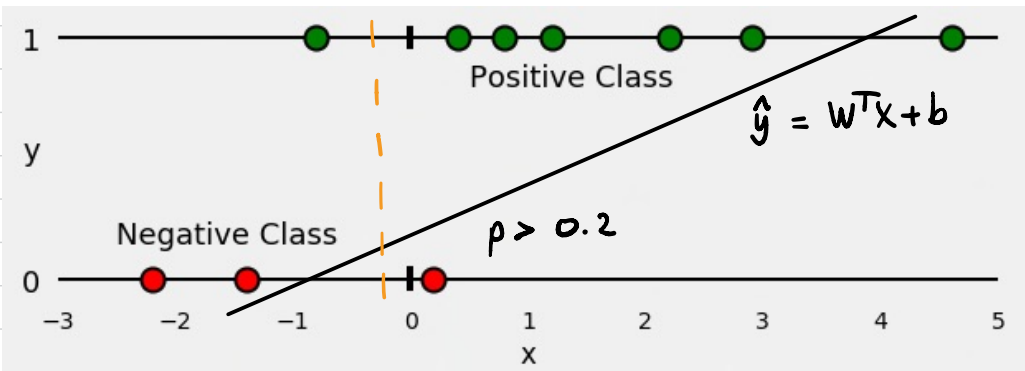
What is the probability of the point being Green?

Ideally, Green points would have a probability of 1.0 (of being Green), while Red points should have a probability of 0.0 (of being Green)

If we fit a model to perform this classification, it will predict a probability of being Green.

Linear regression → Bad fit

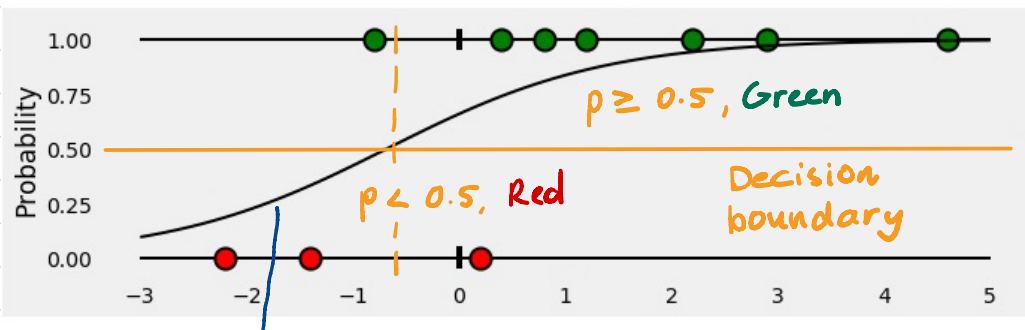
Threshold will change with new data instance, especially with outliers




Logistic Regression

Sigmoid function

Let's train a logistic regression model to classify our points:




↳ Sigmoid function

By default the decision boundary or threshold is set at 0.5. In our example the threshold seems to be placed a bit higher around, 0.75. Meaning, that an $\hat{y} \geq 0.75$ will result in the point being classified as Green, and an $\hat{y} < 0.75$ will result in the point being classified as Red.

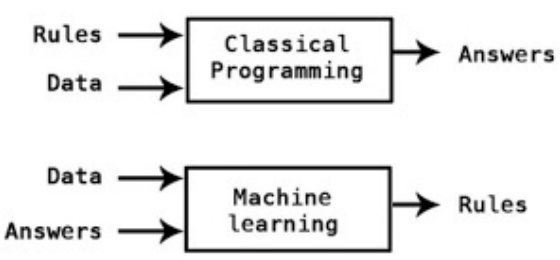
Glossary :

$y$ vs $\hat{y}$

$y$ = 'actual'/'true' label

$\hat{y}$ or sometimes $y_{pred}$ = predicted label (by model)

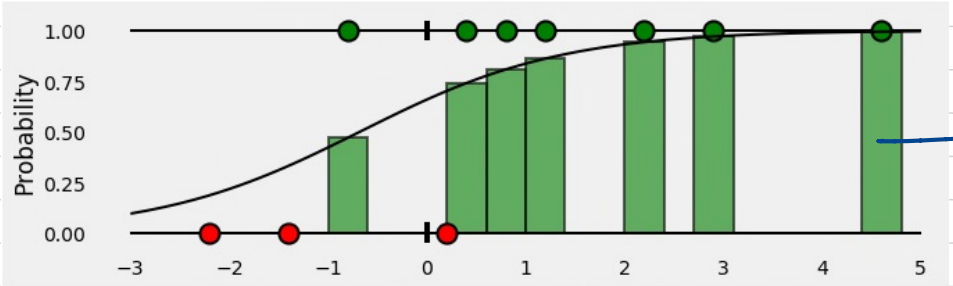
↳ In a supervised learning we already have the 'answers' (a.k.a labels). The task of our model is to find the rules. Remember:




The model is performing well when the difference between $y$ and $\hat{y}$ is small → model is able to find pattern in data and create rules to capture it.

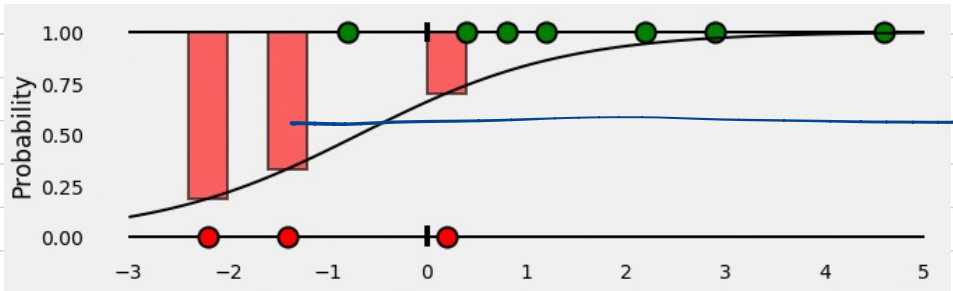
The difference between $y$ and $\hat{y}$ is often expressed in term of loss (e.g. log loss).

## Probabilities




→ p (class = Green)

The probabilities of classifying the points in the positive, Green, class correctly.




→ 1 − p(class = Green)

The probabilities of classifying the points in the negative, Red, class correctly.

## Add sigmoid to linear regression equation

With logistic regression we will transform the output using the sigmoid function to return a probability value between 0.0 and 1.0 :

$z = W^T x + b$ → also known as the logit or log odds

$\sigma = \frac{1}{1 + e^{-z}}$ → p(class = Green)

↳ Ln or $\log_e$ (a.k.a natural logarithm)

$\hat{y} = \frac{1}{1 + e^{-W^T x + b}}$

example :

$z = 1$

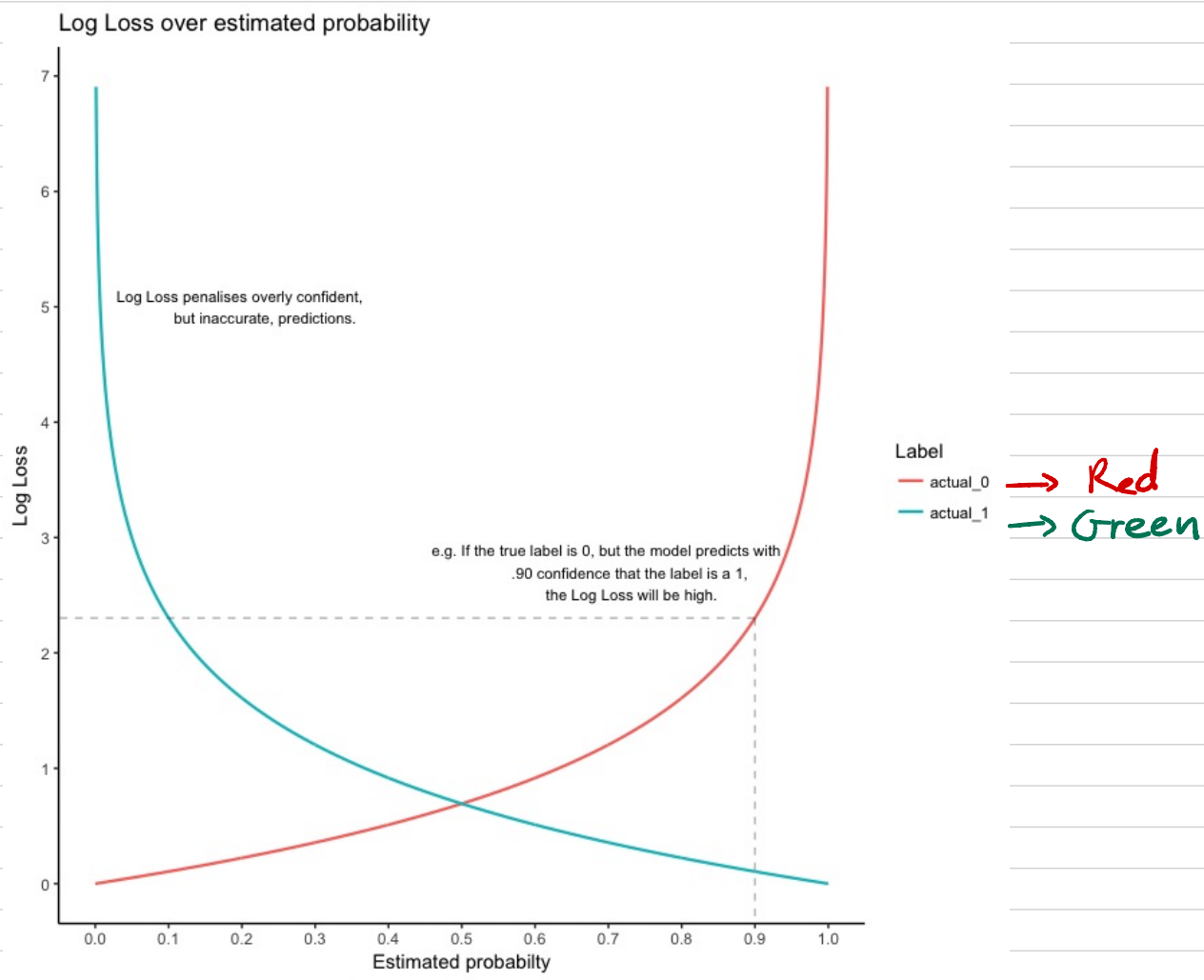
$\hat{y} = \frac{1}{1 + e^{-1}}$

$\hat{y} = 0.73$ → the probability of point being Green, with a logit of 1 = 73%.

→ the probability of the point being Red, with a logit of 1 = 27%.

## Loss / cost function

A cost function (e.g. Binary cross en / log loss) is used to evaluate how good or bad the predicted probabilities are → it returns high values for bad predictions, and low values for predictions.




$\hookrightarrow \hat{y}$

Cost function:

* Error representation in machine learning
* Shows how our model is predicting compared to the original dataset ('true' labels)

$$\text{Log loss} = -\frac{1}{m}\sum_{i=1}^{m}\left[\overbrace{y\log(\hat{y})}+\overbrace{(1-y)\log(1-\hat{y})}\right]$$

$\rightarrow$ log loss for 1 observation

Remember $\hat{y} = \sigma(w^T x + b)$

$y = 0$ (Red) log loss $= -1 * \log(1-\hat{y})$

$y = 1$ (Green) log loss $= -\log(\hat{y})$

When difference between $y$ and $\hat{y}$ is large $\rightarrow$ loss is high

When difference between $y$ and $\hat{y}$ is small $\rightarrow$ loss is low

Confusion matrix

Confusion Matrix :

* Matrix of size 2x2 for binary classification
  - ↳ predicted values
  - ↳ actual values

performance evaluation

* Enables you to evaluate the performance of your binary classification model

| | | ACTUAL (y) | |
|---|---|---|---|
| | | Negative | Positive |
| PREDICTION ($\hat{y}$) | Negative | TRUE NEGATIVE | FALSE NEGATIVE |
| | Positive | FALSE POSITIVE | TRUE POSITIVE |

Case study 1: Breast cancer

1. Case study : The detection of breast cancer

what is the probability of a breast tumor to be maglignant?

| | | ACTUAL | |
|---|---|---|---|
| | | Negative Benign | Positive malignant |
| PREDICTION | Negative Benign | 60 | 8 |
| | Positive malignant | 22 | 10 |

good prediction =

Bad prediction =

**True Positive (TP)** — model correctly predicts the positive class (prediction and actual both are positive). In the above example, **10 people** who have tumors are predicted positively by the model.
**True Negative (TN)** — model correctly predicts the negative class (prediction and actual both are negative). In the above example, **60 people** who don't have tumors are predicted negatively by the model.
**False Positive (FP)** — model gives the wrong prediction of the negative class (predicted-positive, actual-negative). In the above example, **22 people** are predicted as positive of having a tumor, although they don't have a tumor. FP is also called a **TYPE I** error.
**False Negative (FN)** — model wrongly predicts the positive class (predicted-negative, actual-positive). In the above example, **8 people** who have tumors are predicted as negative. FN is also called a **TYPE II** error.

performance metric: Accuracy

When we have a balanced dataset we can use the metric accuracy to measure the performance of our model:

Accuracy : $\frac{T_p + T_n}{T_p + F_p + T_n + F_n}$ → High values for $T_p$ and $T_n$ → high accuracy!

From all the classes (positive and negative), how many of them are predicted correctly.

When the dataset is unbalanced, you want to use other performance metrics, such as precision or recall, instead of accuracy.

Imagine having the following data distribution for y:

| class: | # instances: | Description |
|---|---|---|
| Red | 2 → $T_p$ | Malignant tumor |
| Green | 2000 → $T_n$ | Benign tumor |

You fit an logistic regression model to the data, which outputs an accuracy score of 0.96. Sounds pretty good right? But there is a catch... The model is basically learning the bias: it is very unlikely that the data point is Red.

performance metric: Recall

In situation when you want to minimize the number of $F_n$s (e.g. Breast cancer example), it is good practice to reduce the bias by decreasing the threshold value. This will increase the number of $F_p$s, but lower the number $F_n$s → Type II error

Recall (a.k.a $T_p$ rate): $\frac{T_p}{T_p + F_n}$

Out of all the total positives, what percentage are predicted positive.

Case study 2: spam emails

2. Case study: The detection of spam emails

What is the probability of an email to be no-spam?

Imagine having the following data distribution for y:

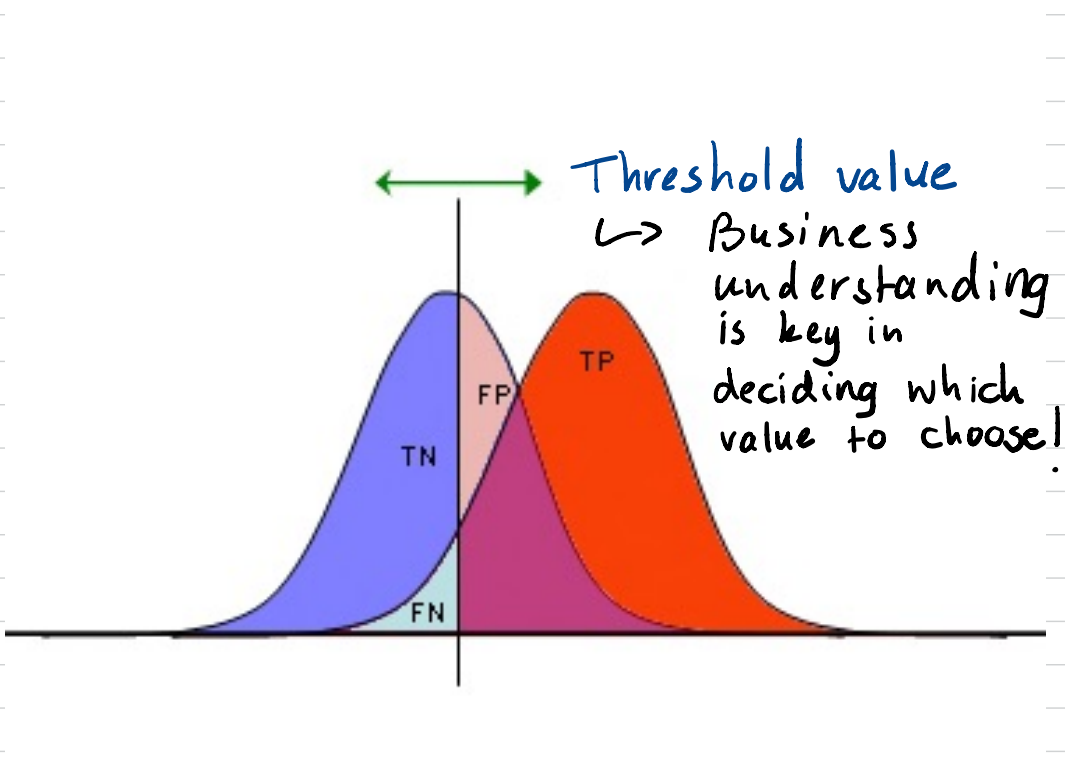
| class: | # instances: | Description |
|---|---|---|
| Red | 2 → $T_p$ | no-spam |
| Green | 2000 → $T_n$ | Spam |

performance metric: precision

In situation when you want to minimize the number of $F_p$s (e.g. spam email example), it is good practice to reduce the bias by increasing the threshold value. This will increase the number of $F_n$s, but lower the number $F_p$s → Type I error

precision : $\frac{Tp}{Tp + Fp}$

Out of all the positive predicted, what percentage is truly positive.




Trade off

Trade off : precision vs. recall

performance metric: F1 score

There are also performance metrics that balance precision and recall. For example :

$$F_1 : 2 * \frac{precision * recall}{precision + recall}$$