Machine Learning with Python
Logistic Regression

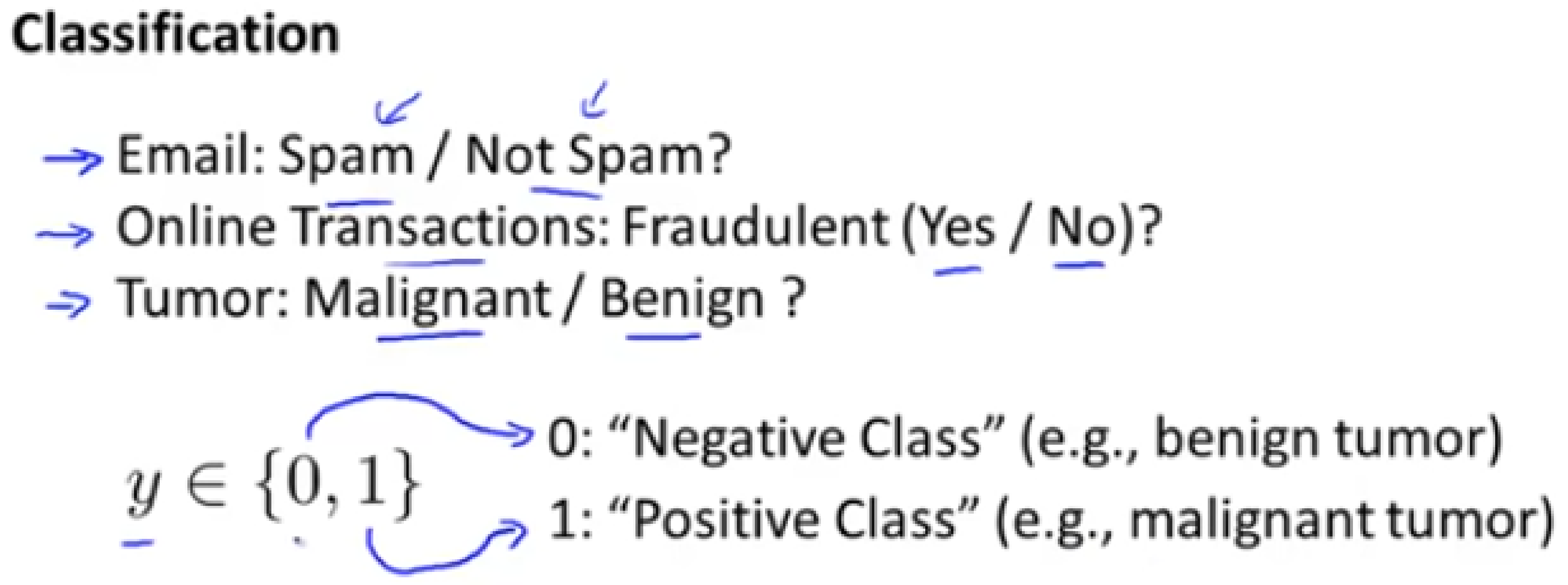
Logistic Regression

- Extension of linear regression where the dependent variable is categorical and not continuous.
- It is a prediction done with categorical variable. It predicts the probability of the outcome variable being true(Y=1) as a function of dependent variables.

In other words, the interest is in predicting which of **two possible events** are going to happen given certain other information

- Is he a prospective customer ie., likelihood of purchase vs non-purchase ?
- Is he a defaulter for bill payment ie., credit worthiness
- Low claim or high risk for an insurance
- Medicine: is it going to work or does not work

Logistic Regression



Classification

→ Email: Spam / Not Spam?

→ Online Transactions: Fraudulent (Yes / No)?

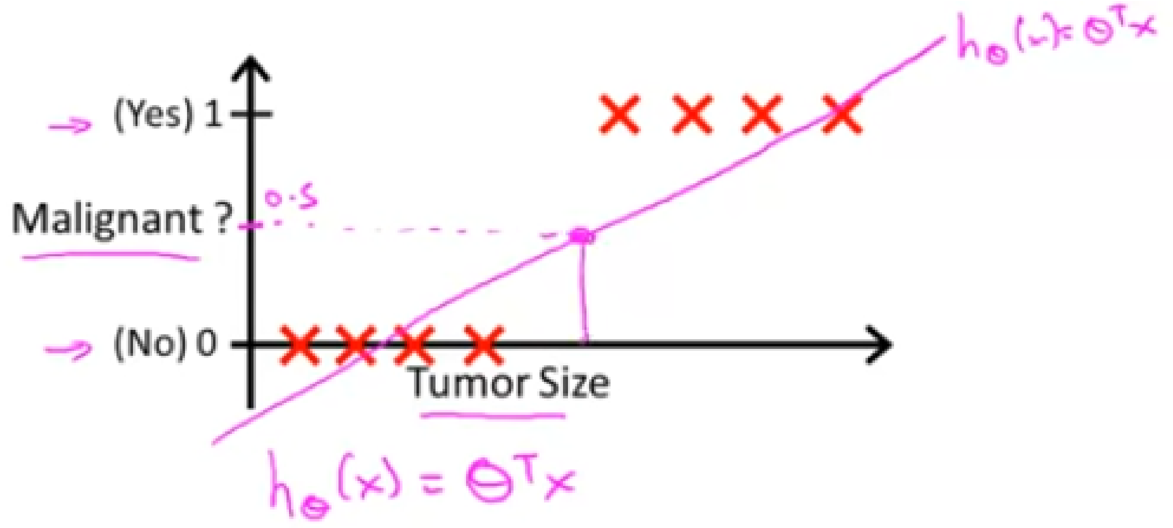
→ Tumor: Malignant / Benign ?

$y \in \{0, 1\}$

0: "Negative Class" (e.g., benign tumor)

1: "Positive Class" (e.g., malignant tumor)

Logistic Regression

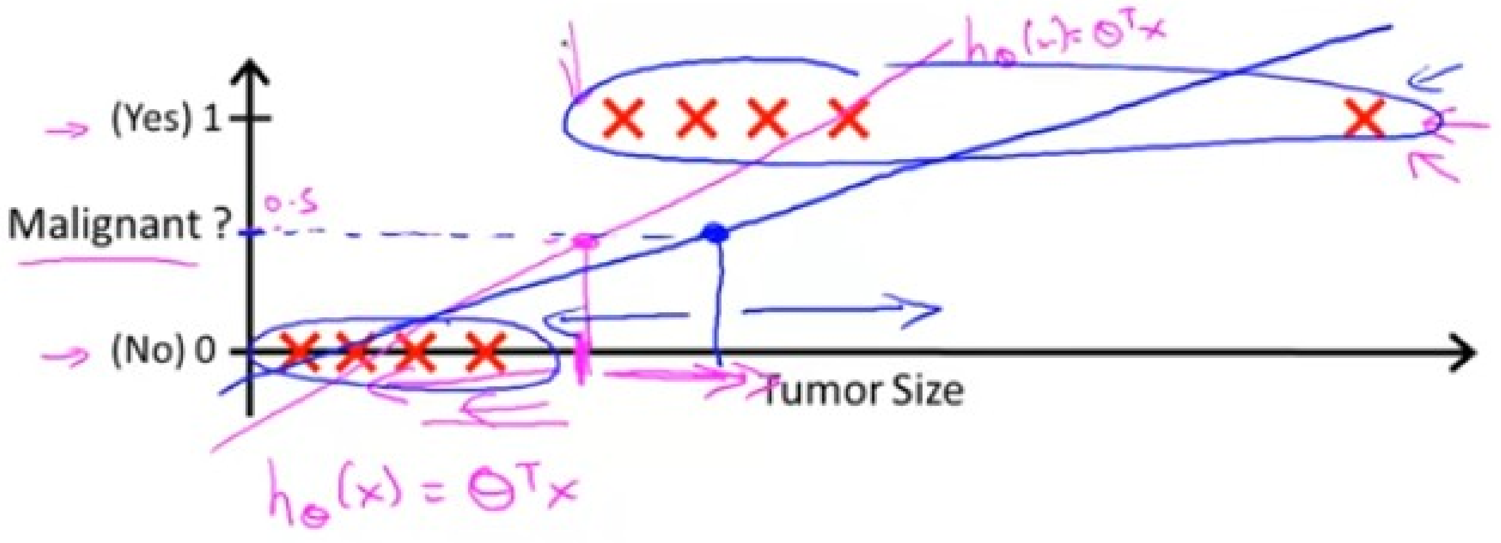


→ Threshold classifier output $h_\theta(x)$ at 0.5:

→ If $h_\theta(x) \geq 0.5$, predict "$\underline{y = 1}$"

If $h_\theta(x) < 0.5$, predict "$\underline{y = 0}$"

Logistic Regression



→ Threshold classifier output $h_\theta(x)$ at 0.5:

→ If $h_\theta(x) \geq 0.5$, predict "y = 1"

If $h_\theta(x) < 0.5$, predict "y = 0"

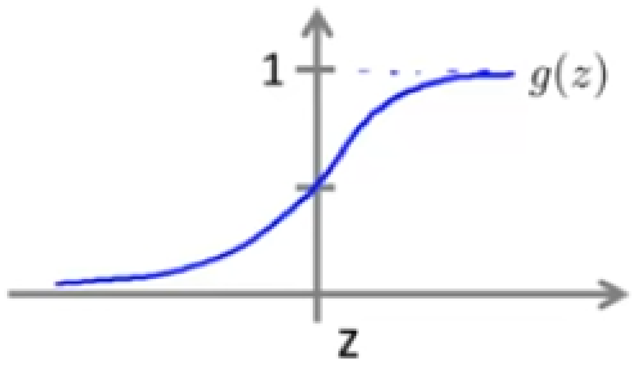
Logistic Regression

Logistic regression

$\rightarrow h_\theta(x) = g(\theta^T x) = P(y=1|x;\theta)$

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

Suppose predict "$\underline{y = 1}$" if $\underline{h_\theta(x) \overset{\swarrow}{\geq} 0.5}$



predict "$\underline{y = 0}$" if $\underline{h_\theta(x) \overset{\swarrow}{<} 0.5}$

Logistic Regression

Logistic regression

$\rightarrow h_\theta(x) = g(\theta^T x) = \underline{P(y=1|x;\theta)}$

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

Suppose predict $\boxed{\text{“}y = 1\text{“ if } h_\theta(x) \geq 0.5}$

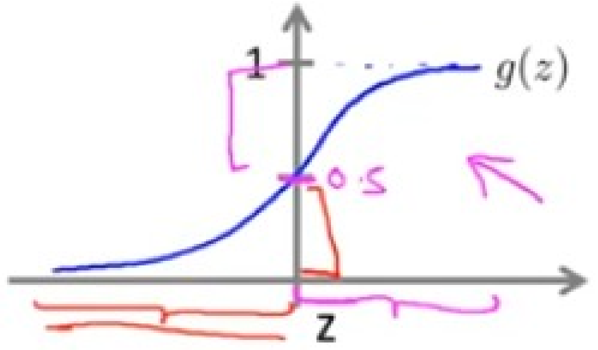
$\rightarrow \theta^T x \geq 0$

predict $\boxed{\text{“}y = 0\text{“ if } h_\theta(x) < 0.5}$

$h_\theta(x) = g(\underline{\theta^T x})$

$\rightarrow \theta^T x < 0$

$\underline{g(z) < 0.5}$



$g(z) \geq 0.5$

when $z \geq 0$

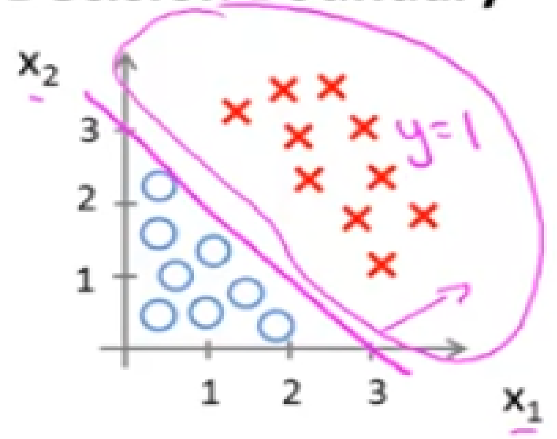
$h_\theta(x) = \underline{g(\theta^T x)} \geq 0.5$

whenever $\theta^T x \geq 0$

$\uparrow$
z

Logistic Regression

Decision Boundary



$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix} \leftarrow$$

$$\rightarrow h_\theta(x) = g(\underset{\underset{-3}{\parallel}}{\theta_0} + \underset{\underset{1}{\parallel}}{\theta_1} x_1 + \underset{\underset{1}{\parallel}}{\theta_2} x_2)$$

x_1, x_2

Predict "$y = 1$" if $\underbrace{-3 + x_1 + x_2 \geq 0}_{\theta^T x}$

$\rightarrow x_1 + x_2 \geq 3$ $\qquad x_1-$

$x_1 + x_2 = 3$

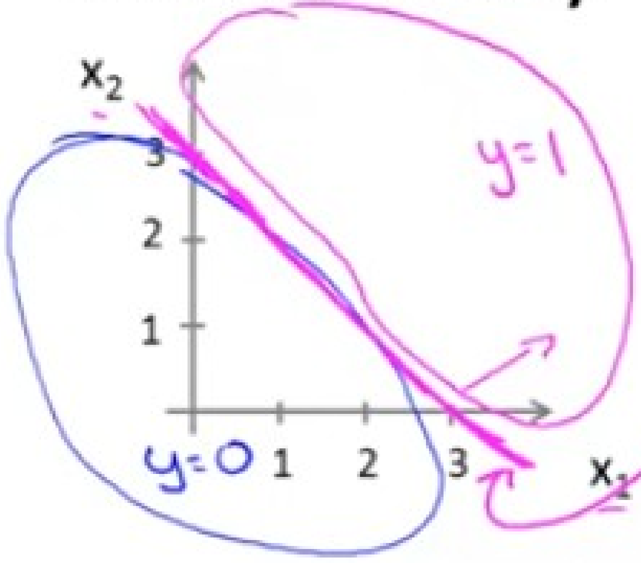
Logistic Regression

Derivative of sigmoid function

https://www.youtube.com/watch?v=CE03E80wbRE

Logistic Regression

Decision Boundary



$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix} \leftarrow$$

$$\rightarrow \boxed{h_\theta(x) = g(\underset{-3}{\theta_0} + \underset{1}{\theta_1} x_1 + \underset{1}{\theta_2} x_2)}$$

Decision boundary

Predict "$y = 1$" if $\underbrace{-3 + x_1 + x_2}_{\theta^T x} \geq 0$

$\rightarrow x_1 + x_2 \geq 3$

x_1, x_2

$\rightarrow h_\theta(x) = 0.5$

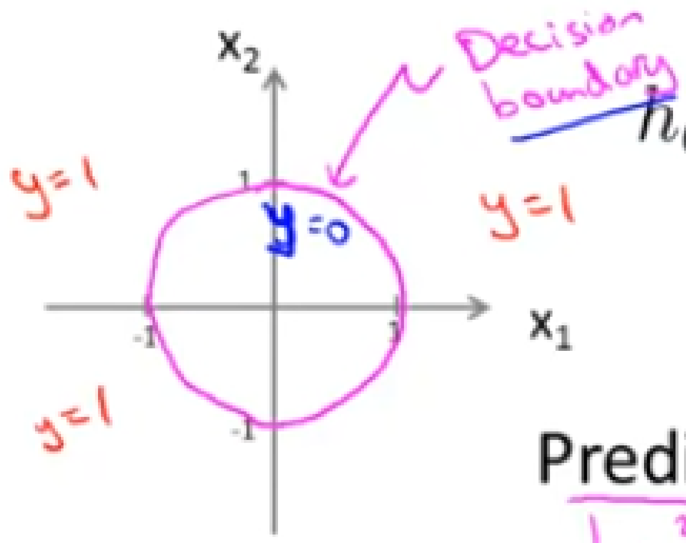
$\boxed{x_1 + x_2 = 3}$

$x_1 + x_2 < 3$

$\rightarrow y = 0$

Logistic Regression

Non-linear decision boundaries



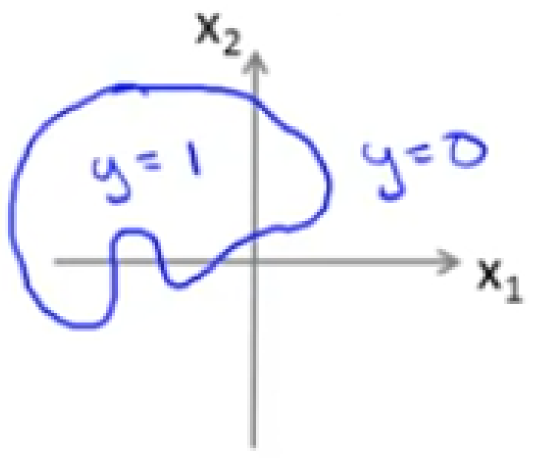
$$h_\theta(x) = g(\overset{-1}{\theta_0} + \overset{0}{\theta_1} x_1 + \overset{0}{\theta_2} x_2 + \underset{1}{\theta_3} x_1^2 + \underset{1}{\theta_4} x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Predict "$y = 1$" if $\underbrace{-1 + x_1^2 + x_2^2 \geq 0}_{x_1^2 + x_2^2 \geq 1}$

$\boxed{x_1^2 + x_2^2 = 1}$

Logistic Regression



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 \underline{x_1^2} + \theta_4 \underline{x_1^2 x_2} + \theta_5 \underline{x_1^2 x_2^2} + \theta_6 \underline{x_1^3 x_2} + \ldots)$$

Logistic Regression

Training set: $\underline{\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}}$

$\underline{\text{m examples}}$ $\quad x \in \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}}\ \mathbb{R}^{n+1} \quad \underline{x_0 = 1}, y \in \{0, 1\}$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

How to choose parameters θ ?

Logistic Regression

Cost function

→ Linear regression: $J(\theta) = \boxed{\frac{1}{m}\sum_{i=1}^{m} \cancel{\underline{\frac{1}{2}}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2} \quad \text{cost}(h_\theta(x^{(i)}), y)}$

$$\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Logistic Regression

Cost function

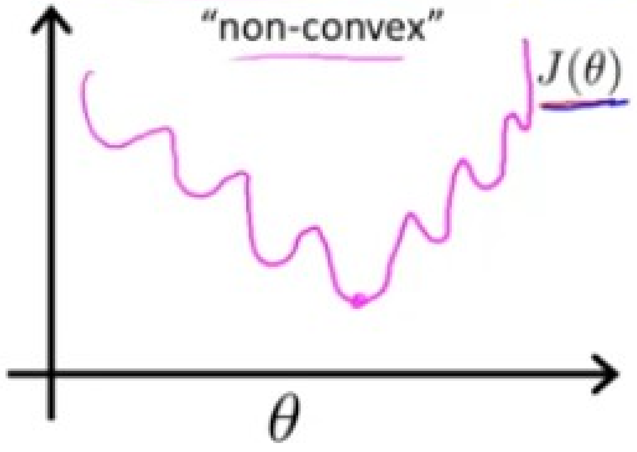
→ ~~Linear~~ regression: $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \cancel{\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}$

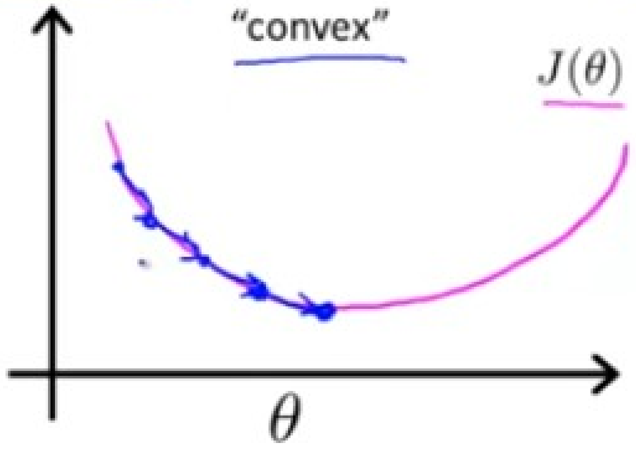
logistic

$\text{cost}(h_\theta(x^{(i)}), y)$

→ $\text{Cost}(h_\theta(x), y) = \frac{1}{2}\left(h_\theta(x) - y\right)^2$ ←

$\frac{1}{1+e^{-\theta^T x}}$

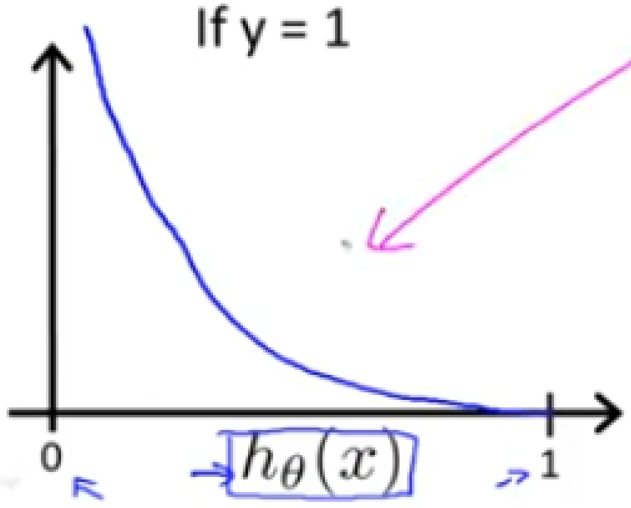


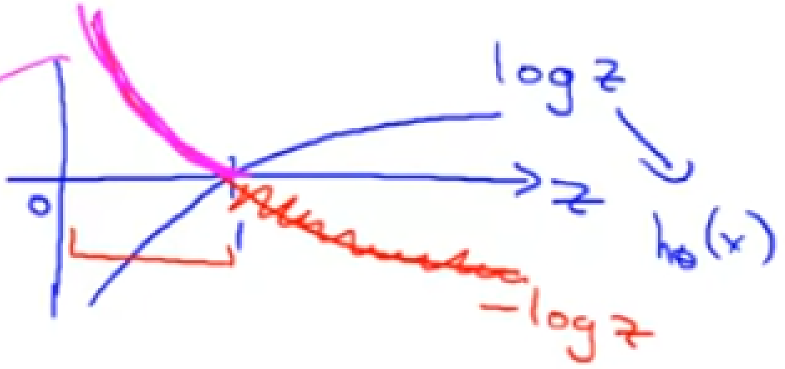


Logistic Regression

Logistic regression cost function

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

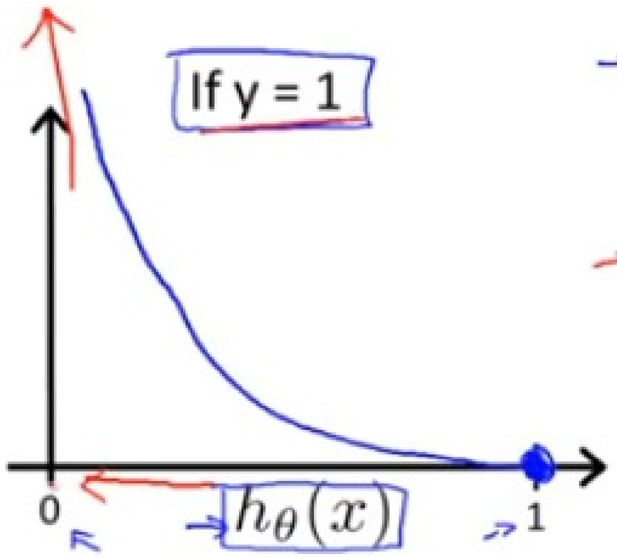




Logistic Regression

Logistic regression cost function

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

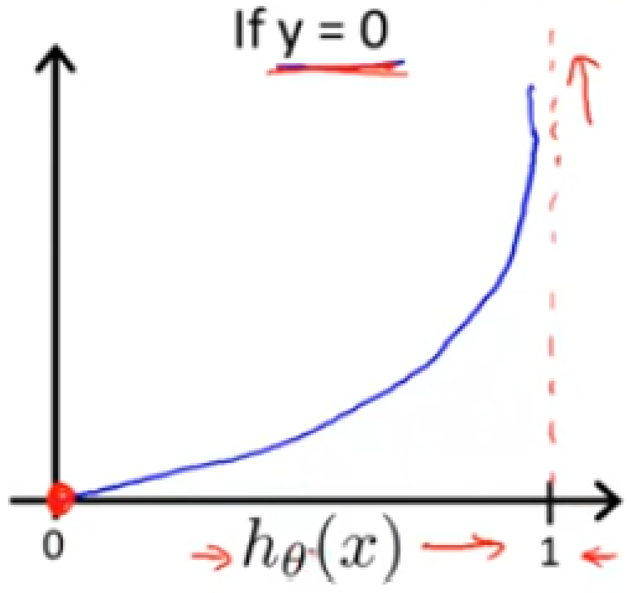


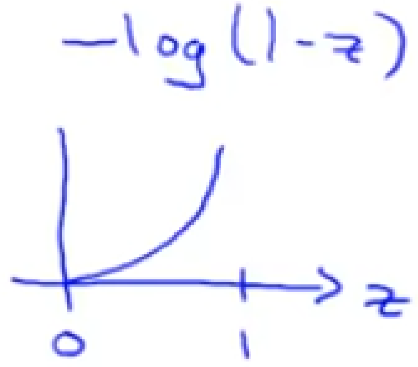
- Cost $= 0$ if $y = 1, h_\theta(x) = 1$
  But as $h_\theta(x) \to 0$
  $Cost \to \infty$
- Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x;\theta) = 0$), but $\boxed{y = 1}$, we'll penalize learning algorithm by a very large cost.

Logistic Regression

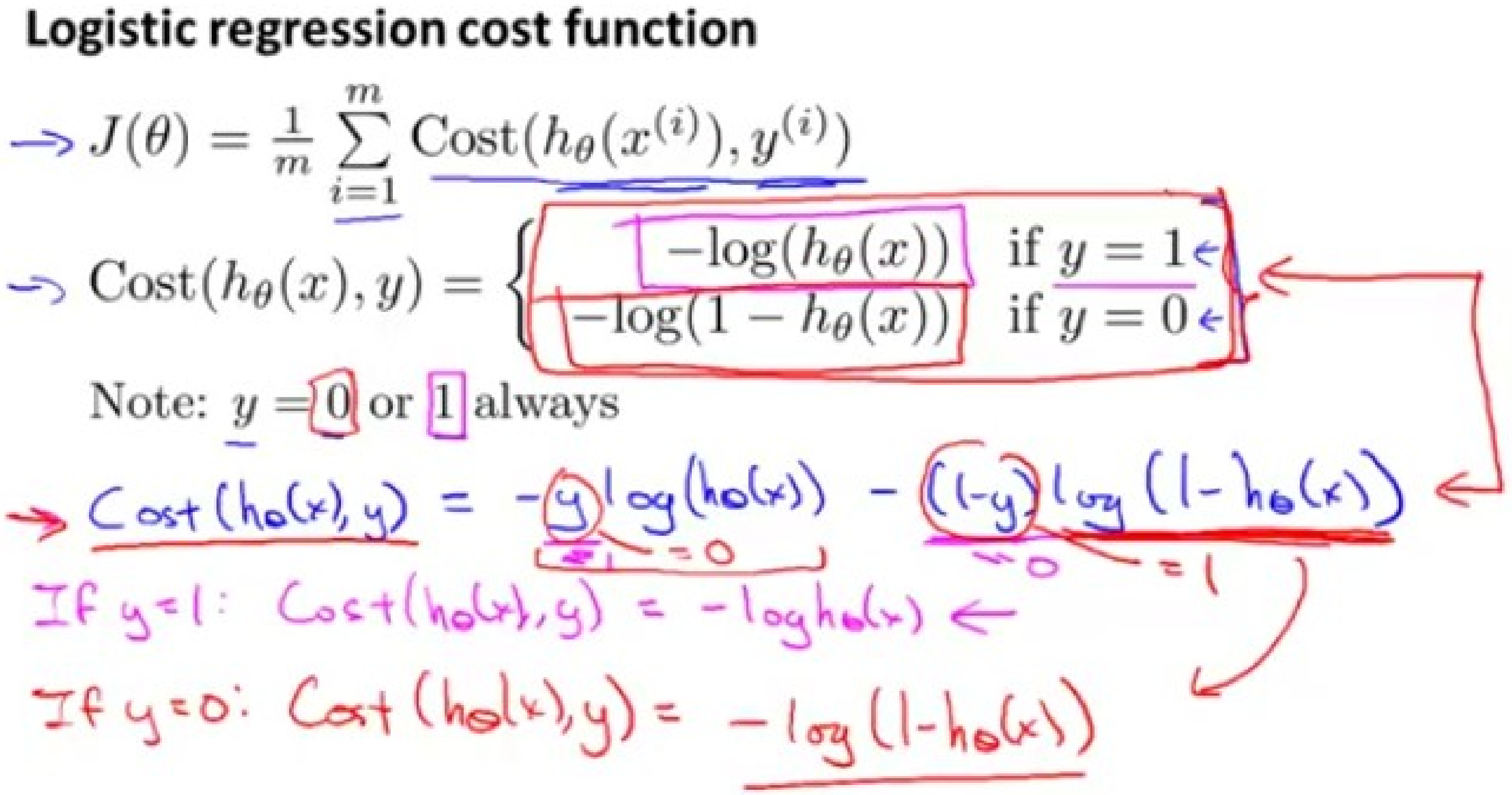
Logistic regression cost function

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





Logistic Regression



Logistic regression cost function

$$\to J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

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

Note: $y = \boxed{0}$ or $\boxed{1}$ always

$$\to \text{Cost}(h_\theta(x), y) = \underbrace{-y \log(h_\theta(x))}_{=0} - \underbrace{(1-y)}_{=0} \underbrace{\log(1 - h_\theta(x))}_{=1}$$

If $y=1$: $\text{Cost}(h_\theta(x), y) = -\log h_\theta(x) \leftarrow$

If $y=0$: $\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x))$

Logistic Regression

IT Bodhi Nurturing Talent

Logistic regression cost function

$$J(\theta) = \underline{\frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})}$$

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

To fit parameters θ:

$$\boxed{\underline{\min_\theta J(\theta)} \quad \text{Get } \underline{\Theta}}$$

To make a prediction given new $\underline{x}$:

Output $\underline{h_\theta(x)} = \frac{1}{1+e^{-\theta^T x}}$ $\qquad$ $\underline{p(y=1 \mid x;\Theta)}$

Logistic Regression
How cost function derived??

Check the Video

https://www.youtube.com/watch?v=CE03E80wbRE

Logistic Regression

Gradient Descent

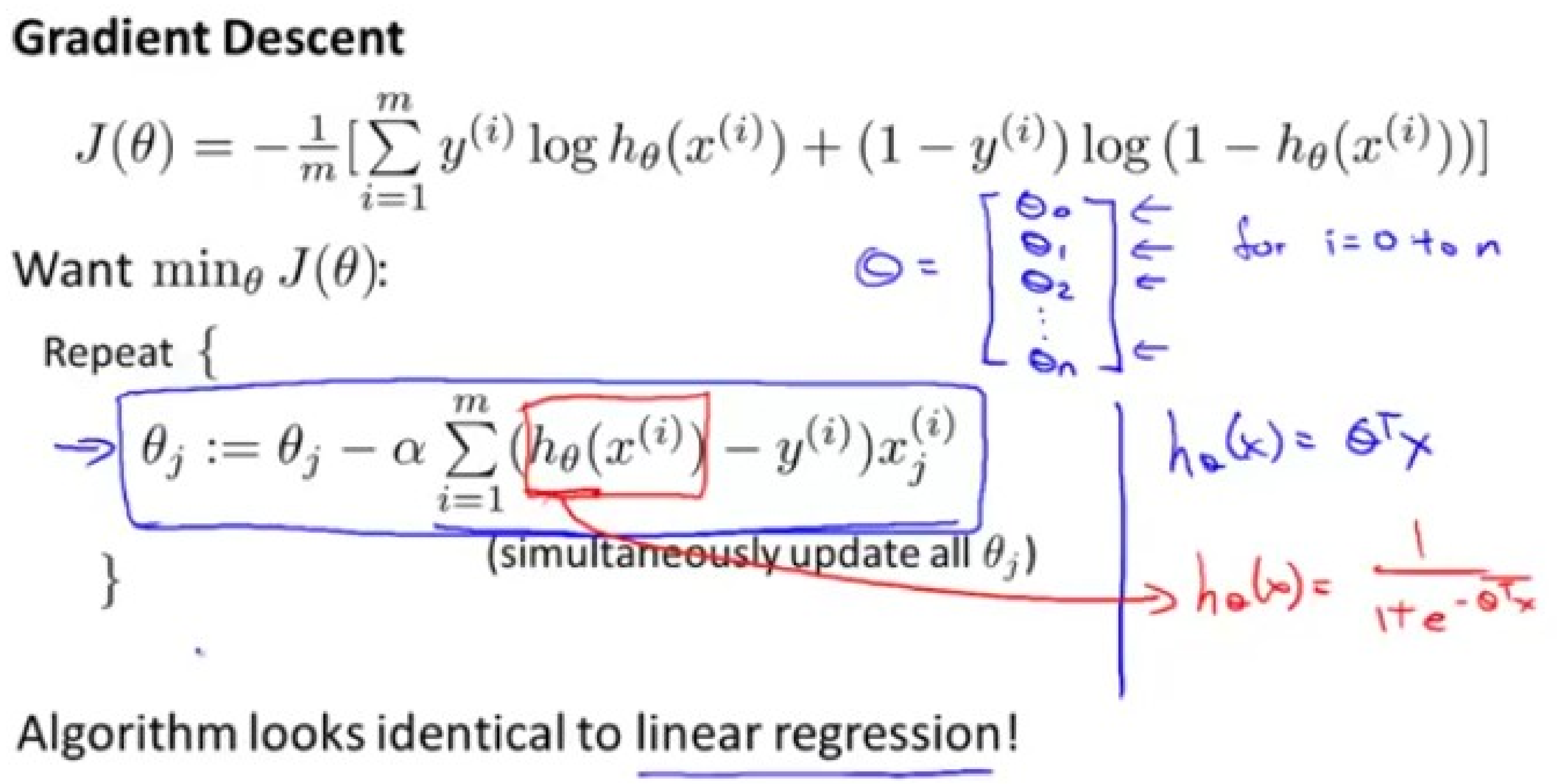
$$\rightarrow J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1-h_\theta(x^{(i)}))\right]$$

Want $\underline{\min_\theta J(\theta)}$:

```
Repeat {
        $\theta_j := \theta_j - \alpha \underline{\frac{\partial}{\partial\theta_j} J(\theta)}$
                        (simultaneously update all $\theta_j$)
}
```

$$\frac{\partial}{\partial\theta_j} J(\theta) = \underline{\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}}$$

Logistic Regression



Gradient Descent

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

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \begin{matrix} \leftarrow \\ \leftarrow \\ \leftarrow \\ \\ \leftarrow \end{matrix} \quad \text{for } i = 0 \text{ to } n$$

Want $\min_\theta J(\theta)$:

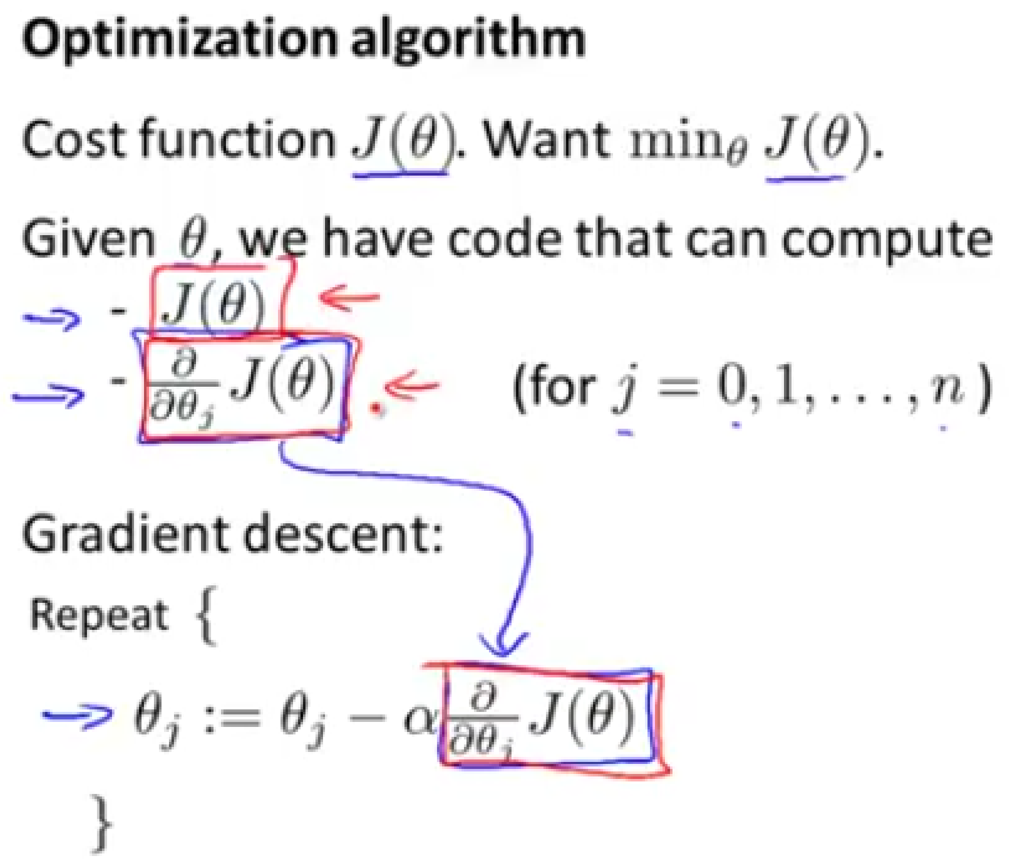
```
Repeat {
    $\rightarrow \boxed{\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (\boxed{h_\theta(x^{(i)})} - y^{(i)}) x_j^{(i)}}$
        (simultaneously update all $\theta_j$)
}
```

$$h_\theta(x) = \theta^T x$$

$$\rightarrow h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Algorithm looks identical to <u>linear regression</u>!

Logistic Regression



Optimization algorithm

Cost function $J(\theta)$. Want $\min_\theta J(\theta)$.

Given θ, we have code that can compute

$\rightarrow$ - $\boxed{J(\theta)}$ $\leftarrow$

$\rightarrow$ - $\boxed{\frac{\partial}{\partial\theta_j}J(\theta)}$ $\leftarrow$ (for $j = 0, 1, \ldots, n$)

Gradient descent:

```
Repeat {
    → $\theta_j := \theta_j - \alpha \boxed{\frac{\partial}{\partial\theta_j}J(\theta)}$
}
```

Logistic Regression

Optimization algorithm

Given θ, we have code that can compute

- $J(\theta)$ $\leftarrow$
- $\frac{\partial}{\partial\theta_j} J(\theta)$ $\leftarrow$ (for $j = 0, 1, \ldots, n$)

Optimization algorithms:

$\rightarrow$ - Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages:

- No need to manually pick $\underline{\alpha}$
- Often faster than gradient descent.

Disadvantages:

- More complex

Logistic Regression

Multiclass classification

Email foldering/tagging: Work, Friends, Family, Hobby

$\uparrow \quad \uparrow \quad \uparrow \quad \uparrow$

$y=1 \quad y=2 \quad y=3 \quad y=4$

Medical diagrams: Not ill, Cold, Flu

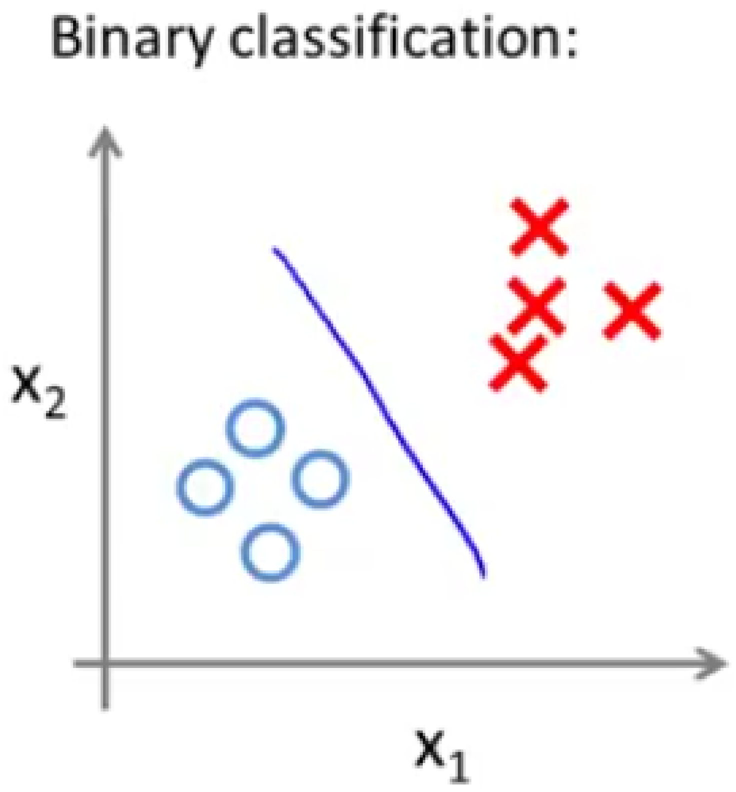
$y=1 \quad 2 \quad 3$

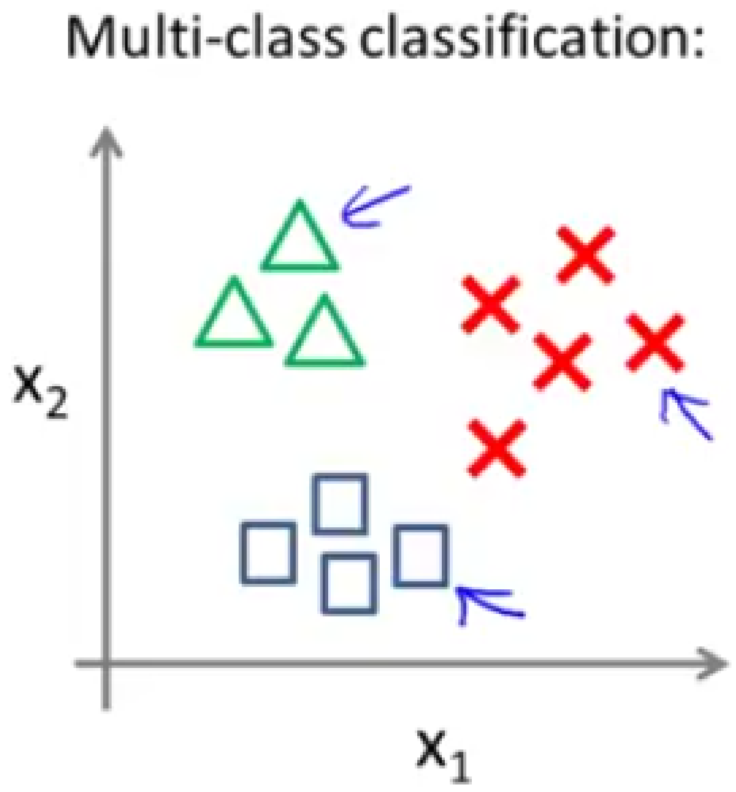
Weather: Sunny, Cloudy, Rain, Snow

$y=1 \quad 2 \quad 3 \quad 4$

$0 \quad 1 \quad 2 \quad 3$

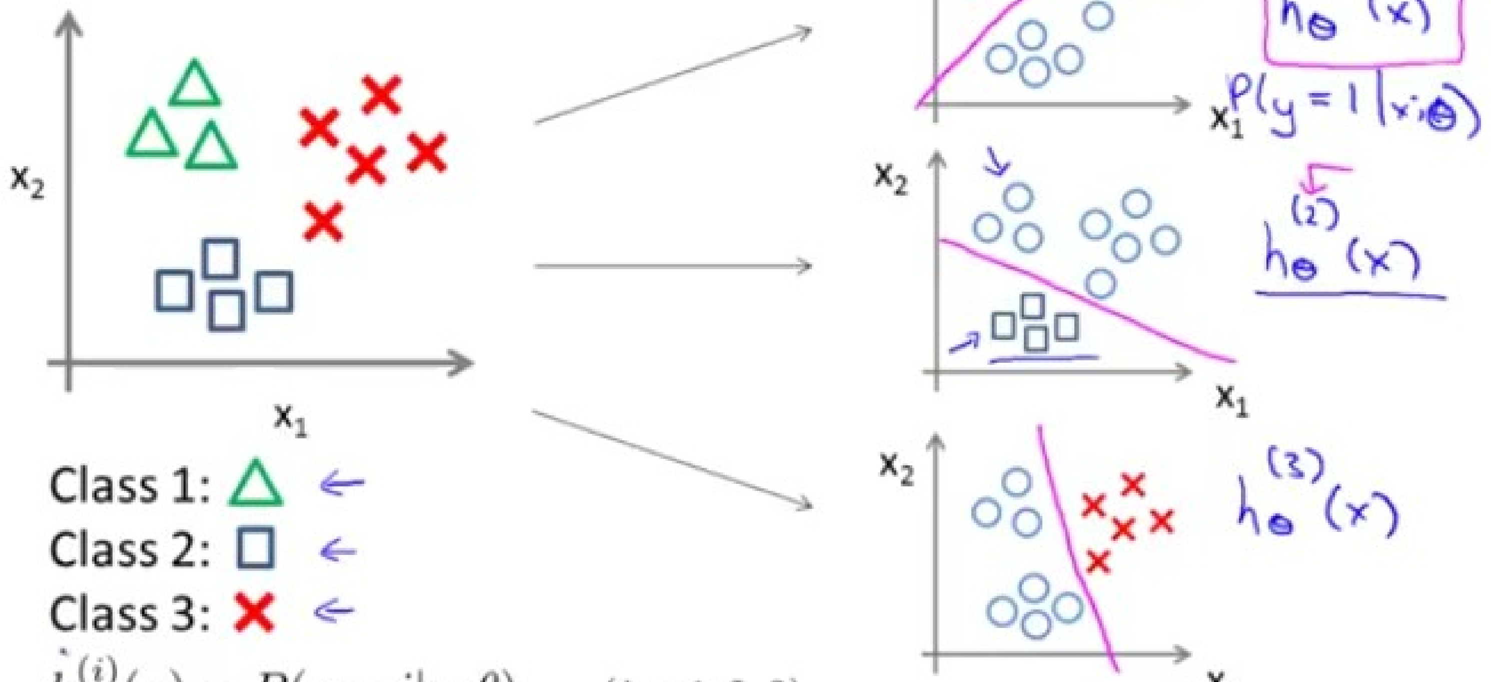
Logistic Regression





Logistic Regression

One-vs-all (one-vs-rest):



Class 1: △

Class 2: □

Class 3: ✕

$h_\theta^{(i)}(x) = P(y = i|x;\theta)$ $(i = 1, 2, 3)$

Logistic Regression

One-vs-all

Train a logistic regression classifier $\underline{h_\theta^{(i)}(x)}$ for each class $\underline{i}$ to predict the probability that $\underline{y = i}$.

On a new input $\underline{x}$, to make a prediction, pick the class i that maximizes

$$\underset{\underline{i}}{\max}\ \underset{\uparrow}{\underline{h_\theta^{(i)}(x)}}$$

Logistic Regression

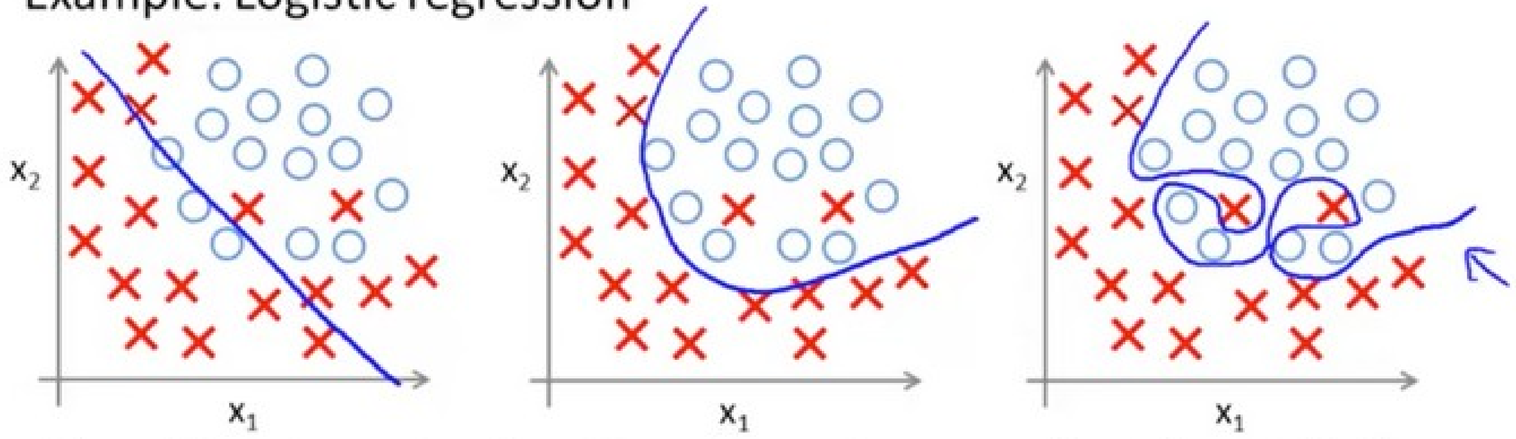
Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

- Overfit
  - High Variance
  - Too many features
  - Fit well but fail to generalize new examples
- Underfit
  - High Bias

Logistic Regression

- Logistic Regression: Overfitting

Example: Logistic regression



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

(g = sigmoid function)

"Underfit"

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$$

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \ldots)$$

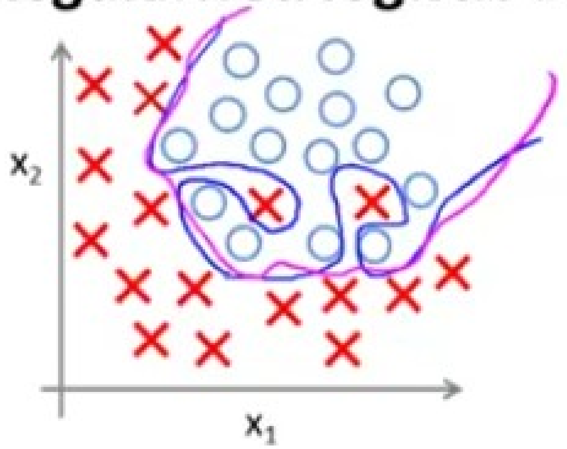
"Overfit"

Logistic Regression

- Solutions to Overfitting
  - Reduce number of features
    - Manually select features to keep
    - Model selection algorithm
  - Regularization
    - Keep all features, but reduce magnitude or values of parameters theta_j
    - Works well when we've a lot of features

Logistic Regression

Regularized logistic regression.



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \ldots)$$

Cost function:

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

$$+ \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \qquad \boxed{\theta_1, \theta_2, \ldots, \theta_n}$$

Logistic Regression

Repeat {

$$\rightarrow \quad \theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\rightarrow \quad \theta_j := \theta_j - \alpha \underbrace{\left[\frac{1}{m}\sum_{i=1}^{m}(\underline{h_\theta(x^{(i)})} - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j\right]}_{\frac{\partial}{\partial\theta_j}\underline{J(\theta)}} \leftarrow$$

$$(j = \cancel{0}, \underbrace{1, 2, 3, \ldots, n}_{\theta_1 \ldots \theta_n})$$

}

$$\underline{h_\theta(x)} = \frac{1}{1+e^{-\theta^T x}}$$

- To check if Gradient Descent is working well

Plot $-[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$ as a function of the number of iterations and make sure it's decreasing.

Performance Metrics

Some of the metrics to evaluate a ML Model performance are:

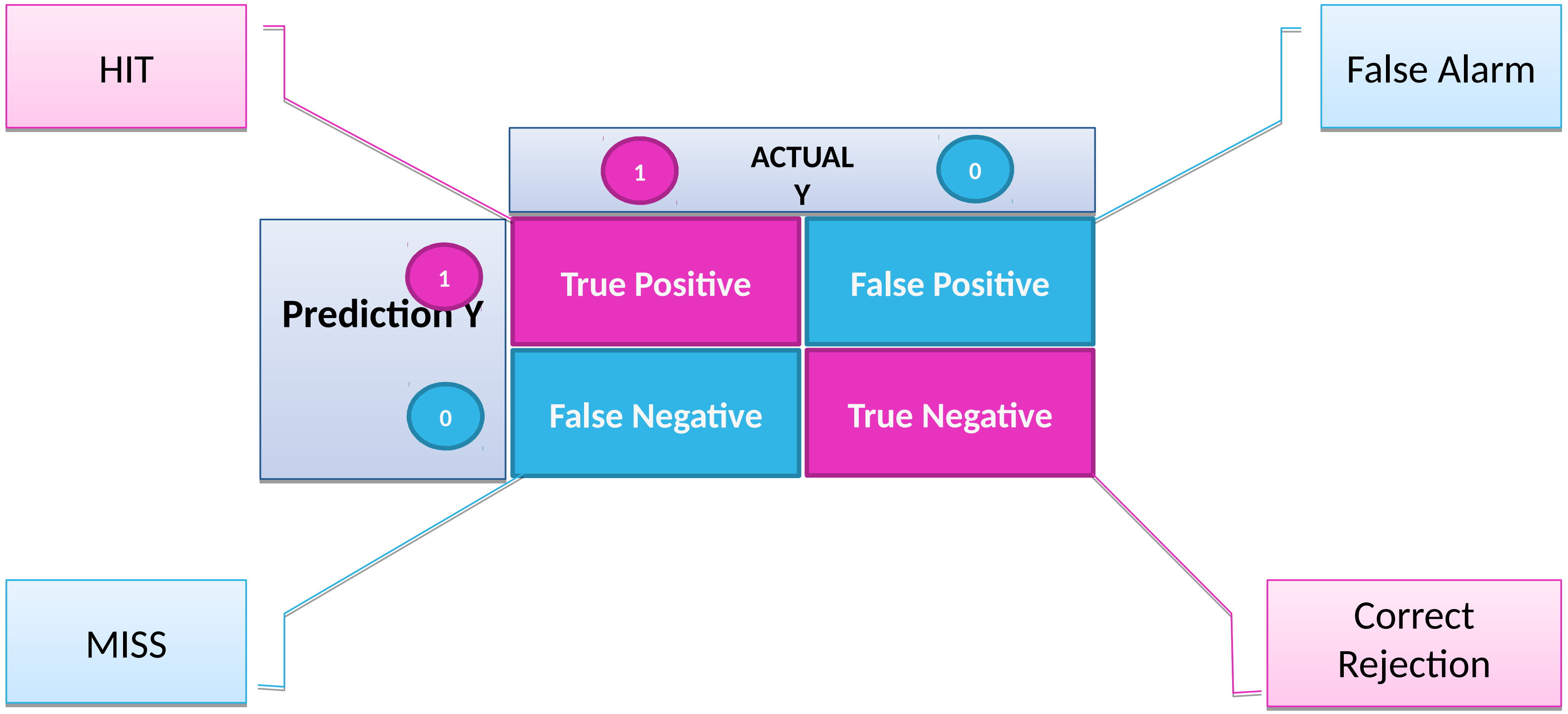
Classification

- Accuracy
- Precision
- Recall
- F1 score

Regression

- Mean Absolute Error
- Mean Squared Error
- R2 Score

CONFUSION MATRIX



Classification Metrics

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

It’s a 2 by 2 matrix which compares the actual outcomes to the predicted outcomes. The rows are labeled with actual outcomes while the columns are labeled with predicted outcomes.

True Positives (TP) - These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN) - These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

False Positives (FP) – When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.
False Negatives (FN) – When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

Classification Metrics

Accuracy : it is simply a ratio of correctly predicted observation to the total observations

Accuracy = TP+TN/TP+FP+FN+TN

It is really only suitable when there are an equal number of observations in each class (which is rarely the case) and that all predictions and prediction errors are equally important, which is often not the case.

Sensitivity or Recall = True Positives/(True Positives + False Negatives)

Out of all the items that are truly positive, how many were correctly classified as positive. Or simply, how many positive items were 'recalled' from the dataset.

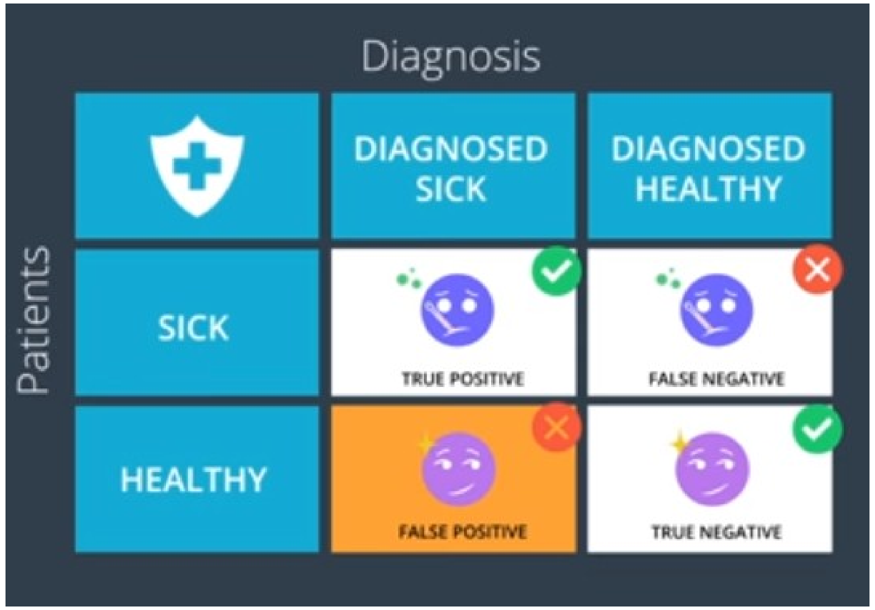
Specificity or Precision = True Positives/(True Positives + False Positives) aka **Precision**

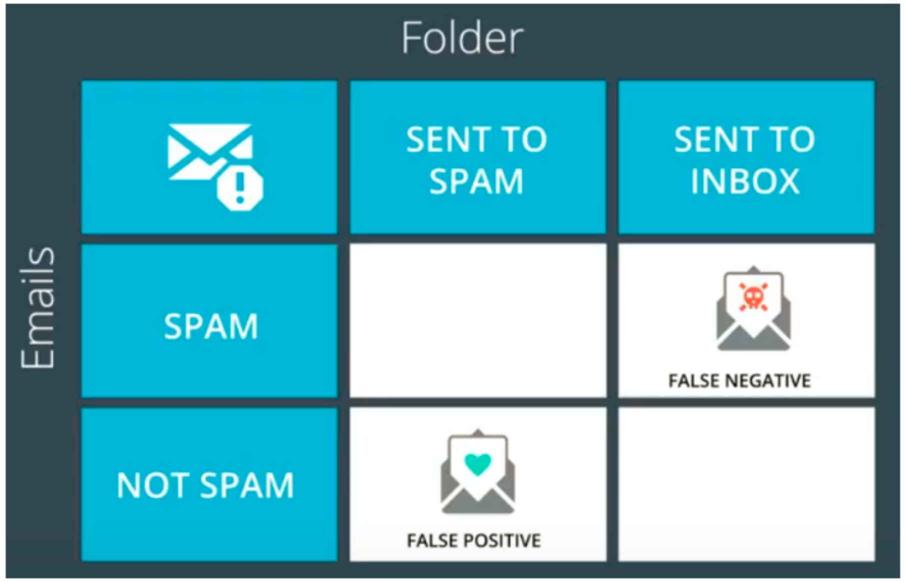
Out of all the items labeled as positive, how many truly belong to the positive class. It measures how *precise you are at predicting a positive label.*

F1-Score: F1 score combines precision and recall relative to a specific positive class. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst at 0:

F1 = 2 * (precision * recall) / (precision + recall)

False Positives and False Negatives



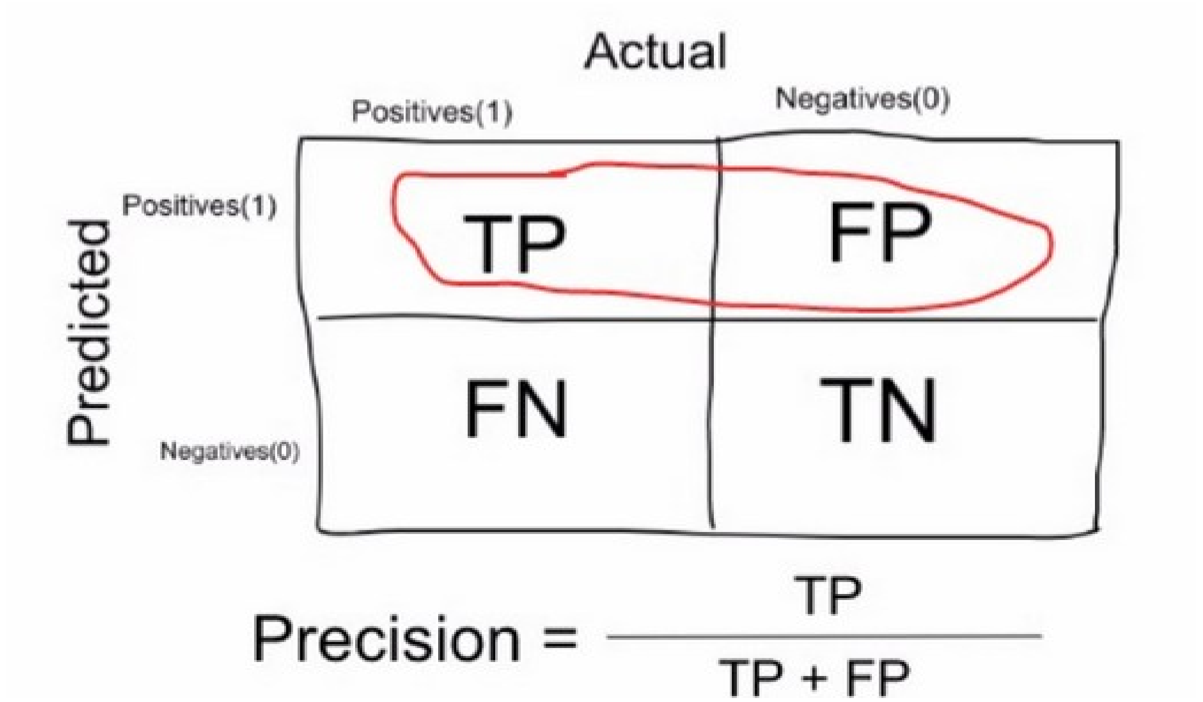


Precision: Out of the patients, diagnosed with illness, how many were **classified correctly**. Out of emails sent to spam folder, how many of them were actually Spam.

Recall: Out of the sick patients, how many did we **correctly diagnosed** as sick? Out of all the spam emails, how many of them were correctly sent to spam?

Precision is the percentage of items in the result set that are relevant.
Recall is the percentage of relevant items that are returned in the result set.

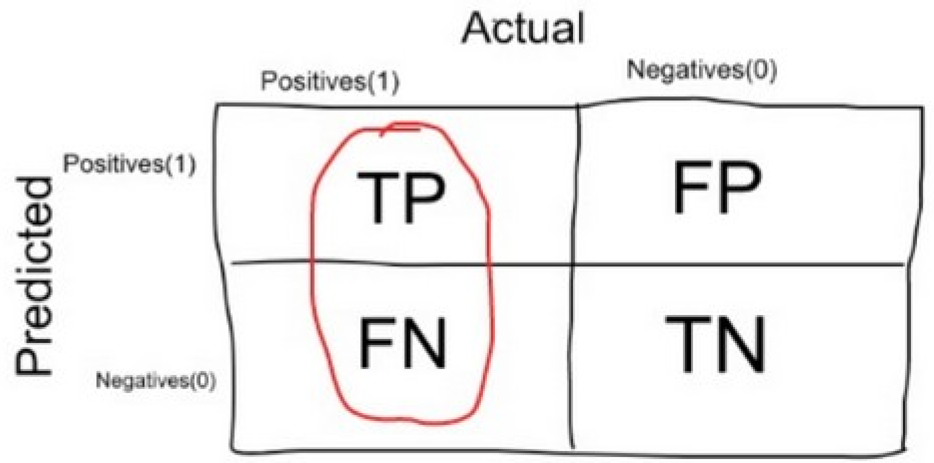
Precision:



Precision is a measure that tells us what proportion of patients that we diagnosed as having cancer, actually had cancer. The predicted positives (People predicted as cancerous are TP and FP) and the people actually having a cancer are TP.

Ex: In our cancer example with 100 people, only 5 people have cancer. Let's say our model is very bad and predicts every case as ***Cancer****. Since we are predicting everyone as having cancer, our denominator(True positives and False Positives) is 100 and the numerator, person having cancer and the model predicting his case as cancer is 5. So in this example, we can say that* ***Precision*** *of such model is 5%.*

Recall or Sensitivity:

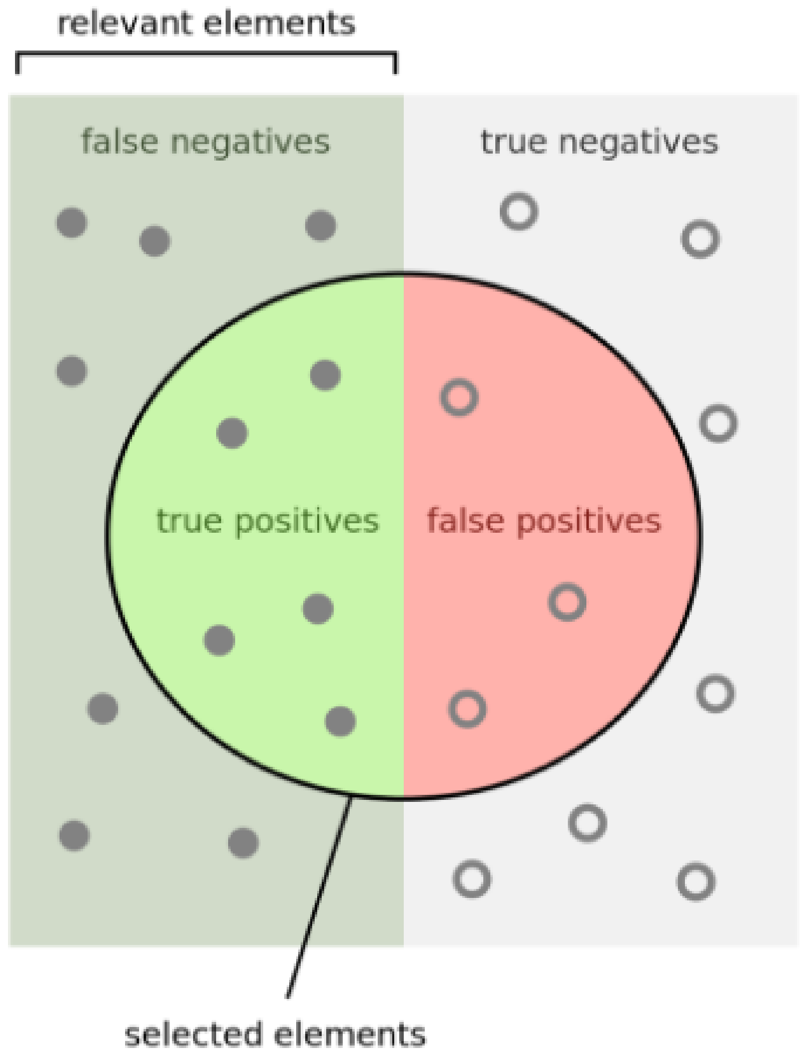


$$\text{Recall} = \frac{TP}{TP + FN}$$

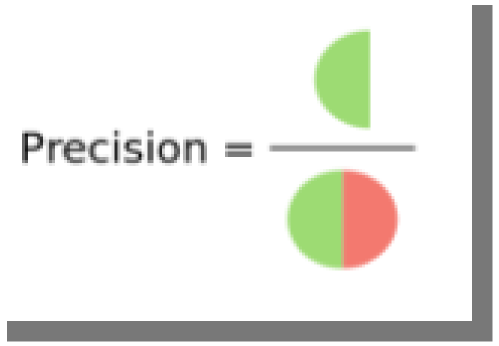
Recall is a measure that tells us what proportion of patients that actually had cancer was diagnosed by the algorithm as having cancer. The actual positives (People having cancer are TP and FN) and the people diagnosed by the model having a cancer are TP. (Note: FN is included because the Person actually had a cancer even though the model predicted otherwise).

Ex: In our cancer example with 100 people, 5 people actually have cancer. Let's say that the model predicts every case as cancer.
So our denominator(True positives and False Negatives) is 5 and the numerator, person having cancer and the model predicting his case as cancer is also 5(Since we predicted 5 cancer cases correctly). So in this example, we can say that the ***Recall*** *of such model is 100%. And Precision of such a model(As we saw above) is 5%*

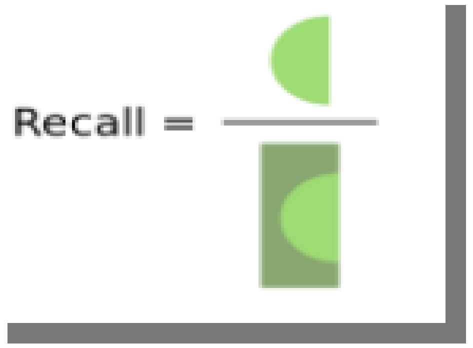
PRECISION & RECALL



How many selected items are relevant

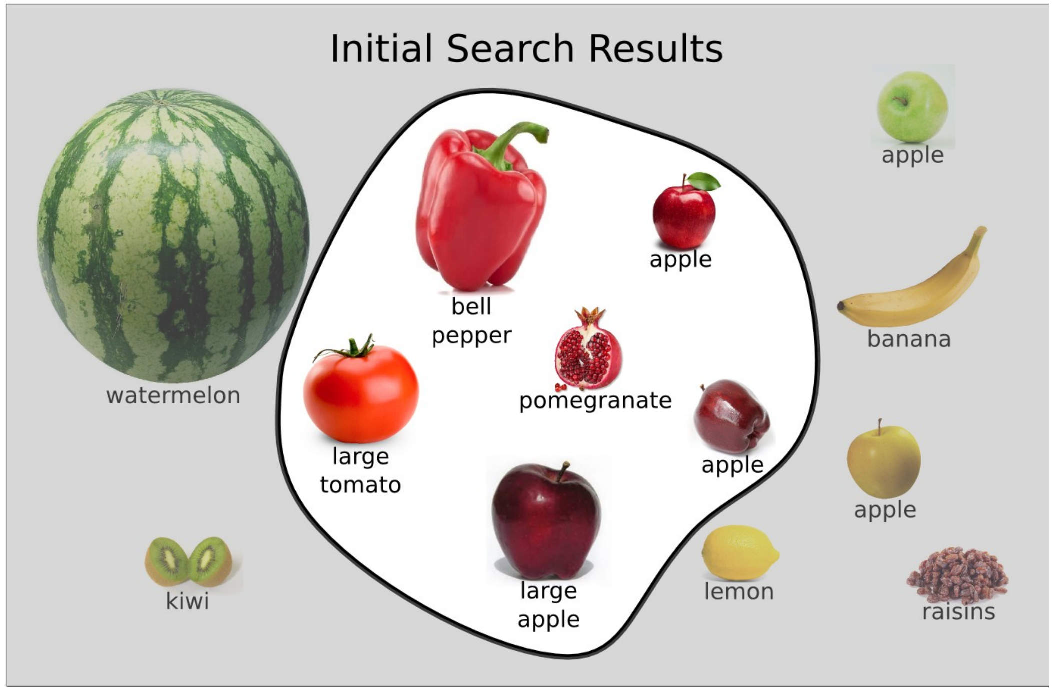


How many relevant items are selected



Precision and Recall

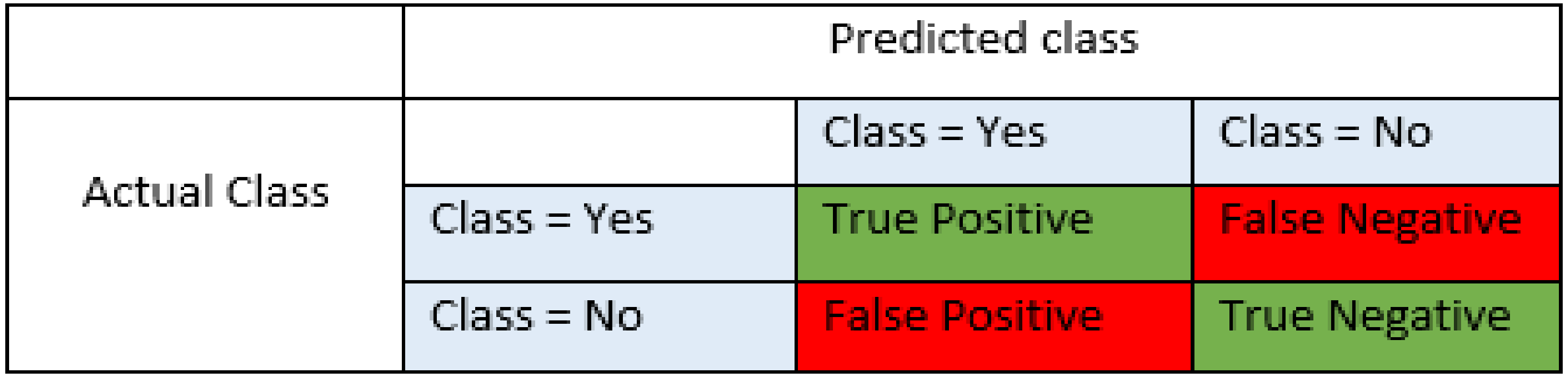
You want to search apples. You issued a query : “red, medium-sized fruit



Precision is: Proportion of No of Apples in the result set = 3/6

Recall is: Out of all apples, how many were returned in the result set = 3/5

Classification Metrics



	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

The ideal scenario that we all want is that the model should give 0 False Positives and 0 False Negatives. But that’s not the case in real life as any model will NOT be 100% accurate most of the times.

When to minimise what?
We know that there will be some error associated with every model that we use for predicting the true class of the target variable. This will result in False Positives and False Negatives(i.e Model classifying things incorrectly as compared to the actual class).
There’s no hard rule that says what should be minimised in all the situations. It purely depends on the business needs and the context of the problem you are trying to solve. Based on that, we might want to minimise either False Positives or False negatives.

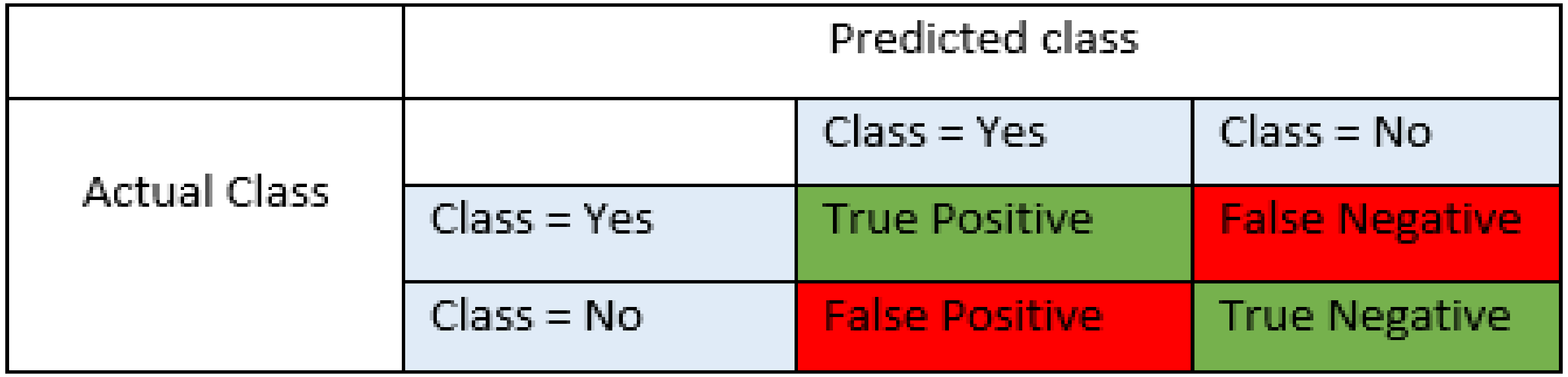
Classification Metrics

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

1. **Minimising False Negatives: C**ancer detection problem

   Out of 100 people, only 5 people have cancer. In this case, we want to correctly classify all the cancerous patients as even a very BAD model(Predicting everyone as NON-Cancerous) will give us a 95% accuracy(will come to what accuracy is). But, in order to capture all cancer cases, we might end up making a classification when the person actually NOT having cancer is classified as Cancerous. This might be okay as it is less dangerous than NOT identifying/capturing a cancerous patient since we will anyway send the cancer cases for further examination and reports. But missing a cancer patient will be a huge mistake as no further examination will be done on them.

Classification Metrics



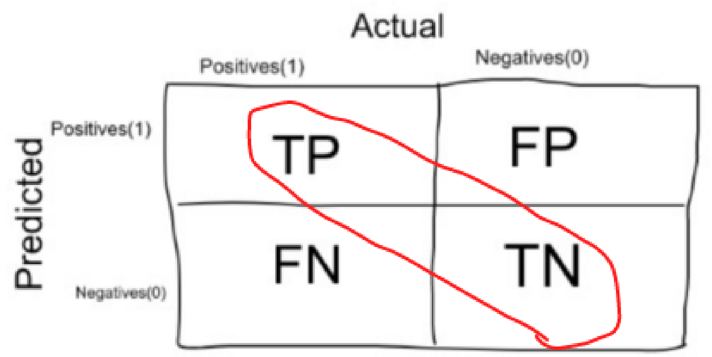
	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

2. **Minimising False Positives:** An email is spam or not

Let’s say that you are expecting an important email like hearing back from a recruiter or awaiting an admit letter from a university. Let’s assign a label to the target variable and say,**1:** “Email is a spam” and **0:**”Email is not a spam”
Suppose the Model classifies that important email that you are desperately waiting for, as Spam(case of False positive). Now, in this situation, this is pretty bad than classifying a spam email as important or not spam since in that case, we can still go ahead and manually delete it and it’s not a pain if it happens once a while. So in case of Spam email classification, minimising False positives is more important than False Negatives.

Accuracy

Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made.



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

When to use Accuracy:

Accuracy is a good measure when the target variable classes in the data are nearly balanced.

Ex:60% classes in our fruits images data are apple and 40% are oranges.

A model which predicts whether a new image is Apple or an Orange, 97% of times correctly is a very good measure in this example

When NOT to use Accuracy:

Accuracy should NEVER be used as a measure when the target variable classes in the data are a majority of one class.

Ex: In our cancer detection example with 100 people, only 5 people has cancer. Let's say our model is very bad and predicts every case as No Cancer. In doing so, it has classified those 95 non-cancer patients correctly and 5 cancerous patients as Non-cancerous. Now even though the model is terrible at predicting cancer, The accuracy of such a bad model is also 95%.

.

When to use Precision and When to use Recall?:

It is clear that **recall** gives us information about a classifier’s performance with respect to false negatives (how many did we miss), while **precision** gives us information about its performance with respect to false positives(how many did we caught).
Precision is about being precise. So even if we managed to capture only one cancer case, and we captured it correctly, then we are 100% precise.
Recall is not so much about capturing cases correctly but more about capturing all cases that have “cancer” with the answer as “cancer”. So if we simply always say every case as “cancer”, we have 100% recall.

So basically if we want to focus more on minimising False Negatives, we would want our Recall to be as close to 100% as possible without precision being too bad.
Andd if we want to focus on minimising False positives, then our focus should be to make Precision as close to 100% as possible.

TPR & FPR

TRUE POSITIVE RATE (HIT RATE)

Sensitivity also called the recall.

It is the number instances from the positive (first) class that actually predicted correctly.

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

FALSE POSITIVE RATE (False Alarm Rate)

1-Specificity

Specificity is also called the true negative rate. Is the number of instances from the negative class (second) class that were actually predicted correctly.

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

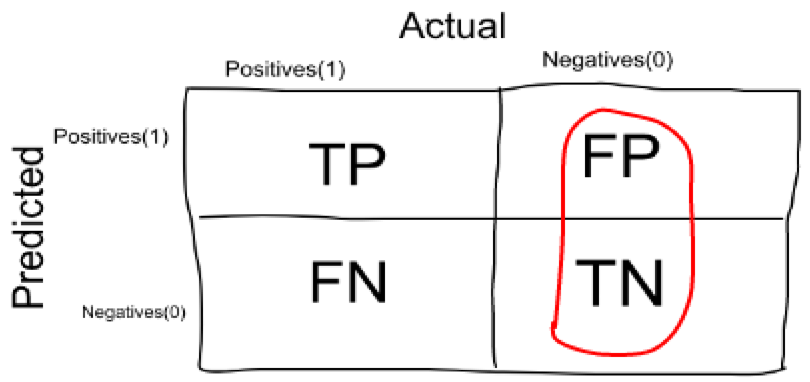
Spam filter (positive class is "spam"): :Optimize for **precision or specificity**

Because false negatives (spam goes to the inbox) are more acceptable than false positives (non-spam is caught by the spam filter)

Fraudulent transaction detector (positive class is "fraud"): Optimize for **sensitivity**

Because false positives (normal transactions that are flagged as possible fraud) are more acceptable than false negatives (fraudulent transactions that are not detected)

Specificity



$$\text{Specificity} = \frac{TN}{TN + FP}$$

Specificity is a measure that tells us what proportion of patients that did NOT have cancer, were predicted by the model as non-cancerous. The actual negatives (People actually NOT having cancer are FP and TN) and the people diagnosed by us not having cancer are TN
Specificity is the exact opposite of Recall.
Ex: In our cancer example with 100 people, 5 people actually have cancer. Let's say that the model predicts every case as cancer.
So our denominator(False positives and True Negatives) is 95 and the numerator, person not having cancer and the model predicting his case as no cancer is 0 (Since we predicted every case as cancer). So in this example, we can that that ***Specificity*** *of such model is 0%.*

Specificity VS Sensitivity

Threshold of 0.5 is used by default (for binary problems) to convert predicted probabilities into class predictions
Threshold can be **adjusted** to increase sensitivity or specificity
Sensitivity and specificity have an **inverse relationship**

Increasing one would always decrease the other

Adjusting the threshold should be one of the last step you do in the model-building process

The most important steps are
Building the models
Selecting the best model

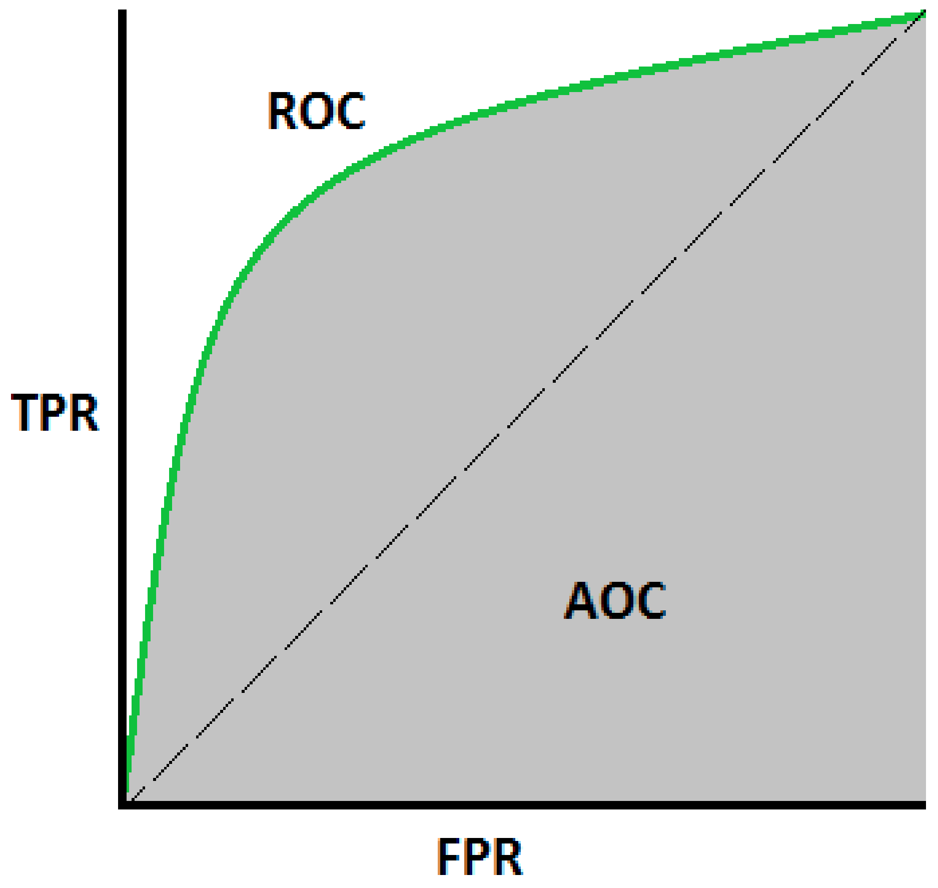
ROC Curve

Receiver Operating Characteristic(ROC) summarizes the model's performance by evaluating the trade offs between true positive rate (sensitivity) and false positive rate(1-specificity).

The TPR defines how many correct positive results occur among all positive samples available during the test. FPR, on the other hand, defines how many incorrect positive results occur among all negative samples available during the test. The TPR defines how many correct positive results occur among all positive samples available during the test. FPR, on the other hand, defines how many incorrect positive results occur among all negative samples available during the test.

The **area under curve (AUC),** referred to as index of accuracy(A) or concordance index, is a perfect performance metric for ROC curve. Higher the area under curve, better the prediction power of the model. Below is a sample ROC curve. The ROC of a perfect predictive model has TP equals 1 and FP equals 0. An area of 0.5 represents a model as good as random

Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.
By analogy, Higher the AUC, better the model is at distinguishing between patients with disease and no disease.



ROC Curve

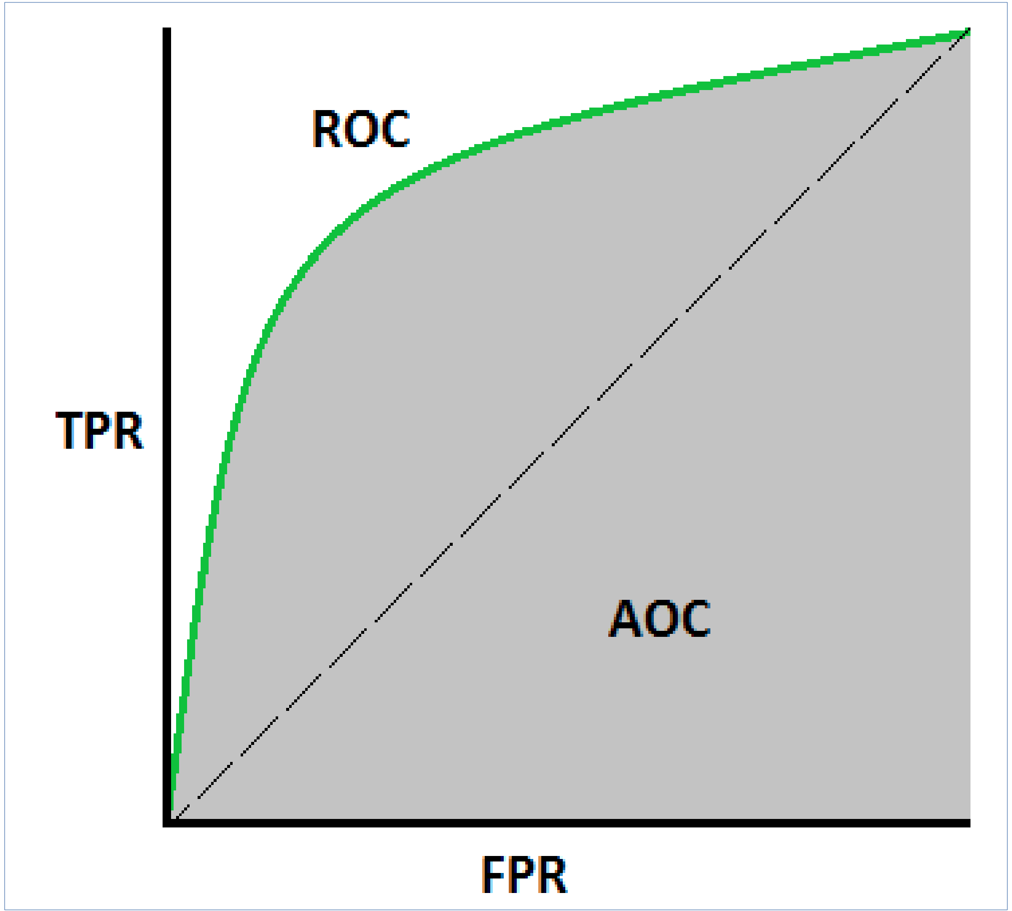
Wouldn't it be nice if we could see how sensitivity and specificity are affected by various thresholds, without actually changing the threshold?
Answer: Plot the ROC curve.

ROC curve can help you to **choose a threshold** that balances sensitivity and specificity in a way that makes sense for your particular context

You can't actually **see the thresholds** used to generate the curve on the ROC curve itself.

AUC is useful as a **single number summary** of classifier performance
Higher value = better classifier
If you randomly chose one positive and one negative observation, AUC represents the likelihood that your classifier will assign a **higher predicted probability** to the positive observation
AUC is useful even when there is **high class imbalance** (unlike classification accuracy)

- Fraud case
  - Null accuracy almost 99%
  - AUC is useful here



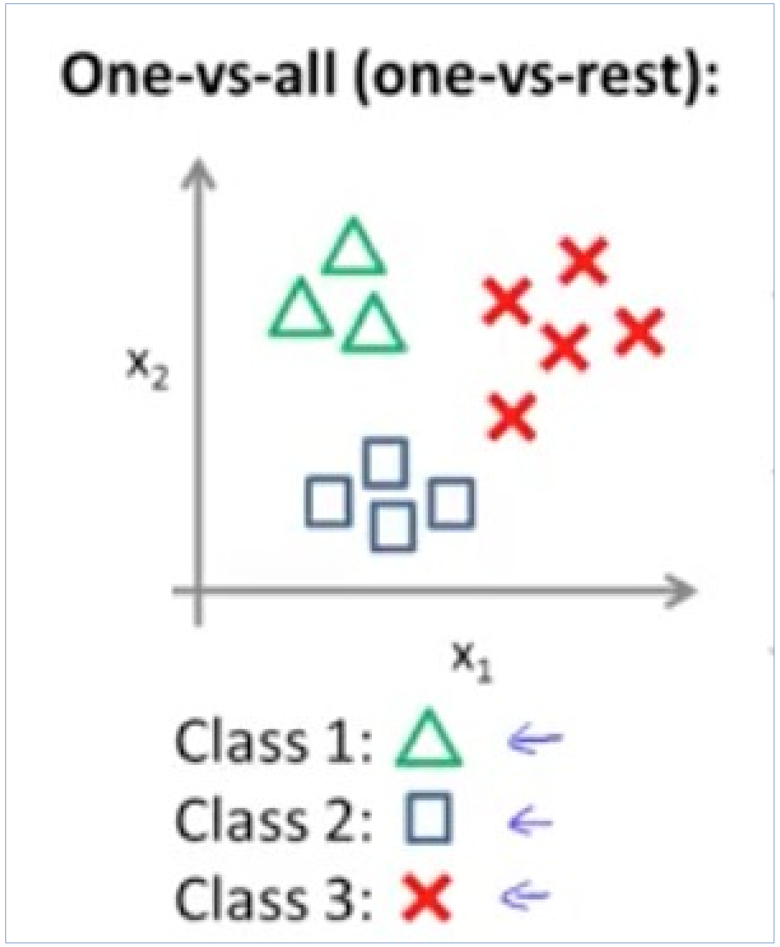
Multi-Class & Multi-label Problems

What features this landscape has ?

It is always a “AND”

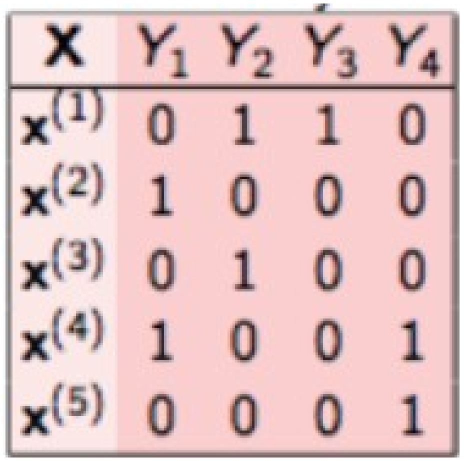
Multi Class has OR operation : Winter or Summer | Movie is Hit or FLOP or Average

Multi-Class problems with Logistic Regression



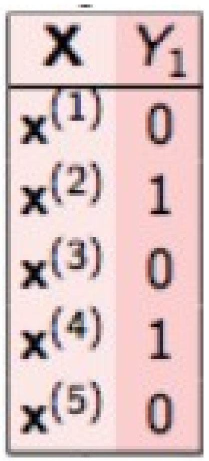
Multi-Label : Binary Relevance

In binary relevance, this problem is broken into 4 different single class classification problems as shown in the figure below.



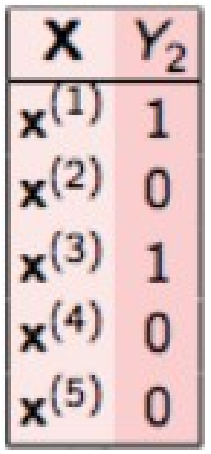
X	Y_1	Y_2	Y_3	Y_4
$x^{(1)}$	0	1	1	0
$x^{(2)}$	1	0	0	0
$x^{(3)}$	0	1	0	0
$x^{(4)}$	1	0	0	1
$x^{(5)}$	0	0	0	1

Classifier : 1



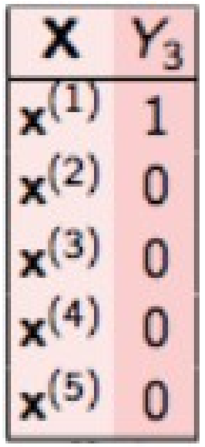
X	Y_1
$x^{(1)}$	0
$x^{(2)}$	1
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	0

Classifier : 2



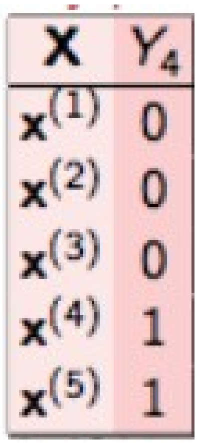
X	Y_2
$x^{(1)}$	1
$x^{(2)}$	0
$x^{(3)}$	1
$x^{(4)}$	0
$x^{(5)}$	0

Classifier : 3



X	Y_3
$x^{(1)}$	1
$x^{(2)}$	0
$x^{(3)}$	0
$x^{(4)}$	0
$x^{(5)}$	0

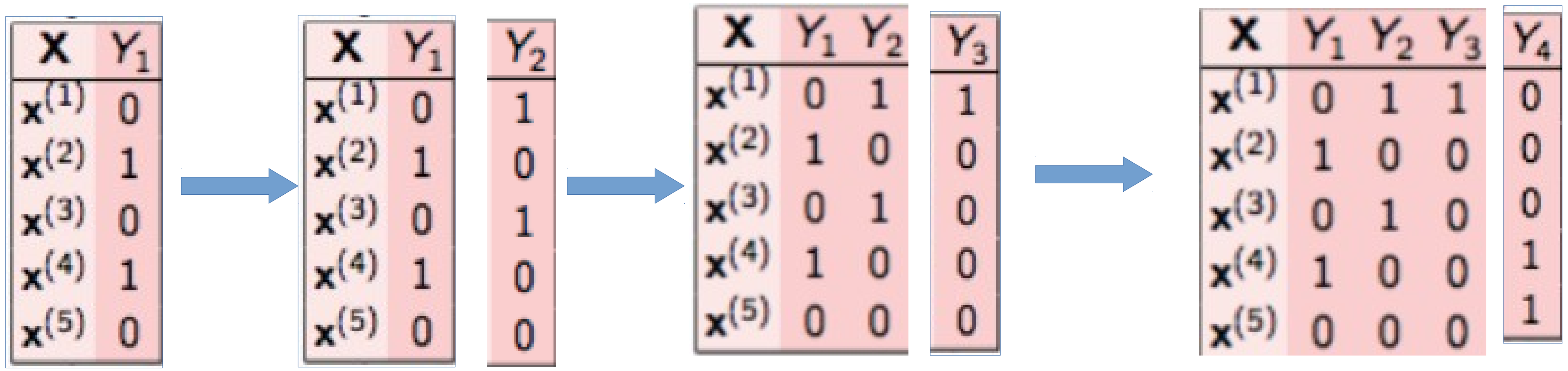
Classifier : 4



X	Y_4
$x^{(1)}$	0
$x^{(2)}$	0
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	1

Multi-Label : Classifier Chains

In classifier chains, this problem would be transformed into 4 different single label problems, just like shown below.



X	Y_1
$x^{(1)}$	0
$x^{(2)}$	1
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	0

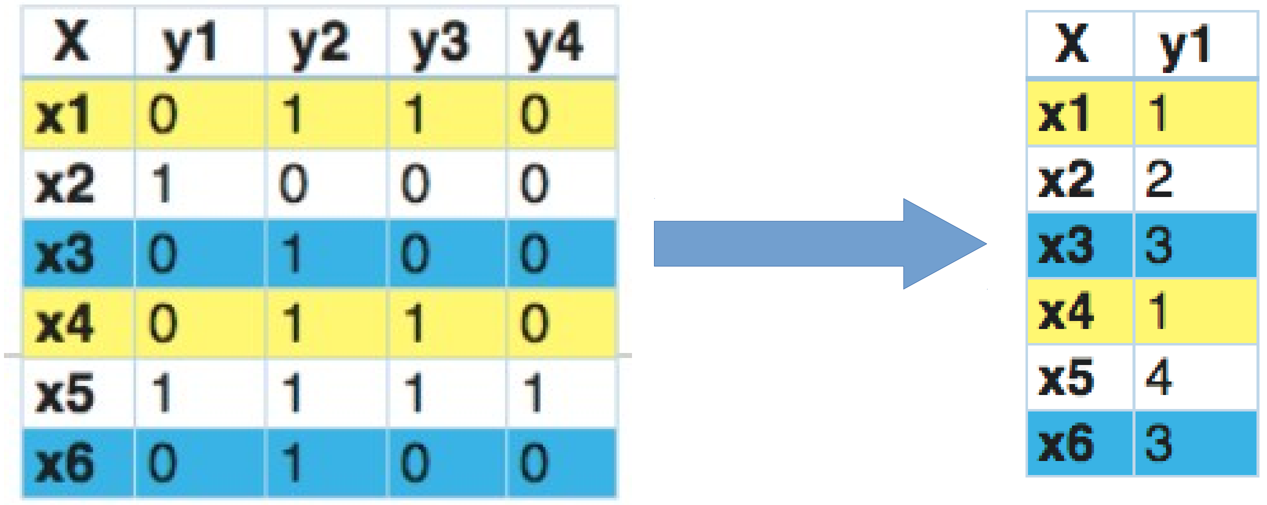
X	Y_1	Y_2
$x^{(1)}$	0	1
$x^{(2)}$	1	0
$x^{(3)}$	0	1
$x^{(4)}$	1	0
$x^{(5)}$	0	0

X	Y_1	Y_2	Y_3
$x^{(1)}$	0	1	1
$x^{(2)}$	1	0	0
$x^{(3)}$	0	1	0
$x^{(4)}$	1	0	0
$x^{(5)}$	0	0	0

X	Y_1	Y_2	Y_3	Y_4
$x^{(1)}$	0	1	1	0
$x^{(2)}$	1	0	0	0
$x^{(3)}$	0	1	0	0
$x^{(4)}$	1	0	0	1
$x^{(5)}$	0	0	0	1

Multi-Label : Label PowerSet

Transform the problem into a multi-class problem with one multi-class classifier is trained on all unique label combinations found in the training data.



X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0
x4	0	1	1	0
x5	1	1	1	1
x6	0	1	0	0

X	y1
x1	1
x2	2
x3	3
x4	1
x5	4
x6	3