PHY654

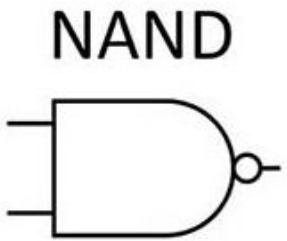
Machine learning (ML) in particle physics

Swagata Mukherjee • IIT Kanpur

17th August 2024

Example code for binary classification

NAND Gate using logistic regression
https://github.com/swagata87/IITKanpurPhy654/blob/main/NAND_gate_Logistic_regression.ipynb



Two features **y**

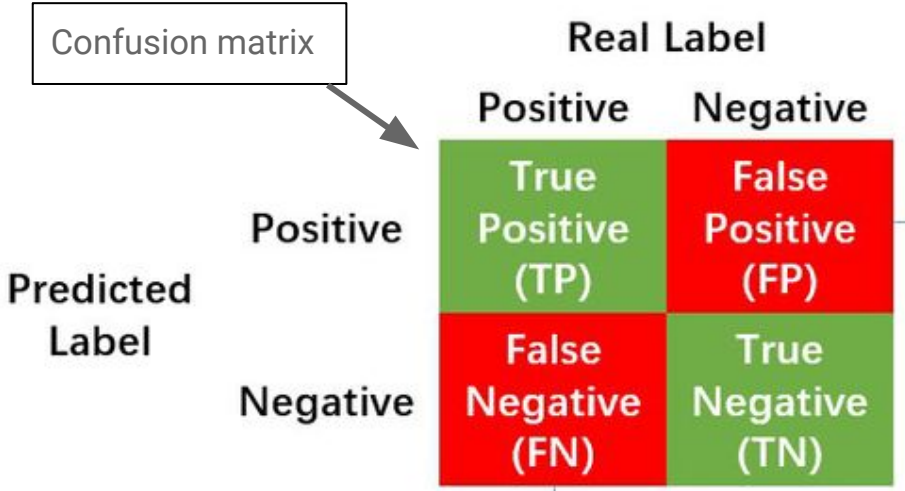
A	B	Output
0	0	1
1	0	1
0	1	1
1	1	0

Try to rewrite the code without using sklearn library

Four training examples
m=4

Useful metrics for evaluating ML models

Confusion matrix



$$\text{Precision} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FP}}$$

Generally used when it is very important for positive predictions to be accurate.

$$\text{F1} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

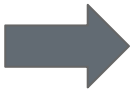
$$\text{Recall} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FN}}$$

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

Not a good measure if the dataset is not balanced

An example

Animal	 Barn owl  Chihuahua  Sheep dog
Not animal	 Apple  Muffin  Mop

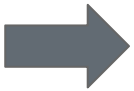


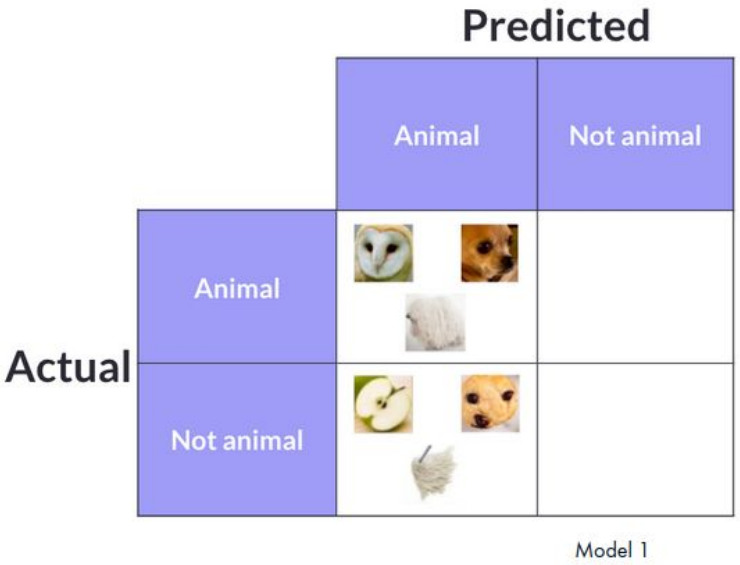
		Predicted: Animal	Predicted: Not animal
Actual	Animal	True Positives	False Negatives
	Not animal	False Positives	True Negatives

Confusion matrix

Look at all these metrics

Animal	 Barn owl  Chihuahua  Sheep dog
Not animal	 Apple  Muffin  Mop



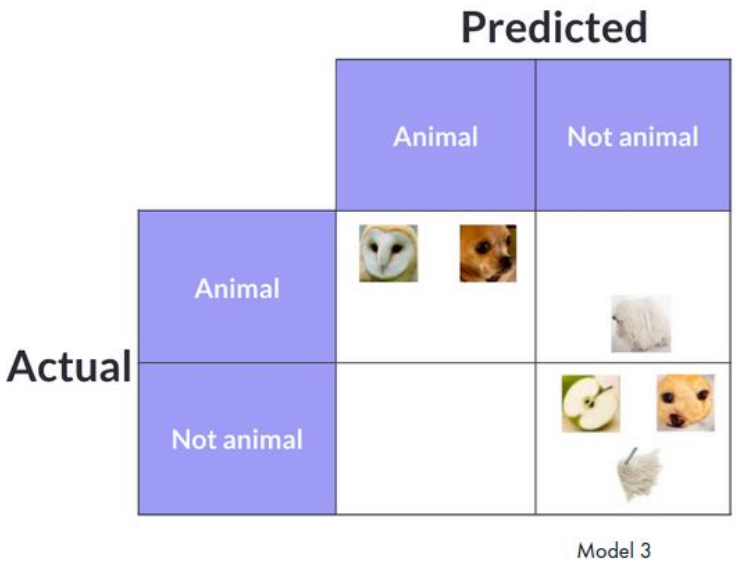


Model 1

True Positives	3
True Negatives	0
False Positives	3
False Negatives	0

Accuracy	50%	$\frac{3+0}{3+0+0+3}$
Precision	50%	$\frac{3}{3+3}$
Recall	100%	$\frac{3}{3+0}$
F1 score	67%	$2 . \frac{0.5 \cdot 1}{0.5+1}$

Look at all these metrics



True Positives	2
True Negatives	3
False Positives	0
False Negatives	1

Accuracy	83%	$\frac{2+3}{2+3+1+0}$
Precision	100%	$\frac{2}{2+0}$
Recall	67%	$\frac{2}{2+1}$
F1 score	80%	$2 \cdot \frac{1 \cdot 0.67}{1+0.67}$

Model 3

These metrics are calculated at a single classification threshold value (predicted output > 0.5 or not).
But if you want to evaluate a model's quality across all possible thresholds, you need different tools.

ROC

Receiver-operating characteristic curve (ROC)

ROC curve is drawn by computing true positive rate (TPR) and false positive rate (FPR) at every possible threshold (in selected intervals), then graphing TPR vs FPR.

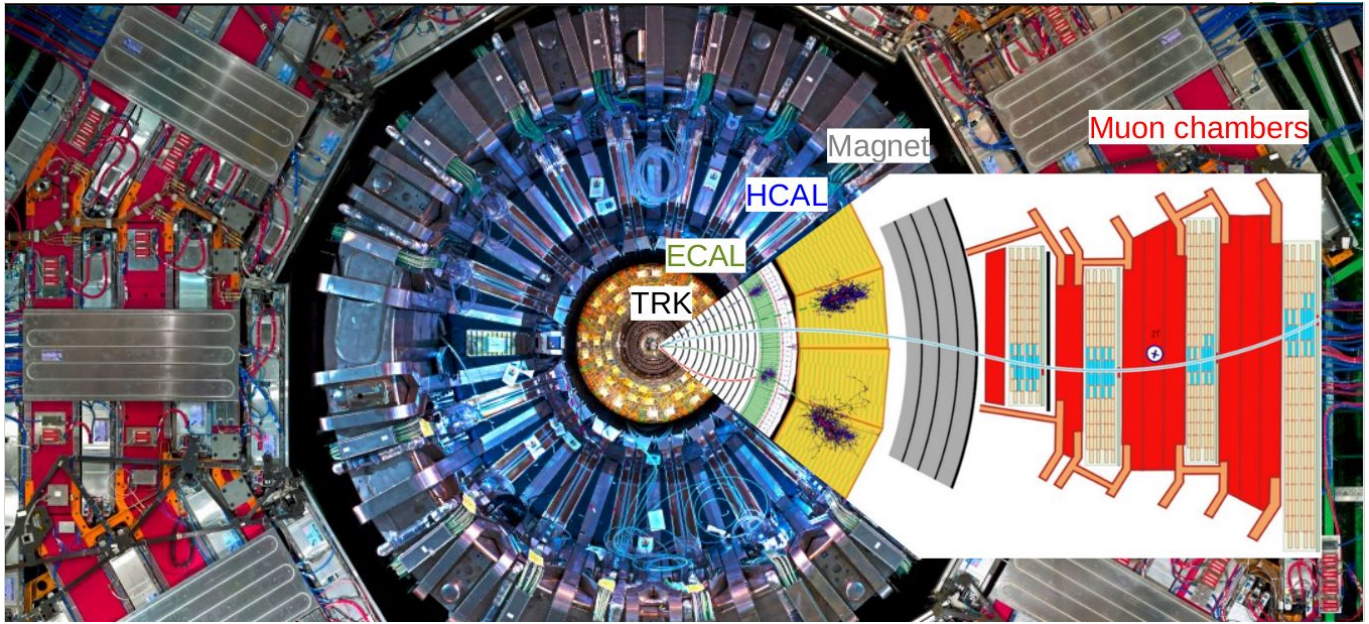
False Positive Rate = False Positives / (False Positives + True Negatives)
True Positive Rate = True Positives / (True Positives + False Negatives)

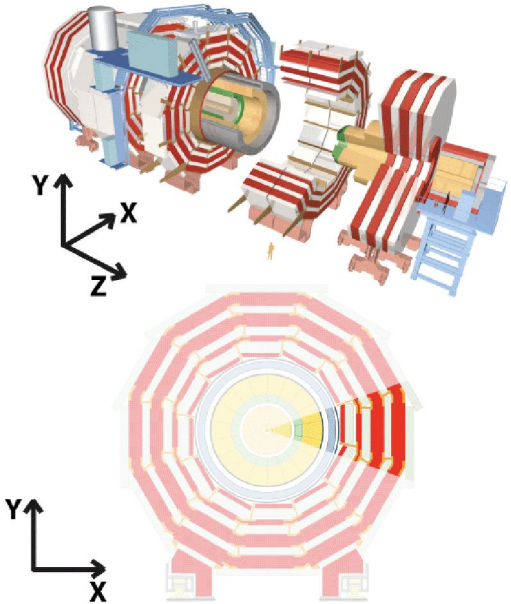
In particle physics, we generally plot **signal efficiency vs background efficiency;** or signal efficiency vs background rejection.

ROC is not specific to ML. It can be plotted for cut-based physics analysis as well. It tells us the usefulness of a cut applied on a variable.

The area under the curve (AUC) can be used as a single-valued metric.

Binary classification in physics experiments



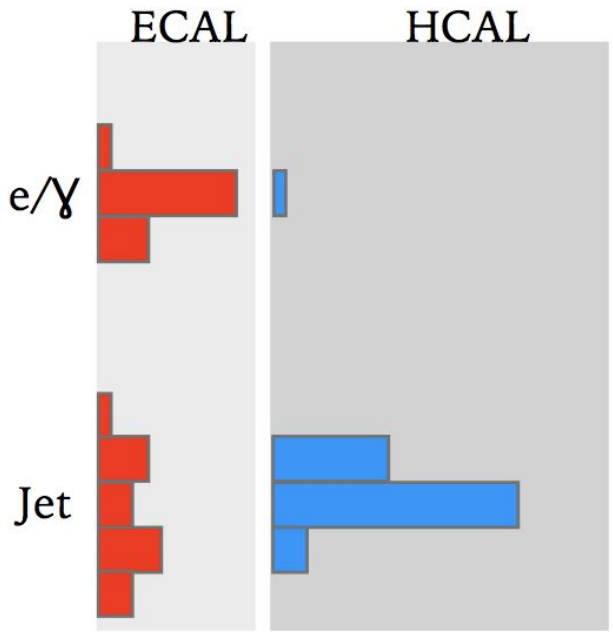


Photon vs hadronic-jet : Binary classification

Photon is signal (y=1) and Hadronic-jet is background (y=0)

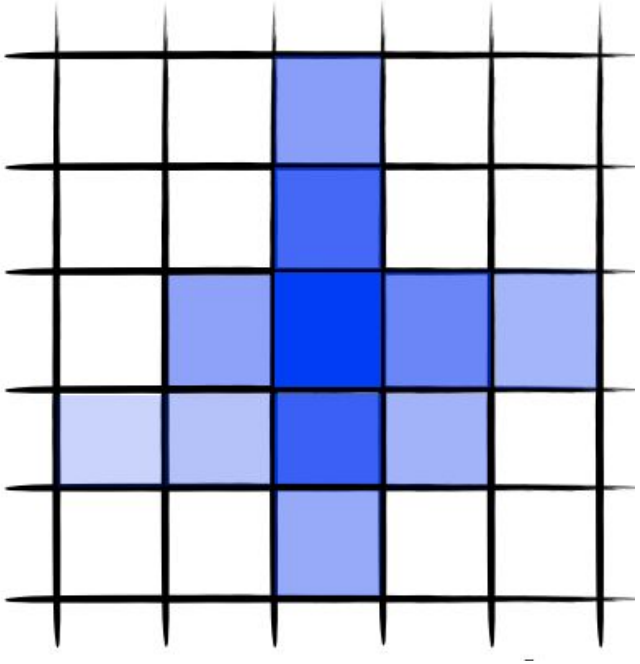
Feature 1 : H/E

Photon vs hadronic-jet : Binary classification



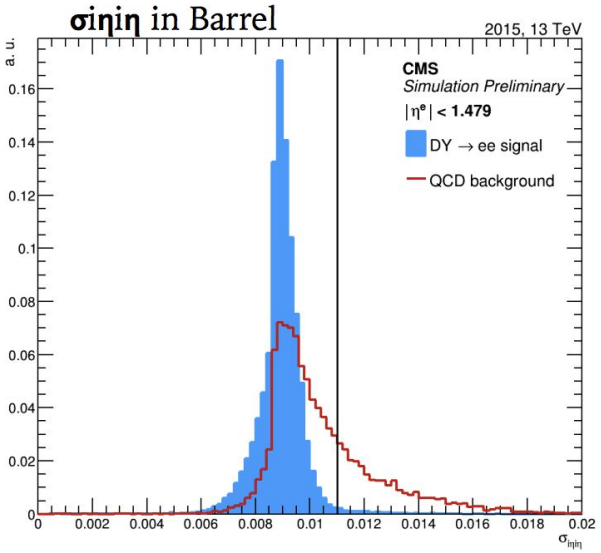
H/E variable is a good discriminator variable.

Feature 2 : Showershape in ECAL



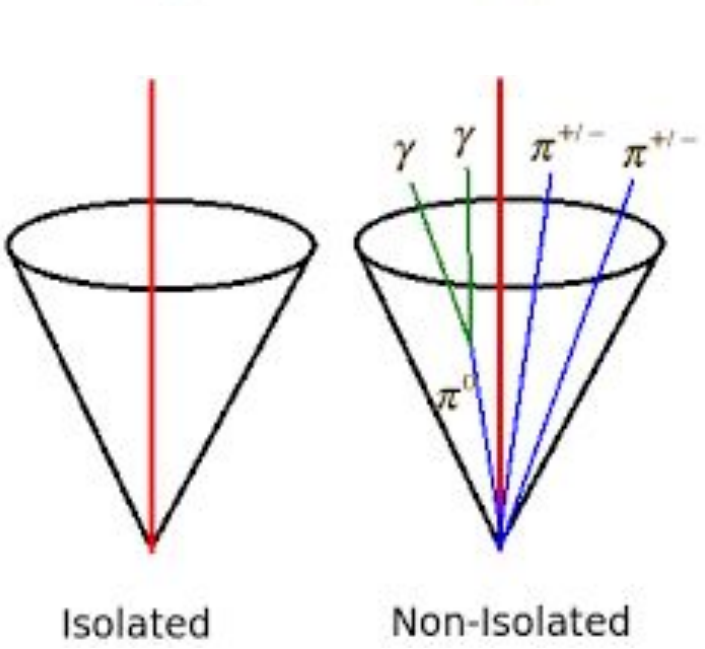
Photon vs hadronic-jet : Binary classification

Showershape is also a good discriminator between signal and background



Feature 3 : Isolation sum

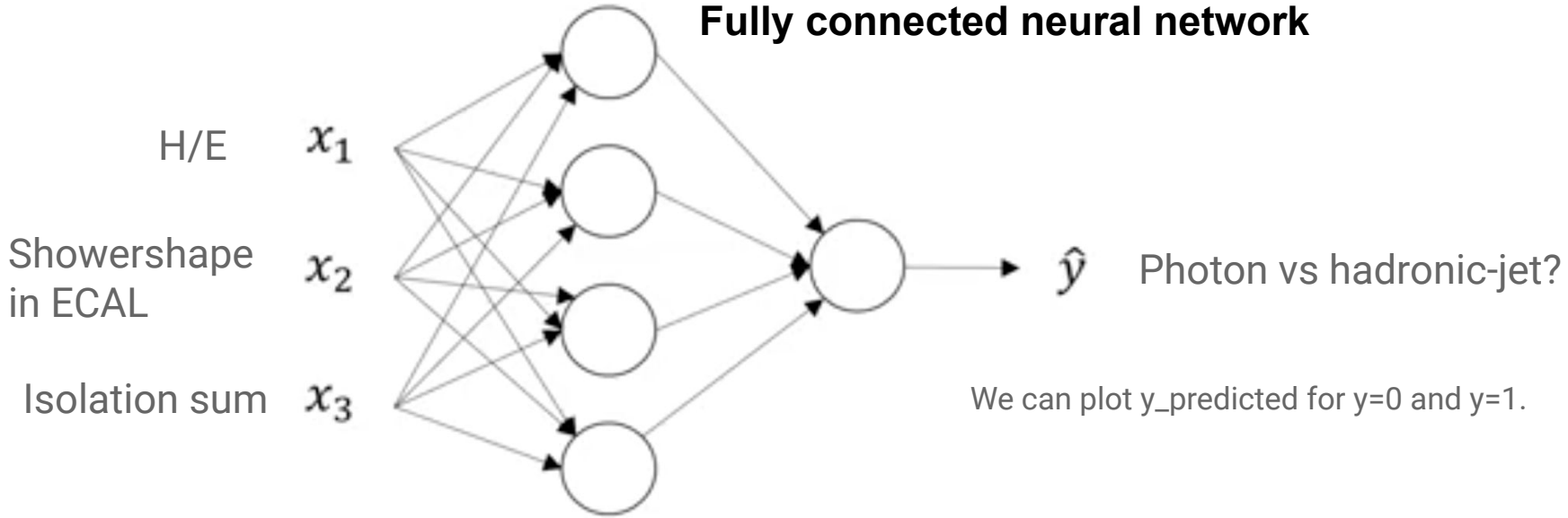
Photon vs hadronic-jet : Binary classification



Isolation sum is also a good discriminator between signal and background

Example use-case in physics experiments

Fully connected neural network



We can plot y_predicted for y=0 and y=1.

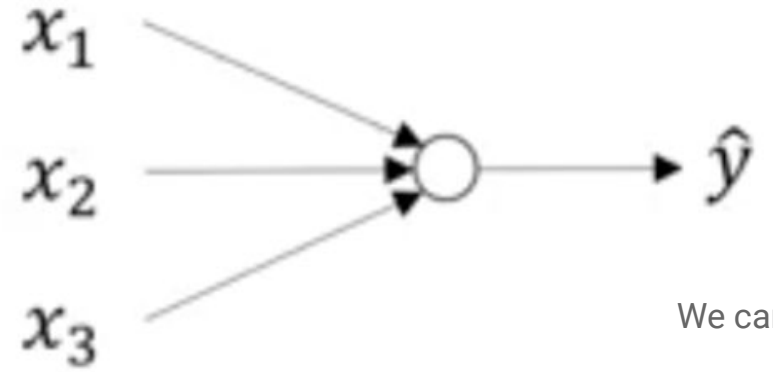
Example use-case in physics experiments

Logistic regression network

H/E

Showershape in ECAL

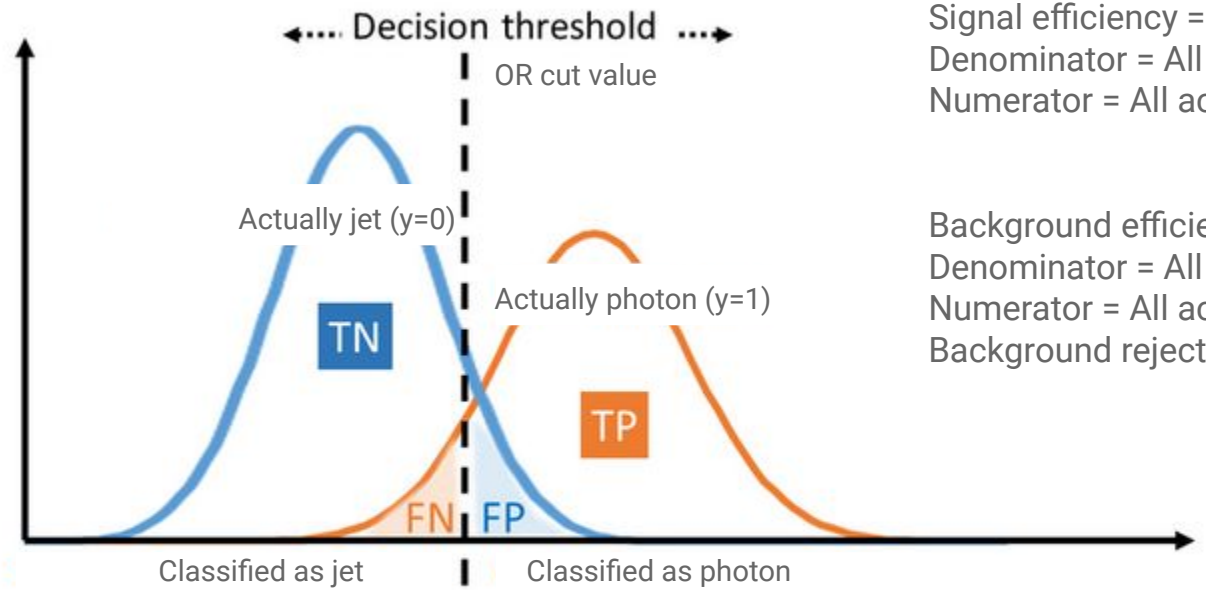
Isolation sum



Photon vs hadronic-jet?

We can plot y_predicted for y=0 and y=1.

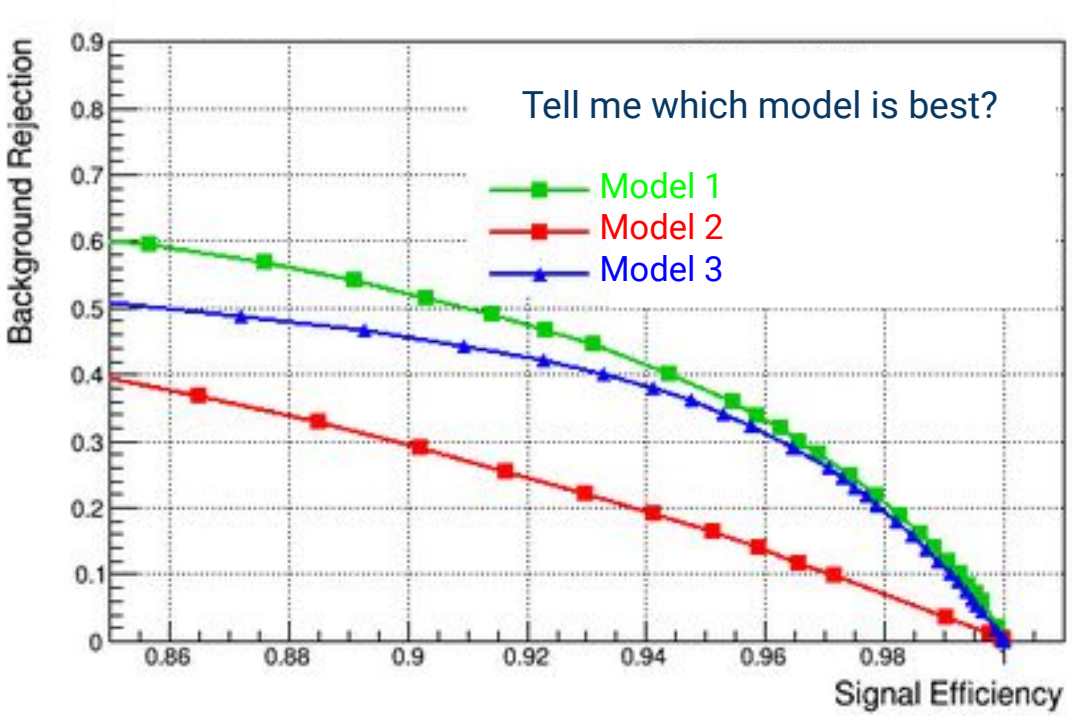
Distribution of predicted output



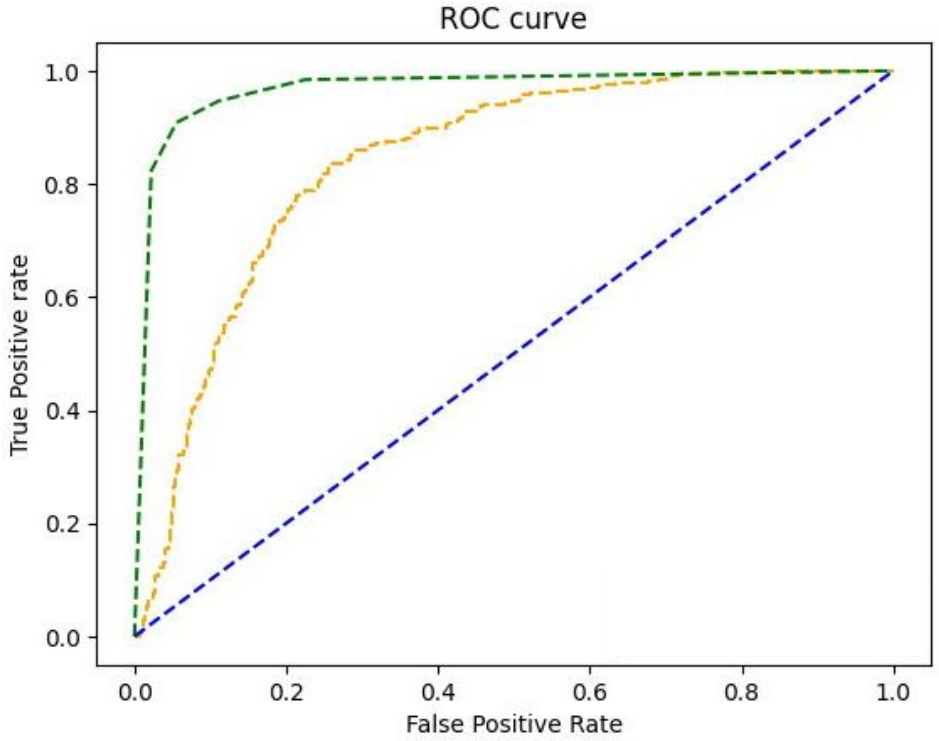
Signal efficiency = numerator / denominator
Denominator = All actual photons
Numerator = All actual photons that pass the cut

Background efficiency = numerator / denominator
Denominator = All actual jets
Numerator = All actual jets that pass the cut
Background rejection = 1 - background efficiency

ROC: an example



ROC: another example



Tell me which model is best?

Some useful links

- Python Tutorial (https://docs.python.org/3.7/tutorial/index.html): an introduction to the Python programming language
- Google Colab (https://colab.research.google.com/): for Python development in your web-browser
- numpy (https://numpy.org/doc/stable/user/quickstart.html): a widely used library for mathematical operations in Python
- Keras (https://keras.io/): a beginner-friendly deep learning library
- Tensor Flow (https://www.tensorflow.org/): a useful backend for deep learning development
- SciKit Learn (https://scikit-learn.org/stable/): helpful machine learning library
- Seaborn (https://seaborn.pydata.org/): a library for creating graphs and figures

A non-physics example of classification

→ Input image →

Binary classifier
1 (cat) vs 0 (non cat)

What kind of data we will be dealing with?

Structured data

H/E	isolation	shower-shape	Matched tracks?	Photon or jet
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-

Unstructured data

Image

Audio

Images as input to a Neural Network?

→ Input image →

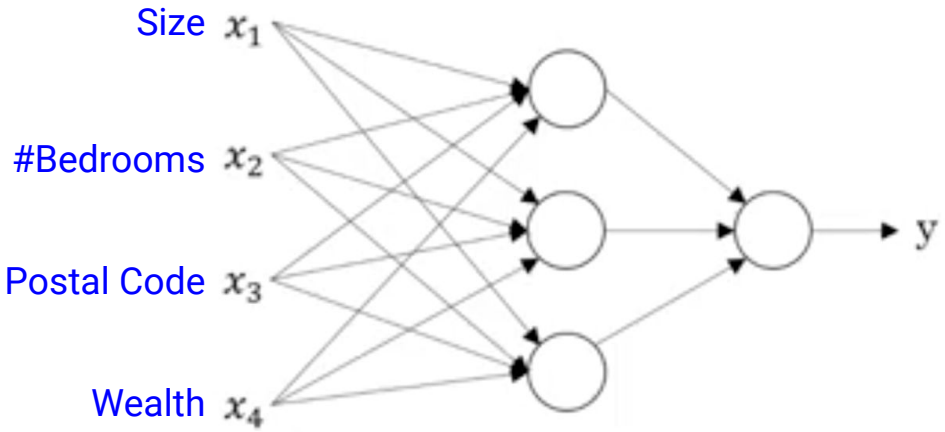
Binary classifier
1 (cat) vs 0 (non cat)

How does a computer "**see**" an image?
→ It sees 3 matrices. These are pixel-intensity values.

→ Each element is a feature. We need to unroll to a column vector.

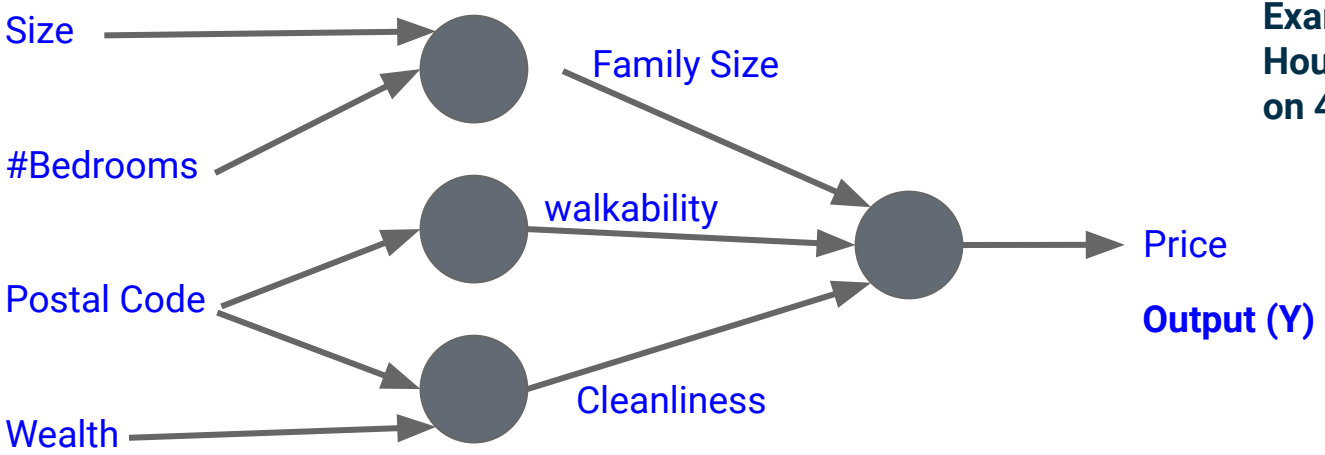
If input image is 64x64 then number of features=64x64x3 = 12288.
So, n=12288.

Hidden layer: what does it do?



Example (regression):
Housing price prediction, based on 4 input variables.

Hidden layer: what does it do?



Example (regression): Housing price prediction, based on 4 input variables.

Output (Y)

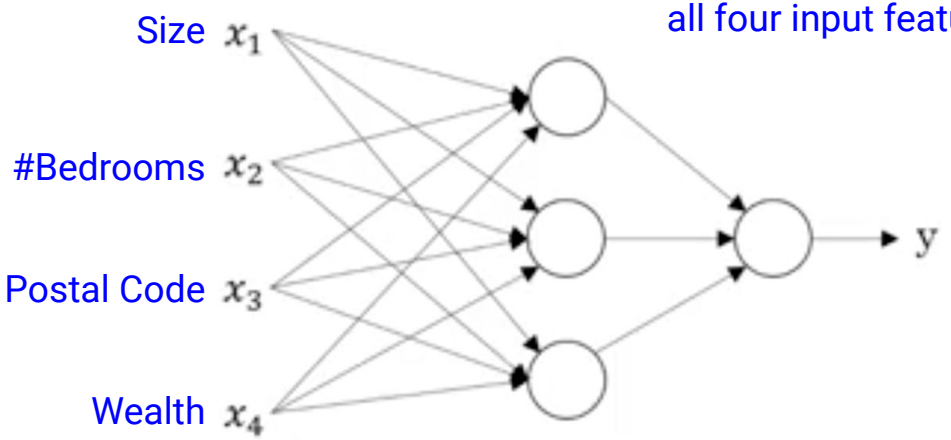
Input features (X)

You will provide X and Y for some training examples, and the hidden layer(s) will figure out all the other things by itself.

Fully connected or densely connected

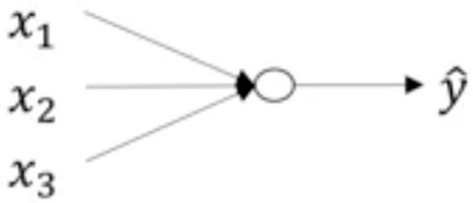
This is what we will actually implement

Each **hidden unit** takes all four input features

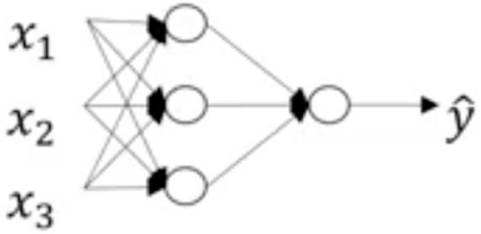


Instead of we deciding that the last hidden unit is “cleanliness”, which depends only on postal code and wealth, we leave it on the neural network to decide what the last hidden unit would be. We just provide it with all input features.

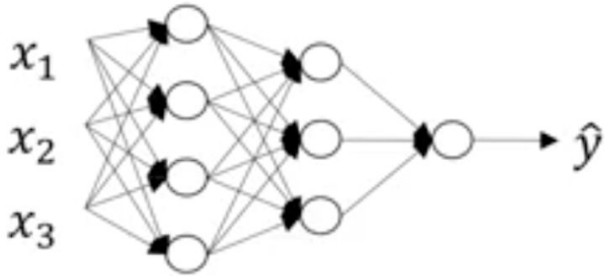
neural network: shallow vs deep



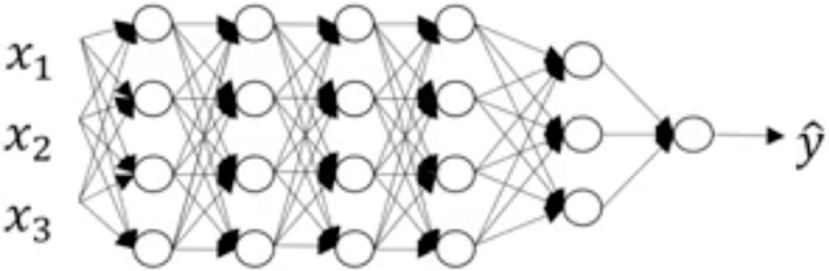
logistic regression



1 hidden layer



2 hidden layers



5 hidden layers