Artificial Intelligence and Machine Learning

Classification / Logistic Regression

Lecture 2: Outline

- Linear Regression (Review)
- Linear Regression (Remaining concepts)
- Intro to Classification
- Logistic Regression (Linear Classifier)
- Classification Metrics
- Other ML Models

Recap

Design your model

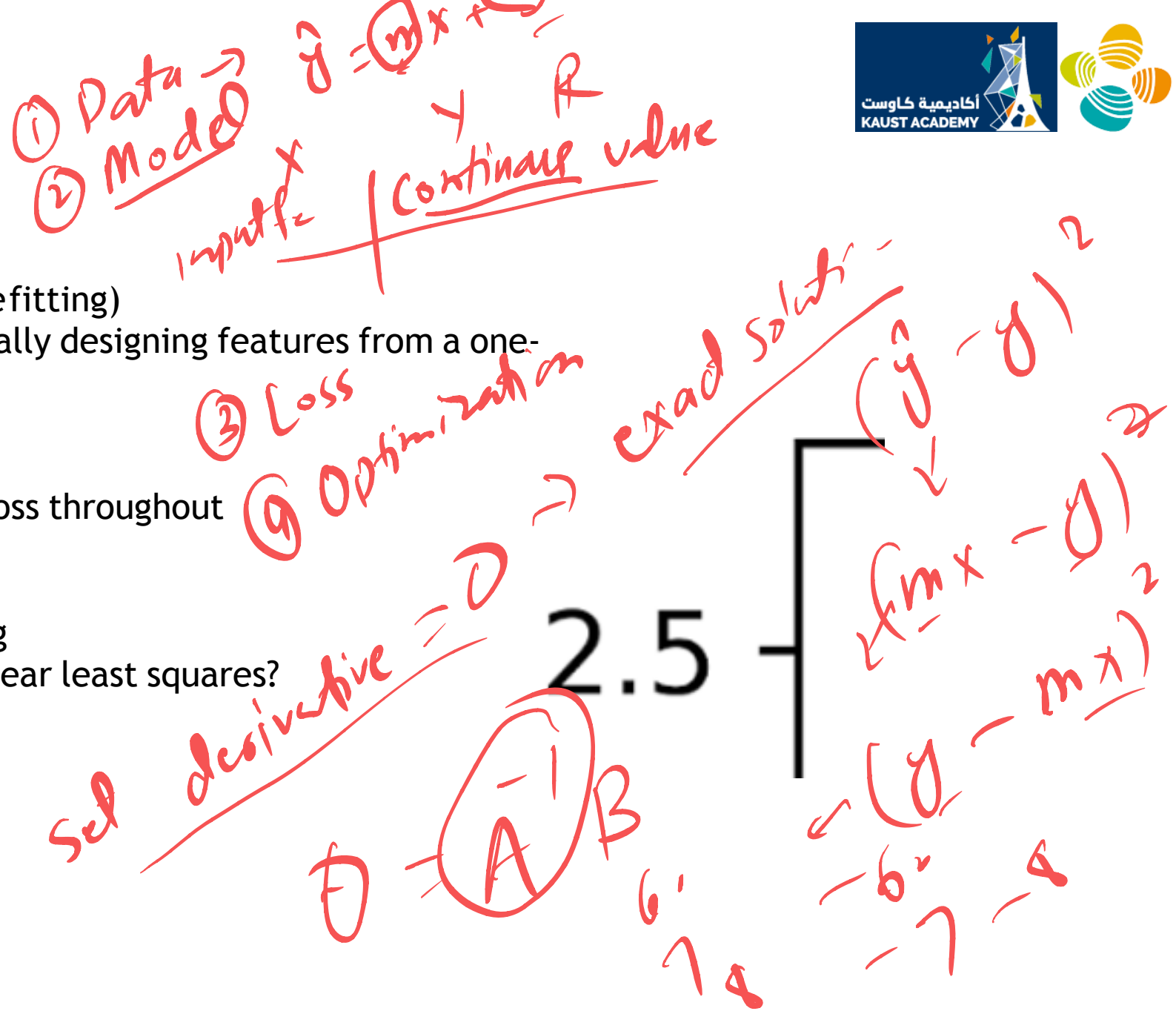
- Input scalar linear model (line fitting)
- Fitting polynomials (synthetically designing features from a one-dimensional input)

Design your loss function

- We used mean squared error loss throughout

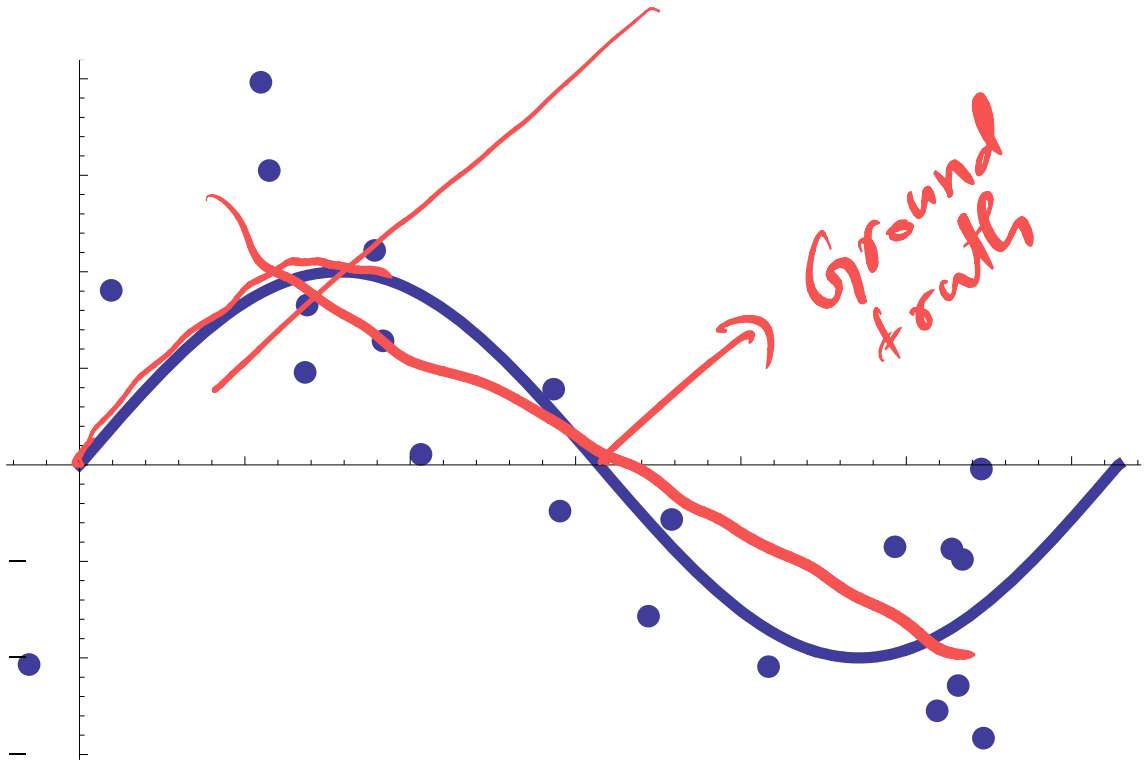
Finding optimal parameter fitting

- Closed form solution to the linear least squares?
- Why is it linear least squares?
- Solution is closed form



Bias and Variance

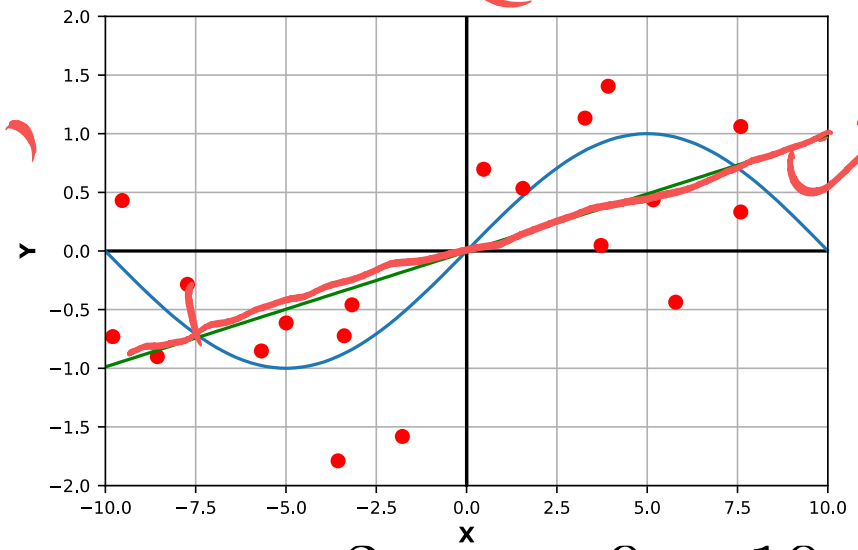
- What if Y has a non-linear response?



- Can we still use a linear model?

→ 2nd feature, 3rd feature

$\{1, x\}$

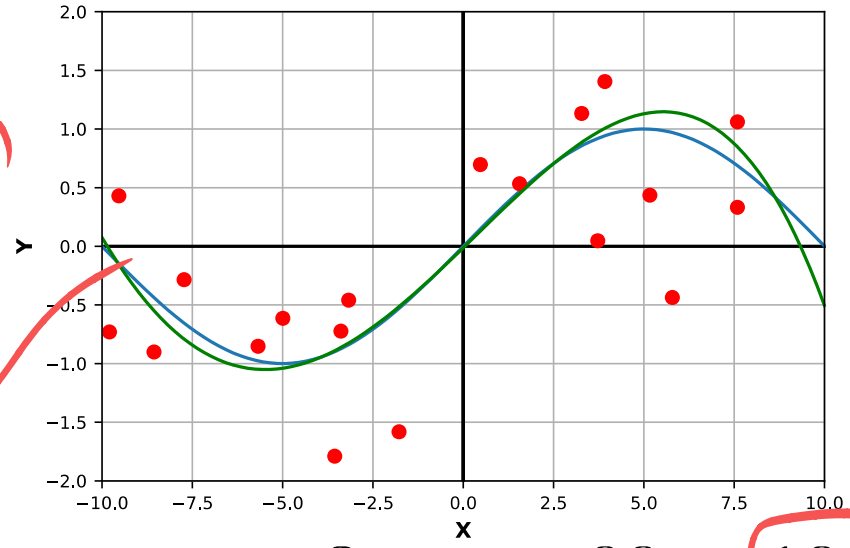


(a)

underfit

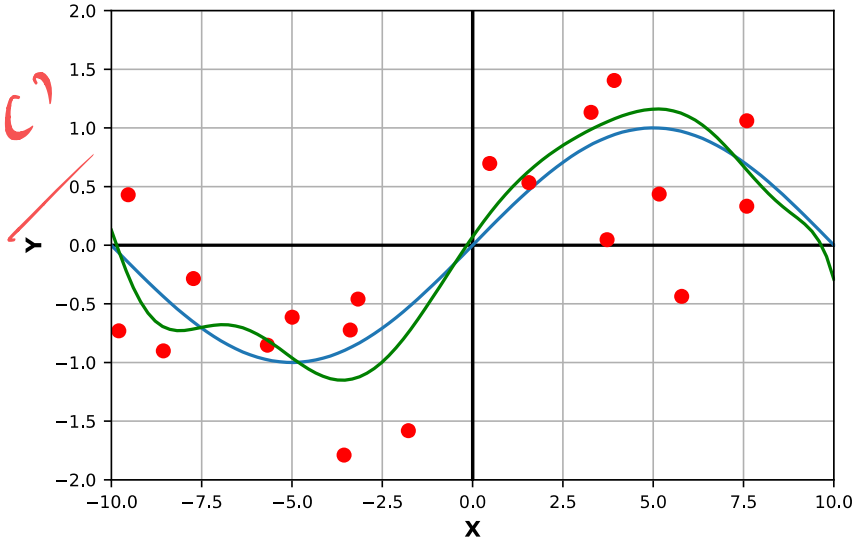
(b)

$\{1, x, x^2, x^3, x^4\}$



green → model
blue → actual

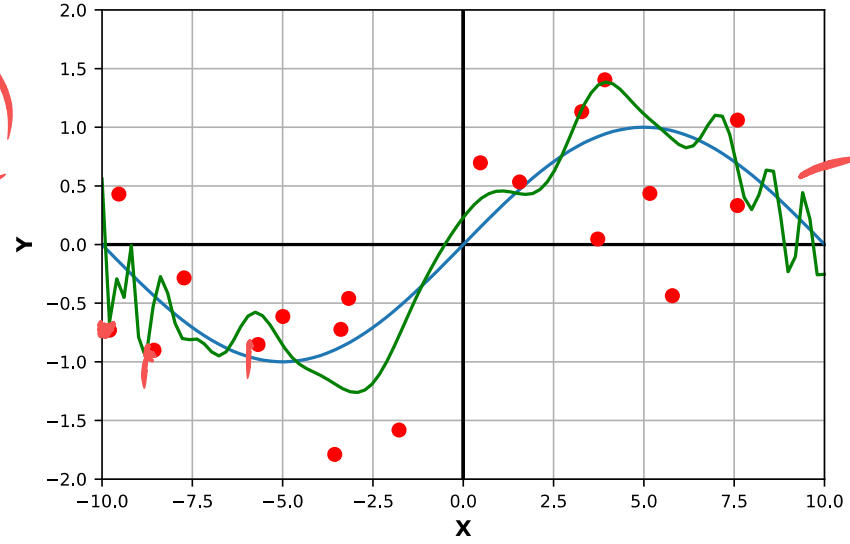
$\{1, x, x^2, ..., x^9, x^{10}\}$



(c)

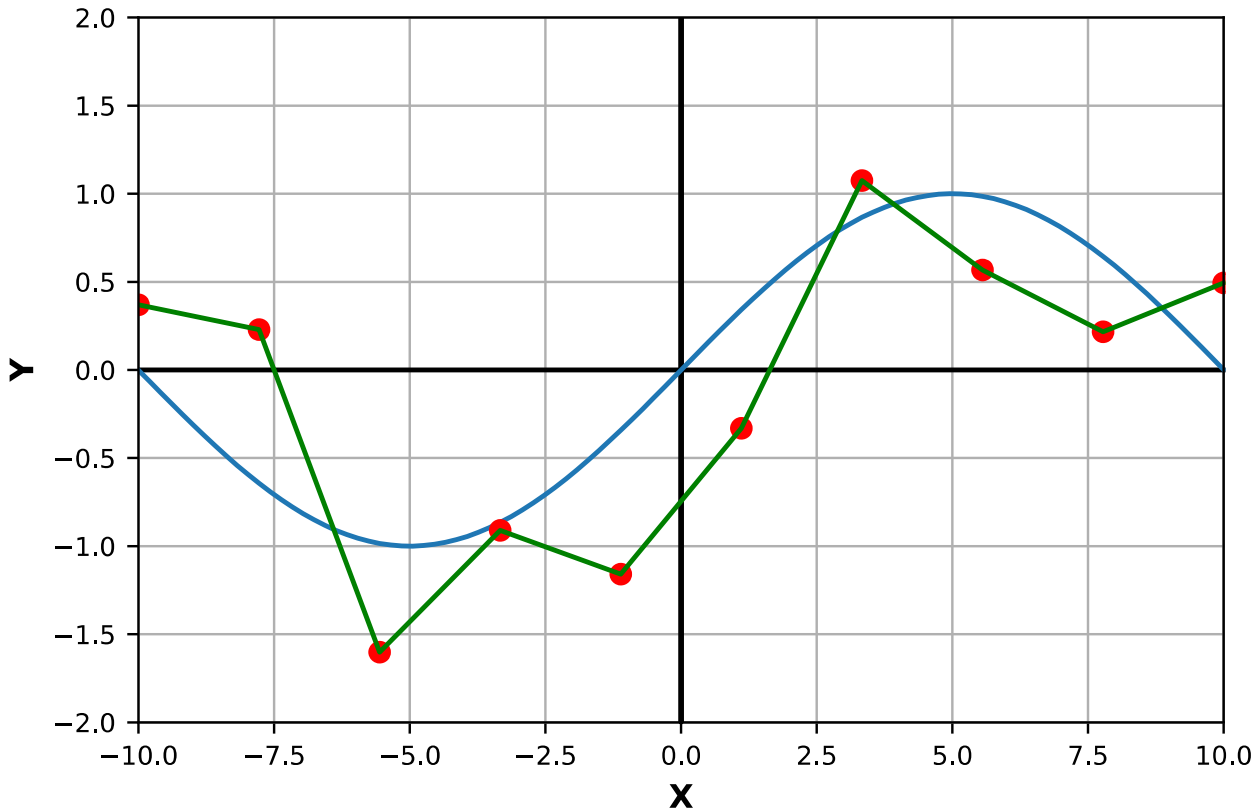
(d)

$\{1, x, x^2, ..., x^{99}, x^{100}\}$



→ overfitting

Real Bad Overfit?



lowered error rate

Bias-Variance Tradeoff

- So far we have minimized the error (loss) with respect to **training data**
  - Low training error does not imply good expected performance: **over-fitting**
- We would like to reason about the **expected loss (Prediction Risk)** over:
  - Training Data: {(y1, x1), …, (yn, xn)}
  - Test point: (y*, x*)
- We will decompose the expected loss into:

$$\mathbf{E}_{D,(y_*,x_*)}\left[(y_* - f(x_*|D))^2\right] = \text{Noise} + \text{Bias}^2 + \text{Variance}$$

Bias Variance Plot

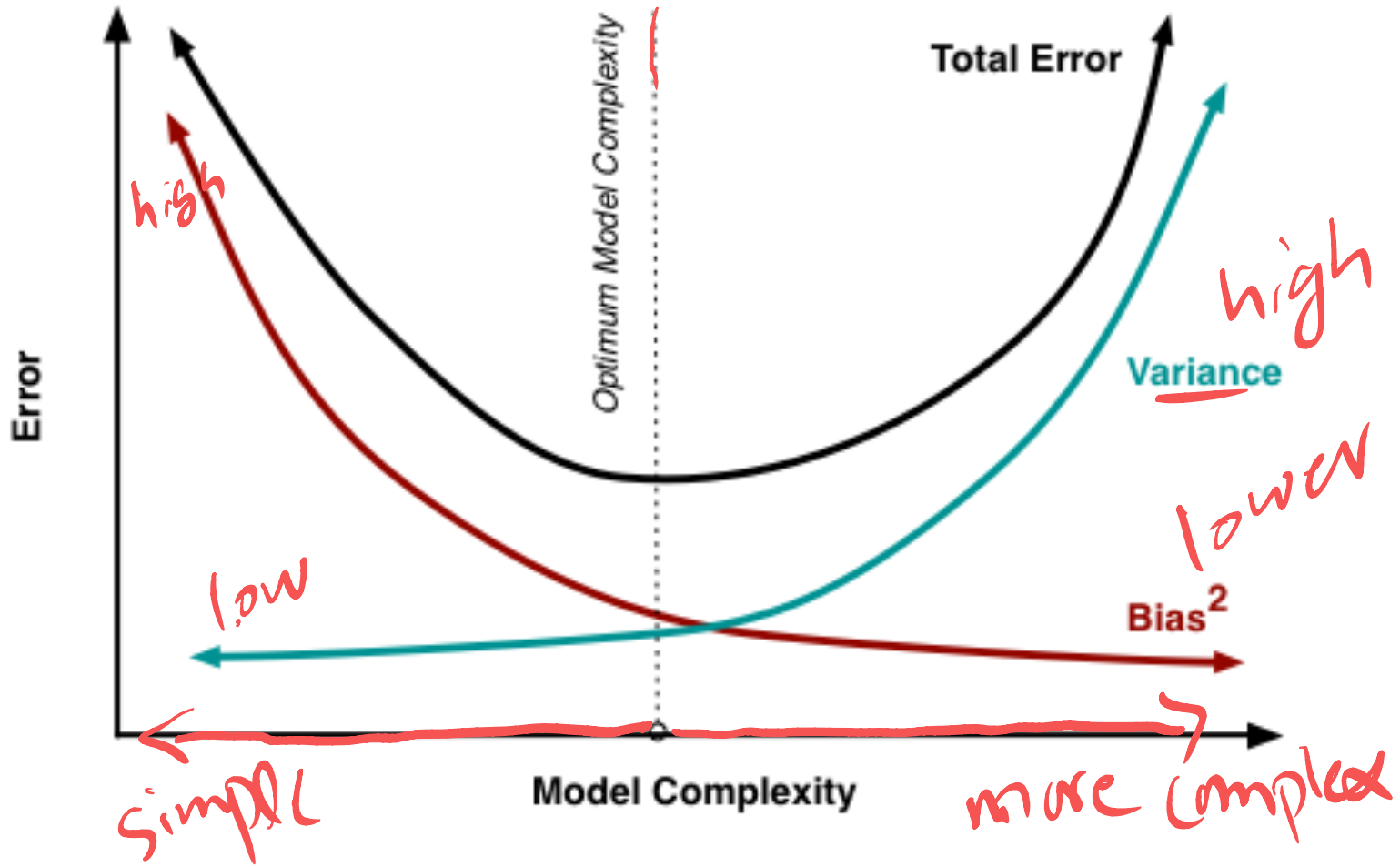


Image from http://scott.fortmann-roe.com/docs/BiasVariance.html

Data Split

- To ensure your model doesn't overfit to the training data, you should have another subset called **testing data.**
- You will evaluate your model against this subset, and based on its **metric score (e.g. accuracy)** you will decide if it's overfitting or not.
- But how should I split my data?

out of 100 → 20 sample

Data Split

- **Hold-out set**:
  - A portion of the dataset set aside and not used during training.
  - E.g. 80% for training and 20% for testing.

Issues:

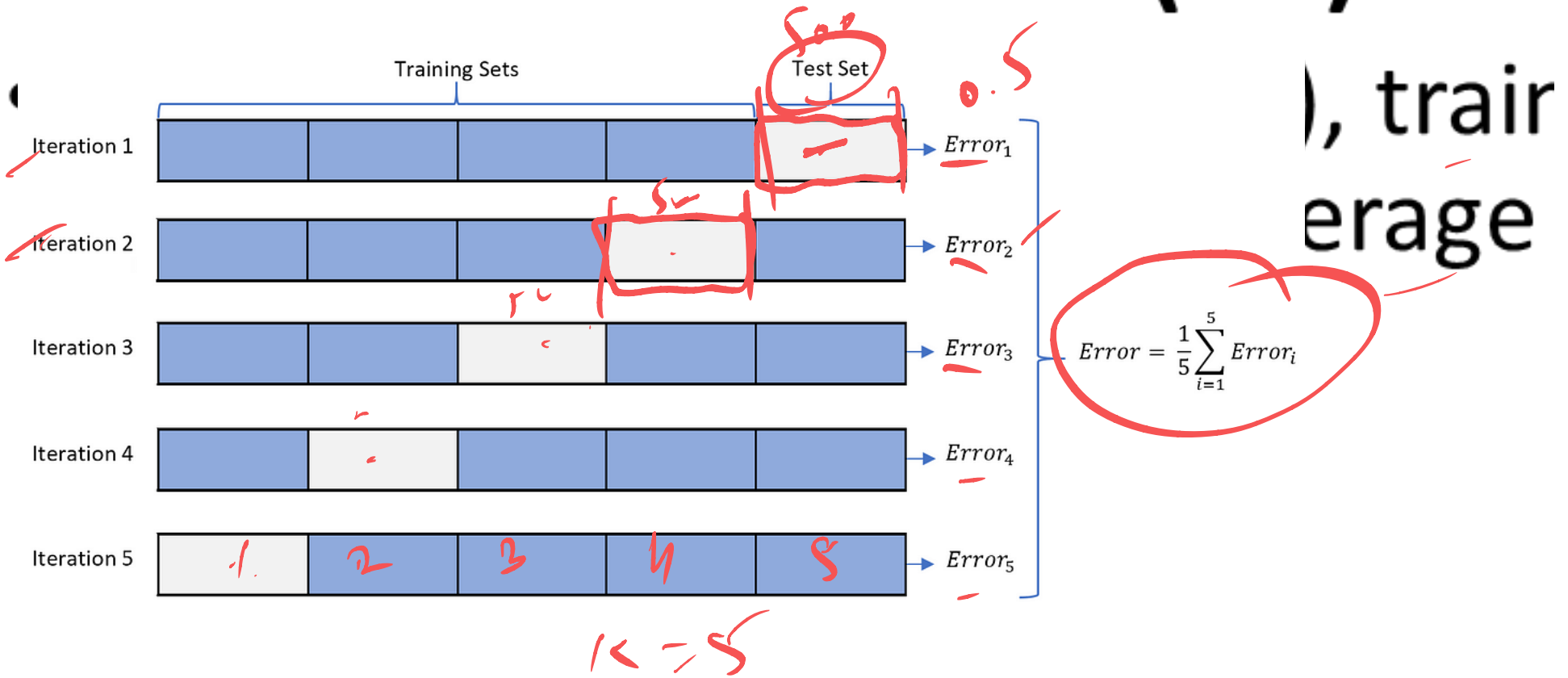
- Imagine you have these labels: [1, 1, 2, 2, 2, 3, 3, 3, 3, 3] and you took last 30% as test: [3,3,3]. You didn't include 1 and 2 in test!

  Solution: Always shuffle before split: [3, 2, 3, 1, 3, 2, 3, 1, 3, 2] → test: [1,3,2]
- My dataset is small. Taking 20% as test would not be representative!

  Solution: Use KFold.

Data Split

- **K-Fold Cross Validation (CV):**

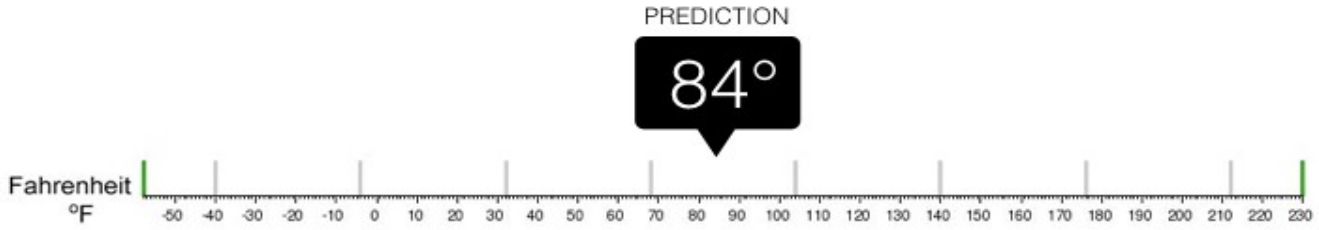


Regression VS classification

logistic regression

Regression

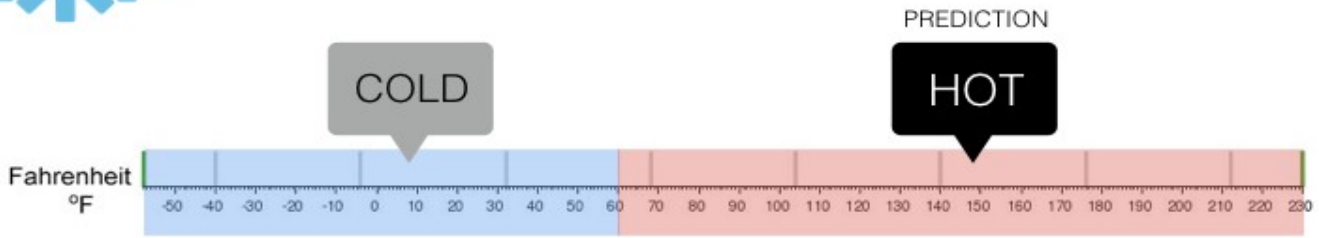
What is the temperature going to be tomorrow?



=> Continuous Values

Classification

Will it be Cold or Hot tomorrow?

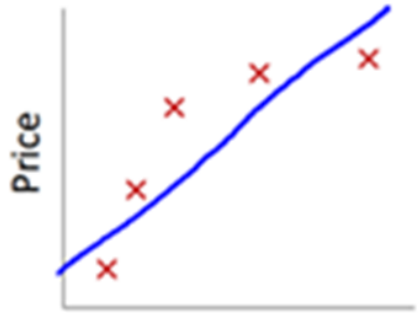


30% rain
70% Clouds

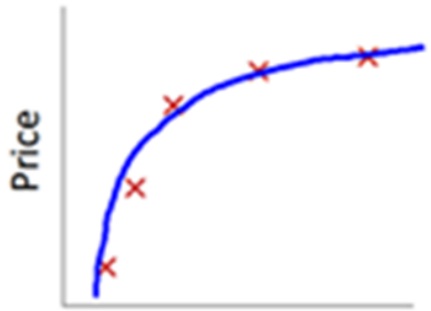
=> Discrete Values

Regression VS classification

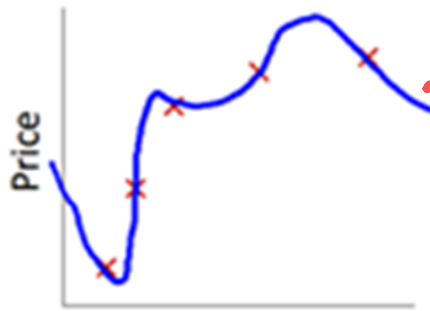
Regression:



Linear (overfitting)

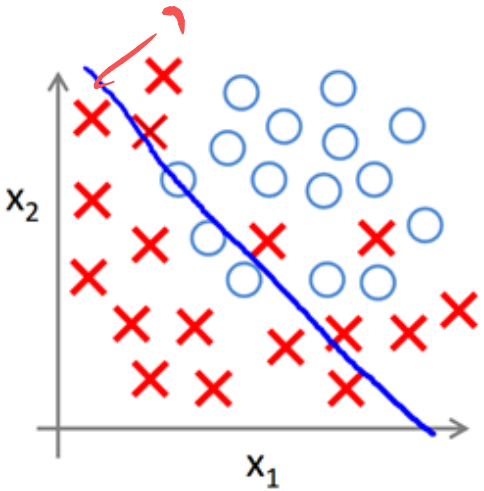


Polynomial

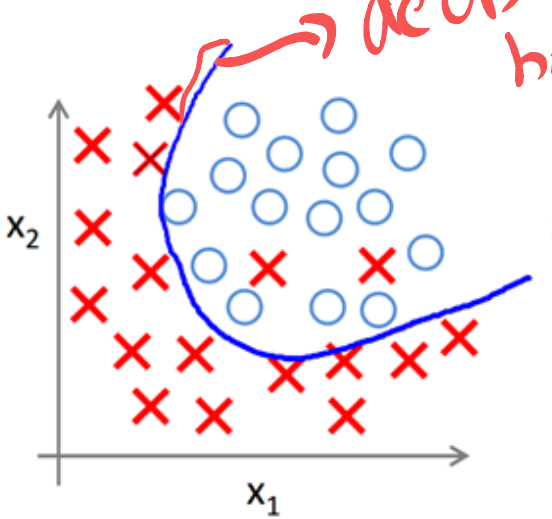


Very complex (overfitting)

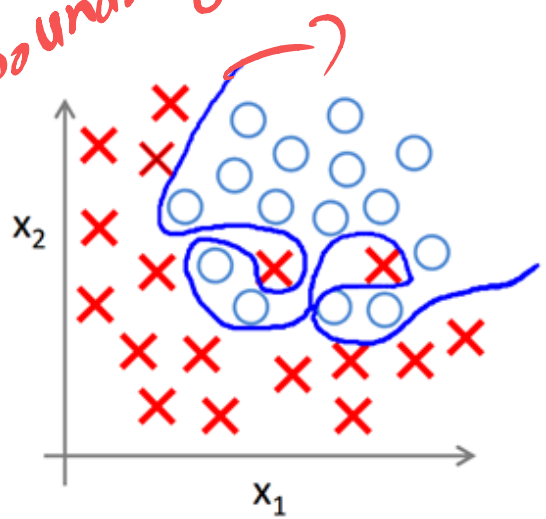
Classification:



Linear (overfitting)



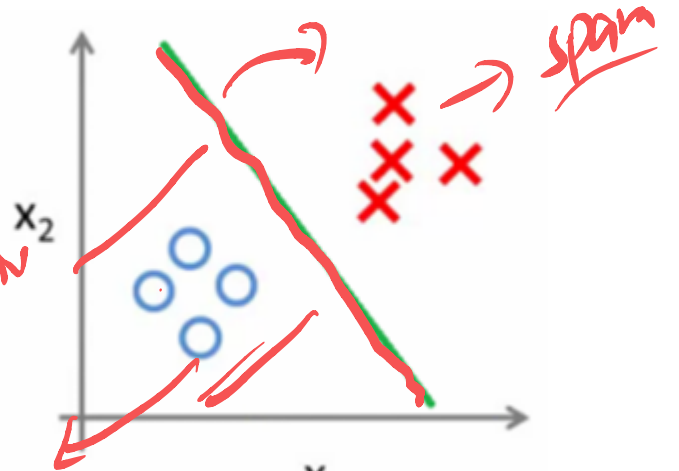
Polynomial



Very complex

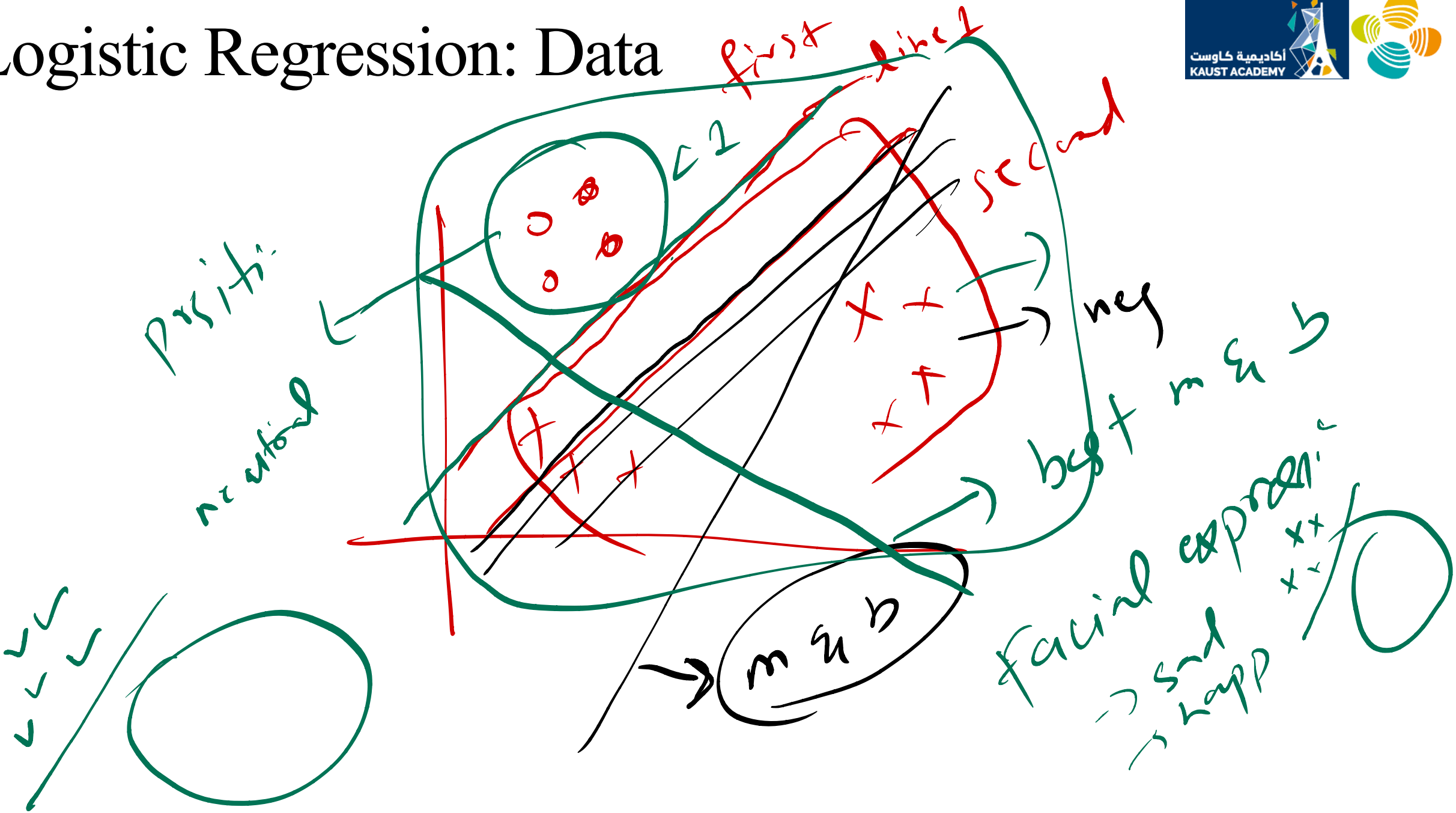
Classification Types

Binary Classification: Two possible outcomes (e.g., Yes/No, Spam/Not Spam).
How? Single Classifier.

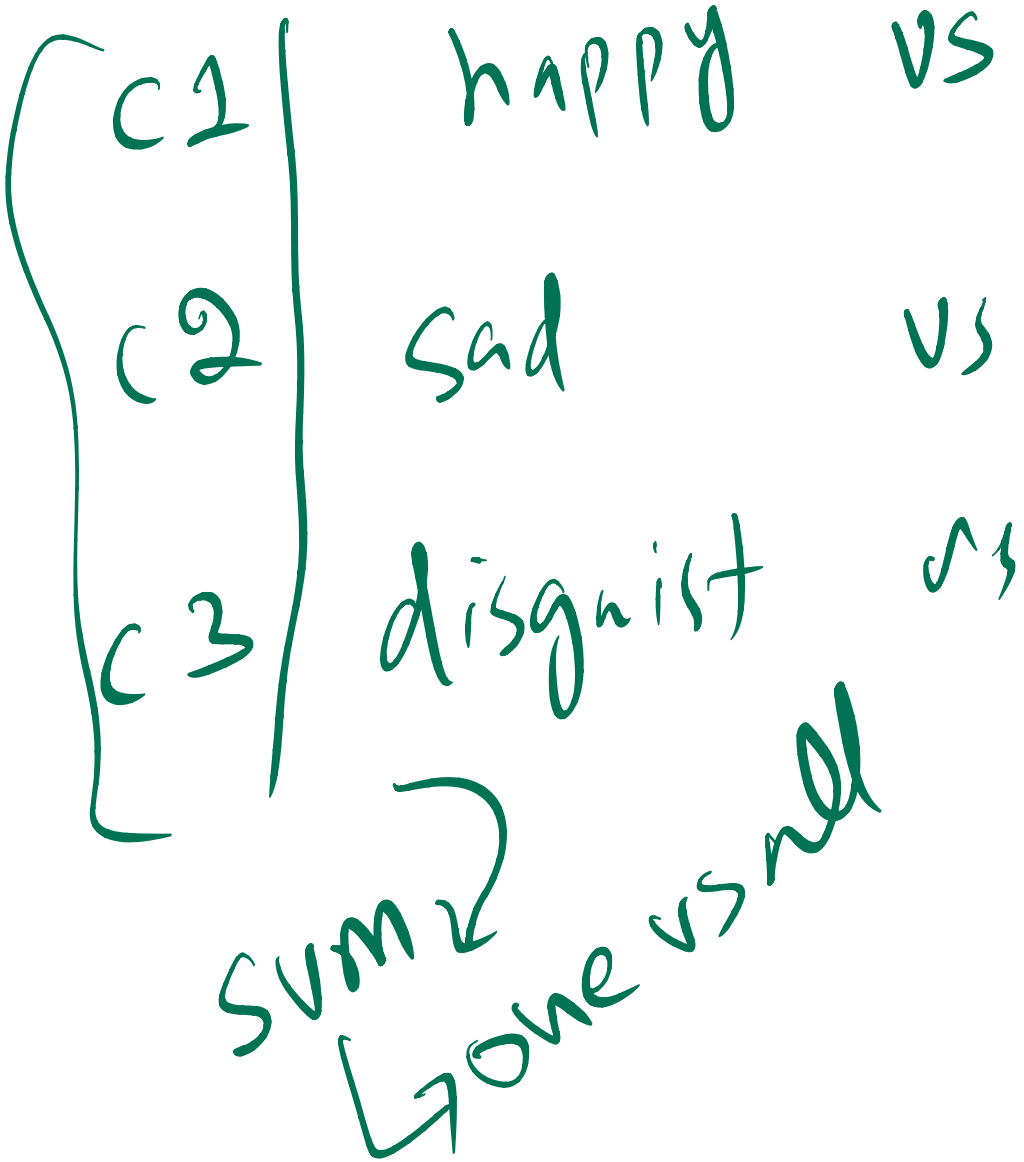


Multiclass Classification: More than two outcomes (e.g., Class A, B, C).
How? Multiple *Single Classifiers*.

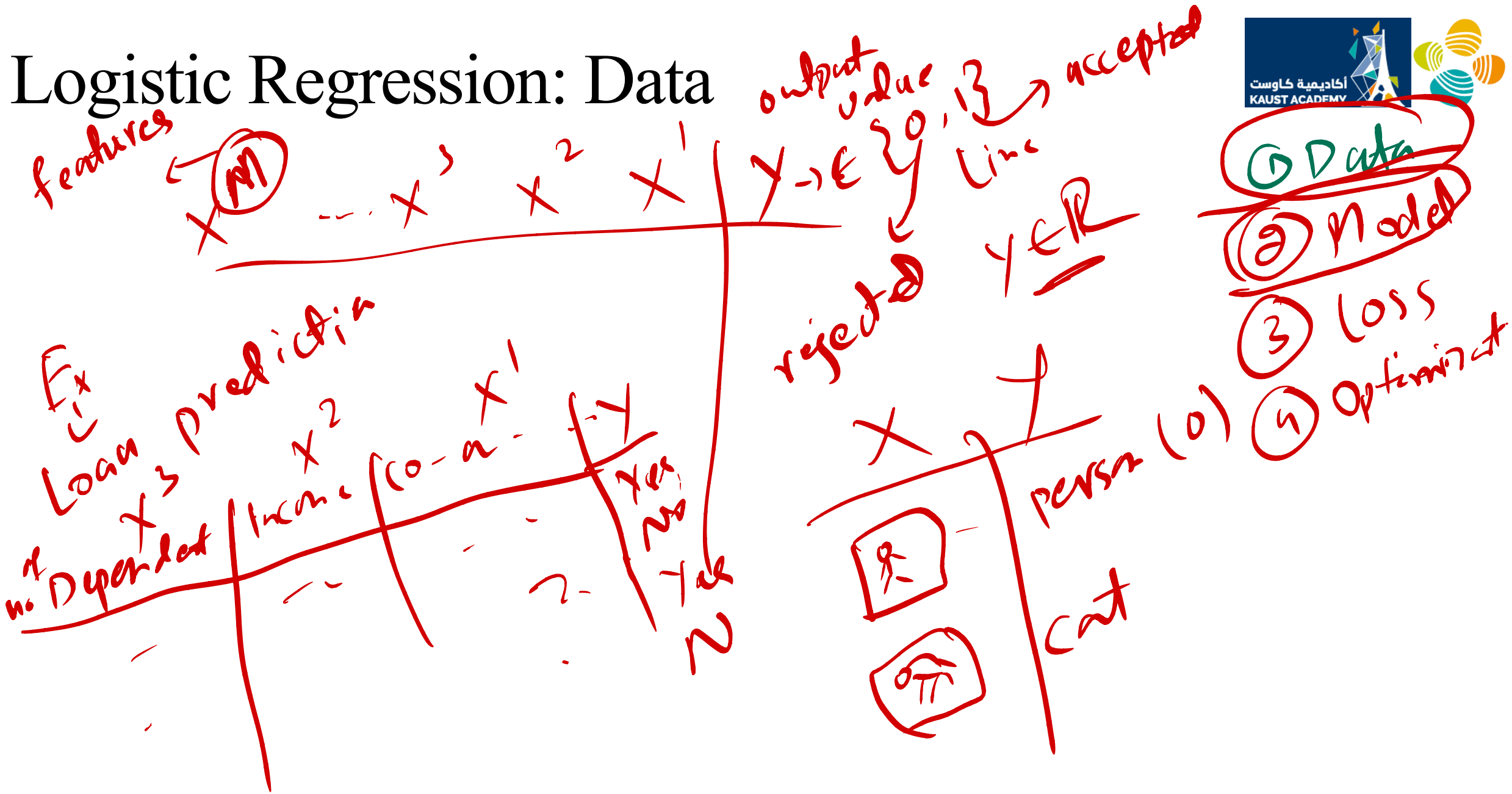
Logistic Regression: Data



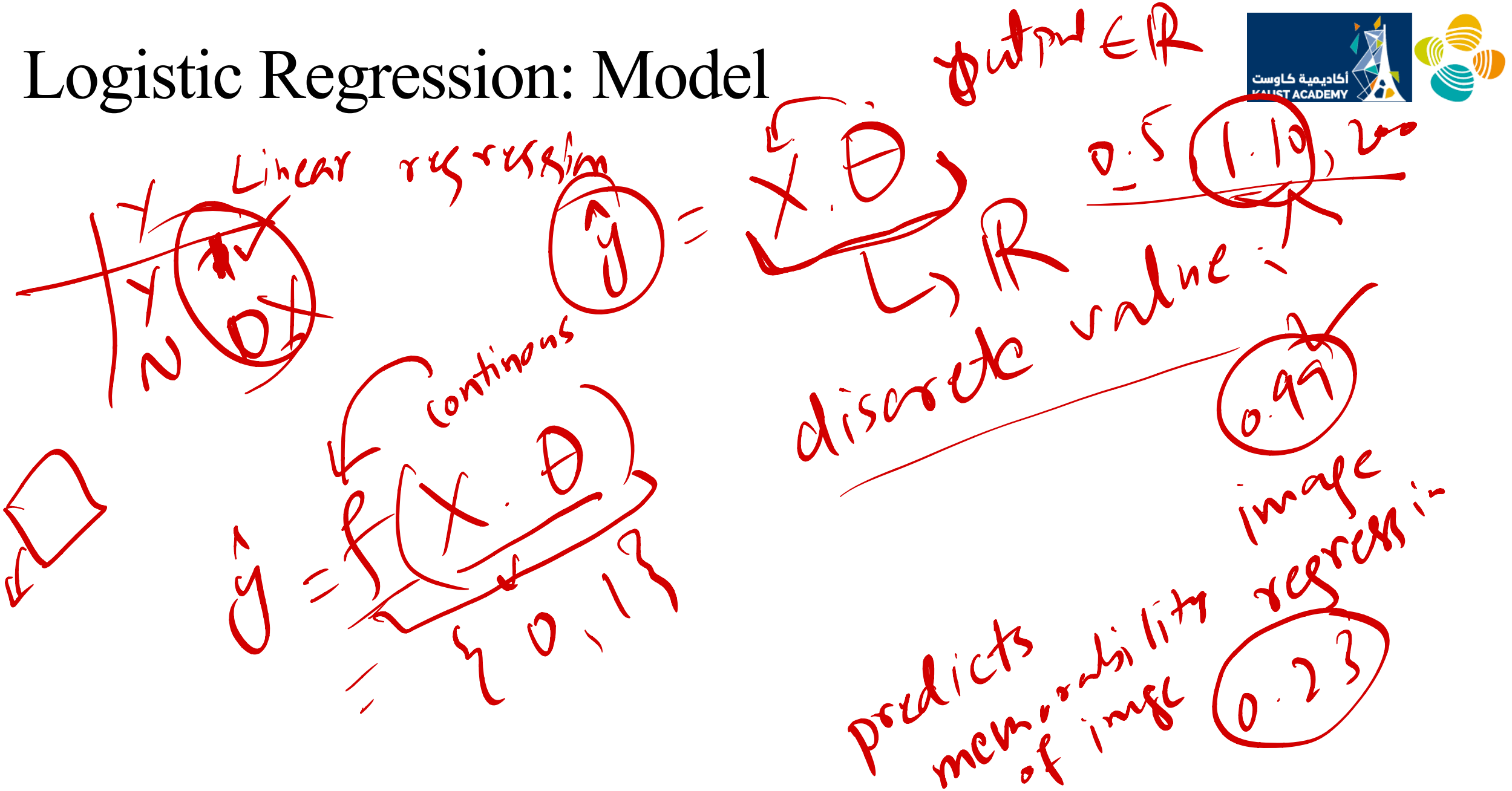
Logistic Regression: Data



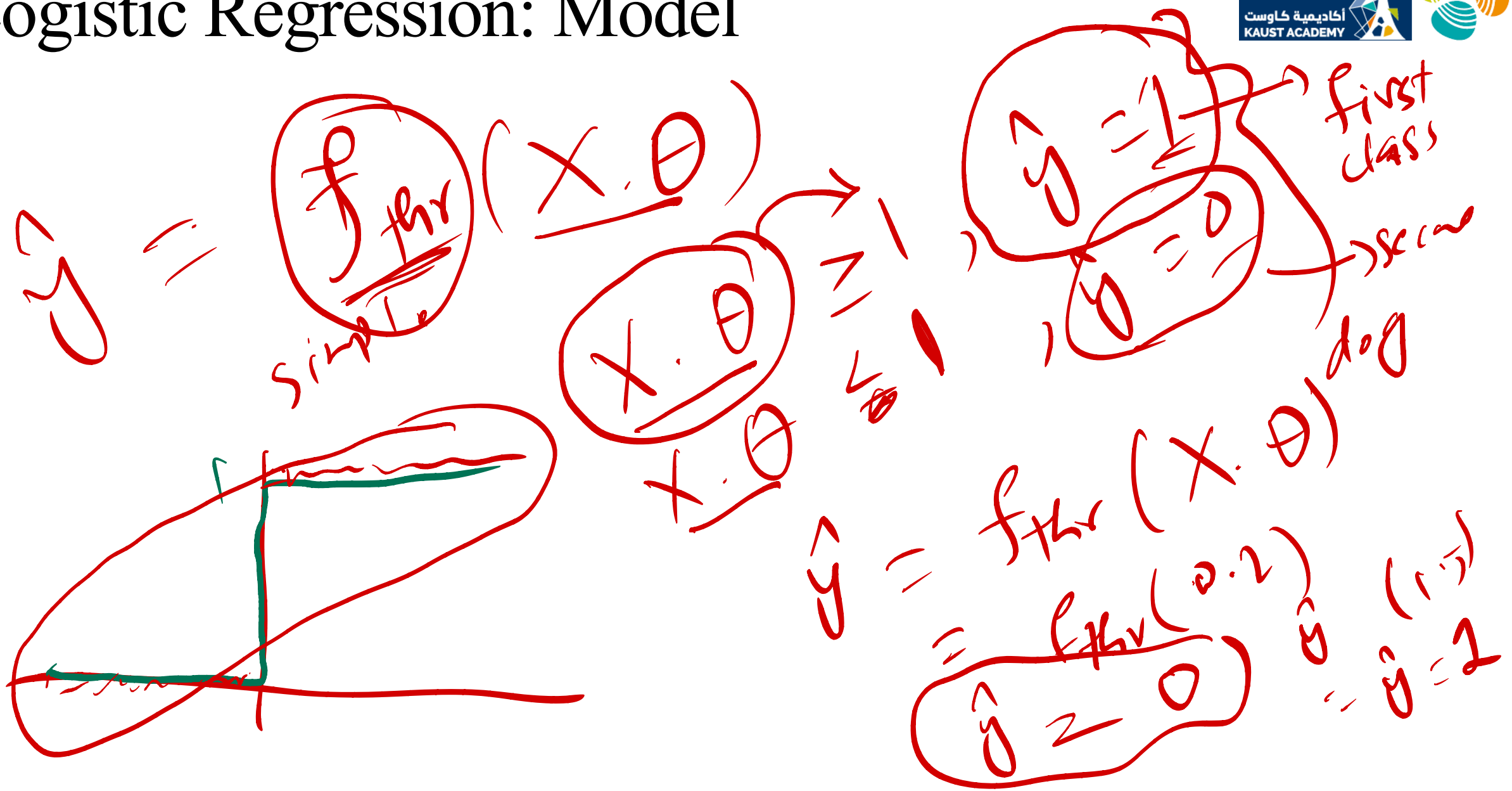
Logistic Regression: Data



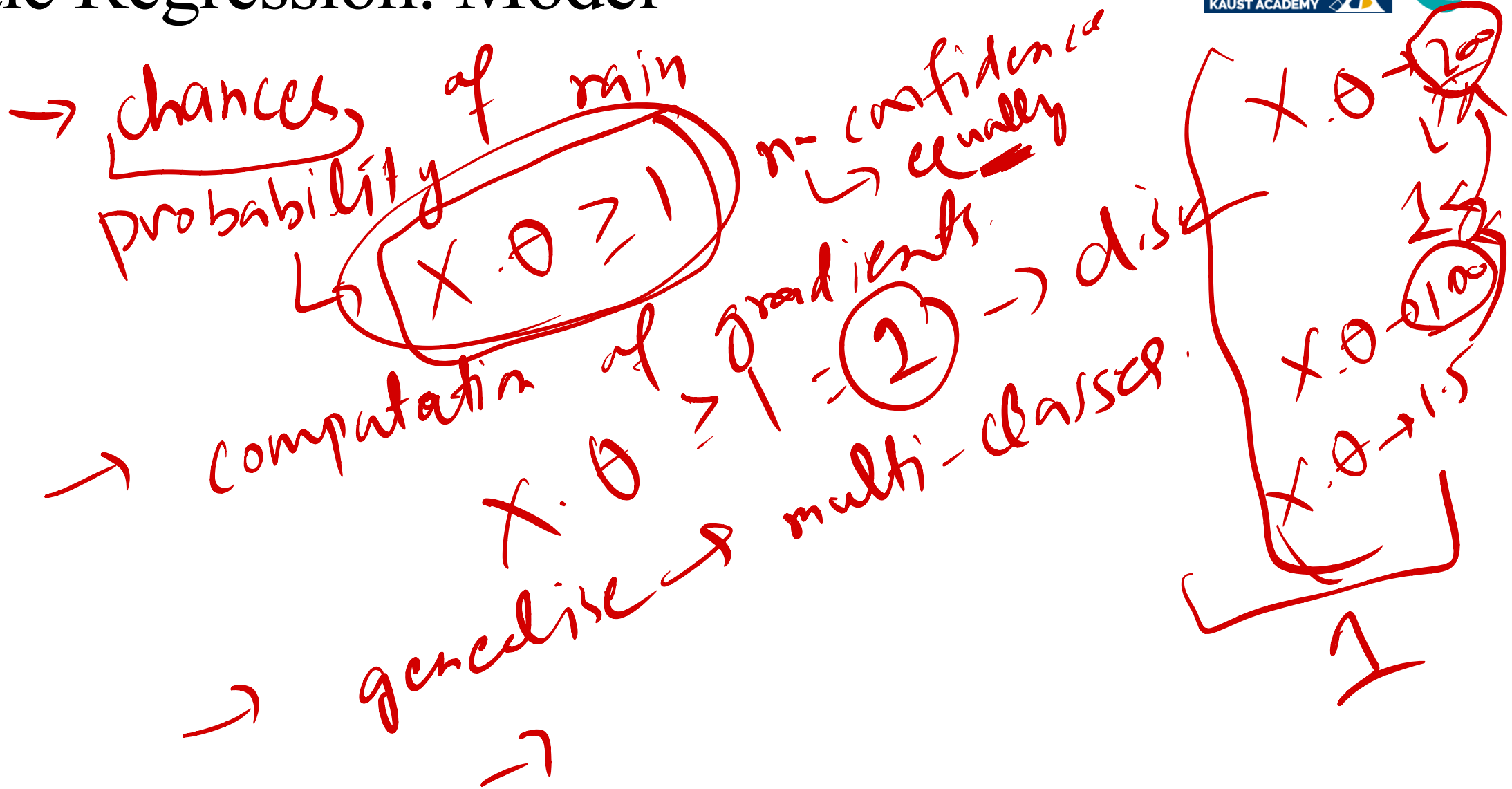
Logistic Regression: Model



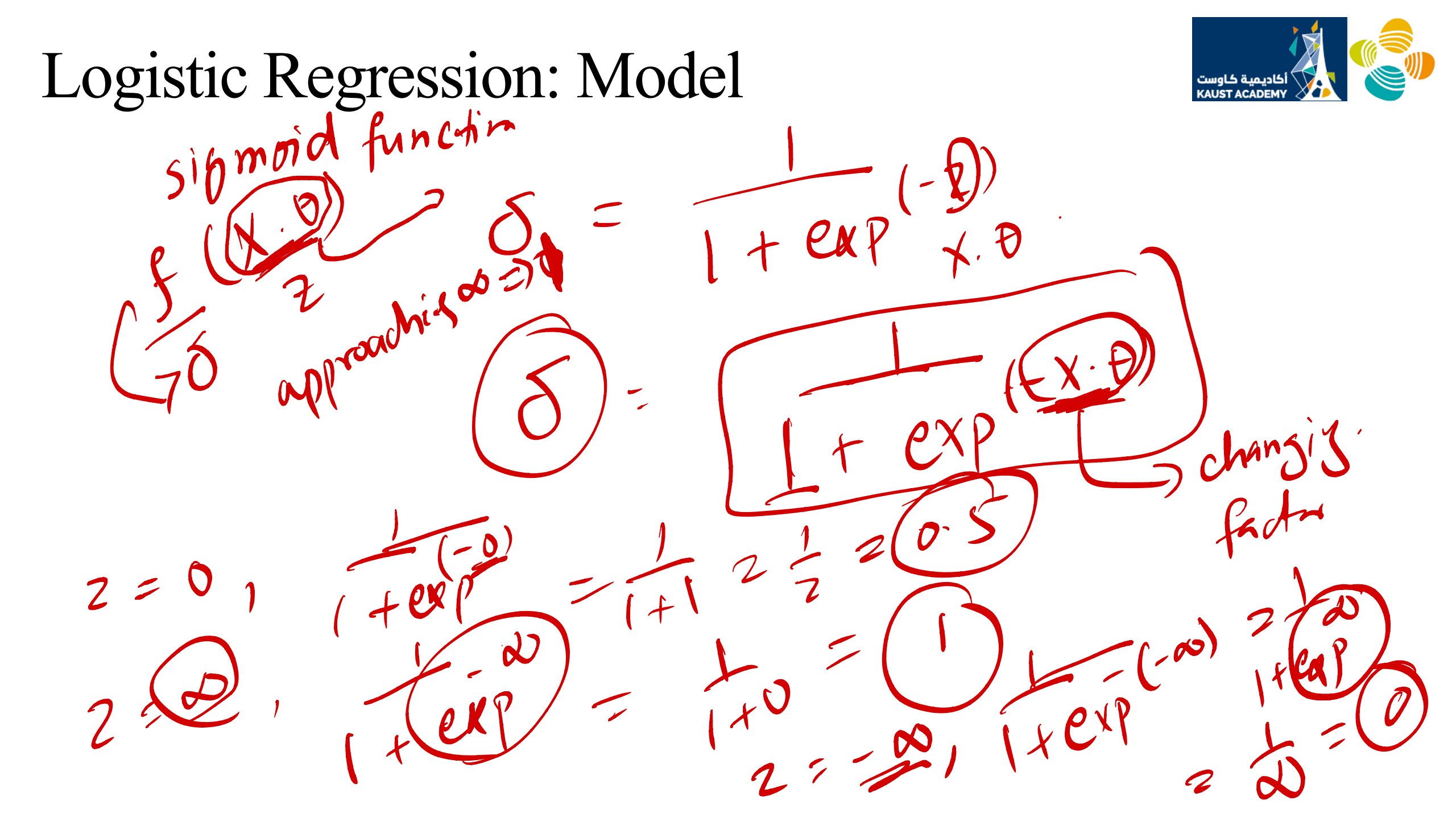
Logistic Regression: Model



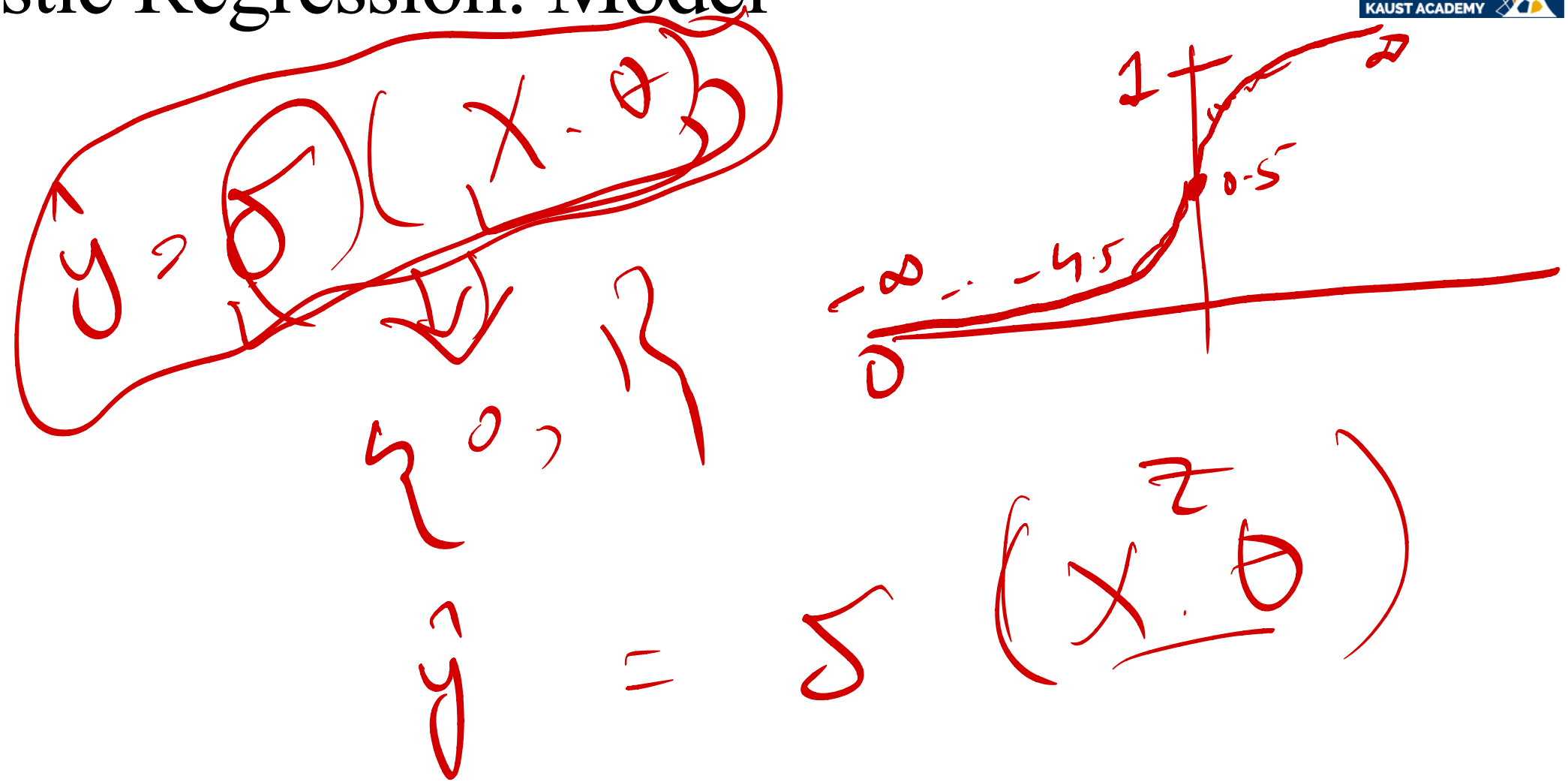
Logistic Regression: Model



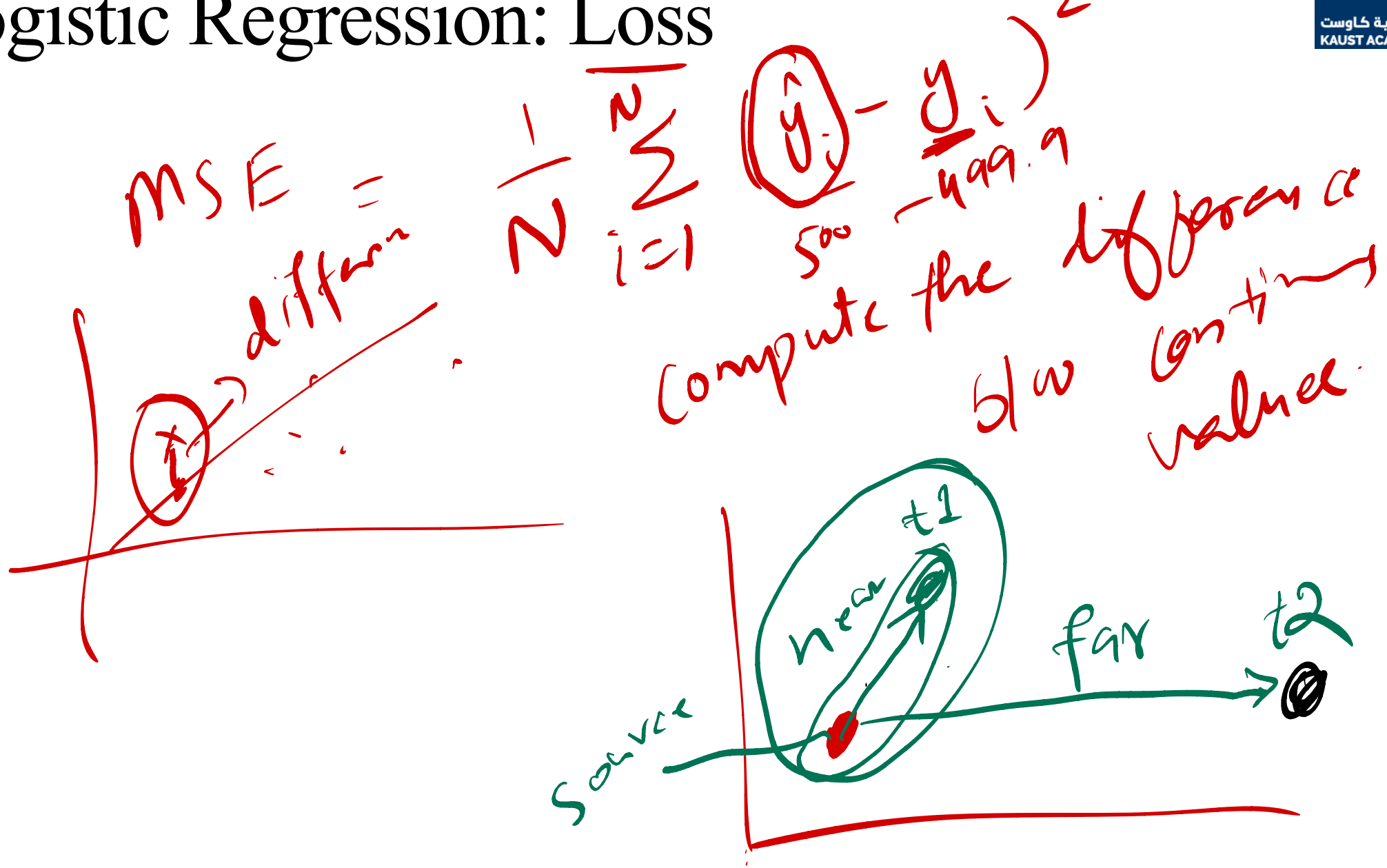
Logistic Regression: Model



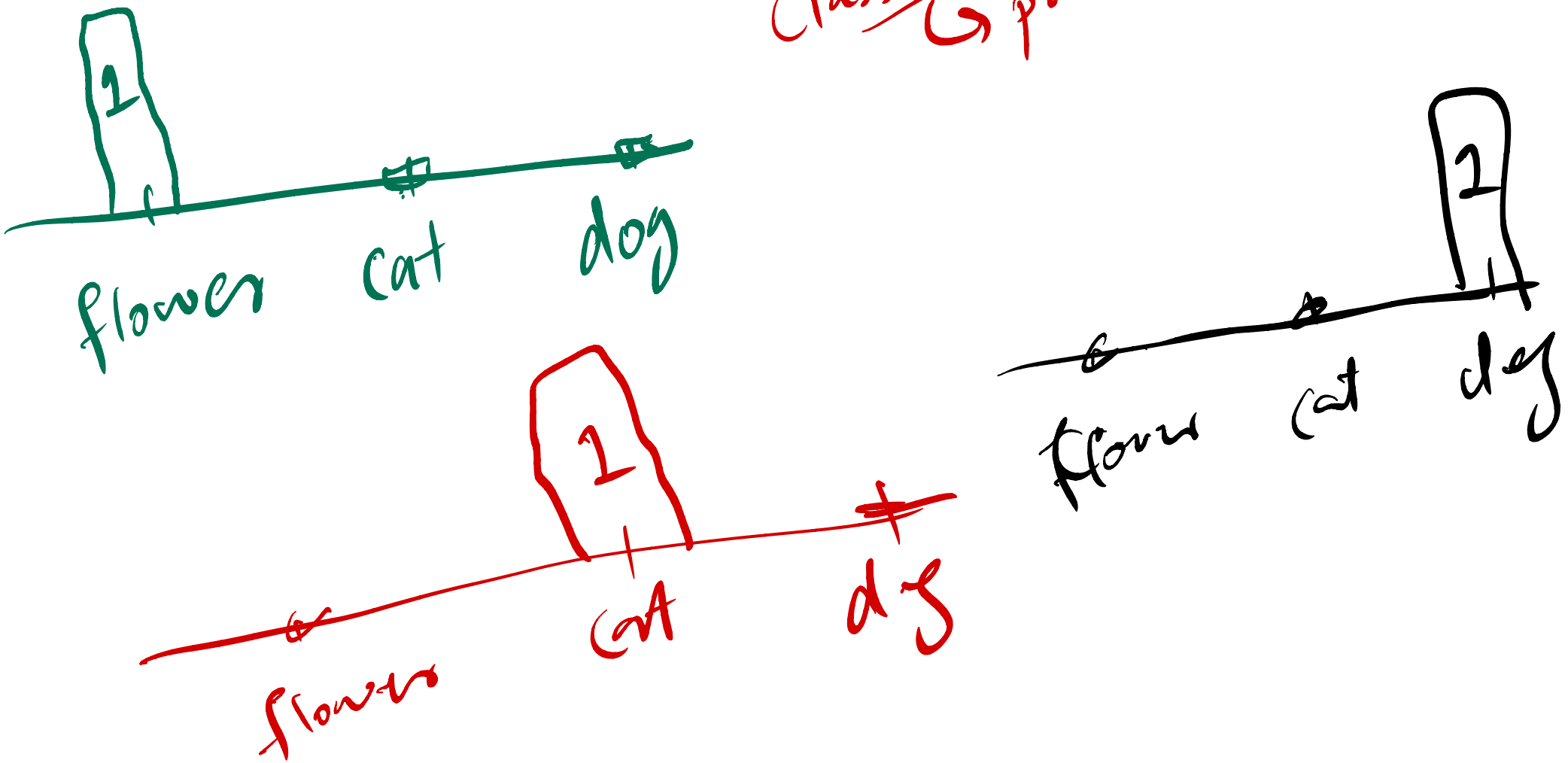
Logistic Regression: Model



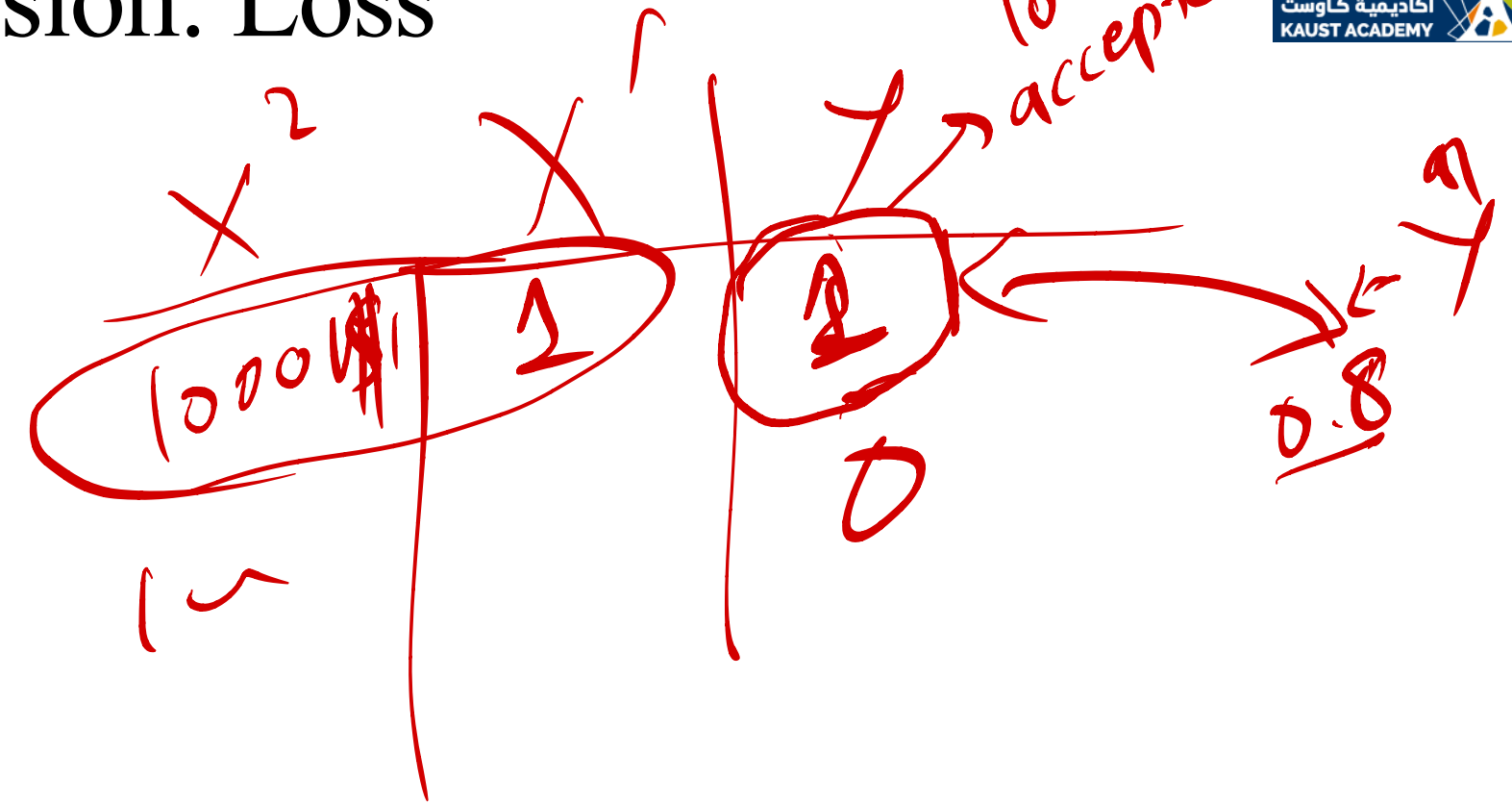
Logistic Regression: Loss



Logistic Regression: Loss



Logistic Regression: Loss



KL divergence

Logistic Regression: Loss

$$\mathcal{L}(\hat{y}, y) = -y \log \hat{y} - (1-y) \log(1-\hat{y})$$

$y = 0$, $\hat{y} \approx 1$

$$\mathcal{L} = -(1-y) \log(1-\hat{y})$$

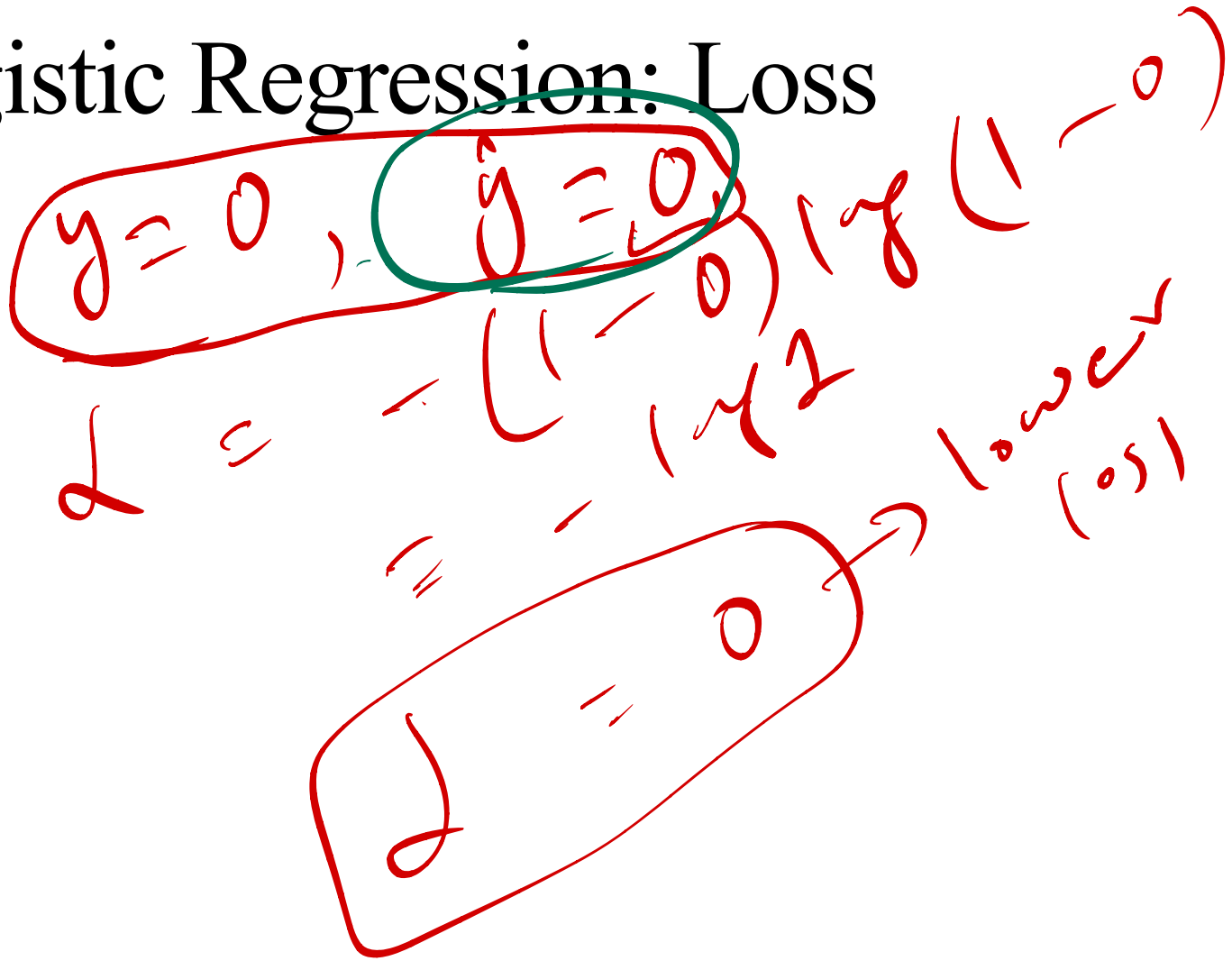
$$\mathcal{L} = -\log(1-1)$$

$$\mathcal{L} = -\log 0$$

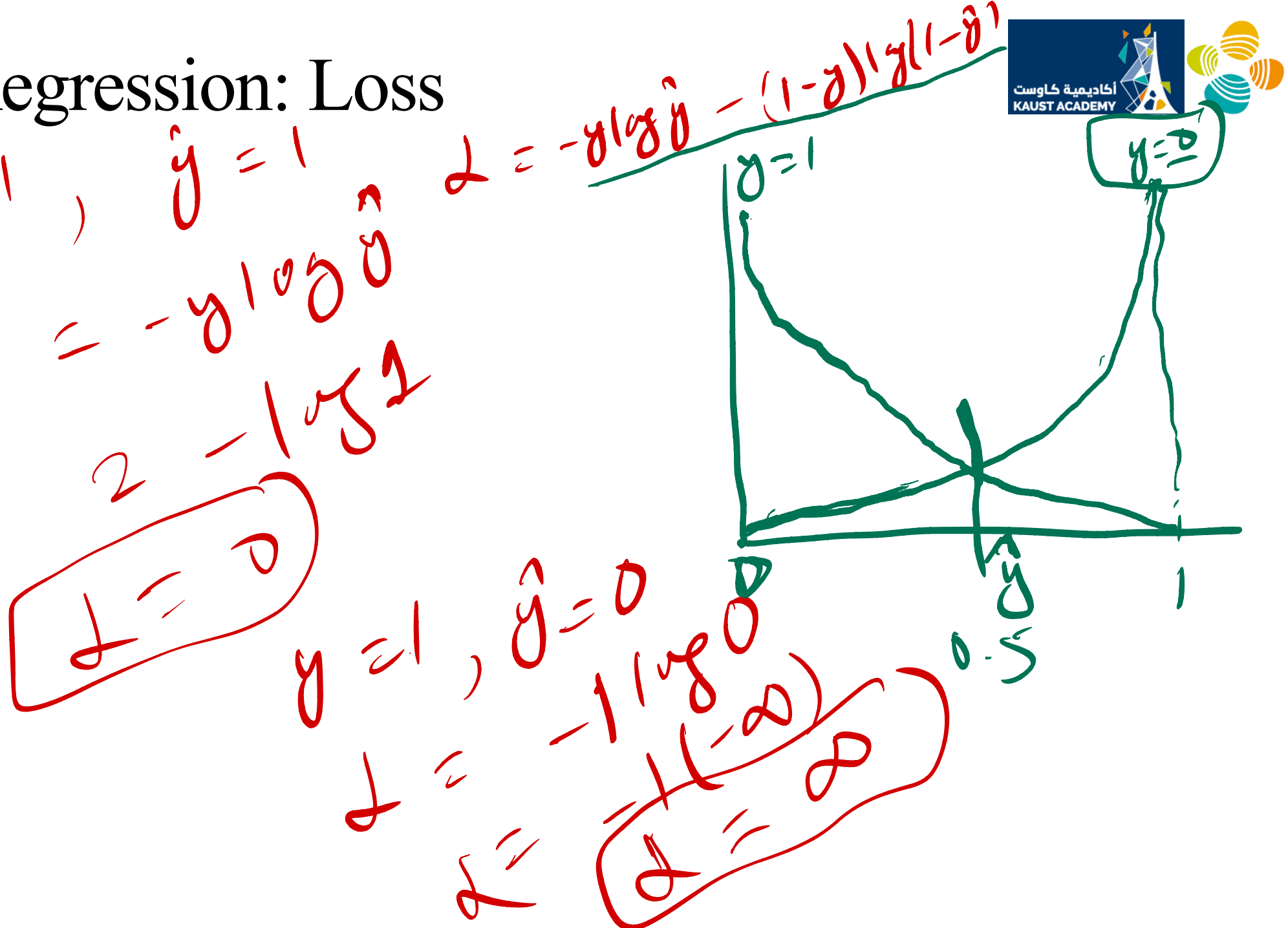
$$\mathcal{L} = +\infty$$

very high loss

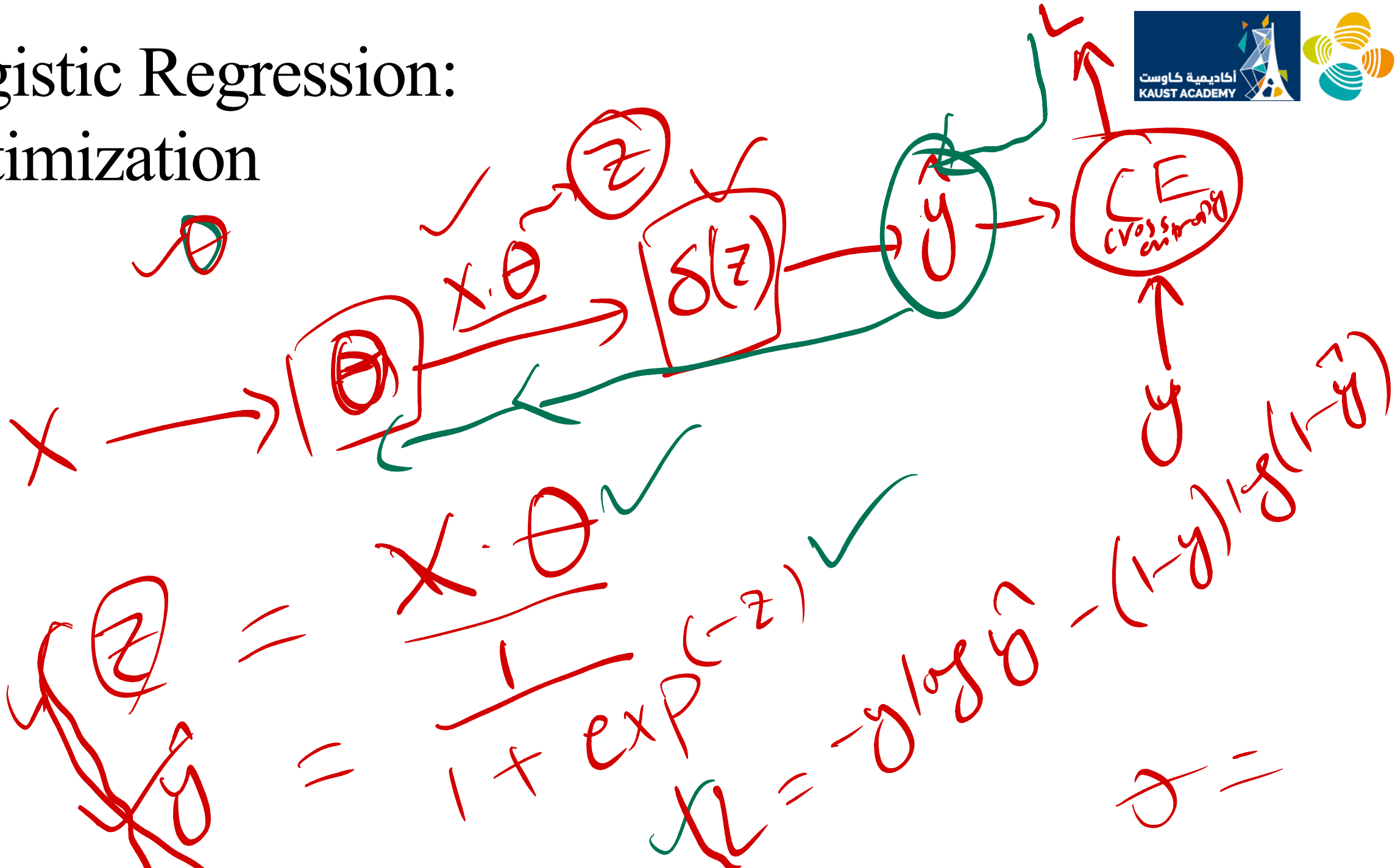
Logistic Regression: Loss



Logistic Regression: Loss



Logistic Regression: Optimization



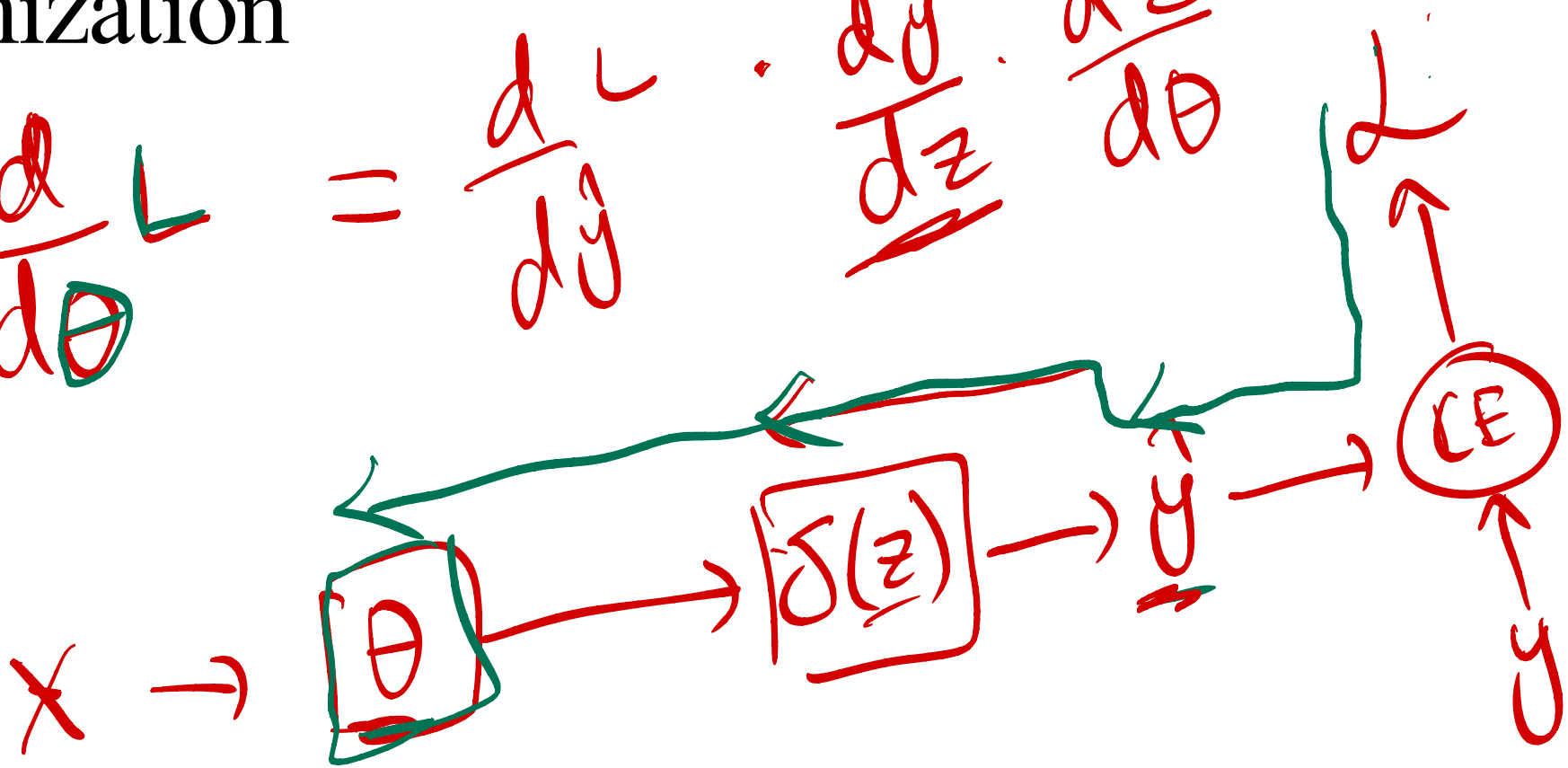
Logistic Regression: Optimization

Linear → set derivative = 0 → system linear equation → closed

non-linearity → open

Logistic Regression: Optimization

$$\frac{d}{d\theta} L = \frac{d}{d\hat{y}} L \cdot \frac{d\hat{y}}{dz} \cdot \frac{dz}{d\theta}$$



Logistic Regression: Optimization

$$\frac{dL}{d\hat{y}} = \frac{d}{d\hat{y}}\left(-y\log\hat{y} - (1-y)\log(1-\hat{y})\right)$$

$$\frac{dL}{d\hat{y}} = \frac{\hat{y} - y}{\hat{y}(1-\hat{y})}$$

Logistic Regression: Optimization

$$\frac{d\hat{y}}{dz} = \frac{d}{dz}\left(\frac{1}{1+\exp(-z)}\right) = \frac{d}{dz}\left(1+\exp(-z)\right)^{-1}$$

$$\frac{d\hat{y}}{dz} = \hat{y}(1-\hat{y})$$

$$\frac{dz}{d\theta} = \frac{d}{d\theta}(X\theta) = X^T$$

Logistic Regression: Optimization

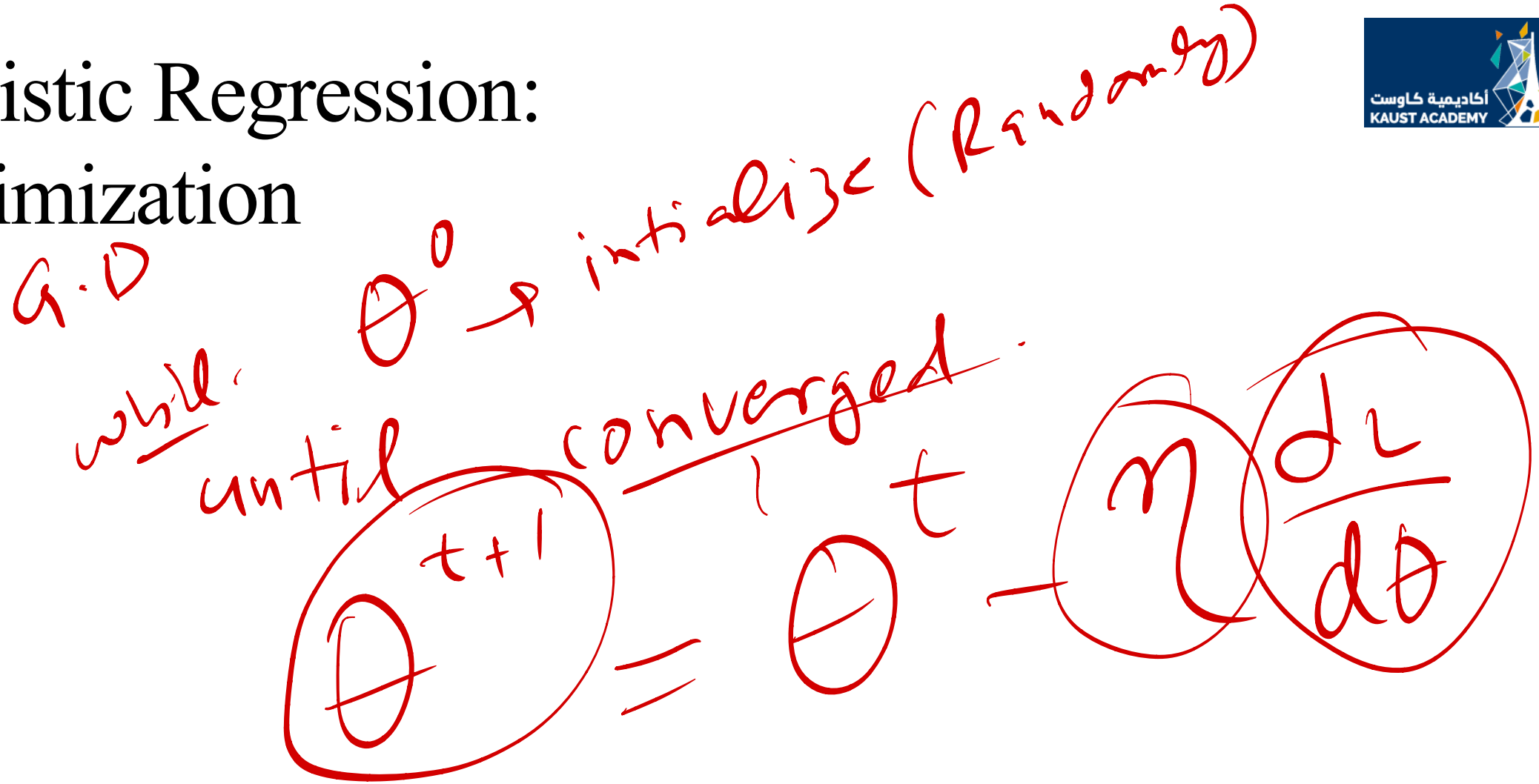
$$\frac{dL}{d\theta} = \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{dz} \cdot \frac{dz}{d\theta}$$

$$\frac{dL}{d\theta} = X^T \left[\frac{(\hat{y} - y)}{\hat{y}(1-\hat{y})} \cdot \hat{y}(1-\hat{y}) \right]$$

linear in θ

$$\frac{dx\theta}{d\theta} = 0$$

Logistic Regression: Optimization



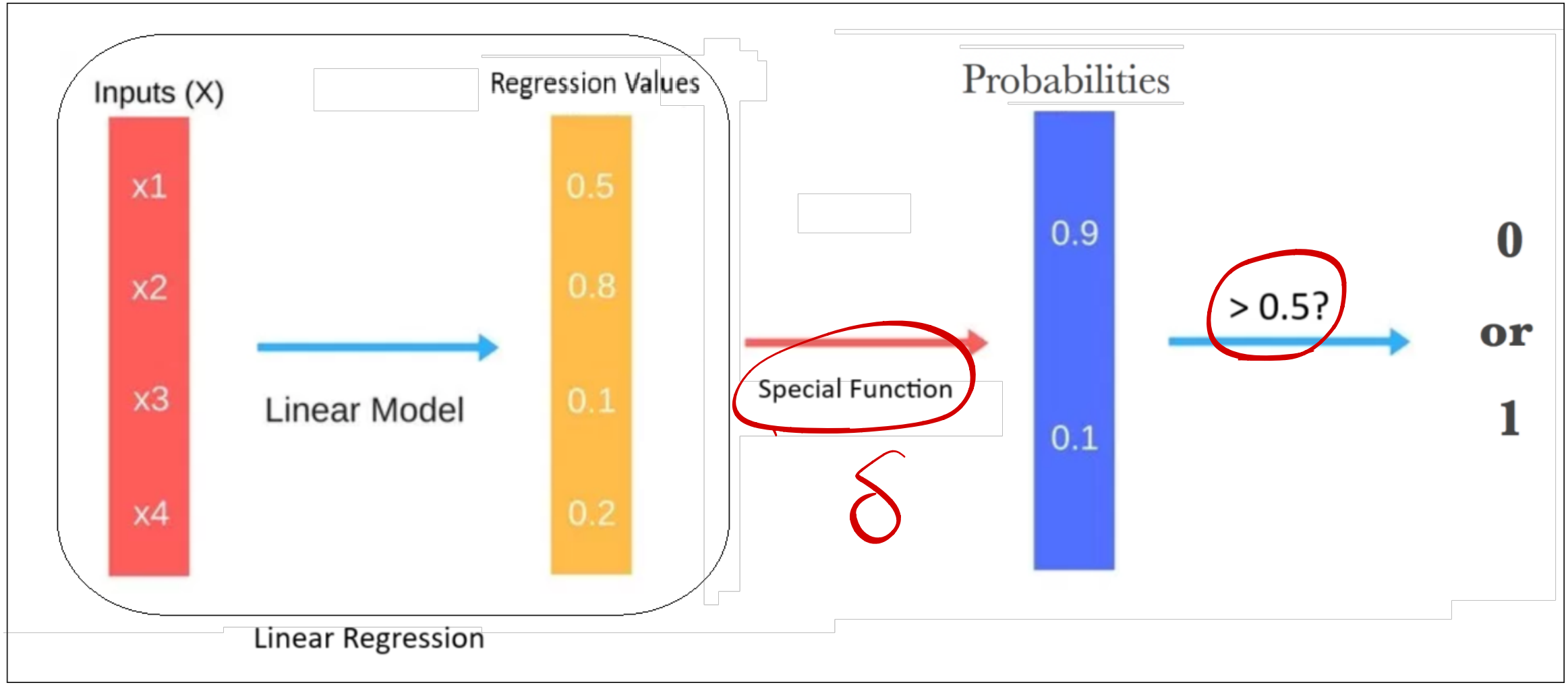
Classifiers

Classifiers are **regression** models with two key modifications:

- Pass the regression outputs through a **special function** to convert them into probabilities.
- Replace the Mean Squared Error loss with a **new loss function** designed for probabilities.

Applying these two to the Linear Regression should give Linear Classifier (called **Logistic Regression**).

Logistic Regression



Special Function

We are searching for a function that have the following characteristics:

1. Could convert any arbitrary input values to **[0,1] range (Probabilities)**.
2. **Smooth and Differentiable**. Because we want to find the minima later, right?

Any ideas?

Sigmoid Function (for Binary Classification)

$$\sigma(z) = \frac{1}{1 + exp(-z)}$$

$$\sigma'(z) = \sigma($$

Assuming you have negative and positive classes, its output would be the probability of the positive class.

But what if we have multiclass problem?🤔

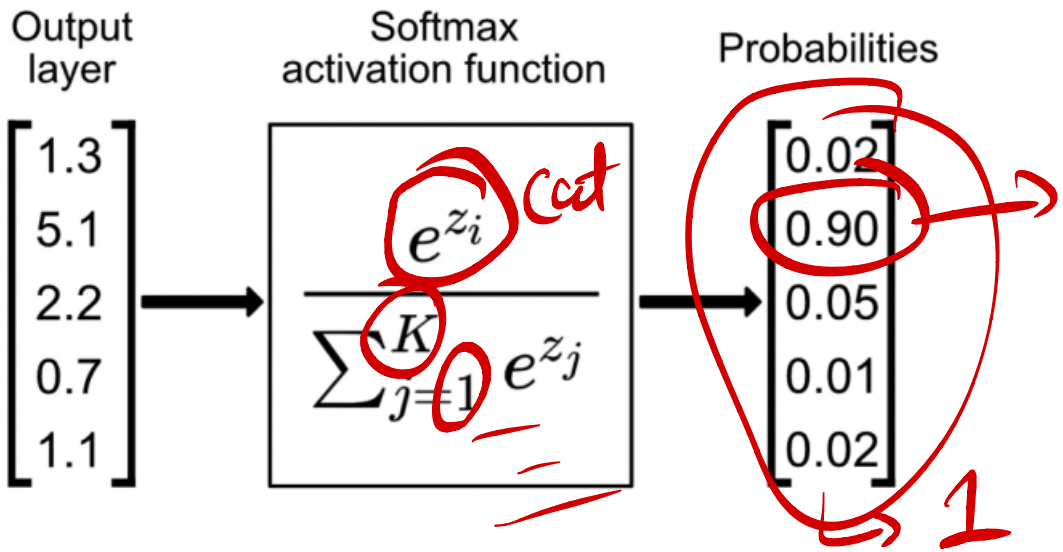
$$\lim_{z \to -\infty} \sigma(z) = 0$$

$$\lim_{z \to \infty} \sigma(z) = 1$$

Softmax Function (for Multiclass Classification)

E.g. Divide the cat value by the cat, dog and deer values to get the cat probability.

Assuming you have K classes, its output would be **the probability of the ith class**. So, you will run it K times to get the probability of each class.



Note: Sigmoid is a special case of Softmax. Optional: Can you prove it?

Classification Loss

We are searching for a loss that have the following characteristics:

1. **Probabilistic behaviour**: Should work with probabilities, not raw numbers.
2. **Sensitivity**: It should heavily penalize incorrect confident predictions (e.g., predicting 0.9 when the true label is 0).
3. **Differentiable**: Must be smooth and differentiable to enable optimization.

Any ideas?

Classification Loss

We are searching for a loss that have the following characteristics:

1. **Probabilistic behaviour**: Should work with probabilities, not raw numbers.
2. **Sensitivity**: It should heavily penalize incorrect confident predictions (e.g., predicting 0.9 when the true label is 0).
3. **Differentiable**: Must be smooth and differentiable to enable optimization.

Any ideas?

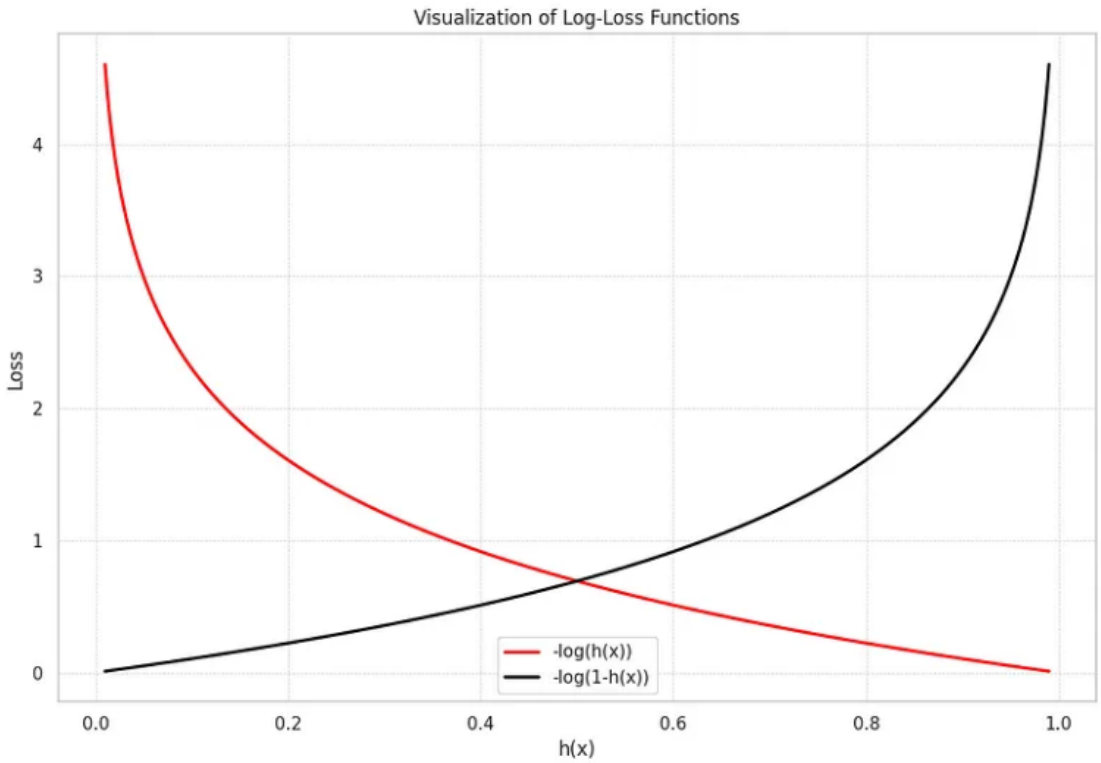
Logarithm is all you need :)

Binary Cross Entropy (LogLoss)

- For one sample, this can be represe

If $y = 1$ and predicti
be close to **zero**. (Grea

If $y = 1$ and predicti
be close to **infinity**. (V



Binary Cross Entropy (LogLoss)

- For one sample, this can be repres

$$loss(y, p) = \begin{cases} - \end{cases}$$

- Let's rewrite it in one line:

$$loss(y, p) = -(y * \log$$

Binary Cross Entropy (LogLoss)

- For one sample, this can be repres

$$loss(y, p) = \begin{cases} \\ - \end{cases}$$

- Let's write it in one line:

$$loss(y, p) = -(y * \log$$

How to find optimal Parameters?

$$J(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\sigma(\mathbf{w}^T\mathbf{x}_i)) + (1-y_i)\log(1-\sigma(\mathbf{w}^T\mathbf{x}_i))$$

Just like before, simply take

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$$

However, this does not have a nice closed solution Just like the MSE case

How to find optimal Parameters?

$$J(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\sigma(\mathbf{w}^T\mathbf{x}_i)) + (1-y_i)\log(1-\sigma(\mathbf{w}^T\mathbf{x}_i))$$

Just like before, simply take

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$$

However, this does not have a nice closed solution Ju

Instead, use gradient descent (optimizer)!

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta\nabla_{\mathbf{w}} J(\mathbf{w}^k)$$

Learning

Logistic Regression

$$J(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\sigma(\mathbf{w}^T\mathbf{x}_i)) + (1-y_i)\log(1-\sigma(\mathbf{w}^T\mathbf{x}_i))$$

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta\nabla_{\mathbf{w}} J(\mathbf{w}^k)$$

Learning rate

- We have a linear model for prediction
- For classification, we want to output a probability
- We map the prediction to probabilities with a sigmoid function
- We have a loss function (BCE) to compare models

Logistic Regression

Linear Regression	Logistic Regression
For Regression	For Classification
We predict the target value for any input value	We predict the probability that the input value belongs to the specific target
Target: Real Values	Target: Discrete values
Graph: Straight Line	Graph: S-curve

Classification Metrics

- **Accuracy:**

  Accuracy measures the proportion of correctly classified samples out of the total number of samples.

$$\text{Accuracy} = \frac{\text{N- of correct samples}}{\text{Overall Samples}}$$

Question: Why not optimizing for the accuracy directly instead of using LogLoss?

Classification Metrics

- **Accuracy:**

  Accuracy measures the proportion of correctly classified samples out of the total number of samples.

Question: Why not optimizing for the accuracy directly instead of using LogLoss?

→ Non-differentiable

Anything else?

Classification Metrics

But accuracy isn't always enough!

Imagine an imbalanced dataset with 95% negative labels (e.g., "not spam").

A model predicting "negative" all the time will be 95% accurate but useless.

Classification Metrics

But accuracy isn't always enough!

Imagine an imbalanced dataset with 95% negative labels (e.g., "not spam").

A model predicting "negative" all the time will be 95% accurate but useless.

Different goals require different metrics

- Do you care more about **catching all positives** (e.g., cancer detection)?
- Or avoiding **false alarms** (e.g., fraud detection)?

Classification Metrics

- **Precision:**

  Among all the positive predictions the model made,
  how many are actually correct?
  important when **minimizing false alarms** is critical,
  like in fraud detection or spam filtering.

$$Tr$$

Classification Metrics

- **Recall:**

  Among all the actual positives, how many did the model correctly identify?

  important when **catching all positives** is vital, such as in cancer detection.

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

Classification Metrics

- **F1 Score:**

  What if both precision and recall are important?

  Take their (harmonic) mean.

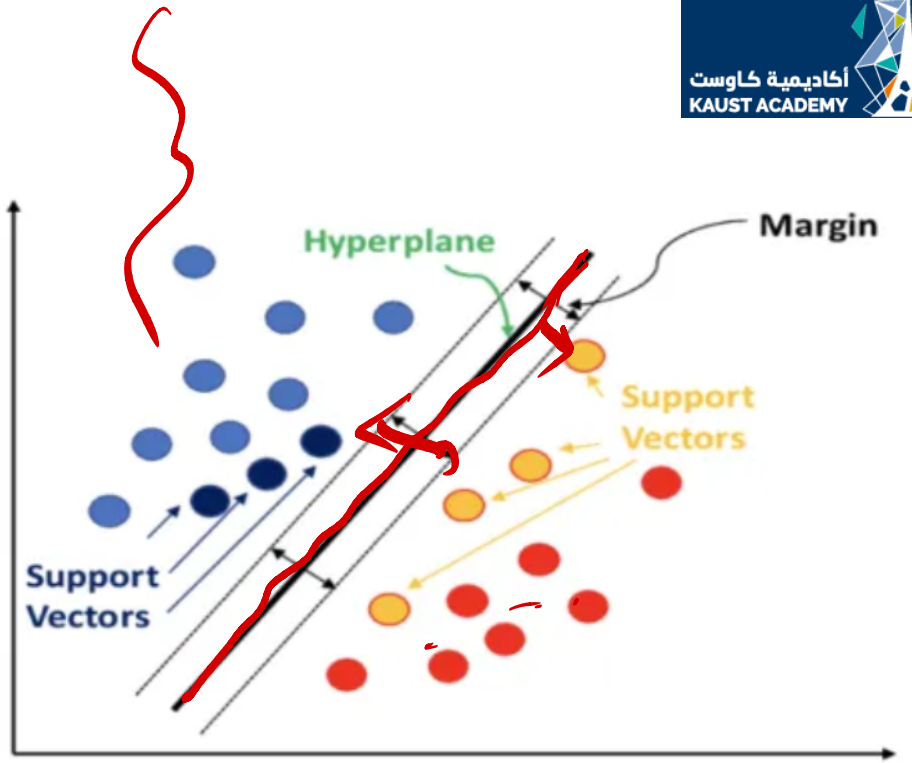
$$F1\ score = \text{—}$$

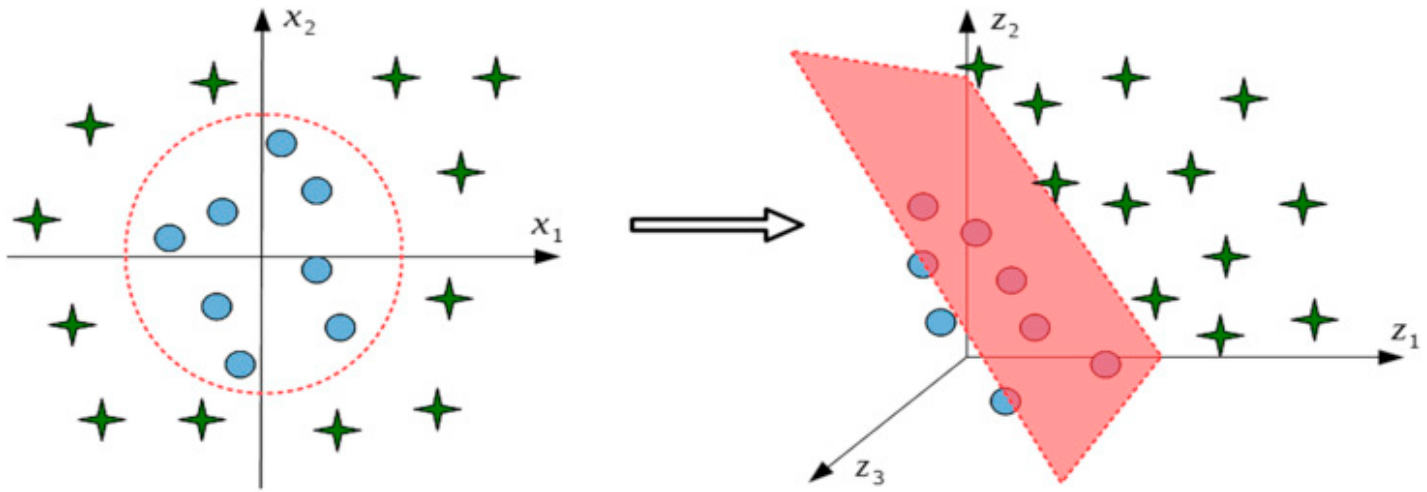
Other ML Models

- **Support Vector Machine (SVM):**

  Key Idea:

- SVM finds the best boundary (hyperplane) that separates classes by maximizing the margin between them.
- Works well with high-dimensional data and small datasets.
- Can be computationally expensive for large datasets.



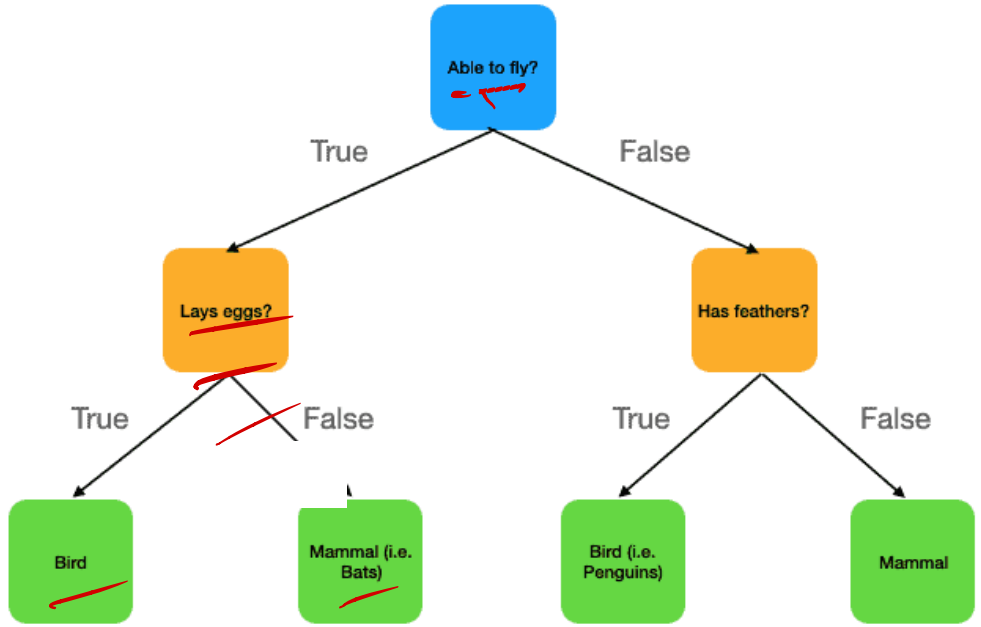


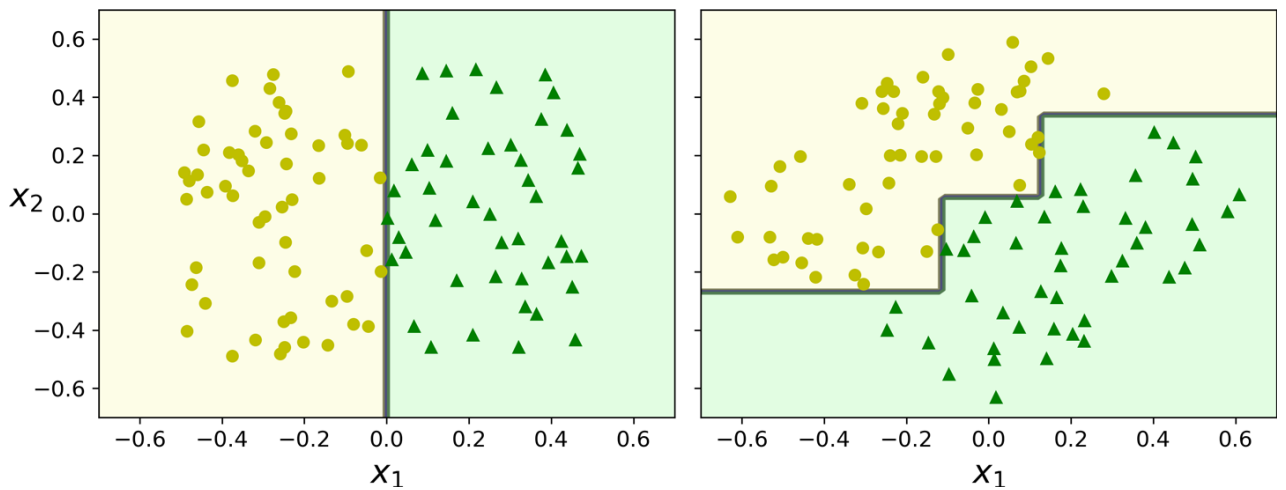
Other ML Models

- **Decision Trees:**

  Key Idea:

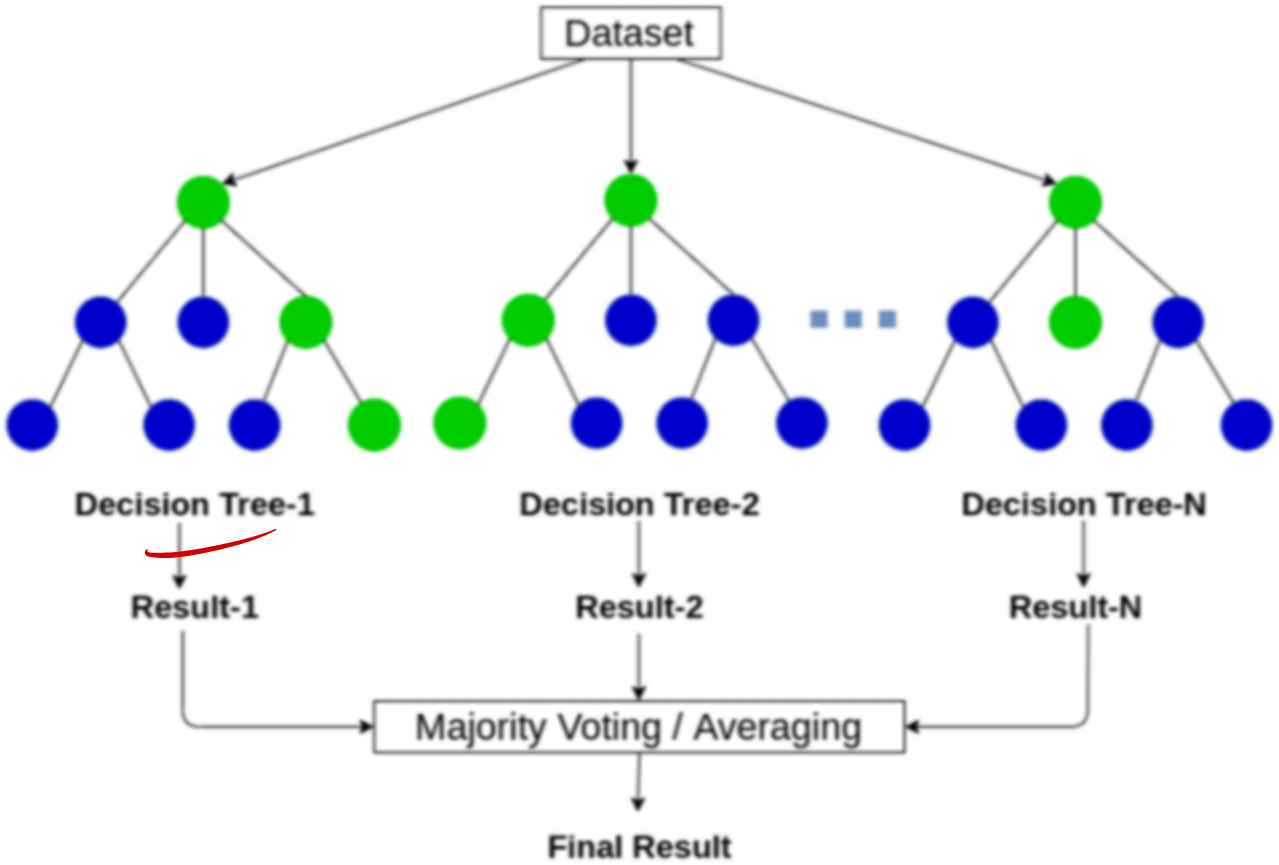
- A tree-based model splits data into subsets based on feature thresholds, creating a flowchart-like structure for decision-making.
- Overfits very easily.
- could work bad if relation between features and the target is linear. (Check the right figure)





Other ML Models

- **Random Forest:**
  Instead of training one tree, let's train a forest :)
  Key Idea:
- Training K decision trees independently, where each one is fed with a different subset of samples and features.
- Then average the results (Regression) or Take the mode (Classification).
- Powerful model.

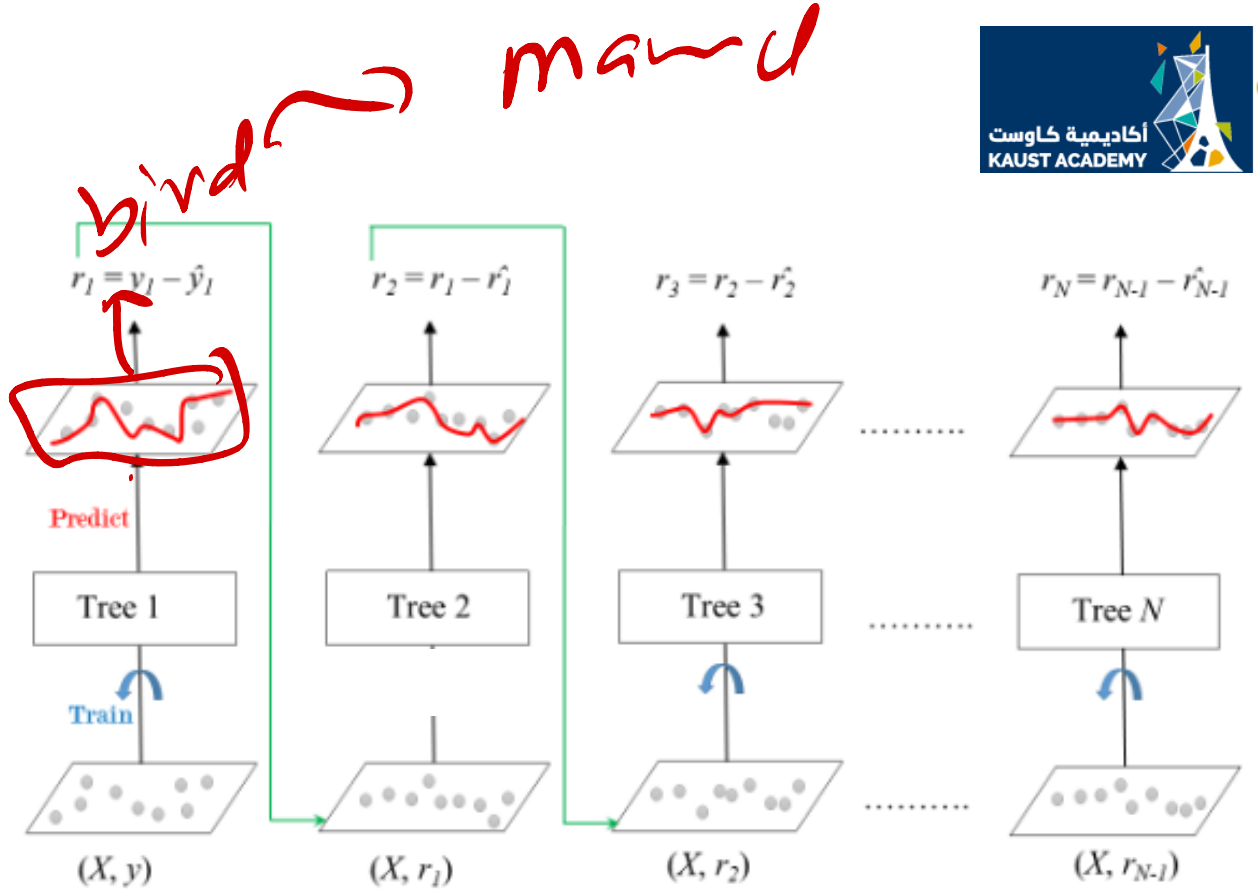


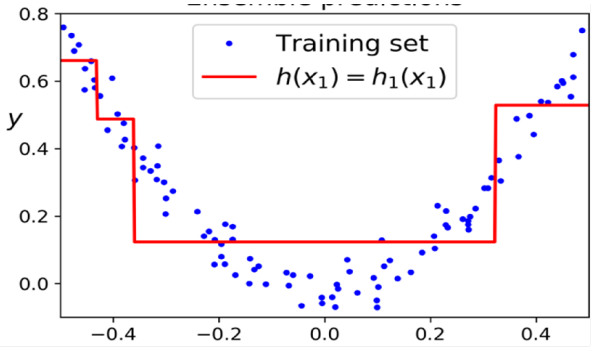
Other ML Models

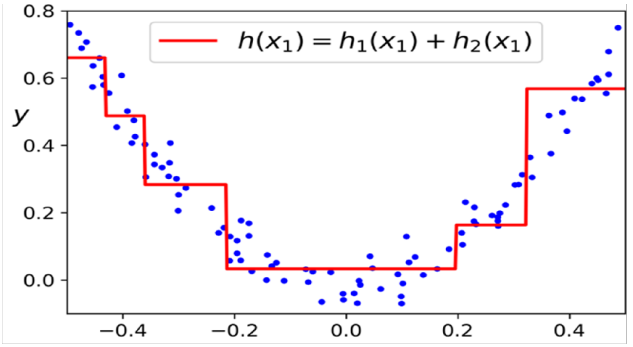
- **Gradient Boosting:**

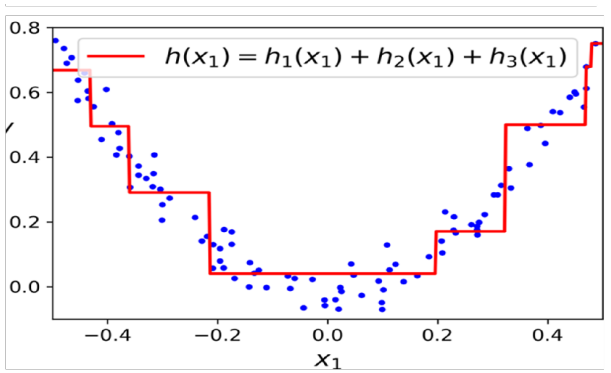
  Key Idea:

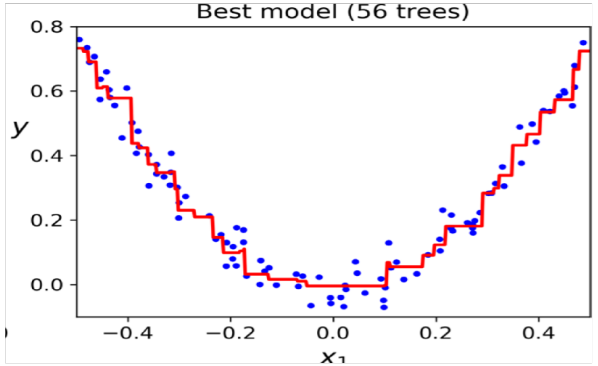
- Training K decision trees dependently, where each tree corrects the errors of the previous one.
- Many versions: XGBoost, LightGBM, CatBoost,…etc
- Top ML models.











Questions?