→ Classification Evaluation

		Predicted	
		Yes	No
Actual	Yes	TP	FN
	No	FP	TN

- Accuracy = TP + TN / Total
- Precision = TP / TP + FP
- Recall = TP / TP + FN
- F1_score = 2P.R / P + R
- TPR = TP / TP + FN . FPR = FP / FP + TN
- Roc curve : x_axis : FPR , y_axis : TPR
  ↳ each point on the curve correspond to a specific threshold

→ Clustering

- Euclidean Distance : $d\|x,y\| = \sqrt{\sum (x_i - y_i)^2}$
- cosine Similarity : $Sim(A,B) = \cos(\theta) = \frac{A.B}{\|A\| \|B\|} = \frac{\sum A_i B_i}{\sqrt{\sum A_i^2} \sqrt{\sum B_i^2}}$

A) K_means:
1) Initialize centers , 2) Compute distance , 3) Assign clusters
4) Update centers (until no more change)

B) Nearest Neighbor Clustering
1) Compute distance 2) Compare to threshold (t) ⟷ ≤t merge / >t don't merge

→ NBC

- $P(C/X_i) = P(X_1/C) \times P(X_2/C) \times \dots \times P(C)$
- Continuous Features
  1) Calculate M_feature_yes and σ_feature_yes (for each numerical feature) M_feature_no and σ_feature_no

$$F(X) = \frac{1}{\sigma\sqrt{2\pi}} \times e^{\frac{-(x-M)^2}{2\sigma^2}} \qquad M = \sum X_i / n \;,\; \sigma^2 = \frac{\sum (X_i - M)^2}{n-1}$$

- if $P(X/C) = 0 \rightarrow P(X=a/C) = \frac{nc + m \cdot p}{n+m}$

nc : nb of example for which $X = a$ and $C = c_i$
n : nb of example for which $C = c_i$

midpoint $\frac{A+B}{2}$

→ Logistic Regression

- $p = P(yes) / total P$
- $odds = p/(1-p)$
- $logit(p) = \ln(odds)$

1) | X | odds | logit(p) = y | $(x_i - \bar{x})^2$ |

2) $y = logit(p) = \hat{\beta}_0 + \hat{\beta}_1 X$

$$\hat{\beta}_1 = \frac{n\sum x_i y_i - \sum x_i \cdot \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \quad , \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

3) $p = 1 / 1 + e^{-(\hat{\beta}_0 + \hat{\beta}_1 X)}$ → > 0.05 / < 0.05

4) $SSE = \sum (y_i - \hat{y}_i)^2$, $SSR = \sum (\hat{y}_i - \bar{y})^2$, $SST = \sum (y_i - \bar{y})^2$

$= SSR + SSE$

$s^2 = SSE/n-2 \Rightarrow s = \sqrt{SSE/n-2}$

5) $S\hat{\beta}_1 = \frac{s}{\sqrt{\sum (x_i - \bar{x})^2}}$ $\quad S\hat{\beta}_0 = s \times \sqrt{\frac{\sum x_i^2}{n\sum (x_i - \bar{x})^2}}$

6) $CI(\hat{\beta}_1) : \hat{\beta}_1 \pm t \times S\hat{\beta}_1$ $\quad CI(\hat{\beta}_0) : \hat{\beta}_0 \pm t \times S\hat{\beta}_0$

7) $H_0 : \beta_1 = 0$ $\quad H_1 : \beta_1 \neq 0$

$$F = \frac{MSR}{MSE} = \frac{SSR}{SSE/n-2}$$

→ $> F_{critical}$ → reject H_0

→ $< F_{critical}$ → don't reject H_0

8) t_test:

$$t = \frac{\hat{\beta}_1}{S\hat{\beta}_1}$$

→ $|t| > t\text{-critical}$ → reject H_0

→ $|t| \leq t\text{-critical}$ → fail to reject H_0

0,0 → 0
0,1 → 1

→ Feed Forward

* First Layer (Inputs = original data X)

1) Weighted Sum (Z^1): $Z^1 = W^1 \times X + B^1$ (weights, Inputs, biases)

2) Apply Activation Function: $A^1 = G(Z^1)$

* Subsequent Layers (Inputs = activation functions from previous layer)

1) Weighted Sum (Z^L): $Z^L = W^L \times A^{L-1} + B^L$

2) Activation Function: $A^L = G(Z^L)$

→ Cost Functions

1) MSE: (predicted value (after applying activation function($\hat{y}$)))

$$J(A_i) = \frac{1}{m} \sum_{i=1}^{m} (\hat{A}_i - Y_i)^2, \quad \frac{dJ}{dA_i} = \frac{2}{m}(A_i - Y_i)$$

m: nb. of samples; Y_i: actual value

2) Cross-Entropy:

$$J(A_i) = \frac{1}{m} \sum_{i=1}^{m} -Y_i \log(A_i) - (1-Y_i)\log(1-A_i)$$

$$dJ/dA_i = \frac{1}{m}\left(-Y_i/A_i + \frac{1-Y_i}{1-A_i}\right)$$

→ Back propagation

1) Output Layer

L: nb. of layer

* $$\frac{dJ}{dw_{ij}^L} = \underbrace{\frac{dJ}{dA_i^L}}_{\text{base on the cost func.}} \cdot \underbrace{\frac{dA_i^L}{dZ_i^L}}_{\text{base on the activation func.}} \cdot \underbrace{\frac{dZ_i^L}{dw_{ij}^L}}_{\rightarrow A_j^{L-1}}$$ (Chain Rule)

i: destination neuron

$$\frac{dJ}{dw_{ij}^L} = J'(A_i^L) \cdot G'(Z_i^L) \cdot A_j^{L-1}$$

j: source neuron

* $$\frac{dJ}{db_i^L} = \frac{dJ}{dA_i^L} \cdot \frac{dA_i^L}{dZ_i^L} \cdot \underbrace{\frac{dZ_i^L}{db_i^L}}_{=1}$$

$$\frac{dJ}{db_i^L} = J'(A_i^L) \cdot G'(Z_i^L)$$

2) Hidden Layer

$$* \frac{\partial J}{\partial w_{ij}^L} = \frac{\partial J}{\partial A_i^L} \cdot \frac{\partial A_i^L}{\partial Z_i^L} \cdot \frac{\partial Z_i^L}{\partial w_{ij}^L} \rightarrow A_j^{L-1}$$

$\hookrightarrow G'(Z_i^L)$ base on the activation

$$\rightarrow \frac{\partial J}{\partial A_i^L} = \sum_K \left[\frac{\partial J}{\partial Z_K^{L+1}} \cdot w_{iK}^{L+1} \right]$$, K: index of neurons in layer "L+1"

$$\cdot \frac{\partial J}{\partial Z_K^{L+1}} = \frac{\partial J}{\partial A_K^{L+1}} \cdot \frac{\partial A_K^{L+1}}{\partial Z_K^{L+1}}$$

$$* \frac{\partial J}{\partial b_i^L} = \left(\sum \frac{\partial J}{\partial Z_K^{L+1}} \cdot w_{iK}^{L+1} \right) \cdot g'(Z_i^L)$$

3) Update Rule

$$w := w - \alpha \frac{\partial J}{\partial w} \quad , \quad b := b - \alpha \cdot \frac{\partial J}{\partial b}$$

(α: learning rate)

→ Activation Functions

1) Sigmoid : $G(x) = 1/1+e^{-x}$, $G'(x) = G(x)(1-G(x))$

2) ReLU : $G(x) = \max(0, x)$: 0 if $x \leq 0$, x if $x > 0$

$G'(x)$: 0 if $x \leq 0$, 1 if $x > 0$

3) tanh: $G(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $G'(x) = 1 - (G(x))^2$

→ Perceptron

1) $y_{pred} = \text{sign}(\sum_i w_i x_i + b)$

2) $y_{actual} \times y_{pred}$
- ≤ 0 misclassified (need update)
- > 0 correct (no update)

3) Update Rules

$w = w + \alpha \cdot y \cdot x$ $\quad\quad$ $b = b + \alpha \cdot y$

→ Gradient Descent

↳ Logistic Regression

1) Linear Combination (Logit)

$z = \sum_{i=1}^{n} w_i \cdot x_i + b$

2) Sigmoid Function

$\hat{y} = h(x) = 1/1 + e^{-z}$ (predicted probability)

3) Cost Function (Binary cross_entropy)

$J(w,b) = -1/m \sum_{i=1}^{m} [y_i \cdot \log(\hat{y}_i) + (1-y_i) \cdot \log(1-\hat{y}_i)]$

4) Gradients (Parial Derivatives)

. Gradient w.r.t. weights (wi)

$\frac{\partial J}{\partial w} = 1/m \sum_{i=1}^{m} (\hat{y}_i - y_i) x_i$

. Gradient w.r.t bias (b)

$\frac{\partial J}{\partial b} = 1/m \sum_{i=1}^{m} (\hat{y}_i - y_i)$

5) Update Rules

$w := w - \alpha \cdot \partial J/\partial w$ $\quad\quad$ $b := b - \alpha \cdot \partial J/\partial b$

α : learning rate

m : nb of examples

Linear Regression

1) Linear Prediction

$$\hat{y}_i = \sum w_i x_i + b$$

2) Cost Function (Mean Squared Error)

$$J(w) = \frac{1}{m} \sum_{i=1}^{m} (y_i - wx_i)^2$$

3) Gradients

- w.r.t weights: $dJ/dw = -\frac{2}{m} \sum_{i=1}^{m} (y_i - wx_i) x_i$
- w.r.t bias: $dJ/db = -\frac{2}{m} \sum_{i=1}^{m} (y_i - wx_i)$

4) Update rules:

$$w := w - \alpha \cdot \frac{dJ}{dw} \qquad b := b - \alpha \cdot \frac{dJ}{db}$$

Regularization

1) Perceptron

learning rate

- L1: $w := w + \alpha (yx - \lambda \text{sign}(w))$ } Sparse weights
  $b := b + \alpha y$
- L2: $w := (1 - \lambda) w + \alpha y x$ } weight decay, prevents large weight updates
  $b := b + \alpha y$

2) Gradient Descent

- L1: $w = w - \alpha \left(dL/dw + \lambda \text{sign}(w) \right)$
- L2: $w = w - \alpha \left(dL/dw + 2\lambda w \right)$

Multi Class classification using Softmax

1) $z = W^T x$

2) $P(y=i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$

↳ e^z for each z in numerator
↳ their sum in denominator

3) $J(w) = -\sum_i y_i \log P(y_i)$

4) Gradients

$dJ/dw_{ij} = (P(y=j) - y_j) x_i$

4) Update : $w = w - \alpha \cdot dJ/dw_{ij}$