1.

1.1. $$P(X|\lambda) = \frac{\lambda^X}{X!}e^{-\lambda}$$

$$L(\lambda) = \prod_{i=1}^{n} P(X_i|\lambda) = \prod_{i=1}^{n} \frac{\lambda^{X_i}e^{-\lambda}}{X_i!}$$

$$\log L(\lambda) = \log \prod_{i=1}^{n} \frac{\lambda^{X_i}e^{-\lambda}}{X_i!}$$

$$\log L(\lambda) = \sum_{i=1}^{n} \log P(X_i|\lambda) = \sum_{i=1}^{n}(X_i \log \lambda - \lambda - \log X_i!)$$

1.2. $$\frac{d}{d\lambda}\log L(\lambda) = \sum_{i=1}^{n}(\frac{X_i}{\lambda} - 1) = 0$$

$$\hat{\lambda} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

1.3. $$\lambda = (\sum_{i=1}^{n} X_i + \alpha - 1)/(n + \beta)$$

2.

2.1. $$\widehat{Y}_i = \sum_{j=1}^{n} H_{ij}Y_j$$

2.2. $$\widehat{Y}^{(-i)} = argmin \sum_{j \neq i} (Y_j - \widehat{Y}_j^{(-i)})^2$$

$$= argmin \sum_{j} (Z_j - \widehat{Y}_j^{(-i)})^2$$

2.3. $$\widehat{Y}_i^{(-i)} = \sum_{j=1}^{n} H_{ij}Z_j$$

2.4. $$\widehat{Y}_i - \widehat{Y}_i^{(-i)} = \sum_{j=1}^{n} H_{ij}Y_j - \sum_{j=1}^{n} H_{ij}Z_j$$

$$= \sum_{j \neq i}^{n} H_{ij}Y_j + H_{ii}Y_i - \sum_{j \neq i}^{n} H_{ij}Y_j - H_{ii}\widehat{Y}_i^{(-i)}$$

$$= H_{ii}Y_i + H_{ii}\widehat{Y}_i^{(-i)}$$

2.5.

3.

3.1. $P(Y = +) = 4/7$ and $P(Y = -) = 3/7$ so $H(Y) = 0.9852$

3.2. $P(X_1 = T) = 8/21$ and $P(X_1 = F) = 13/21$

so

$P(Y = + | X_1 = T) = 7/8$ and $P(Y = + | X_1 = F) = 5/13$

$P(Y = - | X_1 = T) = 1/8$ and $P(Y = - | X_1 = F) = 8/13$

so

$H(Y | X_1) \approx 0.802 \rightarrow IG(X_1) \approx 0.1831$

$P(X_2 = T) = 10/21$ and $P(X_2 = F) = 11/21$

so

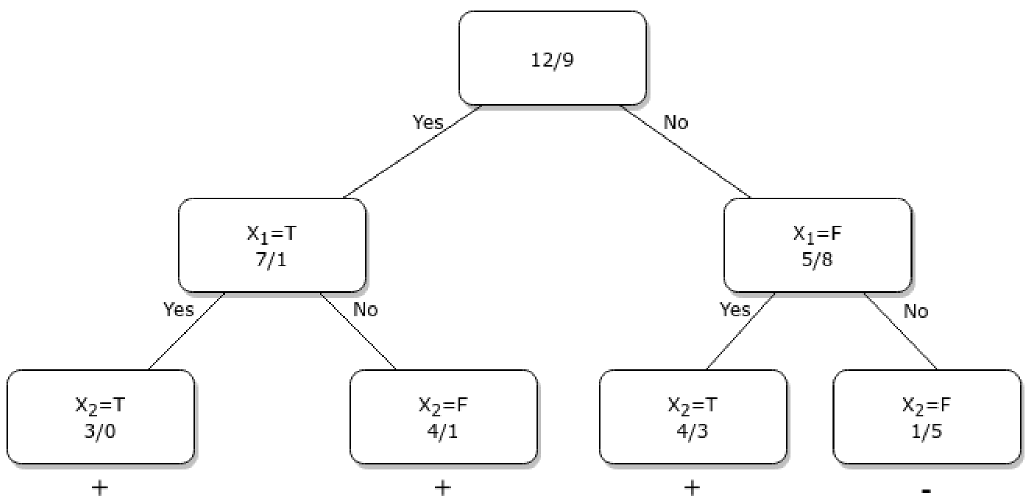
$P(Y = + | X_2 = T) = 7/10$ and $P(Y = + | X_2 = F) = 5/11$

$P(Y = - | X_2 = T) = 3/10$ and $P(Y = - | X_2 = F) = 6/11$

So

$H(Y | X_2) \approx 0.9403 \rightarrow IG(X_2) \approx 0.0449$

3.3.



4.

4.1.

```
# Load dataset and split it into training and test sets
data = load("data_breastcancer.mat")
X_train, y_train, X_test, y_test = split(data)

# Define sigmoid function
function sigmoid(x):
    return 1 / (1 + exp(-x))

# Initialize weight vector with random values
weight = random_vector()

# Set learning rate and maximum number of iterations
alpha = 0.01
max_iter = 1000

# Repeat until either convergence or maximum number of iterations is reached
for i in range(max_iter):
    # For each training example
    for x, y in zip(X_train, y_train):
    # Compute the predicted probability
    pred = sigmoid(weight * x)
    # Compute the error
    error = y - pred
    # Update the weight vector
    weight = weight + alpha * error * x

    # Evaluate the accuracy on the test set
    accuracy = evaluate(weight, X_test, y_test)
    print("Iteration", i, "Accuracy", accuracy)

# To classify a new example x'
function classify(weight, x'):
    # Compute the predicted probability
    pred' = sigmoid(weight * x')
    # Assign it to class 1 if pred' >= 0.5 or class 0 otherwise
    if pred' >= 0.5:
    return 1
    else:
    return 0
```

5.

5.1. Iteration 1: feature component(j) = 4, threshold(c) = 21.0, C1 = 1
Iteration 2: feature component(j) = 4, threshold(c) = 9.0, C1 = 1
Iteration 3: feature component(j) = 3, threshold(c) = 47.0, C1 = 1
Iteration 4: feature component(j) = 4, threshold(c) = 8.0, C1 = 1
Iteration 5: feature component(j) = 3, threshold(c) = 47.0, C1 = 1
Iteration 6: feature component(j) = 3, threshold(c) = 20.0, C1 = 1
Iteration 7: feature component(j) = 5, threshold(c) = 4.0, C1 = 1
Iteration 8: feature component(j) = 4, threshold(c) = 8.0, C1 = 1
Iteration 9: feature component(j) = 3, threshold(c) = 47.0, C1 = 1
Iteration 10: feature component(j) = 4, threshold(c) = 8.0, C1 = 1