# Data Science Advanced

## Lesson02–Logistic Regression using Gradient Descent

# Objective

After completing this lesson you will be able to:

## Logistic Regression:

- Describe hypothesis for a logistic regression
- Understand cost function as a measure to derive logistic regression equation.
- Understand gradient descent algorithm and its working to minimize the cost function for a logistic regression.
- Apply regularization to overcome model overfitting problem in logistic regression.

# Regression Vs. Logistic Regression

| Regression models | Logistic regression models |
|---|---|
| Objective is to estimate the expected or mean value given the independent variables. | Objective is to find the probability of an event given the independent variables. |

The name, logistic regression, is derived from logistic function. Logistic regression or logit model is such that:

$$0 \leq f(x) \leq 1$$

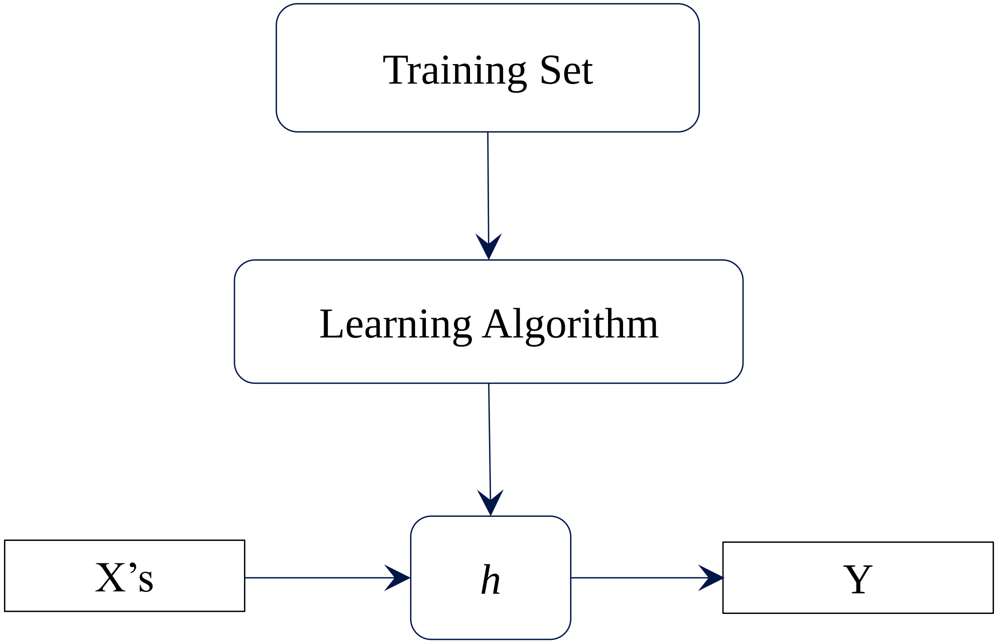
$Y \in \{0,1\}$

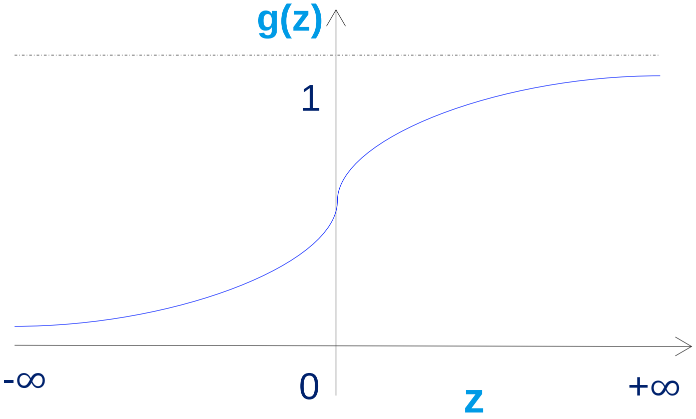
0: “Negative Class”
1: “Positive Class”

# Applications of Logistic Regression

- Logistic regression is one of the most powerful technique to solve classification problem.
  - Email: Spam/Not Spam
  - Online Transaction: Fraudulent/Not Fraudulent (Yes/No)
  - HR Status: Joining/Not Joining
  - Credit Scoring: Defaulter/Non-defaulter

# Logistic Regression–Hypothesis Formulation





In case of Logistic Regression, the hypothesis function is:

$$h_\theta(x) = g(\theta_0 + \theta_1 * x)$$

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

Where z = $\theta_0 + \theta_1 * x$

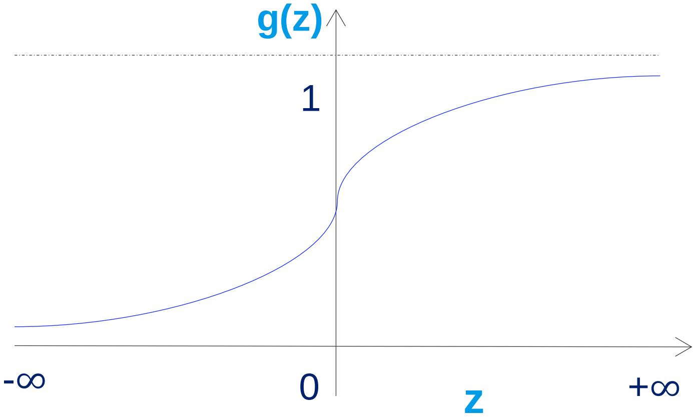
# Logistic Regression–Decision Boundary

$$P(y = 1|x):\ h_\theta(x) = \ g(\theta_0 + \ \theta_1 * x)$$

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

Where z = $\theta_0 + \ \theta_1 * x$

- Predict y =1 if $h_\theta(x) \geq 0.5$
- Predict y =0 if $h_\theta(x) < 0.5$



$$g(z) \geq 0.5\ whenever\ z \geq 0$$

Since

$$h_\theta(x) = g(\theta_0 + \ \theta_1 * x)$$

so

$$h_\theta(x) \geq 0.5\ when(\theta_0 + \theta_1 * x) \geq 0$$

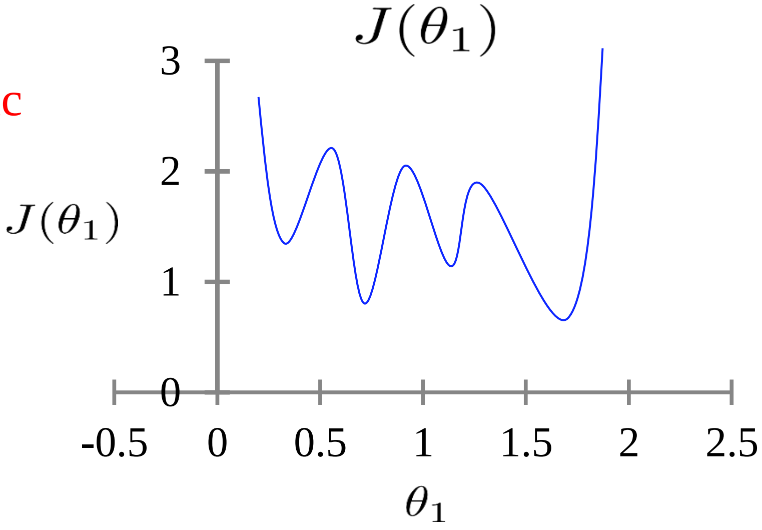
# Logistic Regression – Cost Function

- Choose $\theta_0, \theta_1$ so that $h_\theta(x)$ is close to $\boldsymbol{y}$ for the training examples $(\boldsymbol{x}, \boldsymbol{y})$. Rewriting the linear regression cost function.

$$\boldsymbol{J(\theta)} = 1/m \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^i)) \quad where\ \boldsymbol{Cost(h_\theta(x^{(i)}), y^i)} = 1/2 \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^i)^2$$

- Cost depicts the penalty learning algorithm has to pay when it outputs $h_\theta\left(x^{(i)}\right)$ when actual label is y

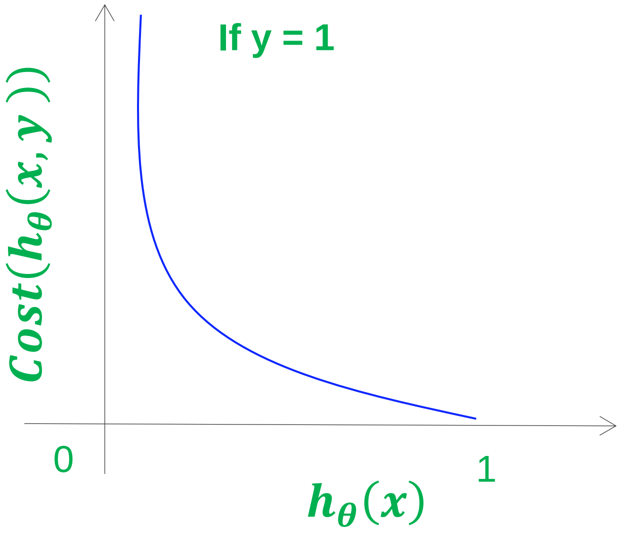
- The above cost function cannot be used for logistic regression as it will be a non-convex function.

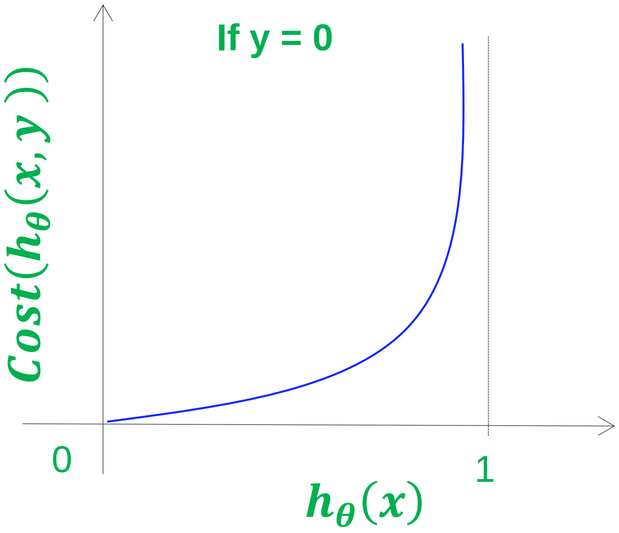


# Logistic Regression – Cost Function

- The penalty, learning algorithm has to pay when it outputs $h_\theta\left(x^{(i)}\right)$ when actual label is y is

$$\boldsymbol{Cost(h_\theta(x), y\,))} = \left\{\begin{matrix} -\log(h_\theta(x) & \boldsymbol{if\ y = 1} \\ -(\log\left(1 - h_\theta(x)\right) & \boldsymbol{if\ y = 0} \end{matrix}\right\}$$





# Logistic Regression – Simplified Cost Function

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

The simplified cost function is

**The cost function:**

$$J(\theta_0, \theta_1) = -\frac{1}{m}\left(\sum_{i=1}^{m} y^i * \log(h_\theta\left(x^i\right) + (1 - y^{(i)}) * \log(1 - h_\theta\left(x^i\right))\right)$$

**Goal:** $\min_{\theta_0,\theta_1}(J(\theta_0, \theta_1))$

# Logistic Regression–Gradient Descent

- The gradient descent algorithm is:

```
$repeat$ until convergence{
    $\theta_j := \theta_j - \alpha * \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \ \ (for\ j = 0\ and\ j = 1)$
}
```

- Simultaneous update of $\theta_0, \theta_1$ is needed:

$$temp0 := \theta_0 - \alpha * \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$temp1 := \theta_1 - \alpha * \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := temp0$$

$$\theta_1 := temp1$$

- α is the learning rate which decides who big or small the steps of descent will be.
- Near to the local minimum, the gradient descent will automatically take smaller steps.
- At the local optima, the $\theta_0, \theta_1$does not change as derivative term will equal to zero.

# Logistic Regression–Gradient Descent

**The gradient descent algorithm is:**

$repeat$ until convergence{

$$\theta_j := \theta_j - \alpha * \frac{\partial}{\partial\theta_j} J(\theta_0, \theta_1)$$

$$(for\ j = 0\ and\ j = 1)$$

}

**The cost function:**

$$J(\theta_0, \theta_1) = -\frac{1}{m}\left(\sum_{i=1}^{m} \begin{matrix} y^i * \log(h_\theta(x^i)) \\ +(1 - y^{(i)}) * \log(1 - h_\theta(x^i)) \end{matrix}\right)$$

**Goal:** $\min_{\theta_0,\theta_1}(J(\theta_0, \theta_1))$

The derivate term for logistic regression will be:

$$\frac{\partial}{\partial\theta_j} J(\theta\ ) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^{(i)}) * x^{(i)}{}_j$$

# Logistic Regression–Gradient Descent

$repeat$ until convergence{

$$\theta_j := \theta_j - \ \alpha * \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{i}) - y^{(i)}) * x^{(i)}{}_j$$

}

# Bias Versus Variance

The problem of high bias comes up when the model is miss-specified (simple hypothesis is selected).

The problem of high variance comes up when the model fails to generalize.

If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta_0, \theta_1) = -\frac{1}{m}(\sum_{i=1}^{m} y^i * \log(h_\theta(x^i)) + (1 - y^{(i)}) * \log(1 - h_\theta(x^i)))$ $\approx \mathbf{0}$ ), but fail to generalize to new examples (predict the cost of treatment for new patients).

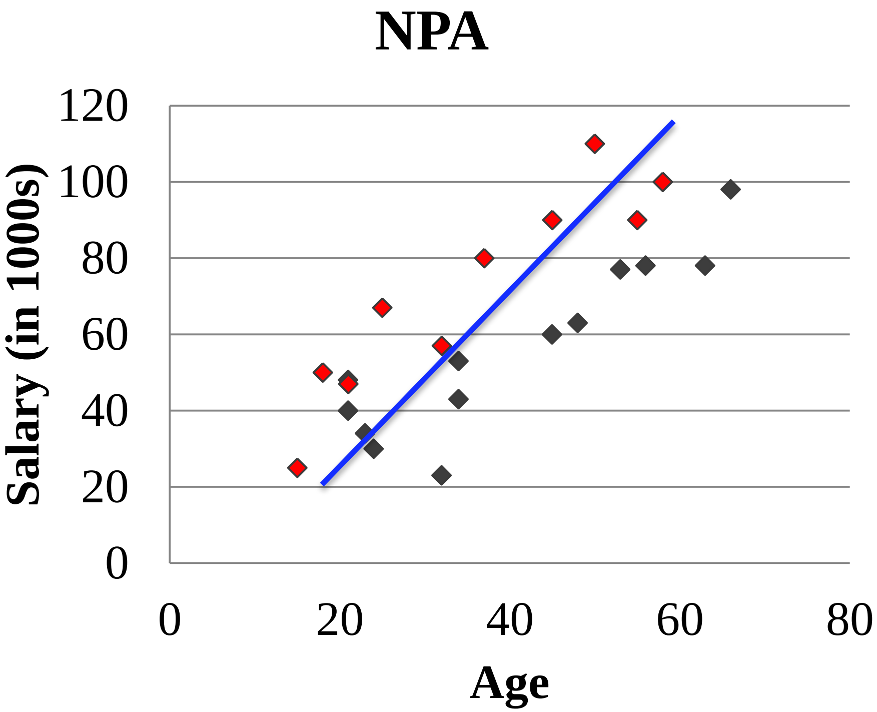
# Reduce Overfitting

1. Reduce number of features.
   - Manually select which features to keep.
   - Model selection algorithm (discussed later).

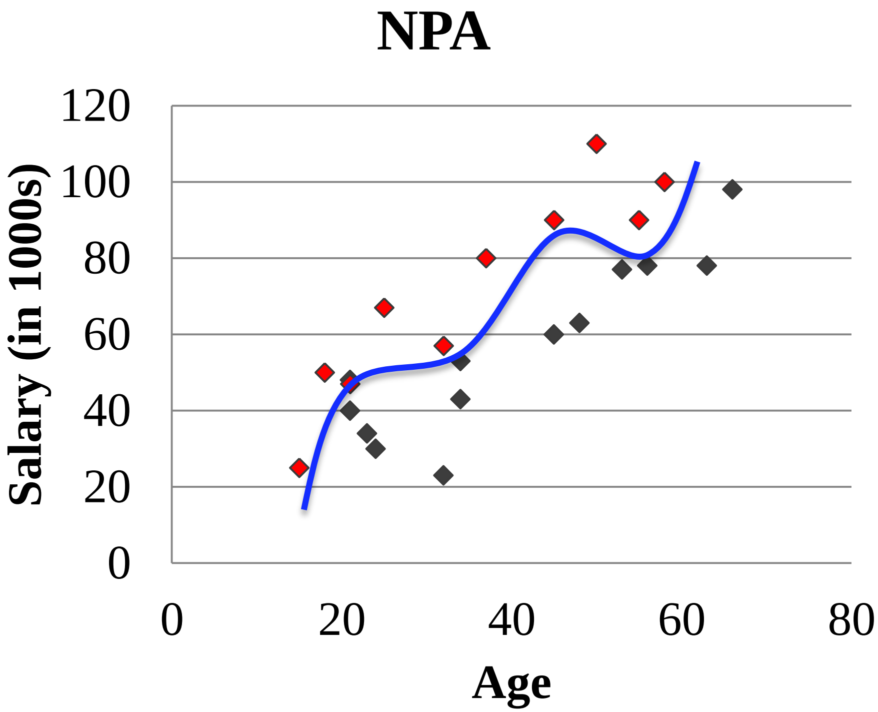
2. Regularization.
   - Keep all the features, but reduce magnitude/values of parameters .
   - Works well when we have a lot of features, each of which contributes a bit to predicting .

# Regularization – Reduce overfitting

*Penalize $\theta_3$ and $\theta_4$ to make them really small.*



$$h_\theta(x) = g(\theta_0 + \theta_1 * Age + \theta_2 * Salary)$$



$$h_\theta(x) = g(\theta_0 + \theta_1 * Age + \theta_2 * Salary$$
$$+\theta_3 * Age^2 + \theta_4 * Age^2 * Salary + \cdots$$

*Modify the cost function to add high weights to $\theta_3$ and $\theta_4$.*
*The cost function can be minimized only if the$\theta_3$ and $\theta_4 \approx 0$.*

# Regularization – Reduce overfitting

$$J(\theta_0,\theta_1) = -\frac{1}{m}\left(\sum_{i=1}^{m} y^i * \log(h_\theta\left(x^i\right) + (1-y^{(i)}) * \log(1-h_\theta\left(x^i\right))\right) + 1000 * {\theta_3}^2 + 1000 * {\theta_4}^2$$

- Since we do not know which feature to penalize. Small values for parameters $\theta_{0,}\theta_{1,}\theta_{2,}\theta_{3,}\ldots$.(Penalize all)

$$J(\theta_0,\theta_1) = -\frac{1}{m}\left(\sum_{i=1}^{m} y^i * \log(h_\theta\left(x^i\right) + (1-y^{(i)}) * \log(1-h_\theta\left(x^i\right))\right) + \frac{\lambda}{2*m}\sum_{j=1}^{n} {\theta_j}^2$$

"Simpler" hypothesis and Less prone to overfitting

The cost function, where the penalty added is sum of $\theta -$ squared is know as **Ridge Regression (L2)**.

In case, the penalty added is absolute value of $\theta$ then it is **Lasso Regression (L1).**

# Regularization – Reduce overfitting

## Ridge versus Lasso Regression

### Ridge

- shrink coefficients towards zero, it can never reduce it to zero.
- all features will be included in the model no matter how small the value of the coefficients.

### Lasso

- able to shrink coefficient to exactly zero.
- reduces the number of features and serve as a feature selection tools at the same time.

Lasso regression useful in cases with high dimension and helps with model interpretability.

# Logistic Regression–Gradient Descent

The gradient descent algorithm is:

```
repeat until convergence{
        $\theta_j := \theta_j - \alpha * \frac{\partial}{\partial\theta_j} J(\theta_0, \theta_1)$
            $(for\ j = 0\ and\ j = 1)$
}
```

The cost function:

$$h_\theta(x) = g(\theta_0 + \theta_1 * x)$$

$$J(\theta_0, \theta_1) = -\frac{1}{m}\left(\sum_{i=1}^{m} \begin{matrix} y^i * \log(h_\theta(x^i) \\ +(1 - y^{(i)}) * \log(1 - h_\theta(x^i)) \end{matrix}\right) + \frac{\lambda}{2 * m}\sum_{j=1}^{n} {\theta_j}^2$$

Goal: $\min_{\theta_0, \theta_1}(\boldsymbol{J(\theta_0, \theta_1)})$

The derivate term for linear regression will be:

$$\frac{\boldsymbol{\partial}}{\boldsymbol{\partial\theta_0}} \boldsymbol{J(\theta_0, \theta_1)} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^{(i)})$$

$$\frac{\boldsymbol{\partial}}{\boldsymbol{\partial\theta_1}} \boldsymbol{J(\theta_0, \theta_1)} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^{(i)}) * {x^{(i)}}_j + \frac{\lambda}{\mathrm{m}} * \sum_{j=1}^{n}\theta_j$$

# Linear Regression–Regularized Gradient Descent

$\mathit{repeat}$ until convergence{

$$\theta_0 := \theta_0 - \ \alpha * \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^i\right) - y^{(i)}\right) * x^{(i)}{}_0$$

$$\theta_j := \theta_j - \alpha * \left[\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^i\right) - y^{(i)}\right) * x^{(i)}{}_j + \frac{\lambda}{\mathrm{m}} * \sum_{j=1}^{n} \theta_j\right]$$

}

The regularization is for all the features excluding $X_0$ and thus gradient descent for $\theta_0$ is separate. Note the summation for regularization starts from index 1.

# Logistic Regression–Model Validity

$$Sensitivity = \left(\frac{TP}{TP + FN}\right) = \frac{4}{7} = 57.1\%$$

$$Specificity = \left(\frac{TN}{TN + FP}\right) = \frac{17}{17} = 100\%$$

| Classification matrix | | |
|---|---|---|
| Observed | Predicted | |
| | Class=1 (Positive) | Class=0 (Negative) |
| Class =1 (Positive) | $f_{11}$= 4 [TP] | $f_{10}$= 3 [FN] |
| Class =0 (Negative) | $f_{01}$= 0 [FP] | $f_{00}$= 17 [TN] |

$$Model\ accuracy = \left(\frac{TP + TN}{TP + TN + FP + FN}\right) = \frac{21}{24} = 87.5\%$$

Sensitivity is the probability that predicted class is 1 when observed class is 1.
Specificity is the probability that the predicted class is 0 when the observed class is 0.

# Logistic Regression–Model Validity

$$Recall = \left(\frac{TP}{TP + FN}\right) = \frac{20}{50} = 40\%$$

$$Precision = \left(\frac{TP}{TP + FP}\right) = \frac{20}{30} = 66\%$$

**Classification matrix**

| | Predicted | |
|---|---|---|
| **Observed** | **Class=1 (Positive)** | **Class=0 (Negative)** |
| Class =1 (Positive) | $f_{11}$= 20 [TP] | $f_{10}$= 3 0[FN] |
| Class =0 (Negative) | $f_{01}$= 10 [FP] | $f_{00}$= 40 [TN] |

Recall is same as sensitivity.

Higher precision means lower recall/sensitivity.

Higher precision means higher specificity.

Precision: Of all the cases where model predicted Y = 1, what fraction actually are positive?
Recall: Of all the cases which are actually Y = 1, what fraction actually are predicted positive?

# Comparing Models–F Score

Measure to compare different models.

$$F\ Score = \frac{(1+\beta^2) * True\ Positives}{(1+\beta^2) * \text{True Positives} + \beta^2 * False\ Negative + False\ Positive}$$

Simplified:

$$F\ Score = (1+\beta^2) * \frac{Precision * Recall}{\beta^2 * \text{Precision} + \text{Recall}}$$

The F-measure was derived so that $F_\beta$"measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision".

# Comparing Models–F 1 Score

Measure to compare different models.

$$F\ 1\ Score = \frac{2 * P * R}{\mathrm{P} + \mathrm{R}}$$

| **Model** | **Precision** | **Recall** | **Average** | **F Score** |
|---|---|---|---|---|
| Logistic | 0.5 | 0.4 | 0.45 | 0.44 |
| Decision Tree | 0.7 | 0.1 | 0.4 | 0.17 |
| Neural Network | .02 | 1.0 | 0.51 | 0.03 |

F1 measure gives a balanced measure to recall and precision (harmonic mean of precision and recall). The threshold/cut-off for F1 roughly corresponds to youden's index. F1 score is also known as Dice similarity coefficient (DSC)

# Comparing Models–F 0.5 Score

Measure to compare different models.

$$F\ Score = 1.25 * \frac{P * R}{0.25 * \mathrm{P} + \mathrm{R}}$$

| **Model** | **Precision** | **Recall** | **Average** | **F0.5 Score** |
|---|---|---|---|---|
| Logistic | 0.5 | 0.4 | 0.45 | 0.47 |
| Decision Tree | 0.7 | 0.1 | 0.4 | 0.32 |
| Neural Network | .02 | 1.0 | 0.51 | 0.02 |

F0.5 measure, which weighs recall/sensitivity lower than precision (by reducing the influence of false negatives)

# Comparing Models–F 2 Score

Measure to compare different models.

$$F\ Score = 5 * \frac{P * R}{4 * \mathrm{P} + \mathrm{R}}$$

| Model | Precision | Recall | Average | F Score |
|---|---|---|---|---|
| Logistic | 0.5 | 0.4 | 0.45 | 0.42 |
| Decision Tree | 0.7 | 0.1 | 0.4 | 0.12 |
| Neural Network | .02 | 1.0 | 0.51 | 0.09 |

F2 measure, which weighs recall/sensitivity higher than precision (by placing more emphasis on false negatives)

# End of Lesson03–Logistic Regression using Gradient Descent