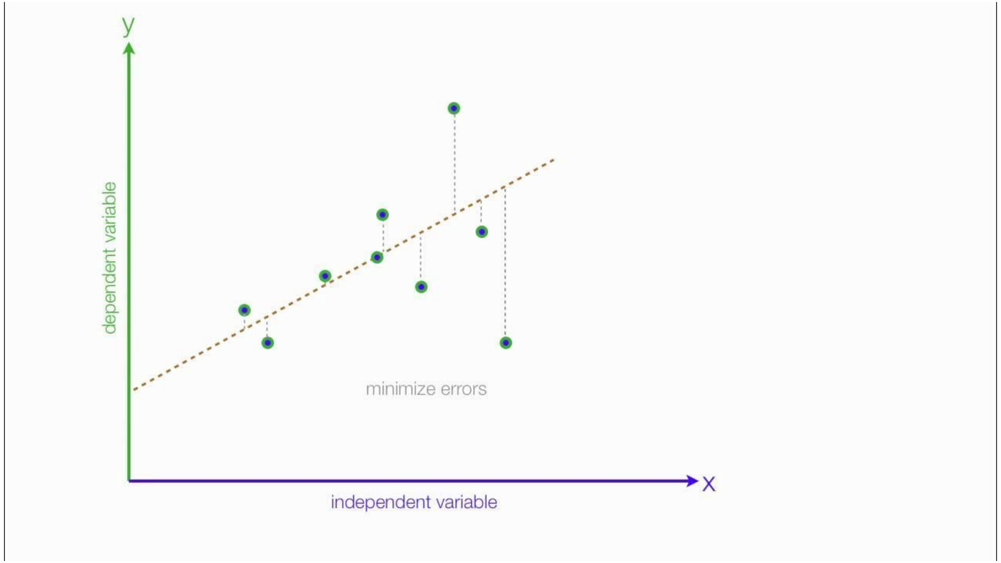
Logistic Regression

Presented by:
Rajendra Baskota

Linear Regression (Recap)

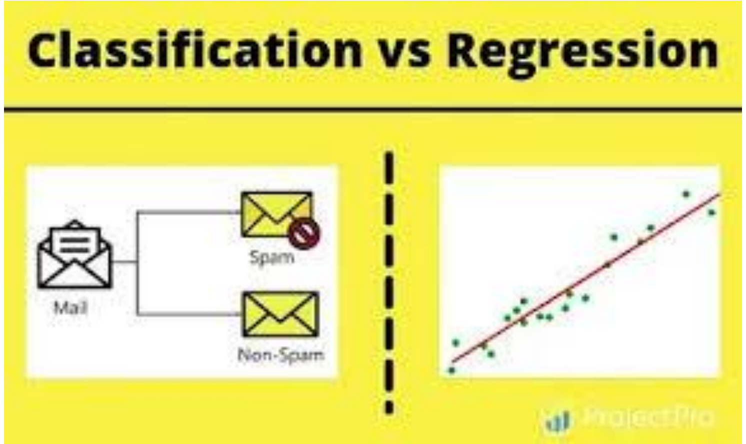
- Regression predicts continuous data.

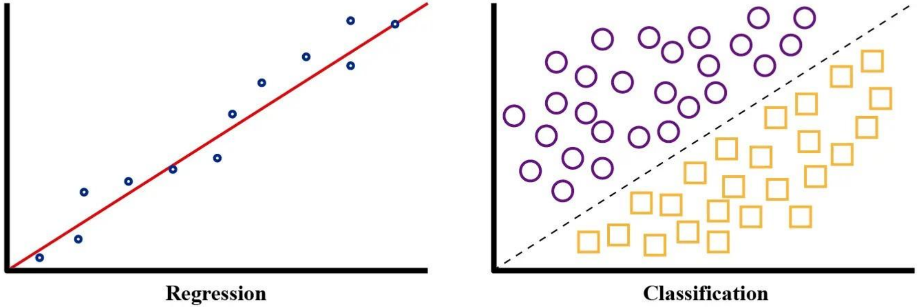
$$\hat{y} = \beta_0 + \beta_1 x$$



Classification

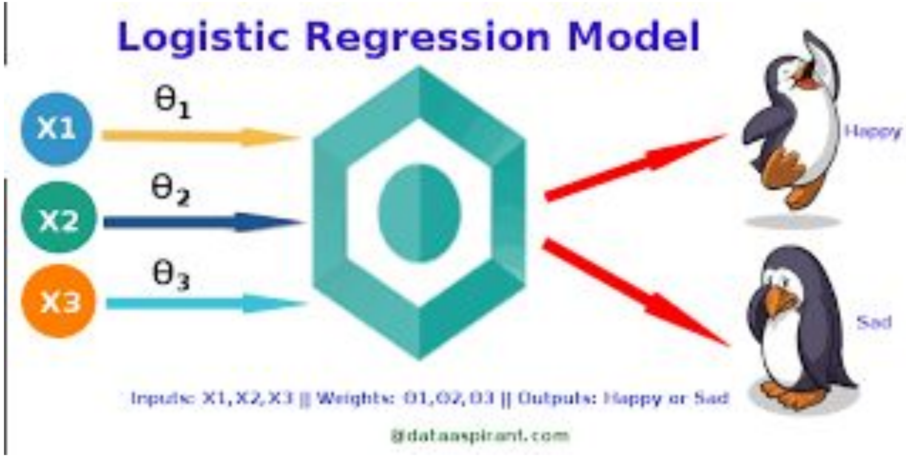
- Classification predicts discrete data.





Logistic Regression: Introduction

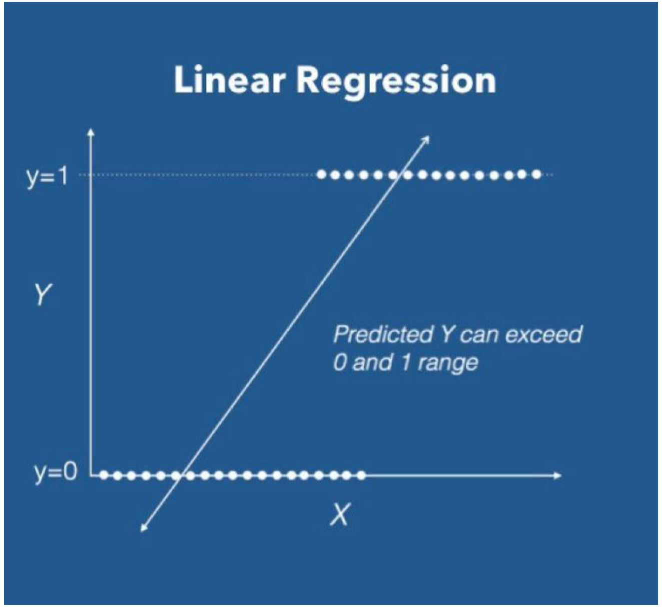
- Logistic regression is a supervised machine learning algorithm used for classification tasks.
- The goal is to predict the **probability** that an instance belongs to a given class or not.


Logistic Regression Model
X1
X2
X3
θ_1
θ_2
θ_3
Happy
Sad
Inputs: X1,X2,X3 || Weights: θ1,θ2,θ3 || Outputs: Happy or Sad
@dataaspirant.com

Using Linear Regression on Classification Task

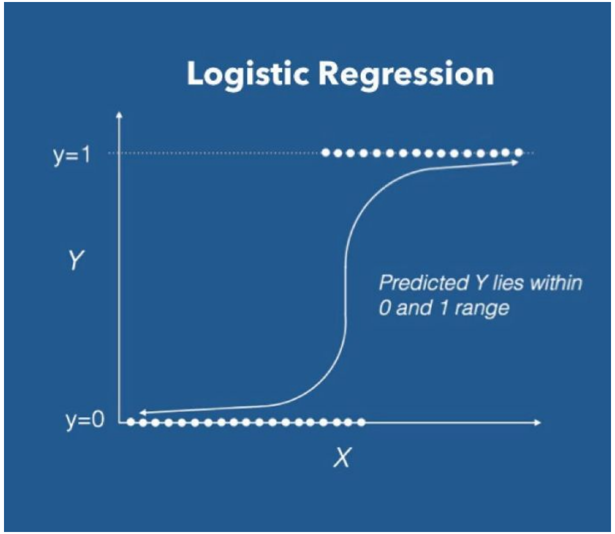
Clearly, this doesn't represent the data points well.

$$\hat{y} = \beta_0 + \beta_1 x$$

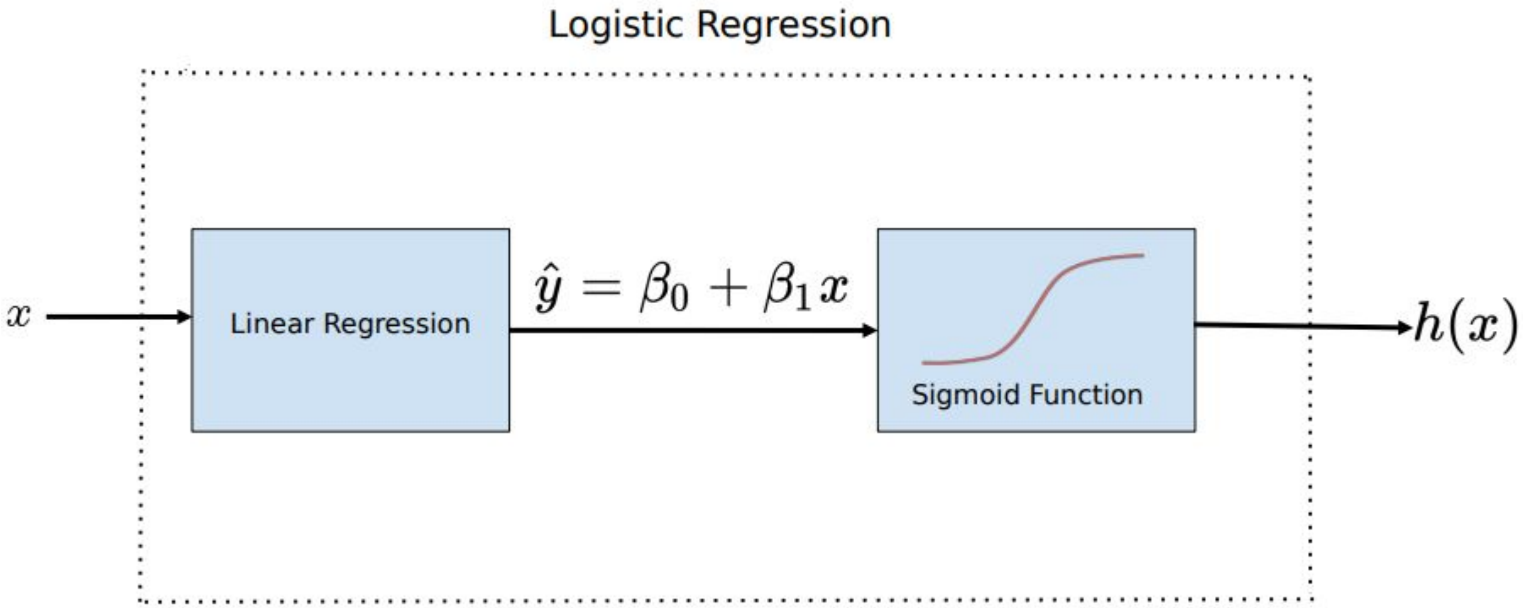


Using Logistic Regression

- This does represent the data points well.
- The predicted value seems to be squeezed in between 0 and 1.

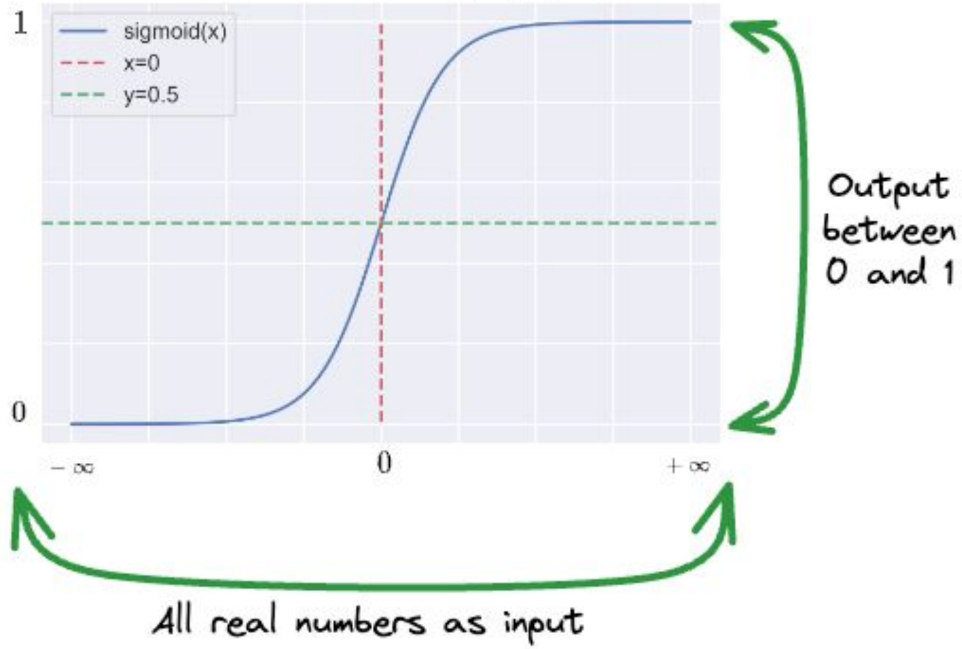


Logistic Regression

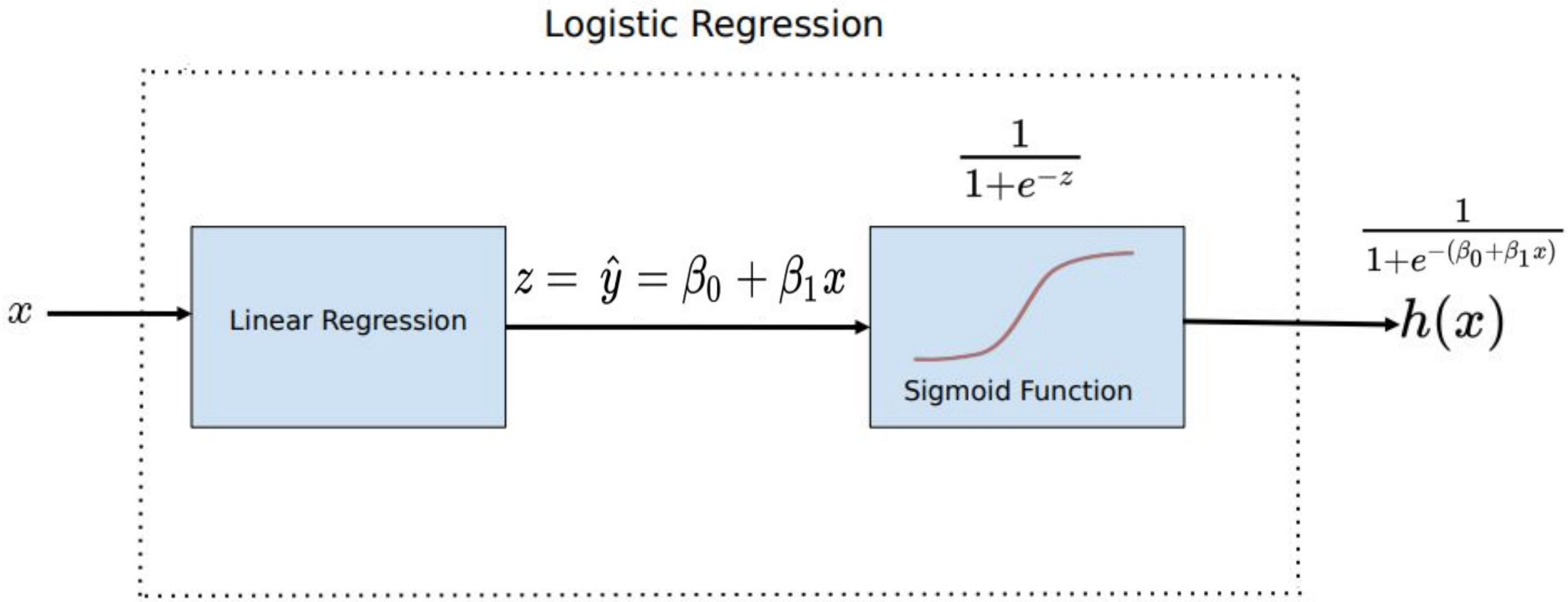


Sigmoid Function

$$\mathrm{sigmoid(x)} = \frac{1}{1+e^{-x}}$$



Logistic Regression Equation



Logistic Regression Output

$$h(x) = P(y = 1 \mid x)$$

$$1 - h(x) = P(y = 0 \mid x)$$

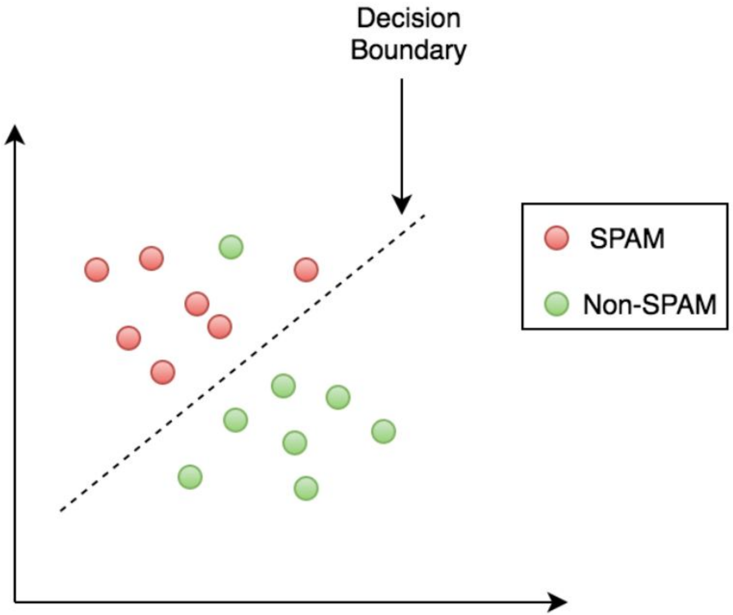
But still h(x) ranges from [0, 1]. What am I supposed to do with that number?

Threshold

```
threshold = 0.5

if h(x) >= threshold,
    predict y = 1
else
    predict y = 0
```

Decision Boundary



Cost Function (Binary Cross-Entropy Loss)

$$Cost(h(x), y) = \begin{cases} -\log(h(x)) & if \quad y = 1 \\ -\log(1 - h(x)) & if \quad y = 0 \end{cases}$$

This can be simplified as:

$$Cost(h(x), y) = -y\log(h(x)) - (1 - y)\log(1 - h(x))$$

For m examples:

$$\mathcal{J}(w) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_w(x^{(i)}), y^{(i)})$$

Significance of Logarithm in the Cost Function

- Logarithm of probabilities penalizes incorrect predictions more severely when the predicted probability is far from the true label.

Loss = -log(P(y | x))

- If P(y | x) is close to 1, the loss is small.
- If P(y | x) is close to 0, the loss becomes large, encouraging the model to improve.

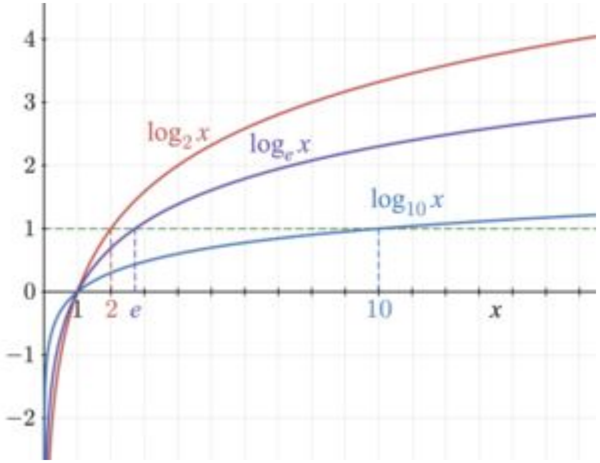


Figure: Log Curve

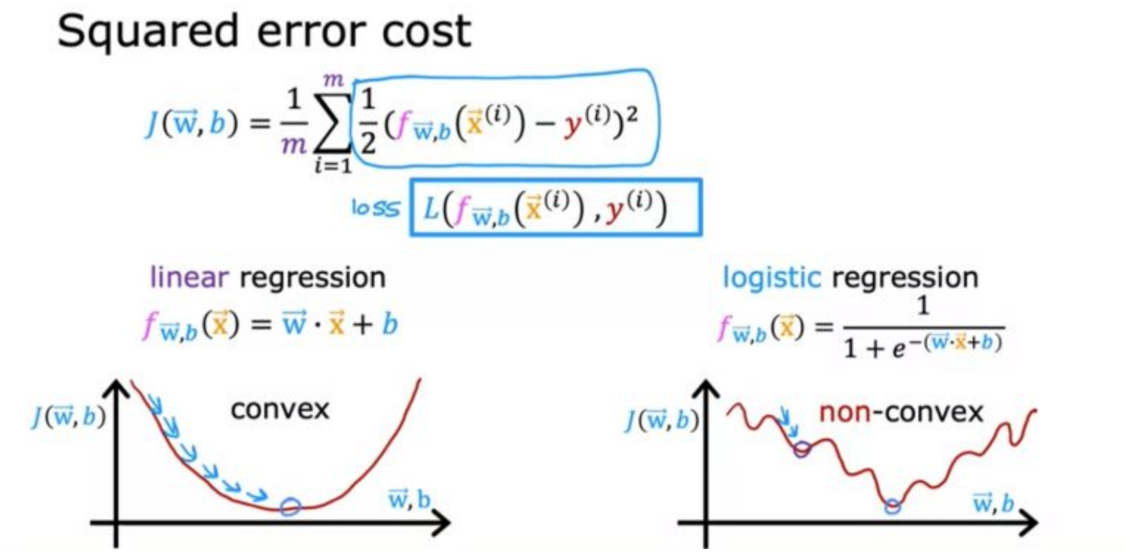
Why can't we use Linear Regression cost function for Logistic Regression?

- Squared Error Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$$

Why can't we use Linear Regression cost function for Logistic Regression?

- The squared cost function for logistic regression is **not convex** and thus shows many local minima.



Squared error cost

$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \boxed{\frac{1}{2}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})^2}$$

loss $\boxed{L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)})}$

linear regression

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

logistic regression

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w}\cdot\vec{x}+b)}}$$

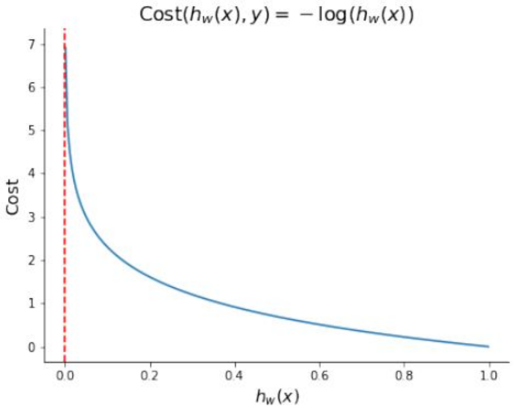
Convexity of Cross-Entropy Loss

- The Cross-entropy is convex.

$$Cost(h(x), y) = \begin{cases} -\log(h(x)) & if \quad y = 1 \\ -\log(1 - h(x)) & if \quad y = 0 \end{cases}$$

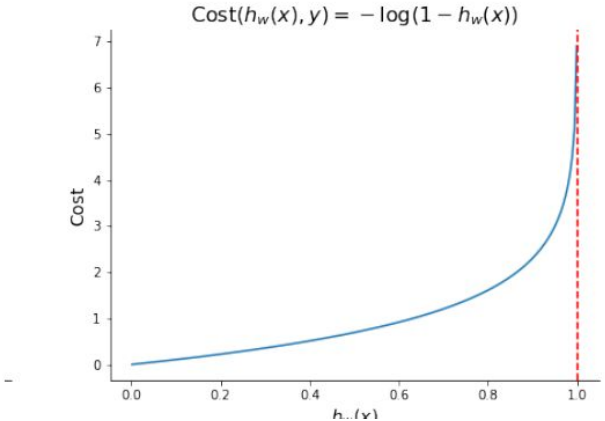
If **y = 1**, the cost of the prediction will be

$$Cost(h(x), y) = -\log(h(x))$$



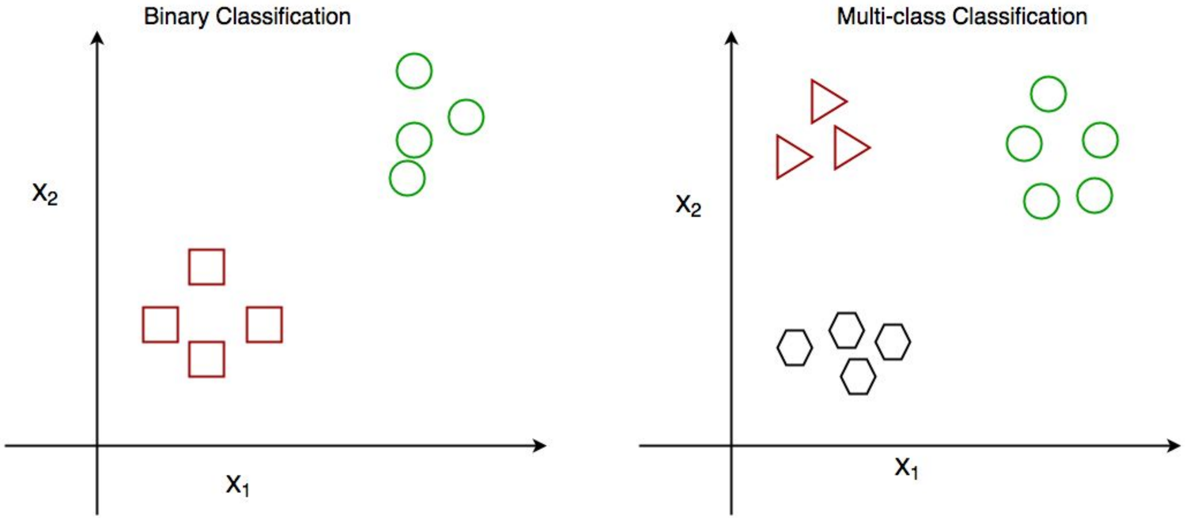
If **y = 0**, the cost of the prediction will be

$$Cost(h(x), y) = -\log(1 - h(x))$$



Multinomial Classification/Multi-class Classification

- Comprising of more than two classes
- Classes should be exclusive



Strategies from Binary Classification

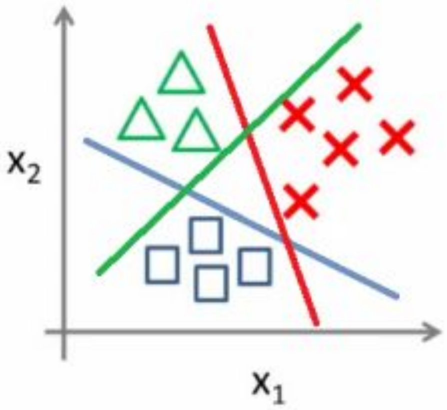
1. One-vs-Rest
2. One-vs-One

One-vs-Rest (OvR)

- Number of classifiers = Number of Classes
- Apply all classifiers
- Predict class with highest confidence score

$$\hat{y} = \begin{cases} 1 & if \quad class \\ 0 & if \quad notclass \end{cases}$$

Multi-class classification:



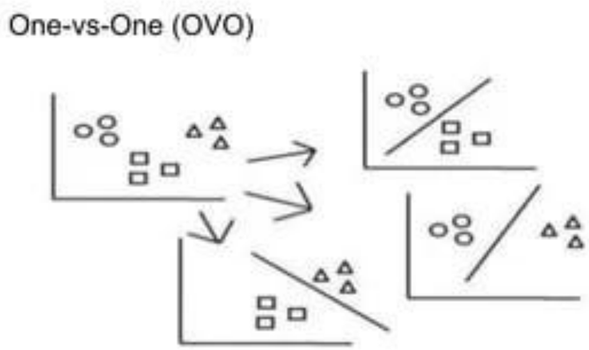
One-vs-One (OvO)

Given N classes,

Total classifiers = **N(N-1)/2**

For classes A, B and C, the three classifiers are:

- Class A vs Class B
- Class A vs Class C
- Class B vs Class C



Multinomial Classification: Cost Function

For m instances, the cross entropy loss is:

$$\mathcal{J}(\boldsymbol{W}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\boldsymbol{y}_k^{(i)}\log\left(\hat{\boldsymbol{y}}_k^{(i)}\right)$$

Where, k=Number of classes

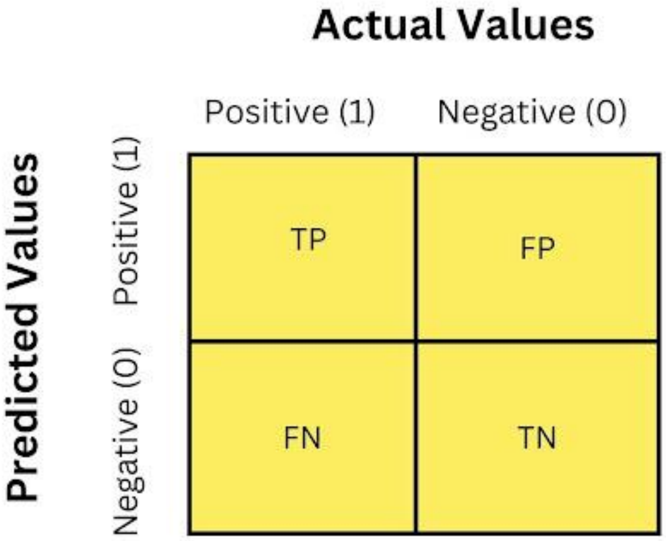
Performance Metrics: Accuracy

- The fraction of predictions that the classifier predicted correctly.

$$\text{Accuracy} = \frac{\text{No. of correct predictions}}{\text{Total no. of predictions}}$$

- Not a viable metric when the dataset is skewed.

Confusion Matrix



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

TN = True Negative

FP = False Positive **Type- I**

FN = False Negative **Type- II**

TP = True Positive

Performance Metrics: Precision

$$\text{Precision} = \frac{\text{True Positives}}{\text{Total no. of predicted positives}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- How often the examples predicted as positive by our classifier are actually positive?
- Used when it's more important to ensure that positive predictions are accurate. Ie, we want to minimize false positives.
- If a model classified a total of 100 samples to be of positive class, and 70 of them actually belonged to the positive class of the dataset (and 30 were negative class samples predicted incorrectly as “positive” by the classifier), then the precision is 70%.

Performance Metrics: Recall (Sensitivity or TPR)

$$\text{Recall} = \frac{\text{True Positives}}{\text{Total no. of actual positives}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

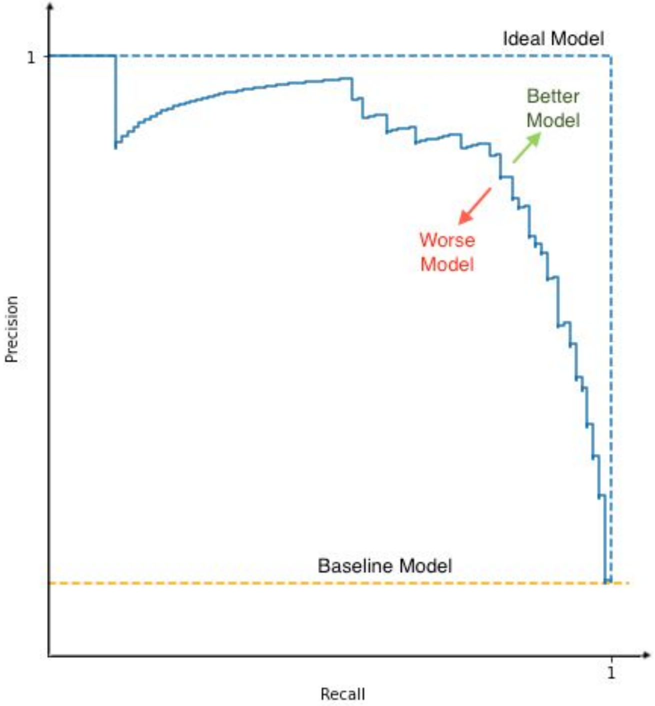
- How often the examples that are actually positive are predicted as positive by our classifier?
- Used when it's crucial to identify all relevant cases, even if it means including some false positives, making it ideal for situations where missing a positive result is more costly than a false positive.
- If the test set of a dataset consists of 100 samples in its positive class, how many of them were identified? If 60 of the positive samples were identified correctly, then the recall is 60%.

When to use precision and when to use recall?

- Use cases of Precision
    - Medical diagnosis (specific treatments of rare conditions whose treatment could cost a lot)
    - Recommendations (eg. YouTube recommendations, FP=videos users dislike but shown on the platform, FN=videos users like but removed from the recommendation)
    - Spam email filtering
    - Legal and compliance system (FP could lead to unwanted legal actions, reputational harm etc)
- Use cases of Recall
    - Medical diagnosis (detection of cancer, COVID-19)
    - Security and threat detection
    - Fraud Detection

But, it all depends on the context and your use cases.

Precision-Recall Trade off

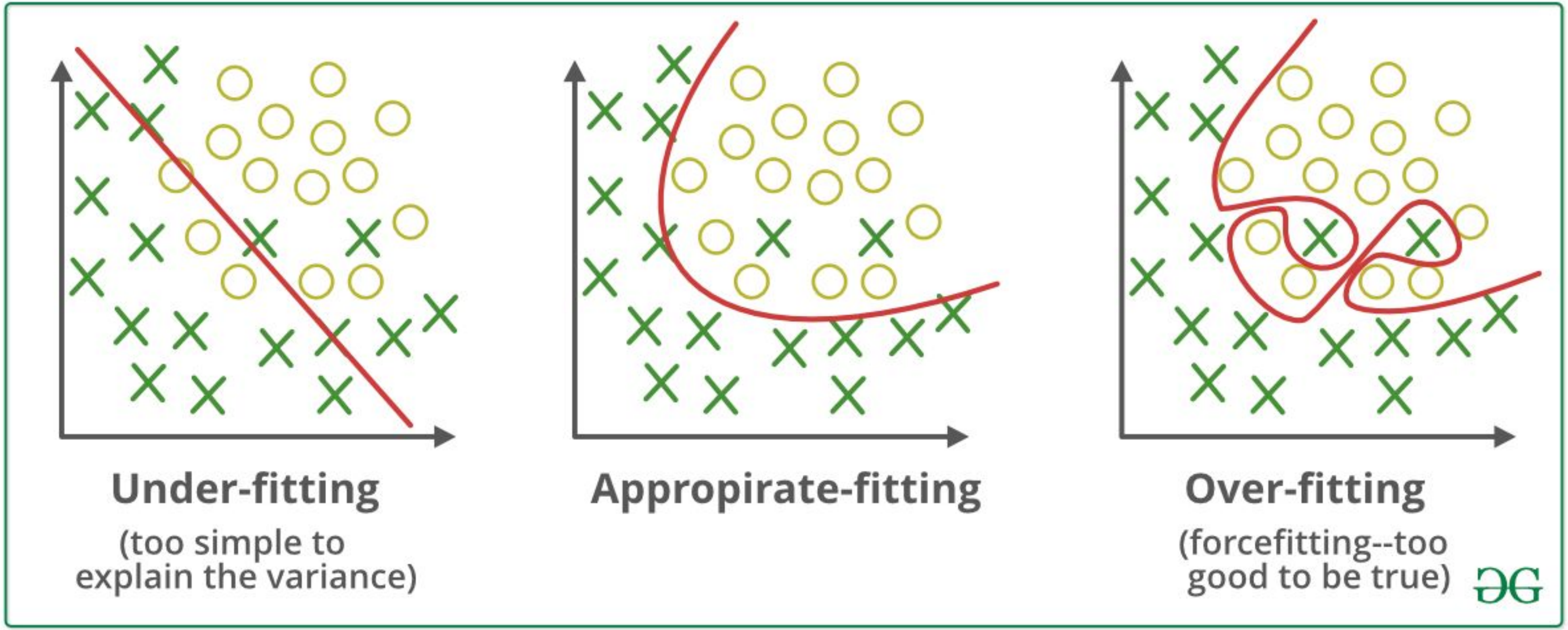


Performance Metrics: F1 Score

- Harmonic mean between precision and recall.
- In general, harmonic mean provides insights into quantities that are inversely proportional.
- Provides a balanced measure between two quantities, precision and recall.
- Ensures that a high value in one metric cannot compensate for a low value in the other, making it a more balanced equation.

$$\text{F1 score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

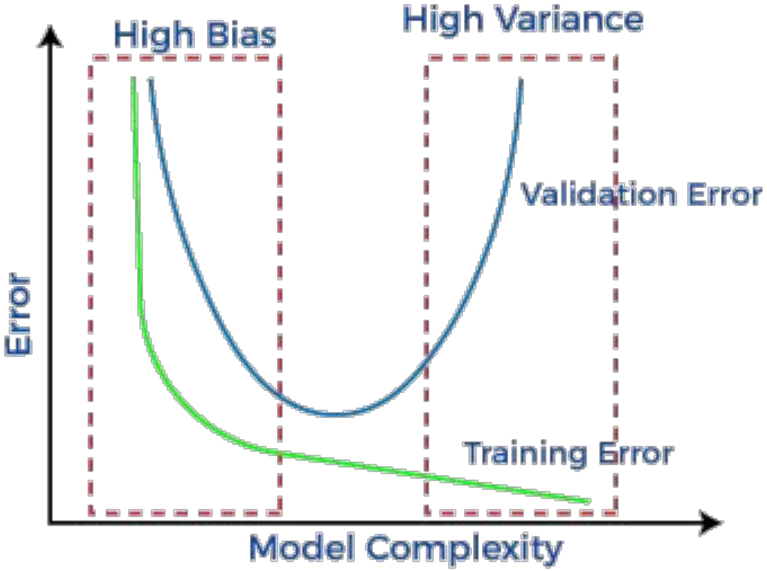
Overfitting and Underfitting


Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)
GG

High Bias

High Variance

Bias-Variance Tradeoff



References

1) https://www.dailydoseofds.com/content/images/2023/08/image-86.png
2) https://cognitree.com/wp-content/uploads/2016/07/logistic-regression-3.png
3) https://upload.wikimedia.org/wikipedia/commons/thumb/8/81/Logarithm_plots.png/300px-Logarithm_plots.png
4) https://media.geeksforgeeks.org/wp-content/uploads/20200702103829/classification1.png
5) https://utkuufuk.com/2018/06/03/one-vs-all-classification/one-vs-all.png
6) https://image.slidesharecdn.com/linearmodelsandmulticlassclassification2-170312171304/85/Linear-models-and-multiclass-classification-25-320.jpg
7) https://i.sstatic.net/XtBWr.png
8) https://i.sstatic.net/CfHCo.png
9) https://velog.io/@brandonnam/ML-7-Cost-Function-for-Logistic-Regression
10) https://h2o.ai/wiki/confusion-matrix/_jcr_content/root/section/par/advancedcolumncontro_1271832721/columns1/image.coreimg.jpeg/1689866291131/confusion-matrix.jpeg

References

11) https://miro.medium.com/v2/resize:fit:874/1*dHeKnaFtqxnMqLfCdJz5bA.png
12) https://i.ytimg.com/vi/zPG4NjIkCjc/maxresdefault.jpg?sqp=-oaymwEmCIAKENAF8quKqQMa8AEB-AH-CYAC0AWKAgwIABABGFIgZShfMA8=&rs=AOn4CLBZnskaRI7tBftWcYmNtmIZv2i-fw
13) https://miro.medium.com/v2/resize:fit:1400/1*G3imr4PVeU1SPSsZLW9ghA.png
14) https://media.geeksforgeeks.org/wp-content/cdn-uploads/20190523171258/overfitting_2.png
15) https://images.javatpoint.com/tutorial/machine-learning/images/bias-and-variance-in-machine-learning.png