



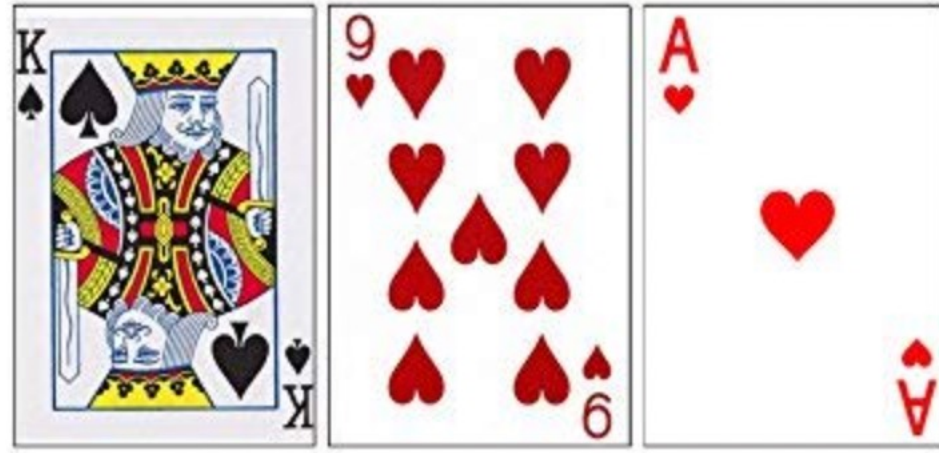
Why do We Learn Loss Functions?

Loss functions are used in neural network training.

Loss Function

- Loss functions compute the distance between the current output of models and the expected output.
- Loss functions are important to optimize training models by minimizing loss.

Actual Values



Guess

9

11

7

Absolute Error

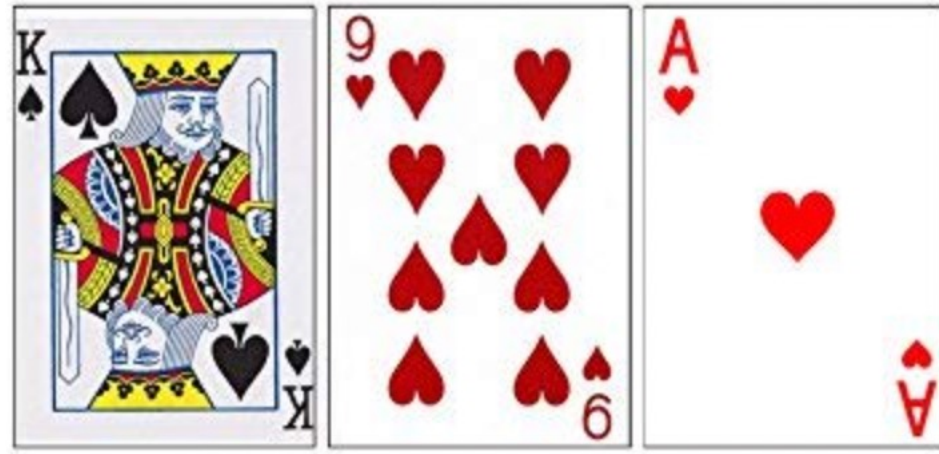
4

2

6

Mean Absolute Error: $(4+2+6)/3 = 4$

Actual Values



Guess

9

11

7

Squared Error

16

4


36

Mean Squared Error: $(4^2 + 2^2 + 6^2)/3 = 18.666666...$

Binary Classification

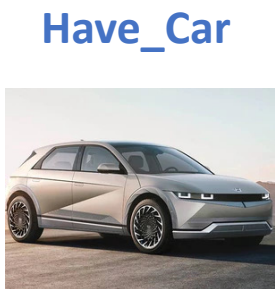
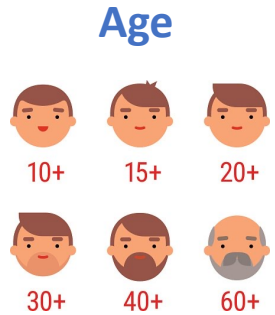
Age	Have_Car	Have_Insurance
21	0	0
48	1	1
28	0	0
19	0	0
56	1	1
65	1	1
32	1	0
24	0	1
43	1	0
22	0	0
53	1	1

Binary Classification



X		y
Age	Have_Car	Have_Insurance
21	0	0
48	1	1
28	0	0
19	0	0
56	1	1
65	1	1
32	1	0
24	0	1
43	1	0
22	0	0
53	1	1

$$y = f(x)$$



X_1

X_2

w_1

w_2

$$y = w_1 * X_1 + w_2 * X_2 + \text{bias}$$

1

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

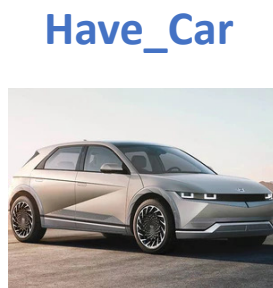
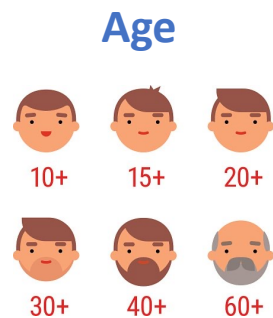


2

$$z = \frac{1}{1 + e^{-y}}$$

z , which is between 0 and 1,
shows the probability of
purchasing insurance

Age	Have_Car	Have_Insurance
21	0	0
48	1	1
28	0	0
19	0	0
56	1	1
65	1	1
32	1	0
24	0	1
43	1	0
22	0	0
53	1	1



32

1

$$w_1=1$$

$$w_2=1$$

$$\text{bias}=0$$

$$y = w_1 * X_1 + w_2 * X_2 + \text{bias}$$

$$y = 1 * 32 + 1 * 1 + 0 = 33$$

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

$$z = \frac{1}{1 + e^{-y}}$$

$$z = \frac{1}{1 + e^{-y}}$$

$$z = 0.9$$

Actual Value = 0

$$\text{error}_1 = 0.99$$

Mean Absolute Error

Loss

Total Error = error₁ + error₂ + error₃ + ...

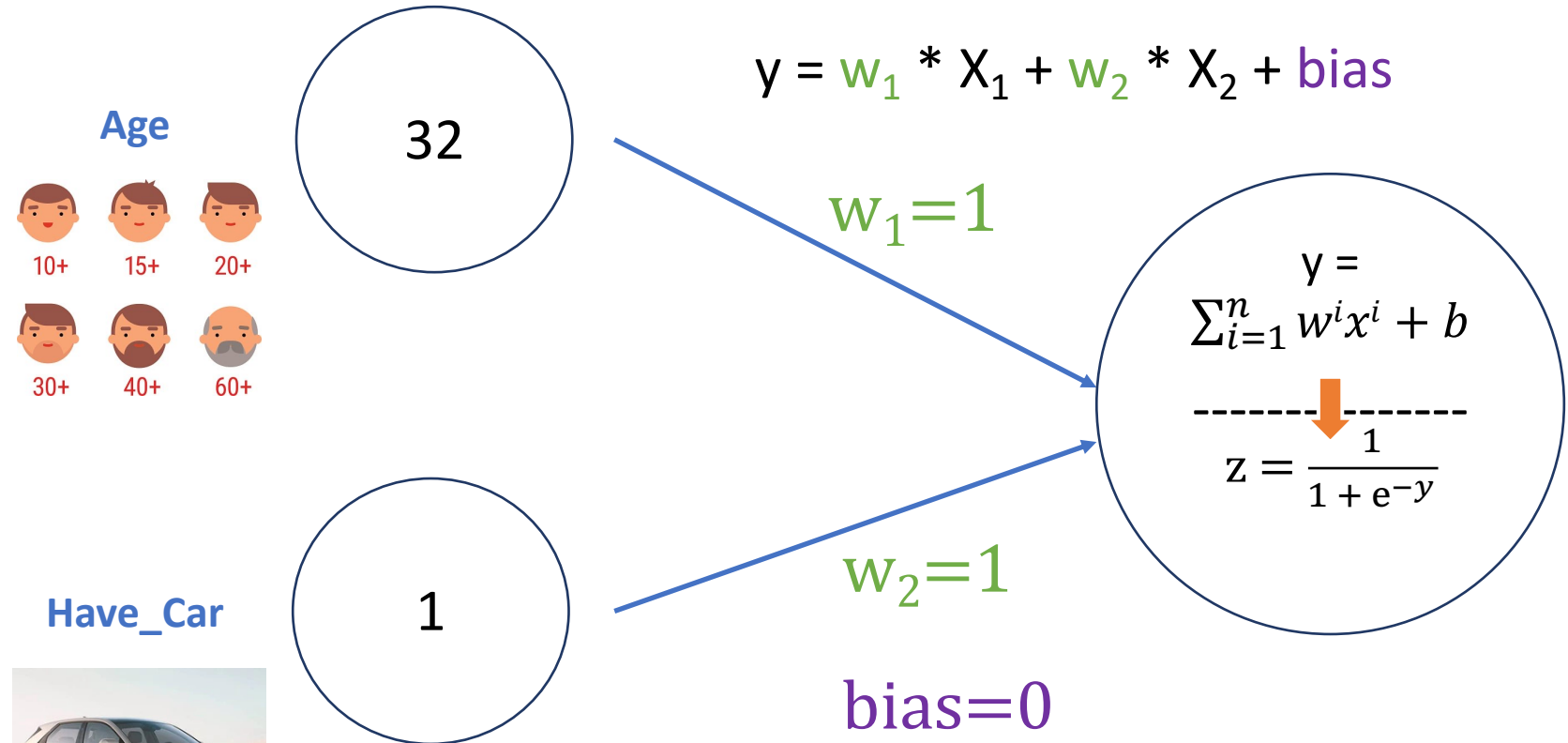
$$= \sum_{i=1}^n \text{abs}(y_i - \hat{y}_i)$$

Mean Absolute Error = $\frac{1}{n} \sum_{i=1}^n \text{abs}(y_i - \hat{y}_i)$


Cost Function

$$MAE = \frac{1}{n} \sum_{i=0}^n abs(y_i - \hat{y}_i)$$


Age	Have_Car	Have_Insurance
21	0	0
48	1	1
28	0	0
19	0	0
56	1	1
65	1	1
32	1	0
24	0	1
43	1	0
22	0	0
53	1	1



Total Error = error₁ + error₂ + error₃ + ...


$$\text{Mean Absolute Error} = \frac{1}{n} \sum_{i=1}^n \text{abs}(y_i - \hat{y}_i)$$

$$\text{Mean Squared Error} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$


$$\begin{aligned} \text{Binary Cross Entropy (Log Loss)} = \\ -\frac{1}{n} \sum_{i=1}^n y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - (\hat{y}_i)) \end{aligned}$$

Python Practice

Mean Absolute Error

$$MAE = \frac{1}{n} \sum_{i=1}^n abs(y_i - \hat{y}_i)$$

```
def mae(true_value, predicted_value):  
    total_error = 0  
    for t, p in zip(true_value, predicted_value):  
        total_error += abs(t - p)  
    print("Total Error:", total_error)  
    mae = total_error / len(true_value)  
    print("MAE:", mae)
```

Mean Squared Error

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

```
def mse(true_value, predicted_value):  
    total_error = 0  
    for t, p in zip(true_value, predicted_value):  
        total_error += (t - p)**2  
    print("Total Squared Error:", total_error)  
    mse = total_error/len(true_value)  
    print("Mean Squared Error:", mse)
```

Binary Cross Entropy (Log Loss)

Binary Cross Entropy (Log Loss) =

$$-\frac{1}{n} \sum_{i=1}^n y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - (\hat{y}_i))$$

```
def log_loss(true_value, predicted_value):  
    predicted_new = [max(i, delta) for i in predicted_value]  
    predicted_new = [min(i, 1-delta) for i in predicted_new]  
    predicted_new = np.array(predicted_new)  
    return -np.mean(true_value*np.log(predicted_new)+(1-true_value)*np.log(1-predicted_new))
```

```
log_loss(true_value, predicted_value)
```

PyTorch Loss Function Examples

- nn.L1Loss
- nn.MSELoss
- nn.CrossEntropyLoss

(<https://pytorch.org/docs/stable/nn.html>)

TensorFlow-Keras Loss Function Examples

- `tf.keras.losses.BinaryCrossentropy()`
- `tf.keras.losses.MeanAbsoluteError()`
- `tf.keras.losses.MeanSquaredError()`

(https://www.tensorflow.org/api_docs/python/tf/keras/losses)