"From Fundamentals to Building Your Own Intelligent System"

AI & Machine Learning Bootcamp 2025

Dr. Fazlul Hasan Siddiqui & Dr. Sabah Binte Noor

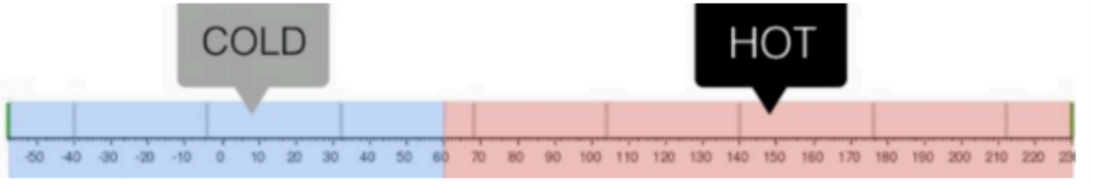
Regression

What will be the temperature tomorrow?

Fahrenheit

Classification

Will it be hot or cold tomorrow?



Fahrenheit

Binary Classification

- Spam
- Not spam

Multiclass Classification

- Dog
- Cat
- Horse
- Fish
- Bird

Multi-label Classification

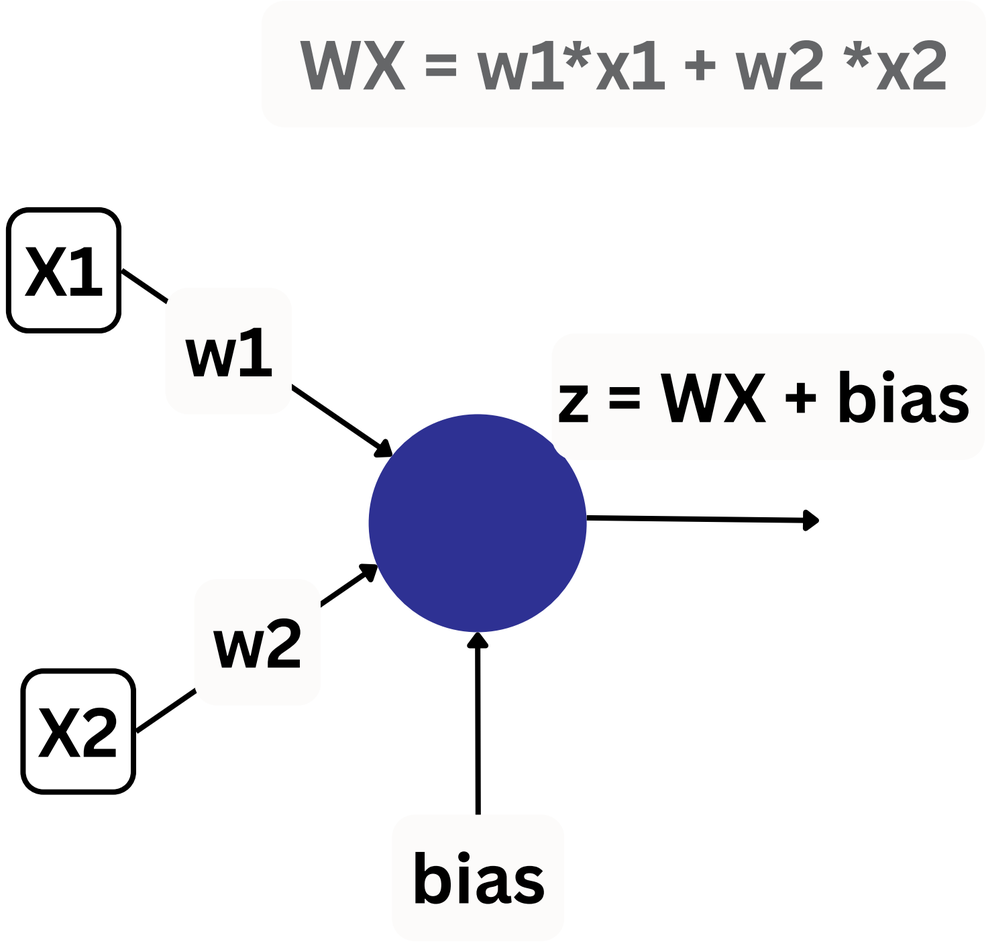
- Dog
- Cat
- Horse
- Fish
- Bird

Perceptron

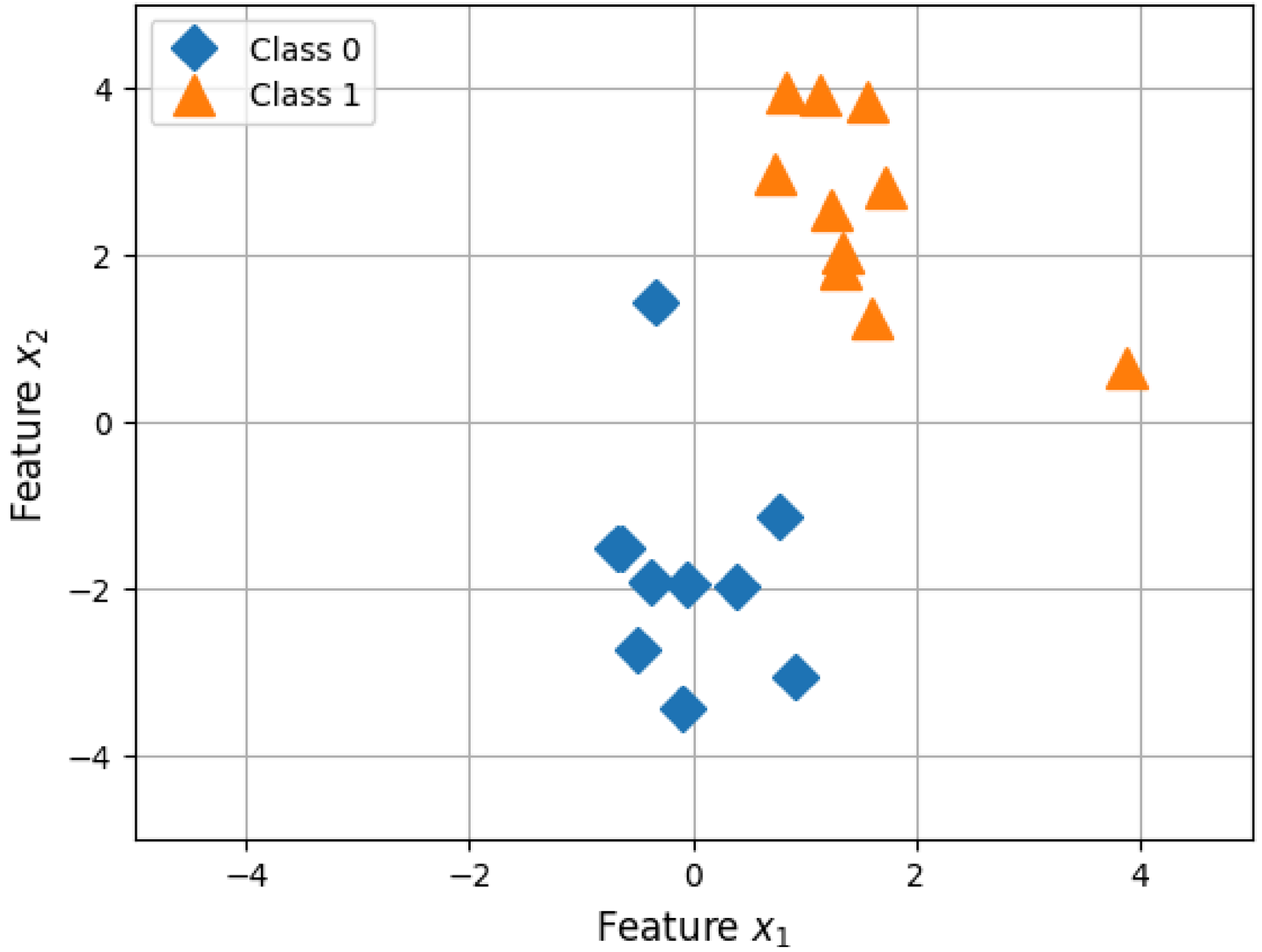
```
class  Perceptron:
        def __init__(self, num_features):
                \\ initialize the weights and bias

        def forward(self, x):
                \\  calculate z

        def update(self, x, y_true):
                \\ calculate error
                 \\ update the weight and bias
```

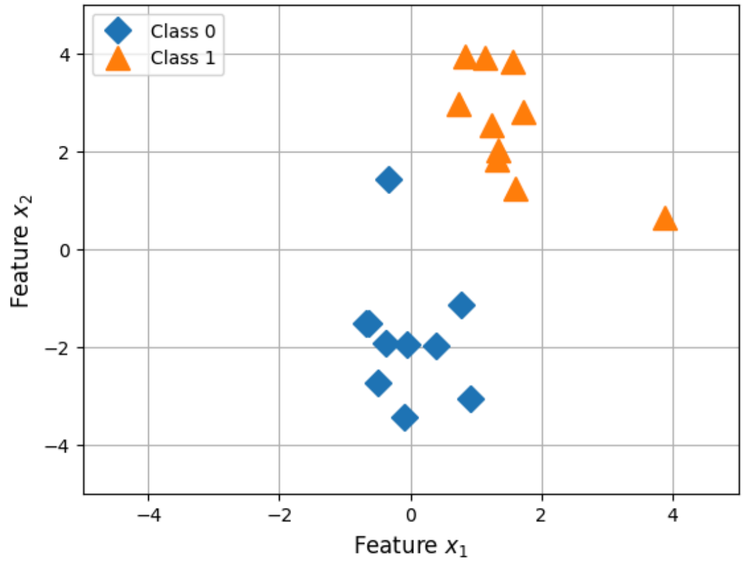


Dataset



Perceptron

```
class  Perceptron:
        def __init__(self, num_features):
                self.w = torch.zeros(num_features, 1)
                self.bias = torch.zeros(1)
```




```
PPn =  Perceptron(2)
```

self.w = torch.zeros(2, 1)
self.w = [[0.0],
[0.0]]
self.bias = [0.]

Perceptron

```
class  Perceptron:
        def __init__(self, num_features):
                self.w = torch.zeros(num_features, 1)
                self.bias = torch.zeros(1)

        def forward(self, x):
                z = self.w.T @ x + self.bias
                if z > 0.0:
                    prediction = 1
                else:
                    prediction = 0
                return prediction

PPn =  Perceptron(2)
PPn.forward(torch.tensor([0.77, -1.14]))
```

w = [[0.0], \\ shape [2,1]
[0.0]]
bias = [0.]
x = [0.77, -1.14] \\ shape [2]
w.T = [[0., 0.]] \\ shape [1,2]

Perceptron

```
class  Perceptron:
      def __init__(self, num_features):
            self.w = torch.zeros(num_features, 1)
            self.bias = torch.zeros(1)

      def forward(self, x):
             z = self.w.T @ x + self.bias
             if z > 0.0:
                 prediction = 1
             else:
                 prediction = 0
              return prediction
```

```
PPn =  Perceptron(2)
PPn.update(torch.tensor([0.77, -1.14]),1)
```

```
def update(self, x, y_true):
      prediction = self.forward(x)
      error = y_true - prediction

      self.bias += error
      self.w += error * x.view(-1,1)
      return error
```

w = [[0.0], \\ shape [2,1]
[0.0]]
bias = [0.]
x = [0.77, -1.14] \\ shape [2]

prediction = 0
error = 1
bias = 1
x = [[0.77], # After x.view(-1, 1)
[-1.14]]