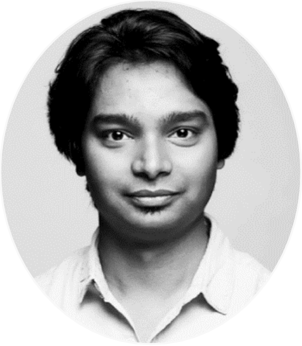
Building, Training, and Evaluating Machine Learning Model

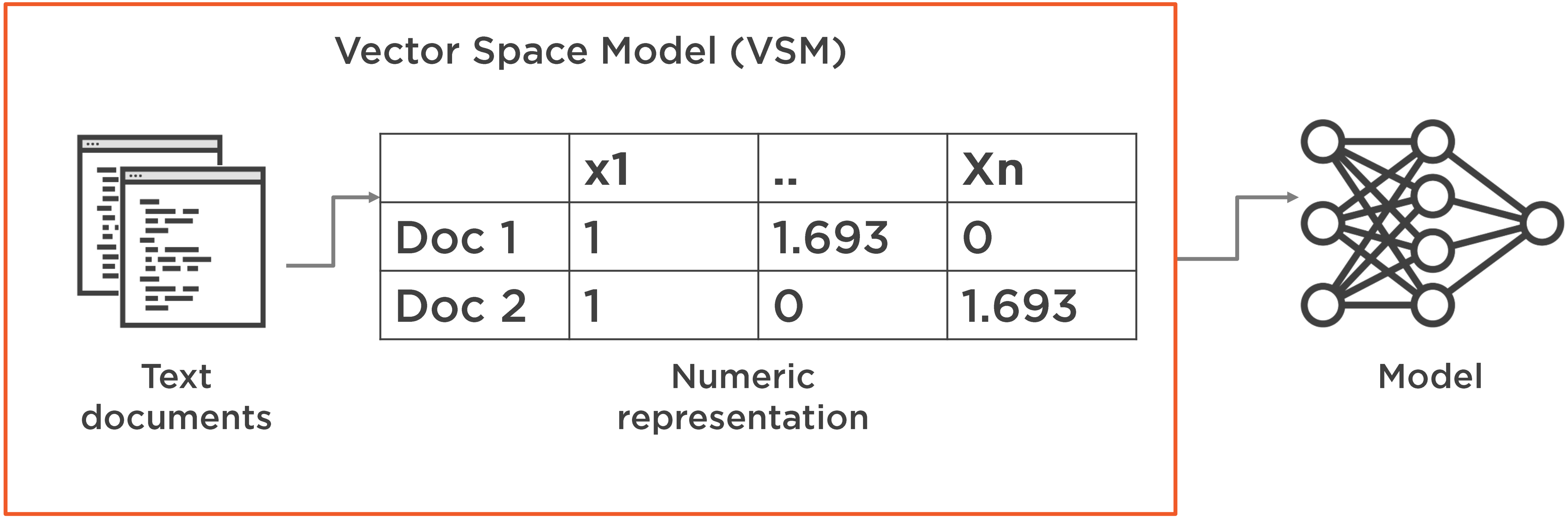


Abhishek Kumar

DATA SCIENTIST | AUTHOR | SPEAKER

@meabhishekkumar

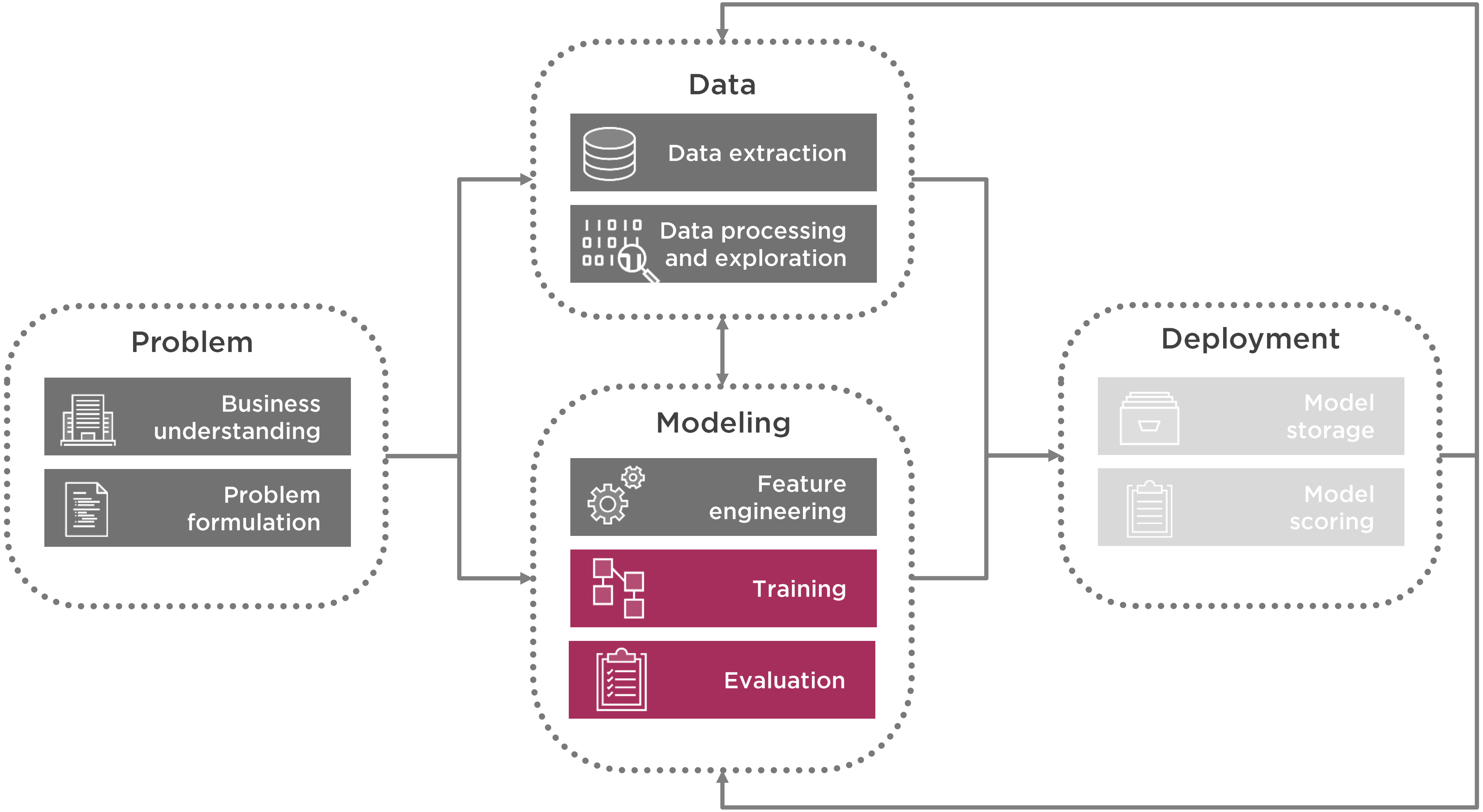
Feature Engineering



Vector Space Model (VSM)

	x1	..	Xn
Doc 1	1	1.693	0
Doc 2	1	0	1.693

Machine Learning Workflow



Overview

Build model

Train model

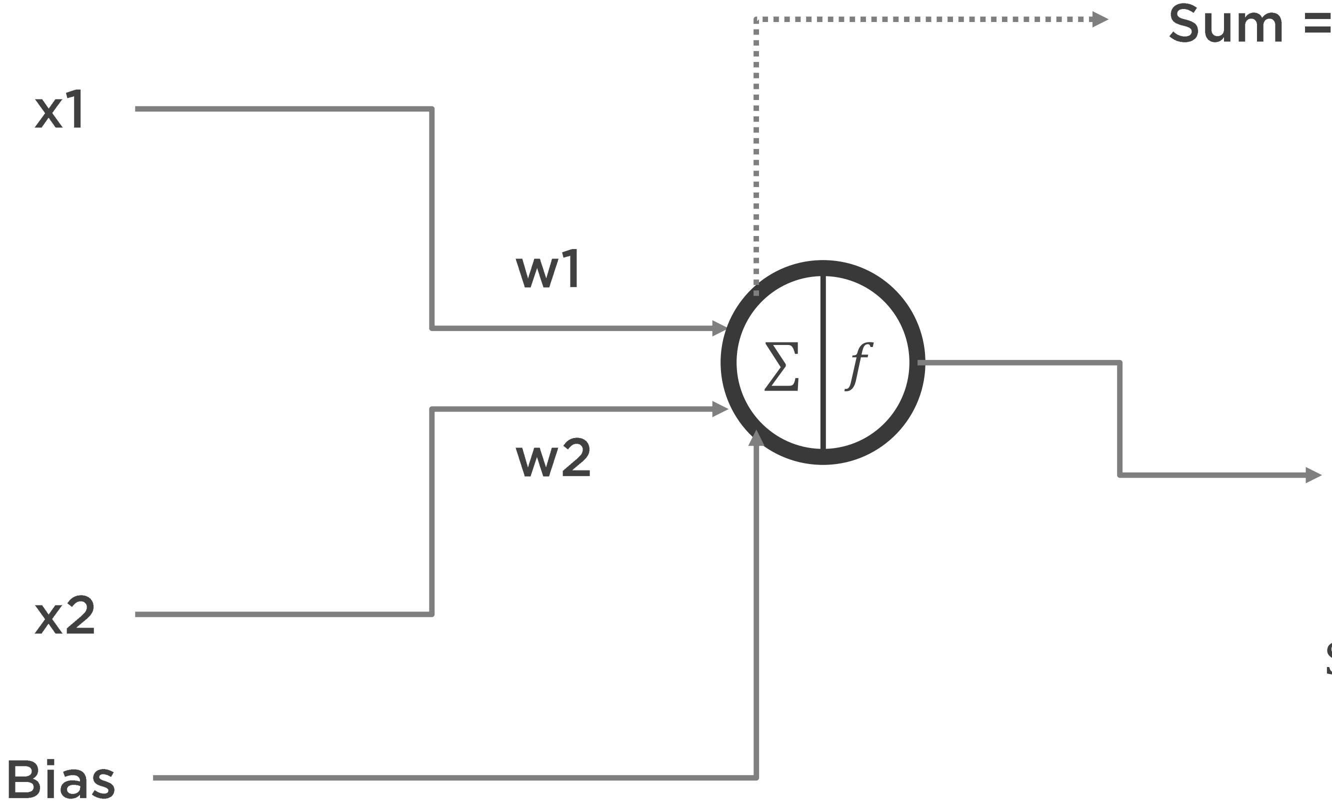
Visualize model training process

Evaluate model

Visualize model evaluation results

Run model training on Node.js

Neuron



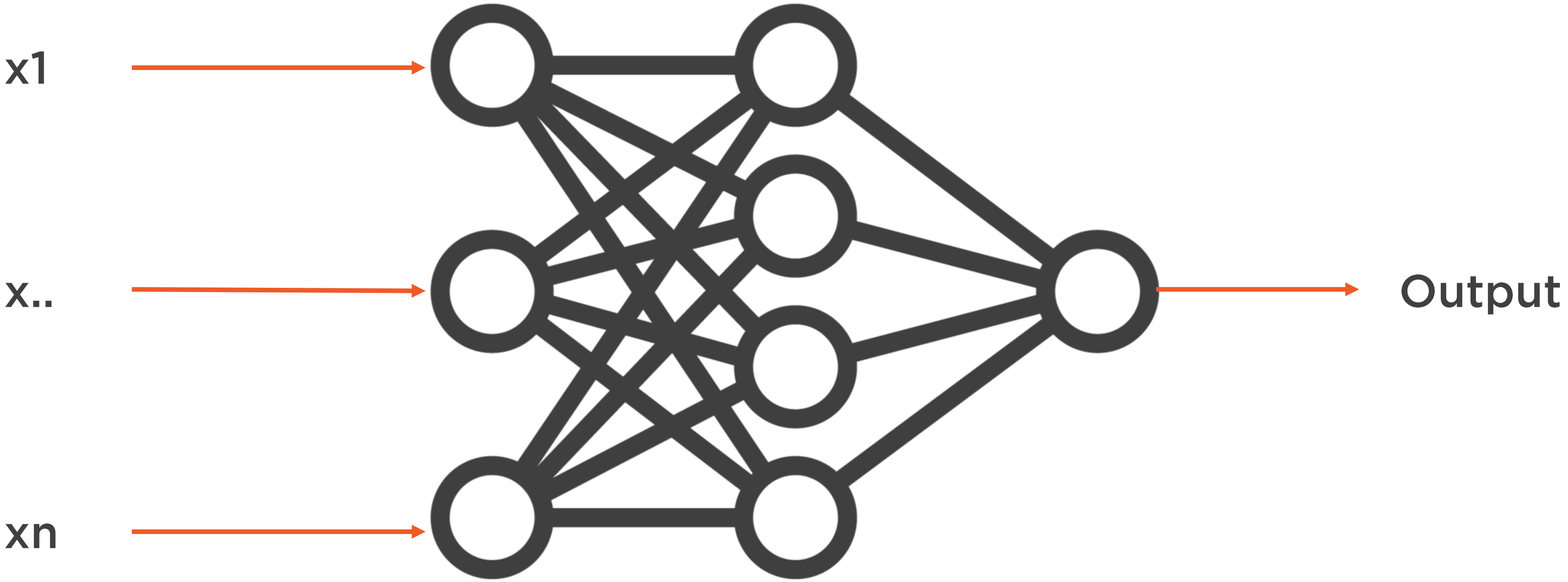
Sum = x1 * w1 + x2 * w2 + b

Output = f(sum)
f → Activation function

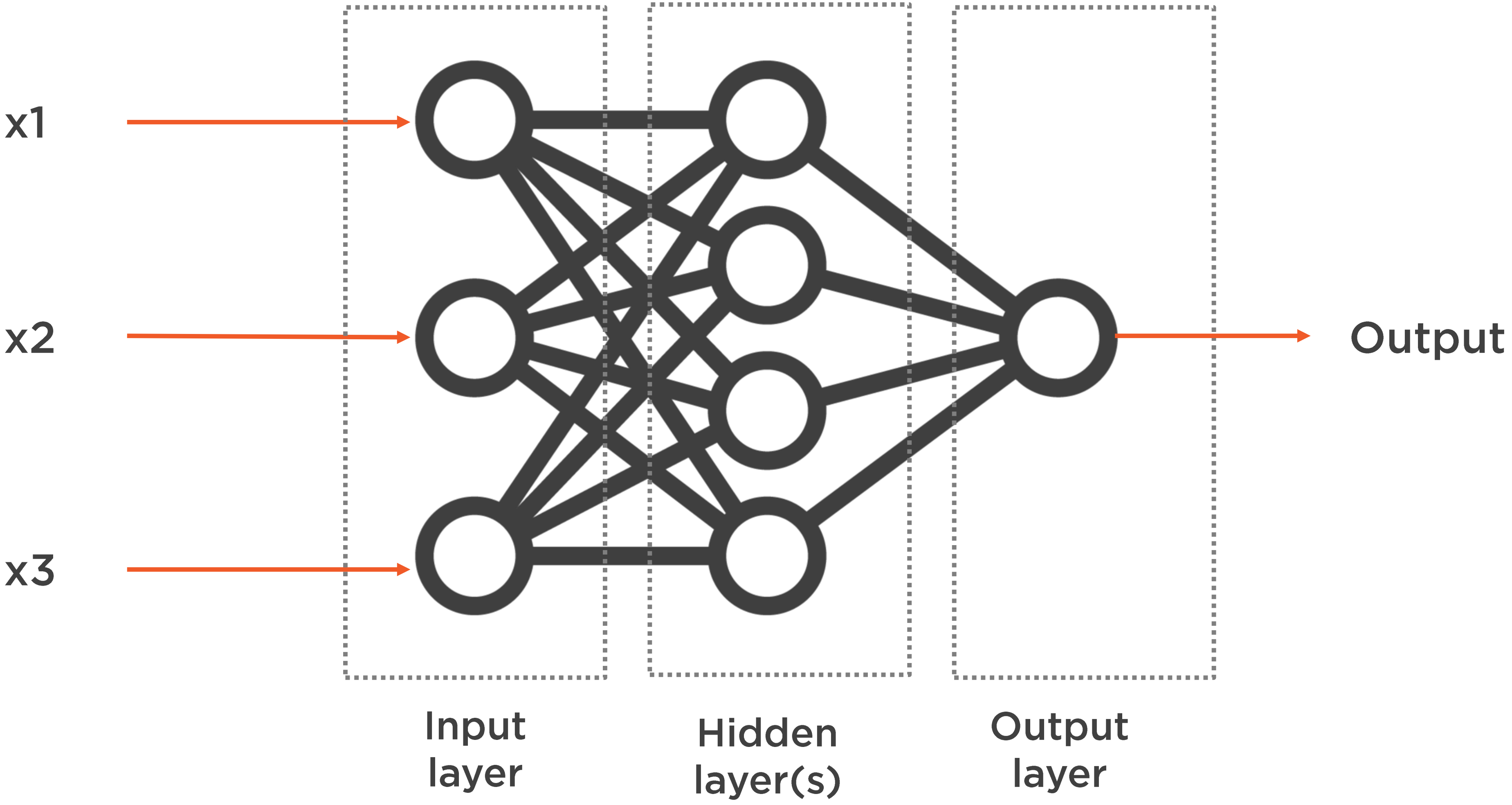
$$\text{Sigmoid :}\ \ \text{f}(sum) = \frac{1}{1 +\ e^{-(sum)}}$$

$$\text{ReLu : f}(sum) = \max(0, sum)$$

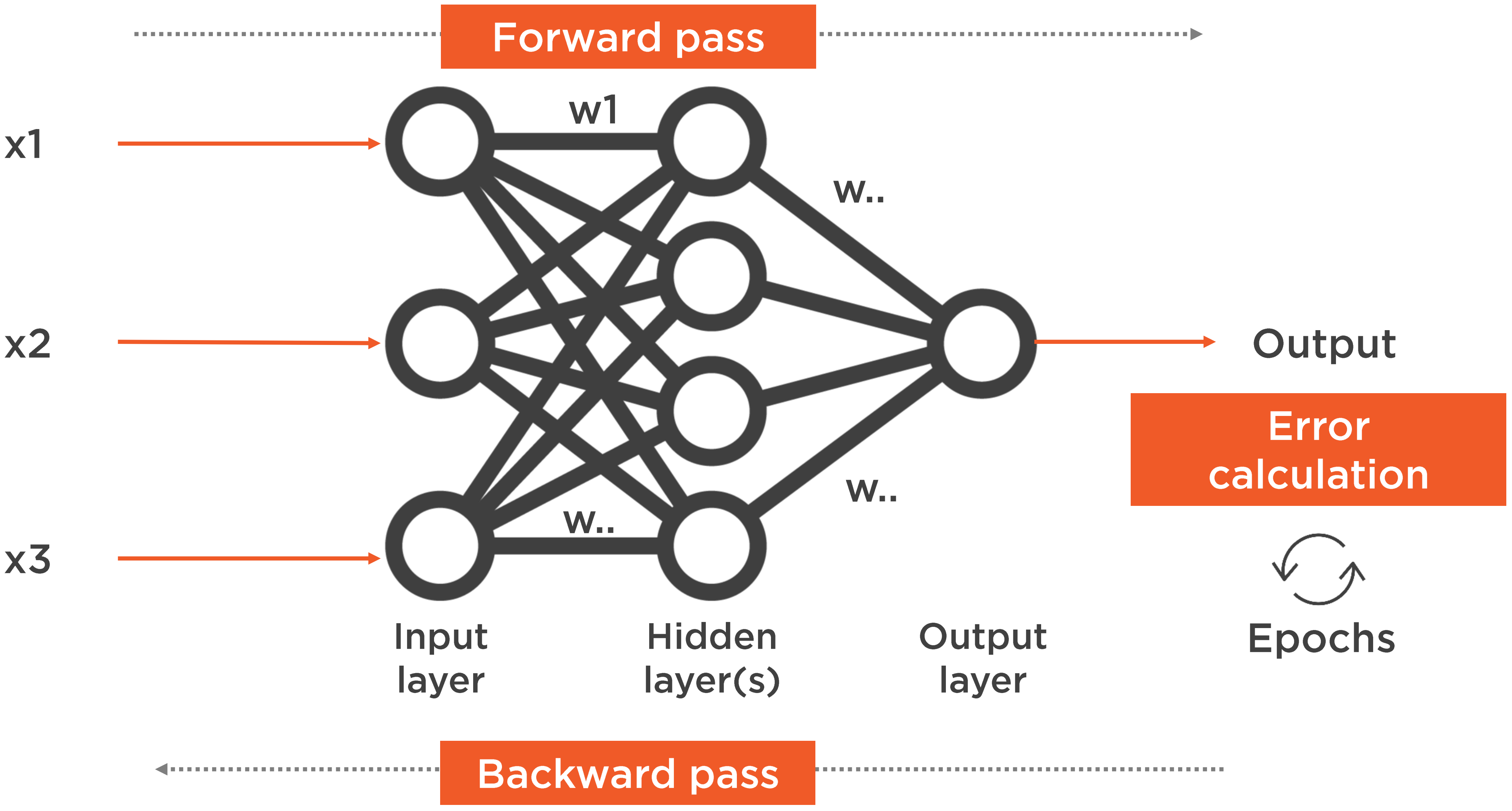
Neural Network



Feed Forward Network



Model Training



```
const model = tf.sequential();

model.add(tf.layers.dense({ inputShape: [N], activation: "relu", units: 5 }));

model.add(tf.layers.dense({ activation: "sigmoid", units: 1 }));

model.compile({ loss: "binaryCrossentropy",

                optimizer: tf.train.adam(0.06),

                metrics: ["accuracy"] });
```

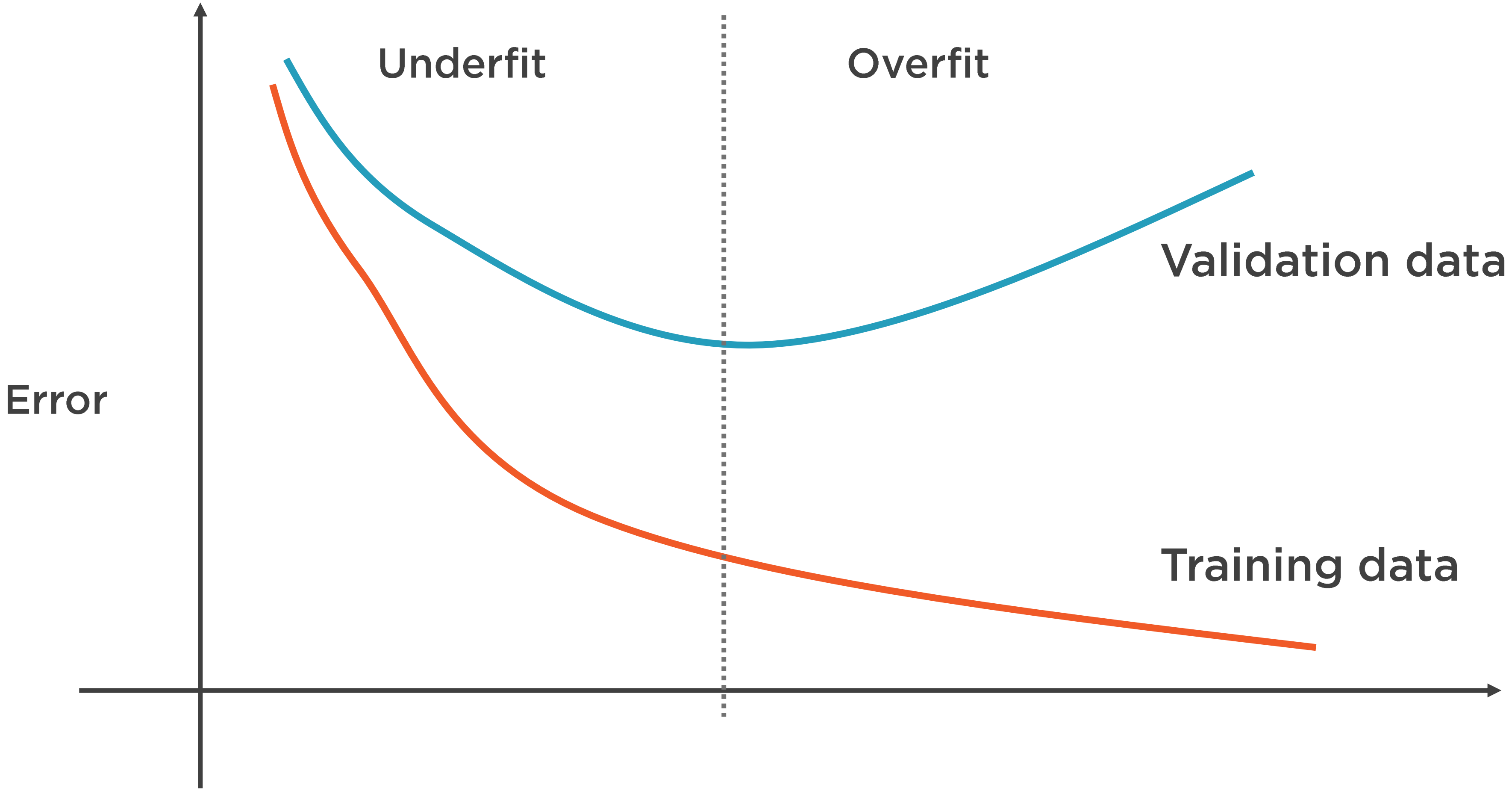
Build Model Using TensorFlow.js

Use layers API

Demo

Building neural network model

Training



```
const trainResult = await model.fitDataset(trainingDataset,
{ epochs: TRAINING_EPOCHS,
  validationData: validationDataset,
});
```

Train Model Using TensorFlow.js

Fit method

```
const trainResult = await model.fitDataset(trainingDataset,
{ epochs: TRAINING_EPOCHS,
  validationData: validationDataset,
  callbacks: {
    onEpochEnd: async (epoch, logs) => { console.log(epoch + “ : " + logs.loss); }},
    });
```

Train Model Using TensorFlow.js

Use callback to track progress

```
const trainResult = await model.fitDataset(trainingDataset,
{ epochs: TRAINING_EPOCHS,
  validationData: validationDataset,
  callbacks: tf.callbacks.earlyStopping({monitor: 'val_acc’})
});
```

Train Model Using TensorFlow.js

Use callback for early stopping

Demo

Training neural network model

Demo

Visualizing training performance

Demo

Evaluating model performance

Accuracy

$$\text{Accuracy} = \frac{\text{Number of true predictions}}{\text{Total number of predictions}} = \frac{80}{100} = 0.8\ (80\%)$$

Confusion Matrix

		Predicted	
		0	1
True	0	True negative	False positive
	1	False negative	True positive

Demo

Visualizing model performance metrics

Demo

Running training in Node.js

Summary

Neural network

Build model

- Layers API

Training

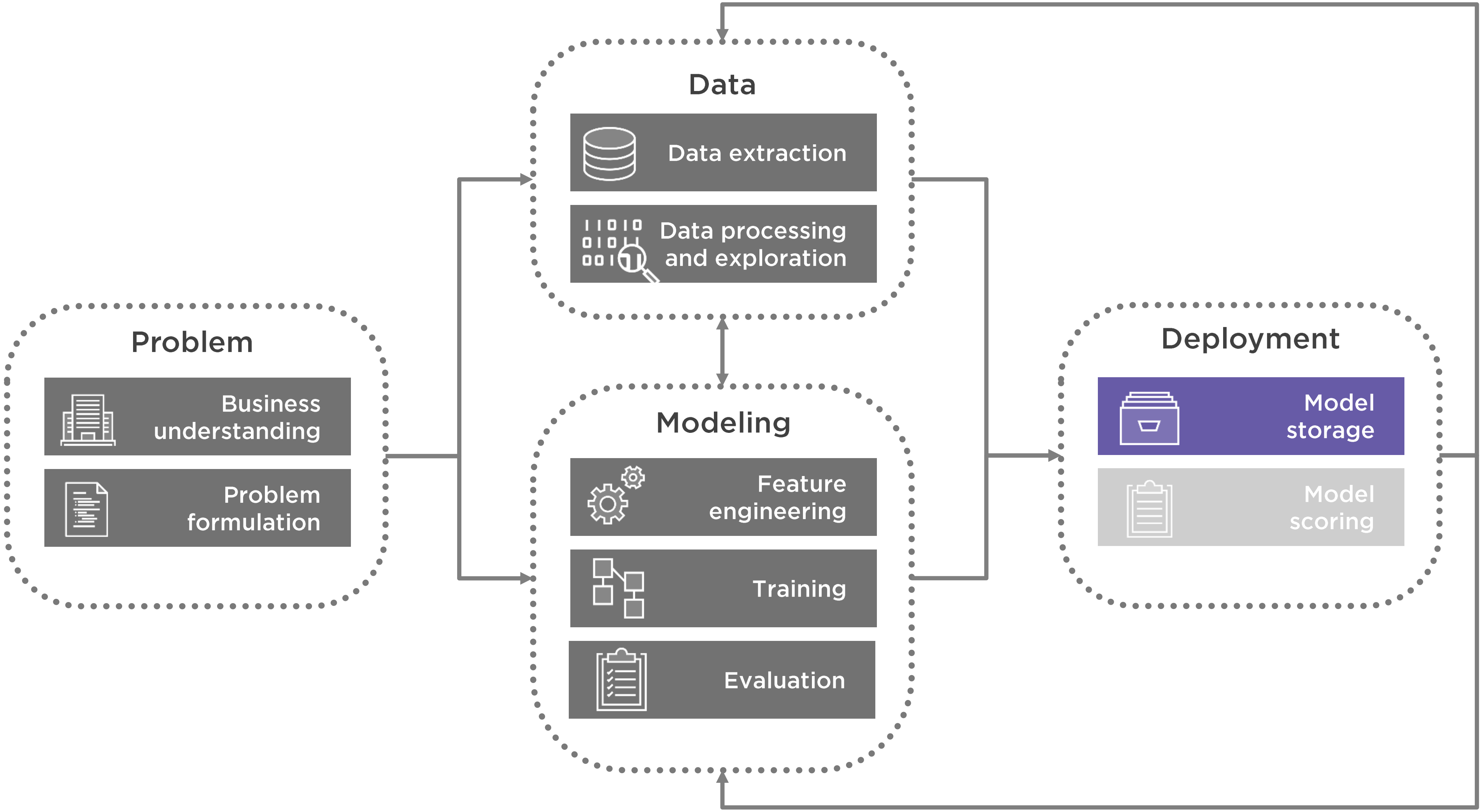
- Using callback to show training process and avoid overfitting
- Visor for visualization

Evaluate

- Accuracy
- Confusion matrix

Server-side execution with Node.js

Machine Learning Workflow



Up Next:
Saving and Loading Machine Learning Model