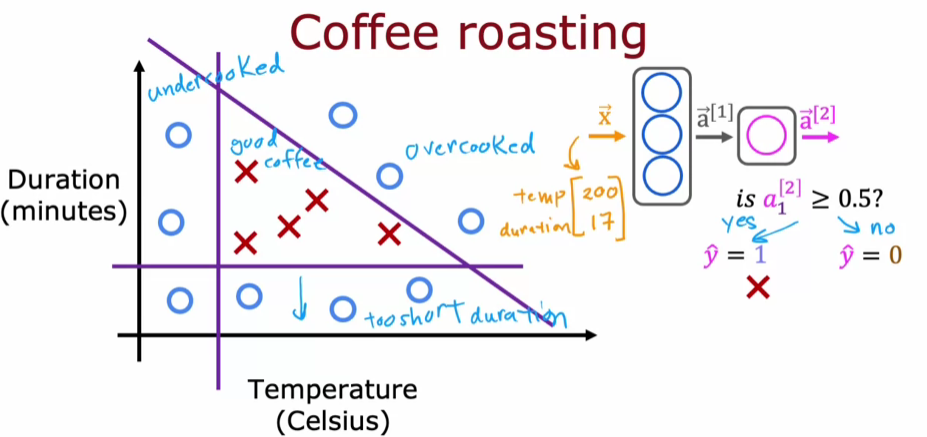
**TENSOR FLOW IMPLEMENTATION**

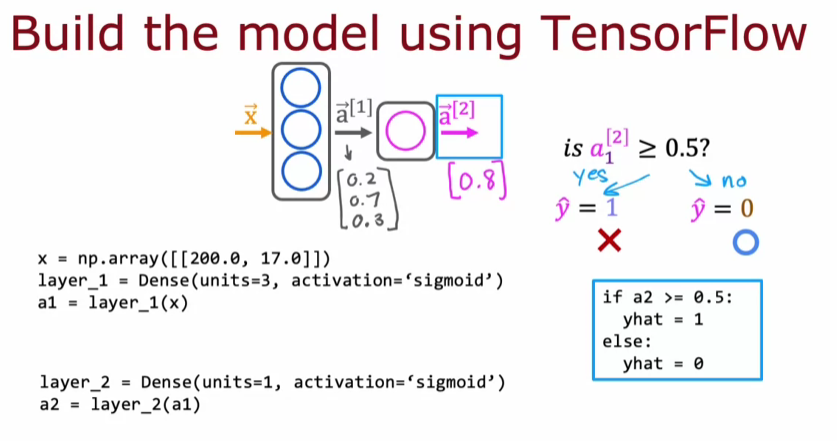
**INFERENCE IN CODE**

**Understanding Neural Networks**

* **Neural networks can apply the same algorithm across different applications, such as optimizing coffee roasting by controlling temperature and duration.**
* **A dataset is created with temperature and duration parameters, labeled to indicate whether the resulting coffee is good or bad.**

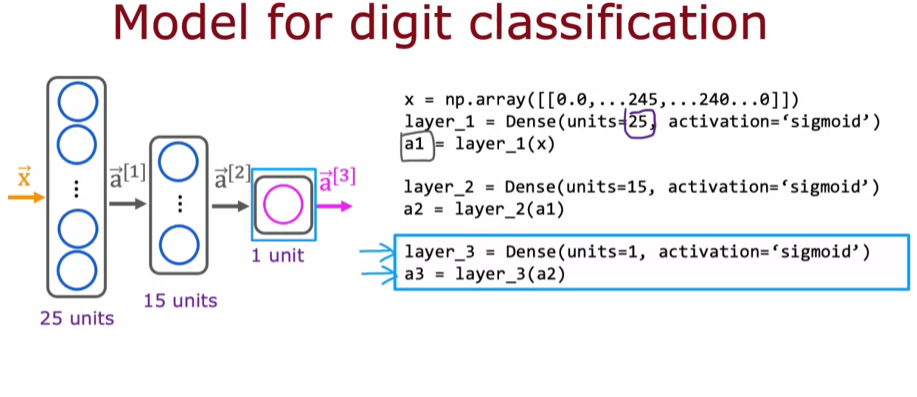
****

**Implementing Inference in TensorFlow**

* **The process involves creating layers in the neural network, starting with a first hidden layer (Layer 1) with three units and a sigmoid activation function.**
* **Forward propagation is performed by applying the layers to input features, resulting in activation values that help determine the quality of the coffee.**

**Example of Handwritten Digit Classification**

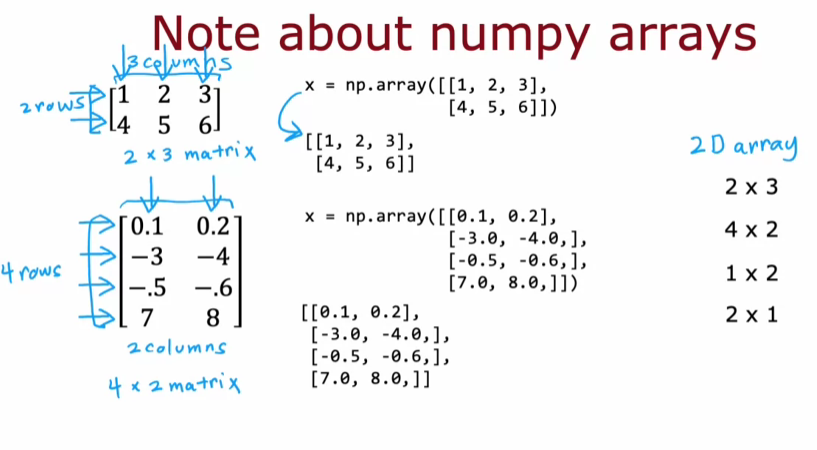
* **Another example involves classifying handwritten digits using pixel intensity values, where multiple dense layers are used to carry out inference.**
* **The final output can be thresholded to make binary predictions, demonstrating the syntax for inference in TensorFlow.**

****

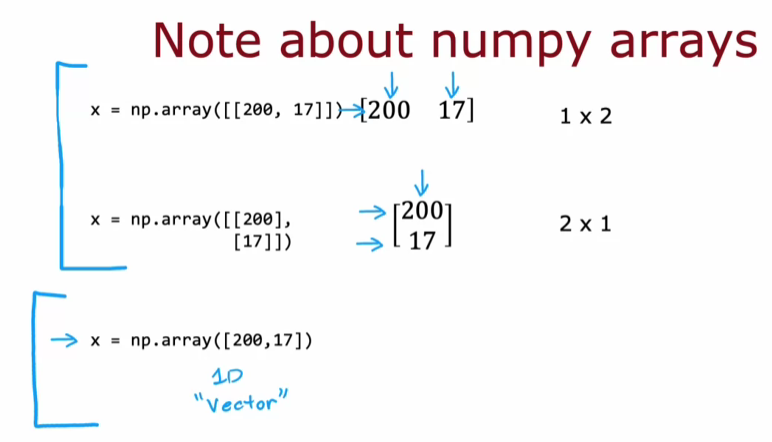
**DATA IN TENSORFLOW**

**Understanding Data Representation**

* **NumPy uses arrays to represent vectors and matrices, with dimensions indicated as rows by columns (e.g., a 2x3 matrix has 2 rows and 3 columns).**
* **TensorFlow, on the other hand, represents data as tensors, which are similar to matrices but optimized for handling large datasets efficiently.**

****

* **The distinction between row vectors (1x2) and column vectors (2x1) is made clear by the use of double versus single square brackets in NumPy.**

****

**Converting Between Formats**

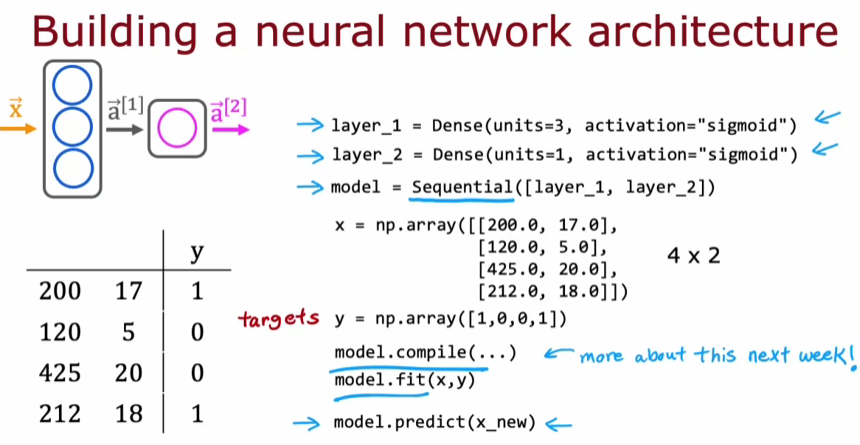
* **When using TensorFlow, a 1x2 matrix is represented as a tensor, and you can convert it back to a NumPy array using the .numpy() method.**
* **Understanding these conversions is crucial for writing efficient code and ensuring compatibility between NumPy and TensorFlow.**

**BUILDING A NEURAL NETWORK**

* **You can create a neural network by sequentially stringing together layers using TensorFlow's Sequential function, which automates the forward propagation process.**
* **Instead of manually passing data through each layer, you can define the model in a more compact way, making the code cleaner and easier to understand.**

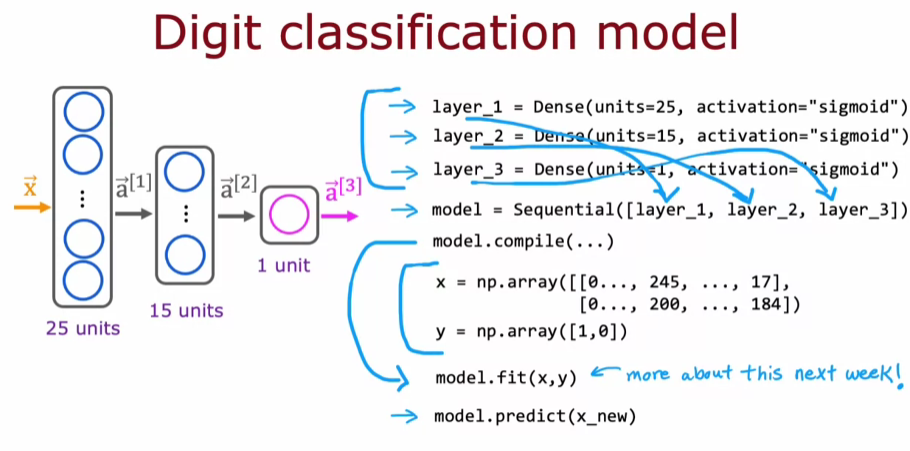
**Training the Model**

* **To train the neural network, you need to compile the model with specific parameters and then fit it to your training data using model.fit(X, Y).**
* **The training data can be organized in a matrix for inputs and a one-dimensional array for target labels, making it straightforward to manage.**

****

**Inference and Predictions**

* **For making predictions with new data, you can simply call model.predict(X\_new), which handles the forward propagation automatically.**
* **This approach allows you to focus on the model's architecture and training without getting bogged down in the details of layer-by-layer computation.**

****