DATA SOCIETY

IntroToNeuralNetworks - IntroToTensorflow - 1

One should look for what is and not what he thinks should be. (Albert Einstein)

Intro To TensorFlow: Topic introduction

In this part of the course, we will cover the following concepts:

- Overview of TensorFlow / Keras building blocks
- Implement and fit a neural network model using Tensorflow on train data
- Evaluating neural network model on test data

Module completion checklist

Objective	Complete
Introduce TensorFlow and Keras	
Describe how to define a neural network model using Tensorflow	

Warm up: What is TensorFlow

- What is TensorFlow?
- Follow the link found ***here*** to watch a short video about TensorFlow
- What feature was the most interesting?
- What applcation was the most surprising?
- In this part of the course, we will explore more about TensorFlow

TensorFlow

- TensorFlow is a base production-grade, deep learning framework that allows for distributed processing and data flow graphs, modeled on tensor data objects (originally from Google)
- Its core is in C++ with a layer of Python around it, abstracting processes into graphs and giving the core instructions
- More information about TensorFlow can be found on its website using the link provided ***here***

Keras

- Keras is a more high-level wrapper around TensorFlow, which makes it easier to build out models that resemble other Python & scikit-learn functions that you are used to
- It is a deep learning library built specifically for Python
- It's a few layers up the abstraction chain from the core in C++, but it allows us to train models easily
- *Since TensorFlow 2.0 fully integrated Keras, there is no need to install Keras separately!*
- More information about Keras can be found on the TensorFlow website using the link provided **here**

TensorFlow and Keras

- TensorFlow and Keras do not HAVE to go together
- Keras can be run on top of other systems like CNTK, and you can code directly in TensorFlow
- If you want to develop your own algorithms, you can do so with TensorFlow

TensorFlow vs Keras

- Although TensorFlow native modules alone can do everything (and then some) that Keras modules are built for, there are a few advantages of using Keras based on the four guiding principles used by Francois Chollet (the author of Keras):
  - **modularity**: a model can be understood as a sequence or a graph alone; all the concerns of a deep learning model are discrete components that can be combined in arbitrary ways
  - **minimalism**: the library provides just enough to achieve an outcome, no frills and maximizing readability
  - **extensibility**: new components are easy to add and use within the framework, intended for researchers to trial and explore new ideas
  - **Python**: no separate model files with custom file formats - everything is native Python

TensorFlow’s APIs

- TensorFlow has APIs that support several languages including:
  - Python
  - JavaScript
  - C++
  - Java
- Additional community-supported languages include C#, Julia, Ruby, Rust, and Scala
- There are several available modules such as Lite and TensorFlow Extended (TFX)
- More information about TensorFlow’s APIs can be found on the TensorFlow website using the link provided ***here***

Tensors

- Tensors are multi-dimensional arrays with a uniform type (called a dtype)
- All tensors are immutable like Python numbers and strings: you can never update the contents of a tensor, only create a new one.
- When working with TensorFlow, it’s fairly safe to assume that you will end up working with tensors at some point

- Tensor vocabulary:
    - **Shape**: The length (number of elements) of each of the dimensions of a tensor.
    - **Rank**: Number of tensor dimensions.
    - **Axis or Dimension**: A particular dimension of a tensor.
    - **Size**: The total number of items in the tensor, the product shape vector

Tensor shape

- You might already be familiar with scalars, vectors, matrices, and tensors, but let's discuss how these translate into TensorFlow

Tensor shape	Math type
0	scalar (magnitude)
1	vector (magnitude and direction)
2	matrix
3	3D-tensor (cube)
4+	multi-dimensional tensor (hypercube)

- More information about tensors can be found on the TensorFlow website using the link provided **here**

Load TensorFlow for Python

- Loading TF is very simple, just run:

```
# Import tensorflow.
import tensorflow as tf
```

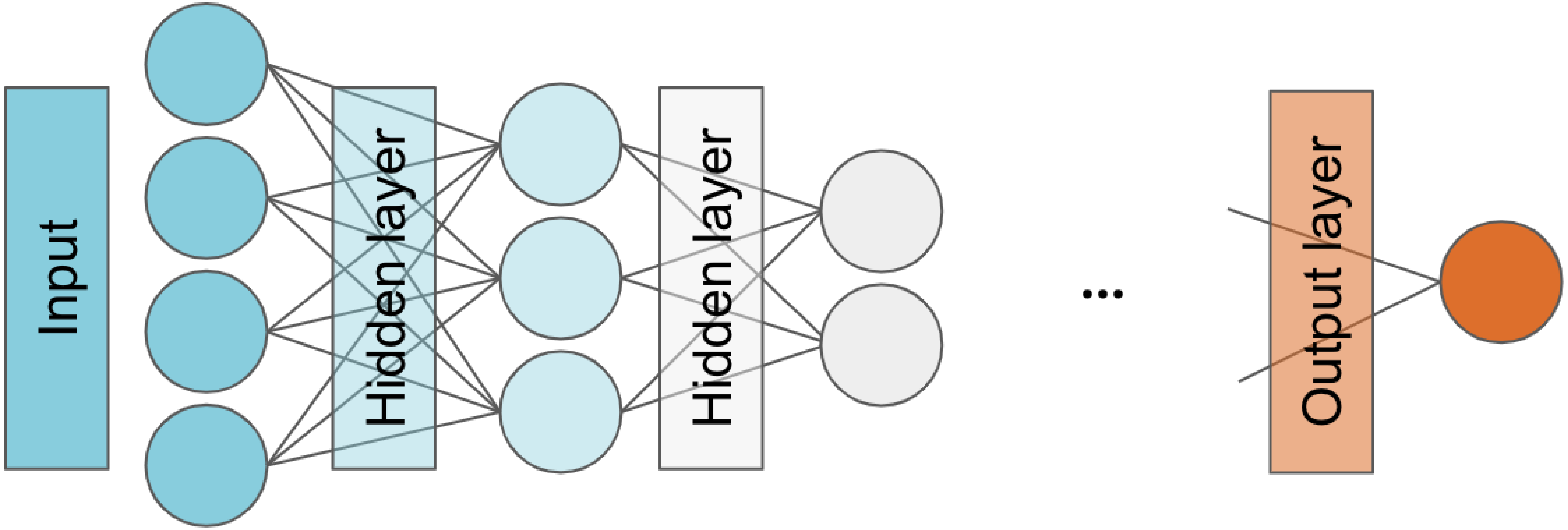
- The `tf` library is the main TensorFlow package that is loaded into the environment

- The list of all modules available through tf can be found on the Tensorflow website using the link ***here***

Module completion checklist

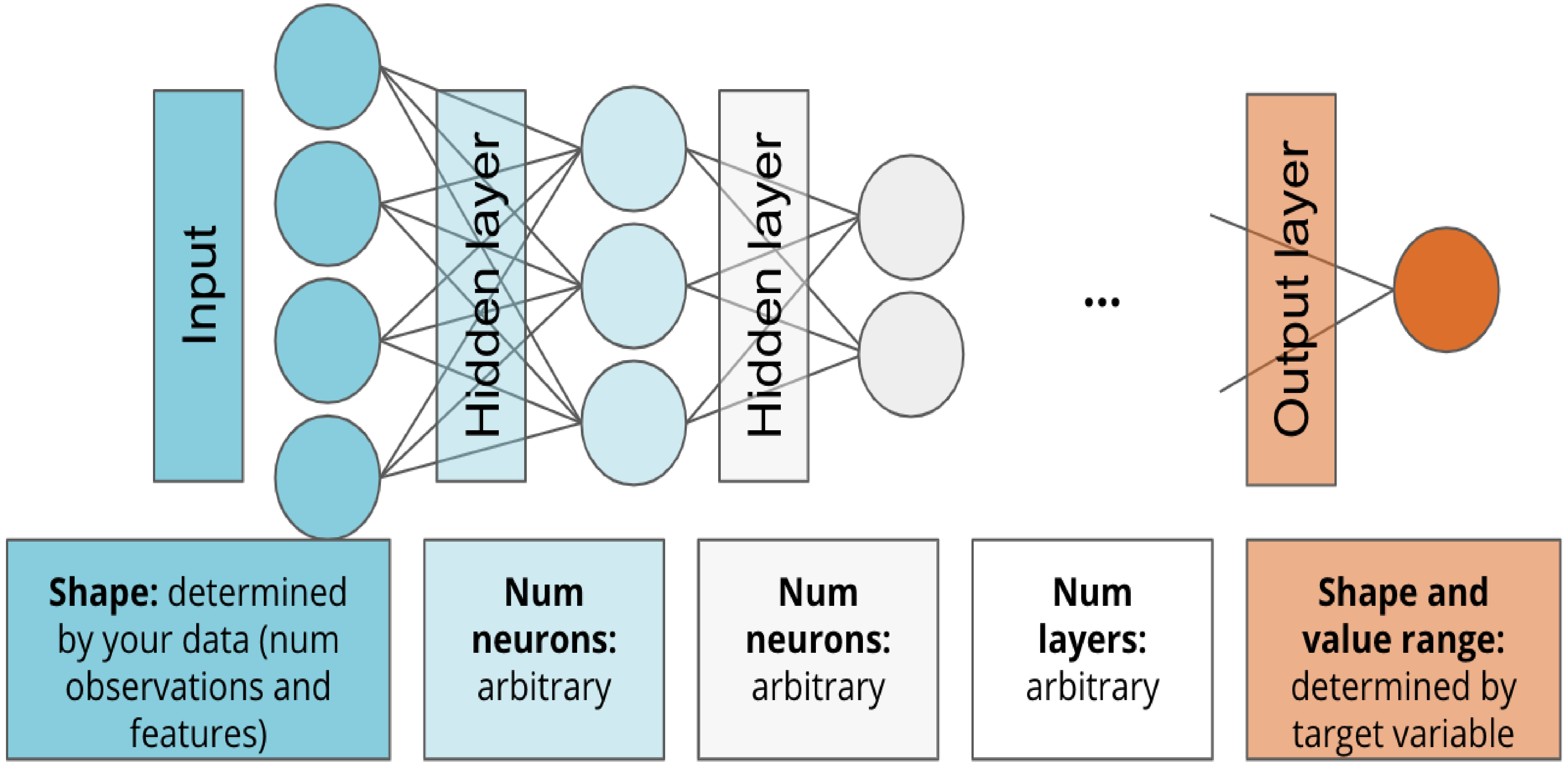
Objective	Complete
Introduce TensorFlow and Keras	✔
Describe how to define a neural network model using Tensorflow	

Define the model: how many layers



- Any neural network modeling starts with the architecture
- We need 1 **input** and 1 **output** layer with *at least* 1 **hidden** layer in-between
- The number of **hidden** layers determines the **depth** of the network and is something that you would need to adjust for every model you build – it's a trial-and-error process

Define the model: layers and neurons



Shape: determined by your data (num observations and features)

Num neurons: arbitrary

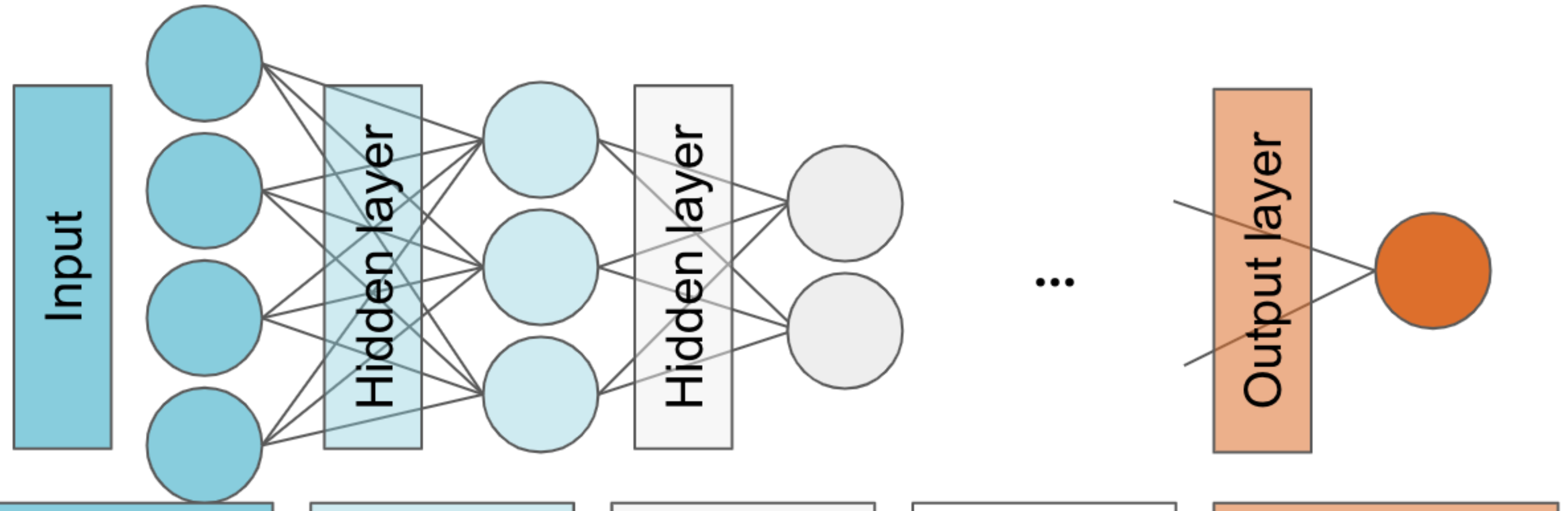
Num neurons: arbitrary

Num layers: arbitrary

Shape and value range: determined by target variable

- The **output** for a binary classification problem will have a **single neuron with a sigmoid** activation function

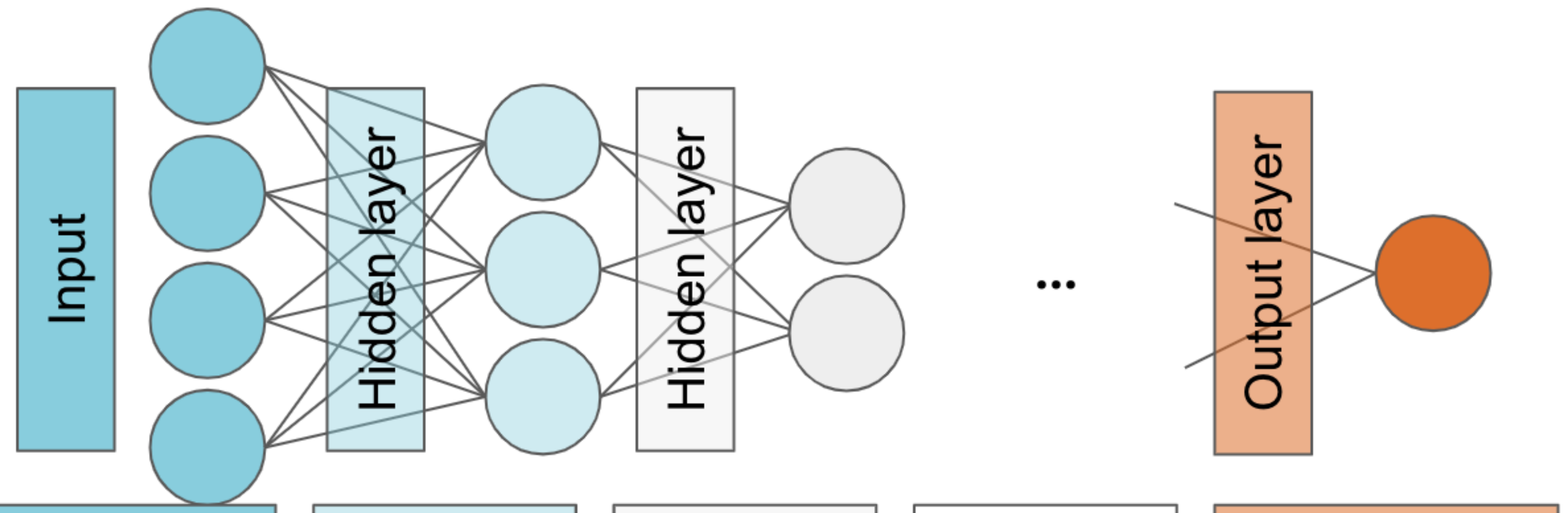
Define the model: implemented in tf.keras



Shape: determined by your data (num observations and features)	**Num neurons:** arbitrary	**Num neurons:** arbitrary	**Num layers:** arbitrary	**Shape and value range:** determined by target variable
Keras layer: *Dense*, where *input_shape* is your data shape, and *number of neurons* is for the *1st hidden layer*		**Keras layer:** *Dense* with *neuron number* for the *2nd hidden layer*	**Keras layers:** just keep adding …	**Keras layer:** *Dense* with *1 neuron*

- Keras model architecture follows this logic:
    - choose model type
    - add your layers one-by-one and make sure to pick your parameters properly

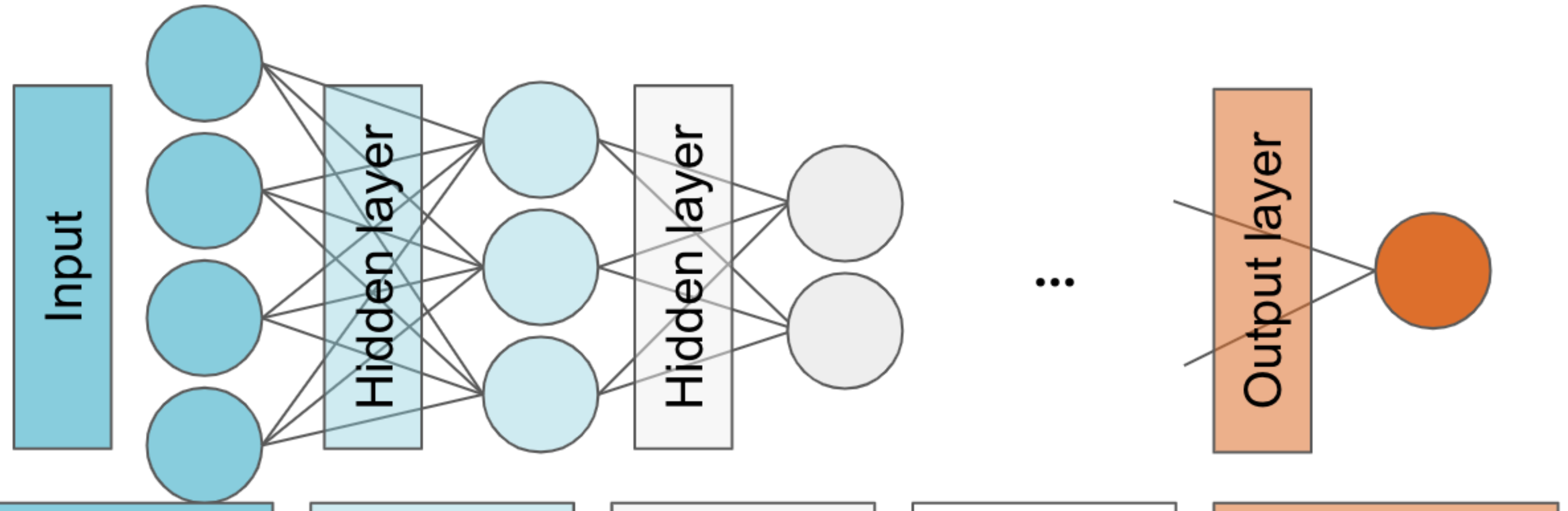
Define the model: implemented in tf.keras (cont'd)



Shape: determined by your data (num observations and features)	**Num neurons:** arbitrary	**Num neurons:** arbitrary	**Num layers:** arbitrary	**Shape and value range:** determined by target variable
Keras layer: *Dense*, where *input_shape* is your data shape, and *number of neurons* is for the *1st hidden layer*		**Keras layer:** *Dense* with *neuron number* for the *2nd hidden layer*	**Keras layers:** just keep adding ...	**Keras layer:** *Dense* with *1 neuron*

- There are two main types of models available in Keras:
  - **Sequential model:** A linear stack of layers with each layer having exactly one input tensor and one output tensor
  - **Functional API:** Uses the same layers as the sequential model but provides more flexibility and can handle models with non-linear topology, shared layers, and multiple inputs or outputs

Define the model: implemented in tf.keras (cont’d)



Shape: determined by your data (num observations and features)	**Num neurons:** arbitrary	**Num neurons:** arbitrary	**Num layers:** arbitrary	**Shape and value range:** determined by target variable
Keras layer: *Dense*, where *input_shape* is your data shape, and *number of neurons* is for the *1st hidden layer*		**Keras layer:** *Dense* with *neuron number* for the *2nd hidden layer*	**Keras layers:** just keep adding ...	**Keras layer:** *Dense* with *1 neuron*

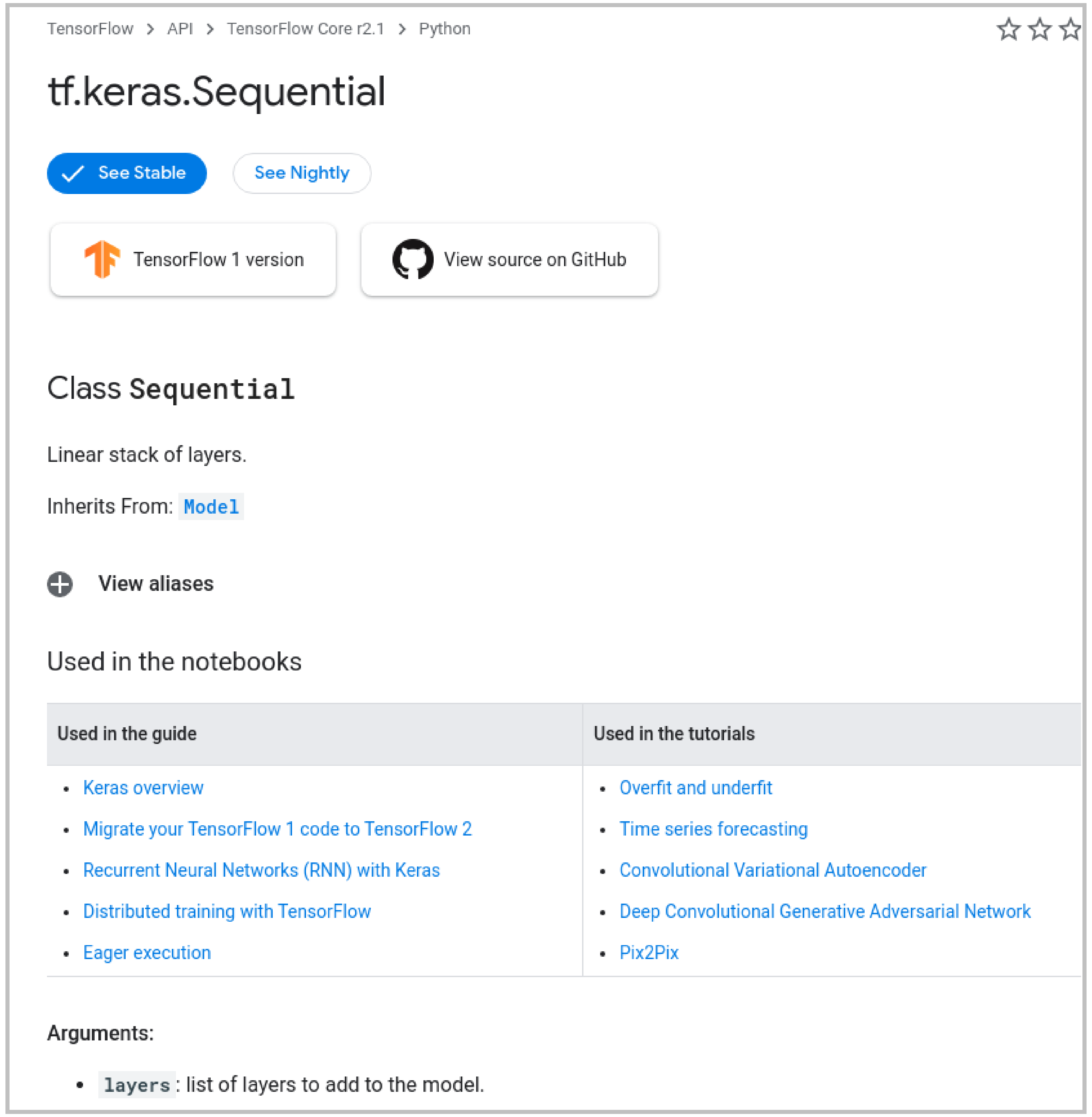
- We will be discussing the `Sequential` model first, as we’ll start by working with one input tensor and one output tensor at each layer
- Each layer will be `Dense`, meaning densely connected to the preceding and following layers
- For each layer we just need to pick appropriate parameters

Keras Sequential model with Dense layers

- To create a `Sequential` model, you must pass a **list of layers** to the model constructor

```
Sequential([
  Dense()
  Dense()
   ...
])
```

- Package documentation is available on the TensorFlow website and can be accessed using ***this link***



tf.keras.Sequential

Class `Sequential`

Linear stack of layers.

Inherits From: `Model`

Used in the notebooks

Used in the guide	Used in the tutorials
Keras overview	Overfit and underfit
Migrate your TensorFlow 1 code to TensorFlow 2	Time series forecasting
Recurrent Neural Networks (RNN) with Keras	Convolutional Variational Autoencoder
Distributed training with TensorFlow	Deep Convolutional Generative Adversarial Network
Eager execution	Pix2Pix

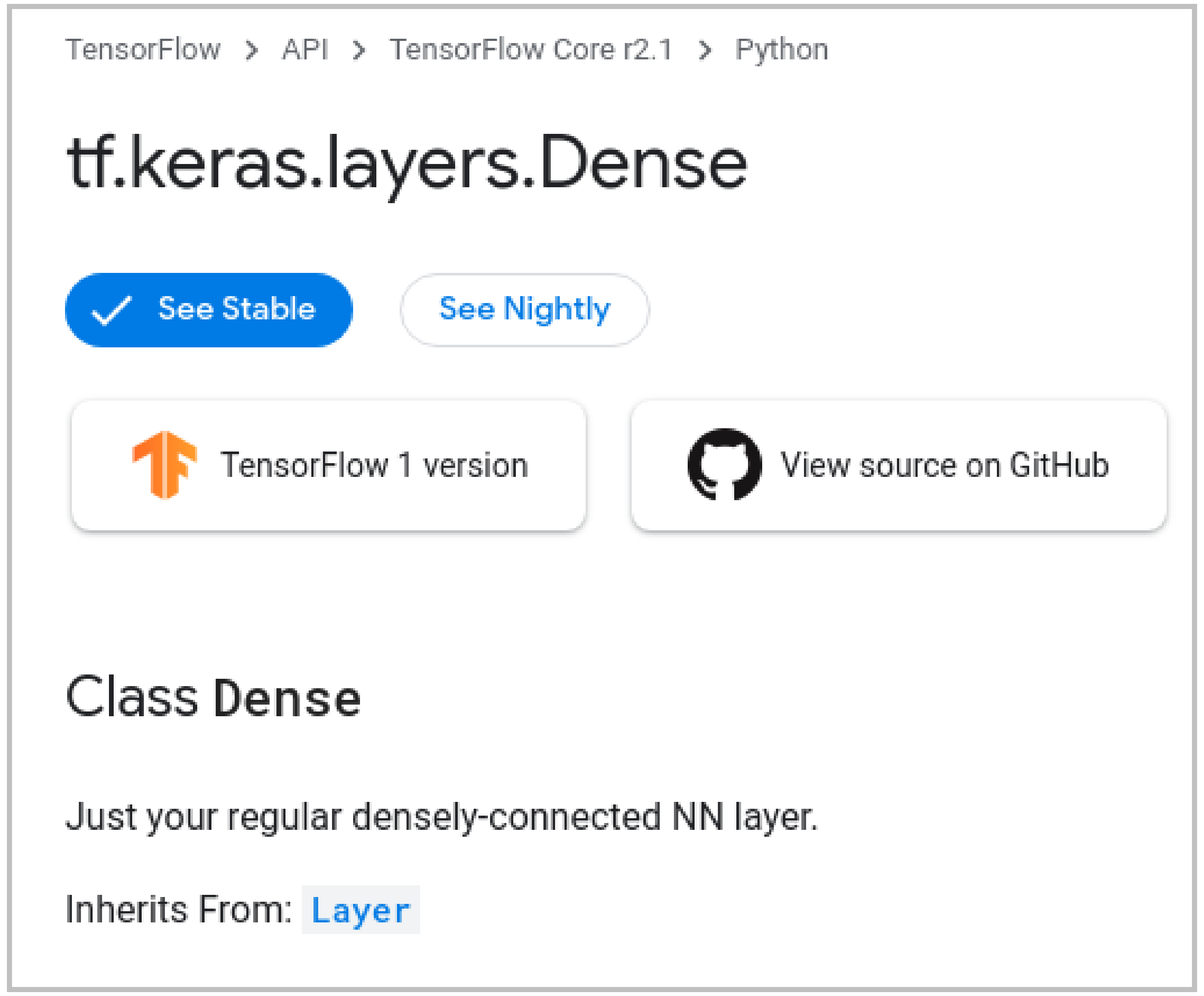
Arguments:

- `layers`: list of layers to add to the model.

Keras Sequential model with Dense layers

- Each `Dense` layer in the list will have the following form

```
Dense(units,                  #<- num neurons
      activation = None,      #<- activation
function
      ...
      )
```



TensorFlow > API > TensorFlow Core r2.1 > Python

tf.keras.layers.Dense

Class **Dense**

Just your regular densely-connected NN layer.

Inherits From: `Layer`

- Package documentation is available on the TensorFlow website and can be accessed using ***this link***

Knowledge check

Link: ***https://forms.gle/APsU9ZVNzG9ji53y5***

Module completion checklist

Objective	Complete
Introduce TensorFlow and Keras	✔
Describe how to define a neural network model using Tensorflow	✔

Congratulations on completing this module!