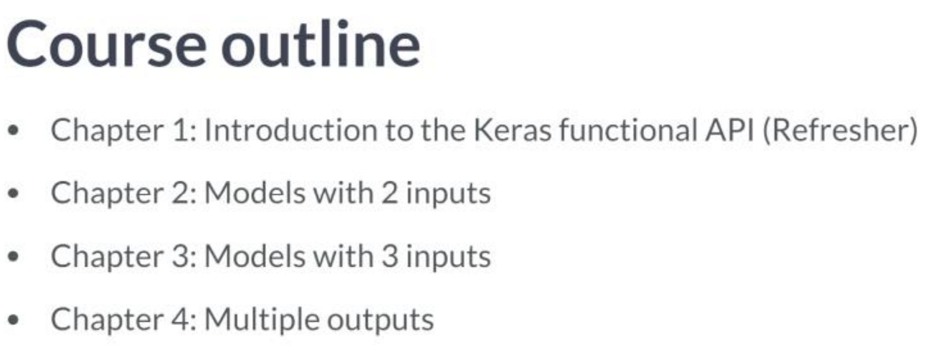
Advance Deep learning with keras

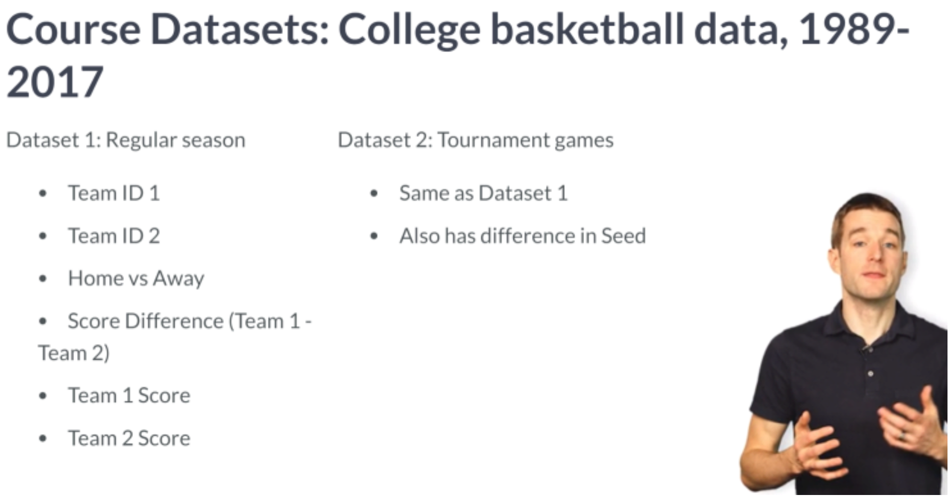
Thursday, 20 August 2020 2:51 PM

The Keras Functional API



Course outline

- Chapter 1: Introduction to the Keras functional API (Refresher)
- Chapter 2: Models with 2 inputs
- Chapter 3: Models with 3 inputs
- Chapter 4: Multiple outputs



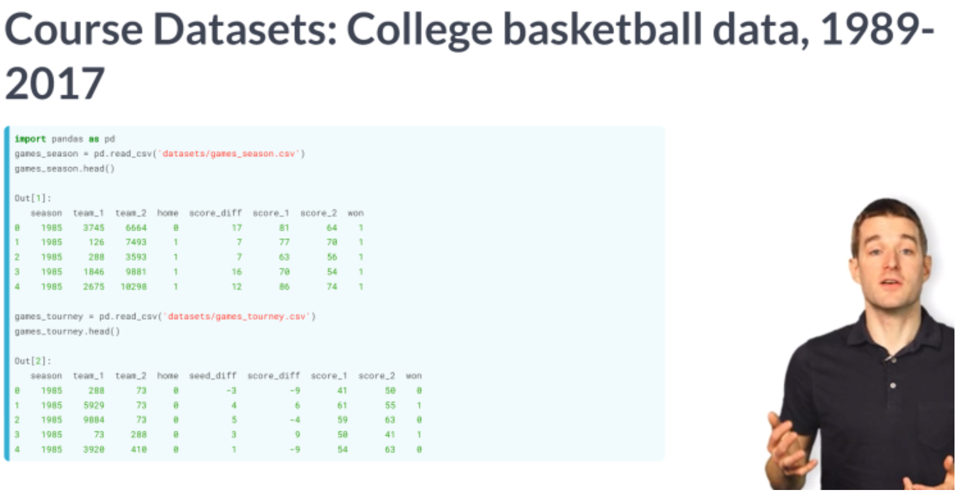
Course Datasets: College basketball data, 1989-2017

Dataset 1: Regular season

- Team ID 1
- Team ID 2
- Home vs Away
- Score Difference (Team 1 - Team 2)
- Team 1 Score
- Team 2 Score

Dataset 2: Tournament games

- Same as Dataset 1
- Also has difference in Seed



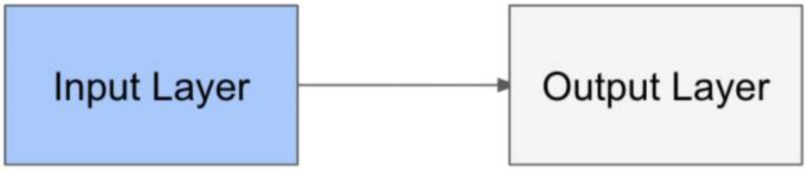
Course Datasets: College basketball data, 1989-2017

```
import pandas as pd
games_season = pd.read_csv('datasets/games_season.csv')
games_season.head()

Out[1]:
   season  team_1  team_2  home  score_diff  score_1  score_2  won
0    1985    3745    6664     0          17       81       64    1
1    1985     126    7493     1           7       77       70    1
2    1985     288    3593     1           7       63       56    1
3    1985    1846    9881     1          16       70       54    1
4    1985    2675   10298     1          12       86       74    1

games_tourney = pd.read_csv('datasets/games_tourney.csv')
games_tourney.head()

Out[2]:
   season  team_1  team_2  home  seed_diff  score_diff  score_1  score_2  won
0    1985     288      73     0         -3          -9       41       50    0
1    1985    5929      73     0          4           6       61       55    1
2    1985    9884      73     0          5          -4       59       63    0
3    1985      73     288     0          3           9       50       41    1
4    1985    3920     410     0          1          -9       54       63    0
```

Inputs and outputs

Two fundamental parts:

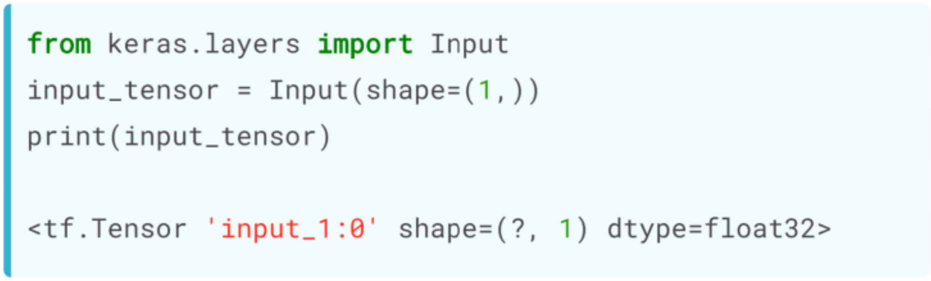
- Input layer
- Output layer



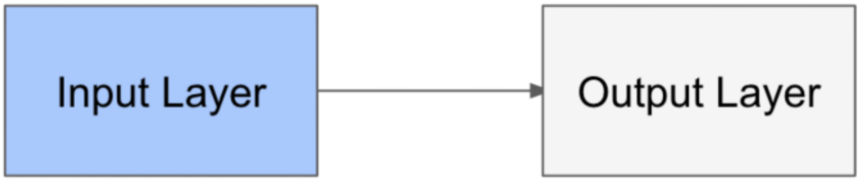
Input Layer

Output Layer

Inputs



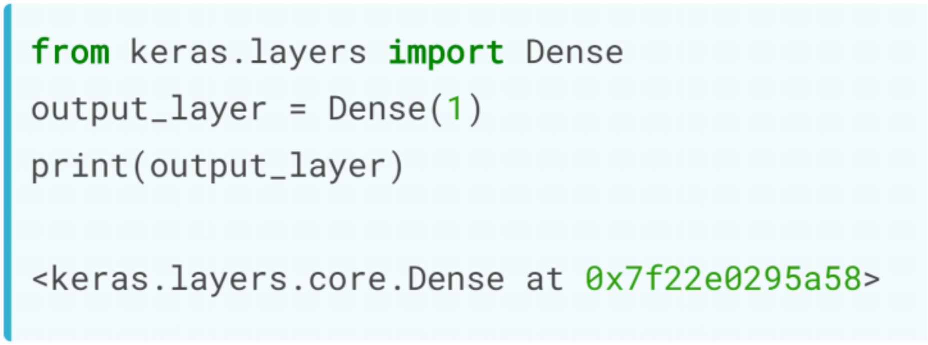
```
from keras.layers import Input
input_tensor = Input(shape=(1,))
print(input_tensor)

<tf.Tensor 'input_1:0' shape=(?, 1) dtype=float32>
```



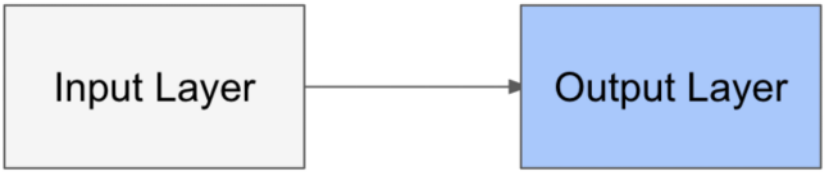
Input Layer

Output Layer

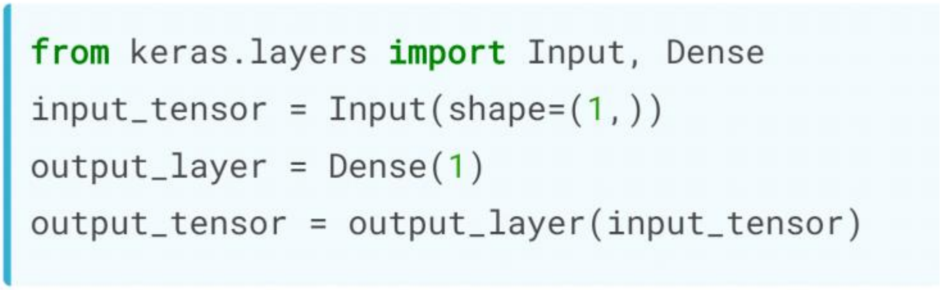
Outputs



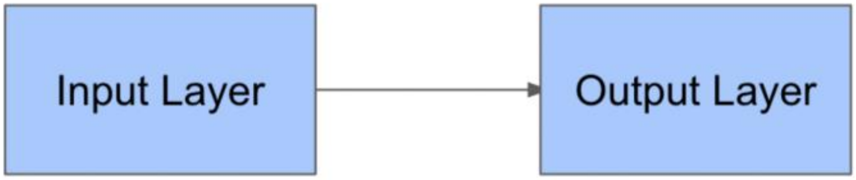
```
from keras.layers import Dense
output_layer = Dense(1)
print(output_layer)

<keras.layers.core.Dense at 0x7f22e0295a58>
```



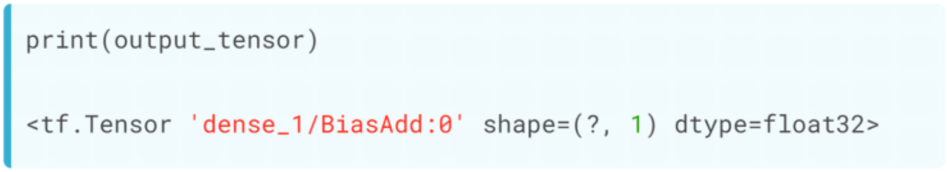
Connecting inputs to outputs



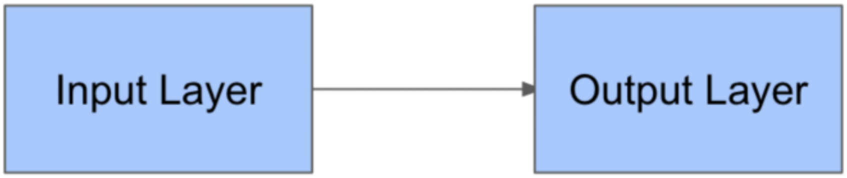
```
from keras.layers import Input, Dense
input_tensor = Input(shape=(1,))
output_layer = Dense(1)
output_tensor = output_layer(input_tensor)
```



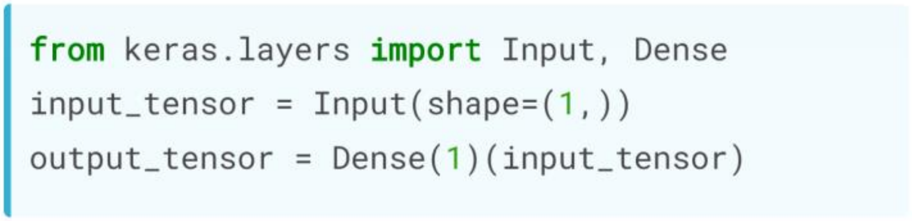
Connecting inputs to outputs



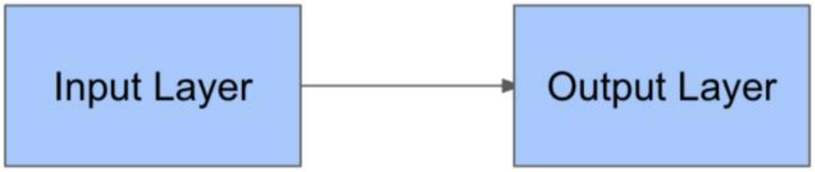
```
print(output_tensor)

<tf.Tensor 'dense_1/BiasAdd:0' shape=(?, 1) dtype=float32>
```



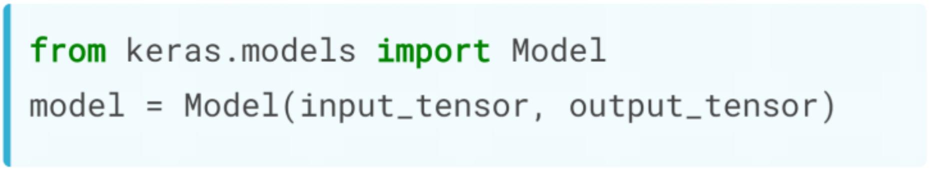
Keras models



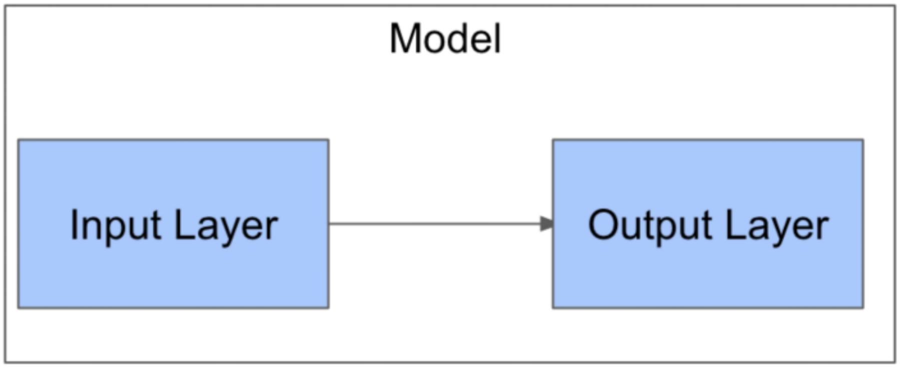
```
from keras.layers import Input, Dense
input_tensor = Input(shape=(1,))
output_tensor = Dense(1)(input_tensor)
```



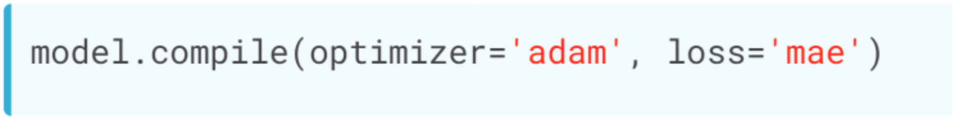
Keras models



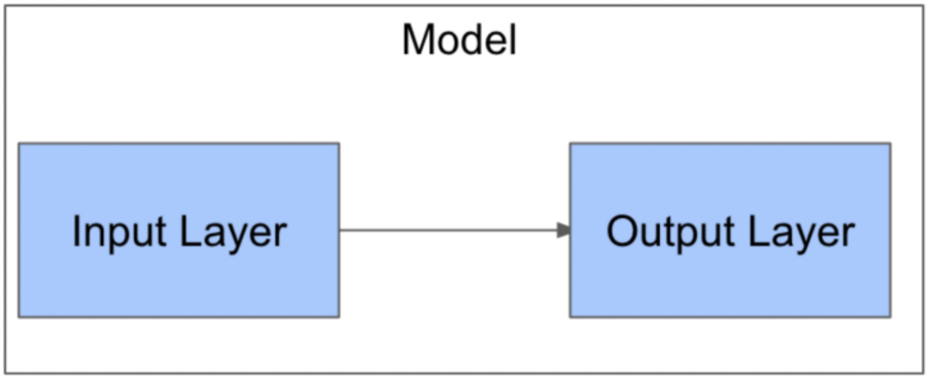
```
from keras.models import Model
model = Model(input_tensor, output_tensor)
```



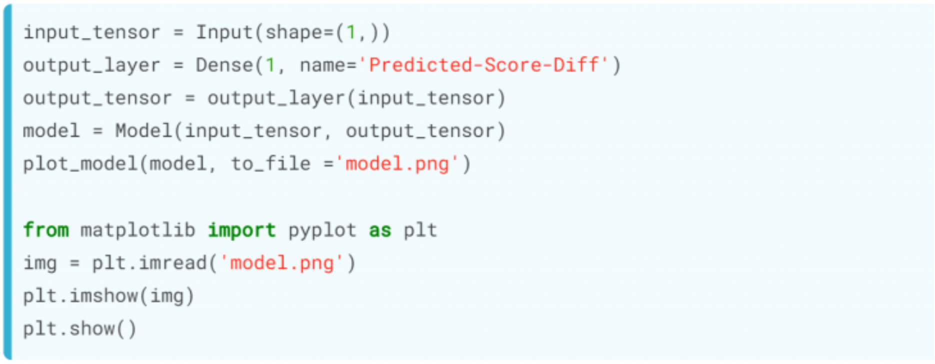
Compile a model



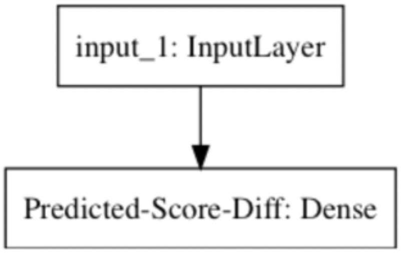
```
model.compile(optimizer='adam', loss='mae')
```

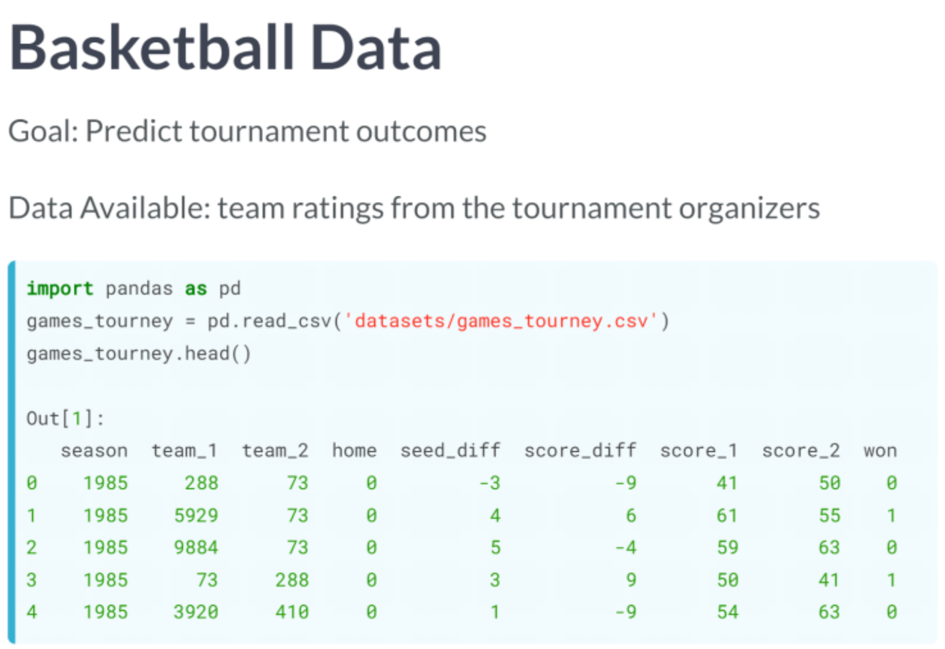


Plot model using keras



```
input_tensor = Input(shape=(1,))
output_layer = Dense(1, name='Predicted-Score-Diff')
output_tensor = output_layer(input_tensor)
model = Model(input_tensor, output_tensor)
plot_model(model, to_file ='model.png')

from matplotlib import pyplot as plt
img = plt.imread('model.png')
plt.imshow(img)
plt.show()
```



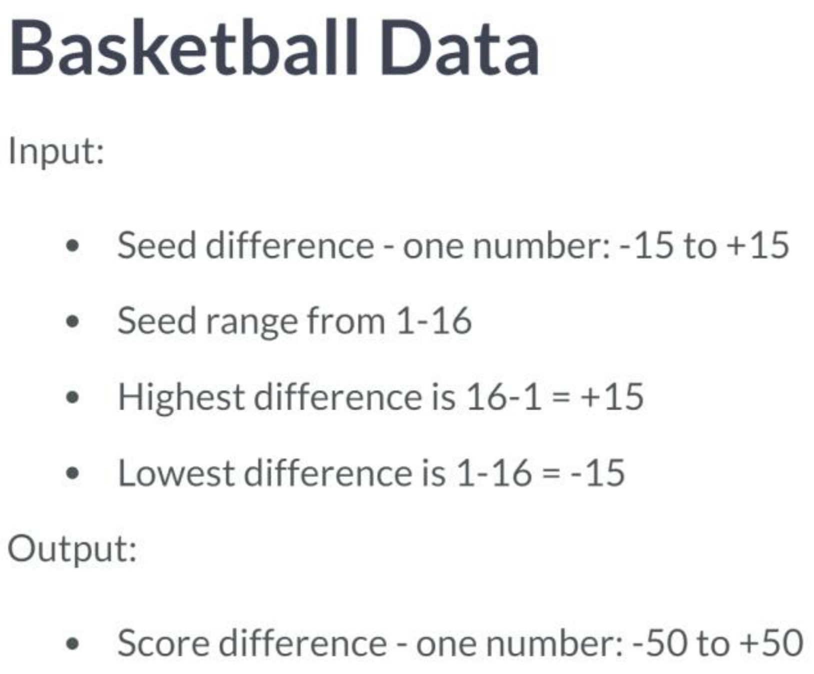


Basketball Data

Goal: Predict tournament outcomes

Data Available: team ratings from the tournament organizers

```
import pandas as pd
games_tourney = pd.read_csv('datasets/games_tourney.csv')
games_tourney.head()

Out[1]:
   season  team_1  team_2  home  seed_diff  score_diff  score_1  score_2  won
0    1985     288      73     0         -3          -9       41       50    0
1    1985    5929      73     0          4           6       61       55    1
2    1985    9884      73     0          5          -4       59       63    0
3    1985      73     288     0          3           9       50       41    1
4    1985    3920     410     0          1          -9       54       63    0
```



Basketball Data

Input:

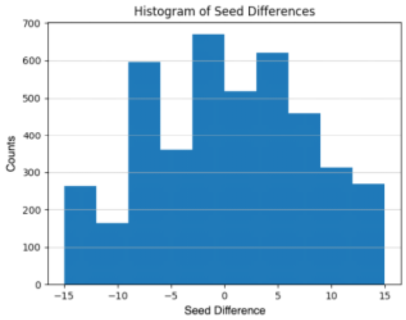
- Seed difference - one number: -15 to +15
- Seed range from 1-16
- Highest difference is 16-1 = +15
- Lowest difference is 1-16 = -15

Output:

- Score difference - one number: -50 to +50

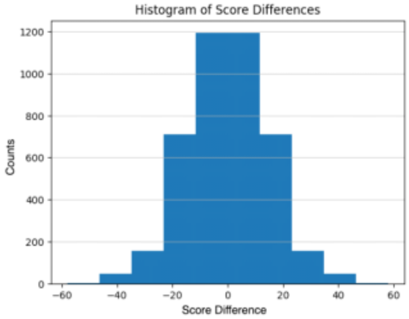
Basketball Data

- Seed difference: 15
  - Team 1: 16
  - Team 2: 1
- Seed difference: -15
  - Team 1: 1
  - Team 2: 16



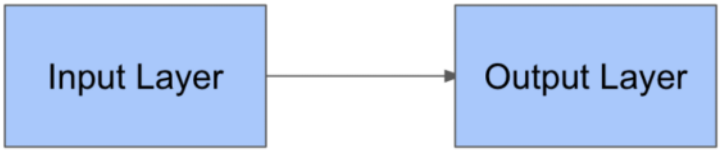
Basketball Data

- Score difference: -9
  - Team 1: 41
  - Team 2: 50
- Score difference: 6
  - Team 1: 61
  - Team 2: 55



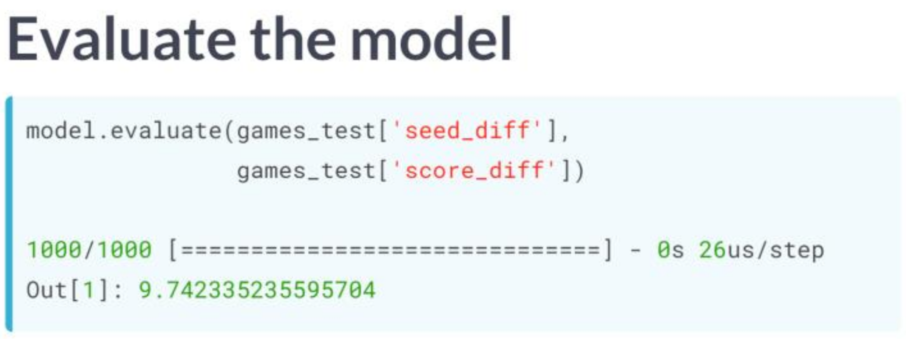
Fit the model

```
from pandas import read_csv
games = read_csv('datasets/games_tourney.csv')
model.fit(games['seed_diff'],
          games['score_diff'],
          batch_size=64,
          validation_split=.20,
          verbose=True)
```



Input Layer

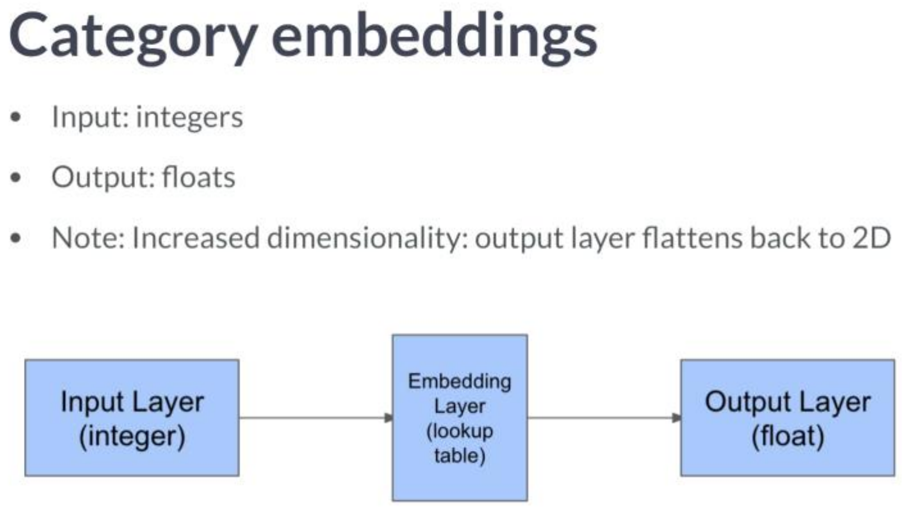
Output Layer



Evaluate the model

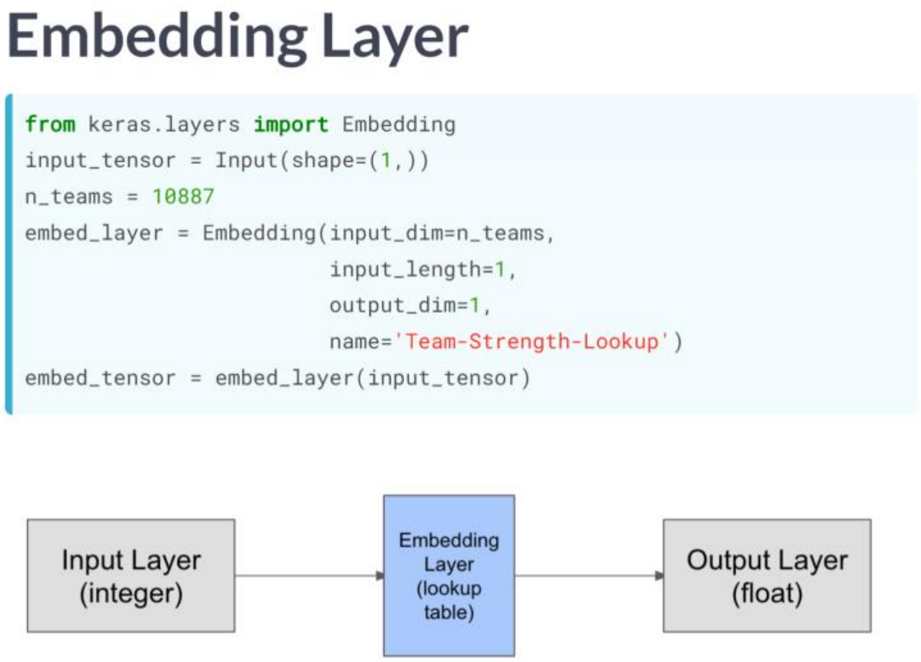
```
model.evaluate(games_test['seed_diff'],
               games_test['score_diff'])

1000/1000 [==============================] - 0s 26us/step
Out[1]: 9.742335235595704
```

Two Input Networks Using Categorical Embeddings, Shared Layers, and Merge Layers



Category embeddings

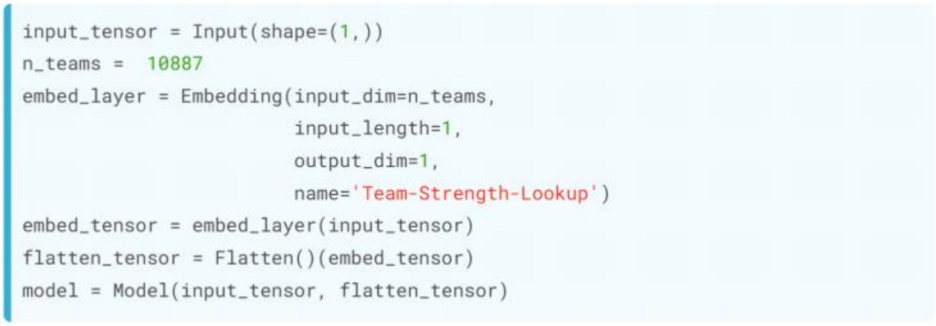
- Input: integers
- Output: floats
- Note: Increased dimensionality: output layer flattens back to 2D



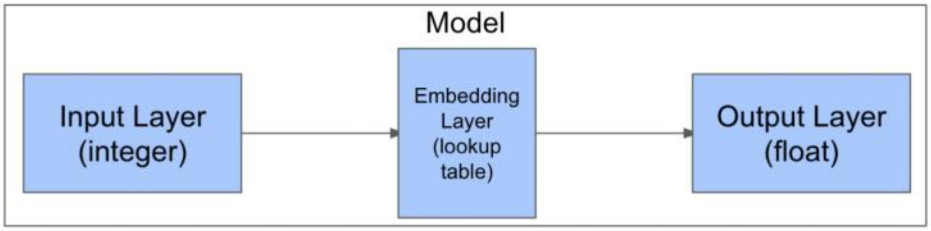
Embedding Layer

```
from keras.layers import Embedding
input_tensor = Input(shape=(1,))
n_teams = 10887
embed_layer = Embedding(input_dim=n_teams,
                        input_length=1,
                        output_dim=1,
                        name='Team-Strength-Lookup')
embed_tensor = embed_layer(input_tensor)
```

Put it all together

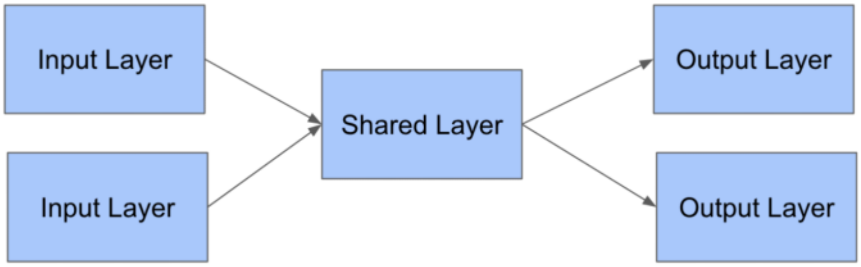


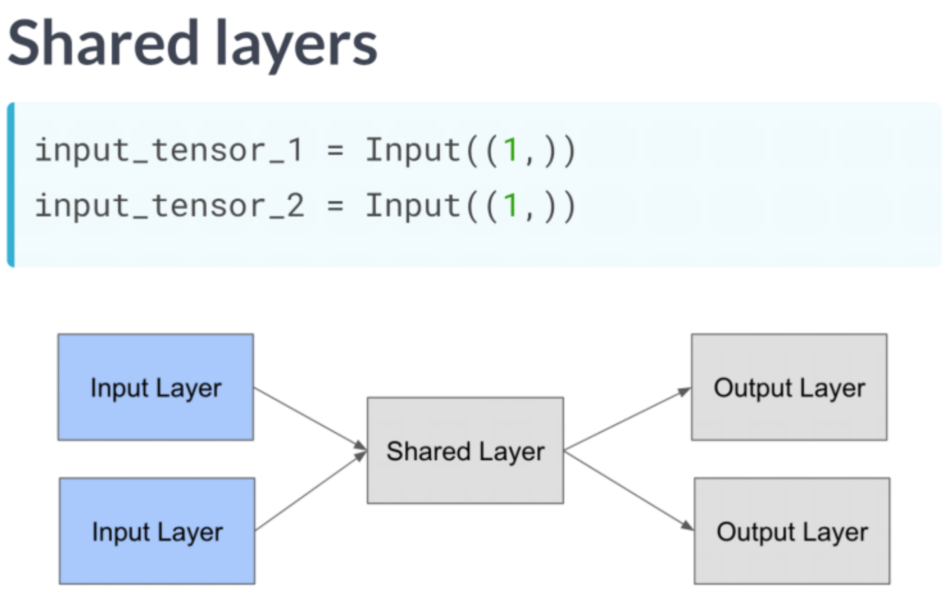
```
input_tensor = Input(shape=(1,))
n_teams =  10887
embed_layer = Embedding(input_dim=n_teams,
                        input_length=1,
                        output_dim=1,
                        name='Team-Strength-Lookup')
embed_tensor = embed_layer(input_tensor)
flatten_tensor = Flatten()(embed_tensor)
model = Model(input_tensor, flatten_tensor)
```



Shared layers

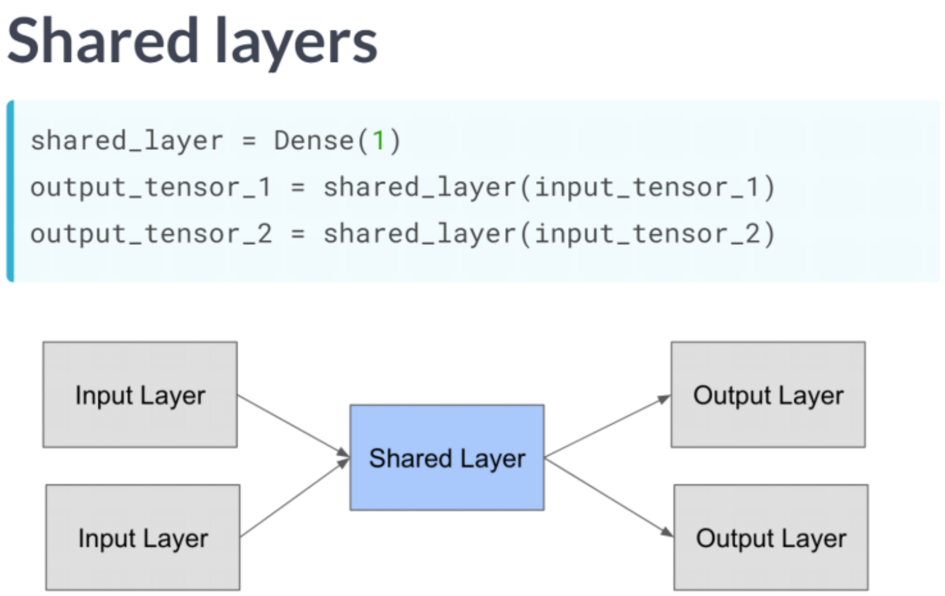
- Require the functional API
- Very flexible





Shared layers

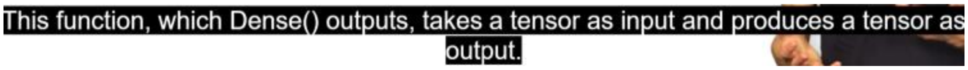
```
input_tensor_1 = Input((1,))
input_tensor_2 = Input((1,))
```



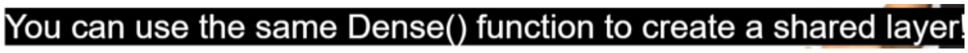
Shared layers

```
shared_layer = Dense(1)
output_tensor_1 = shared_layer(input_tensor_1)
output_tensor_2 = shared_layer(input_tensor_2)
```

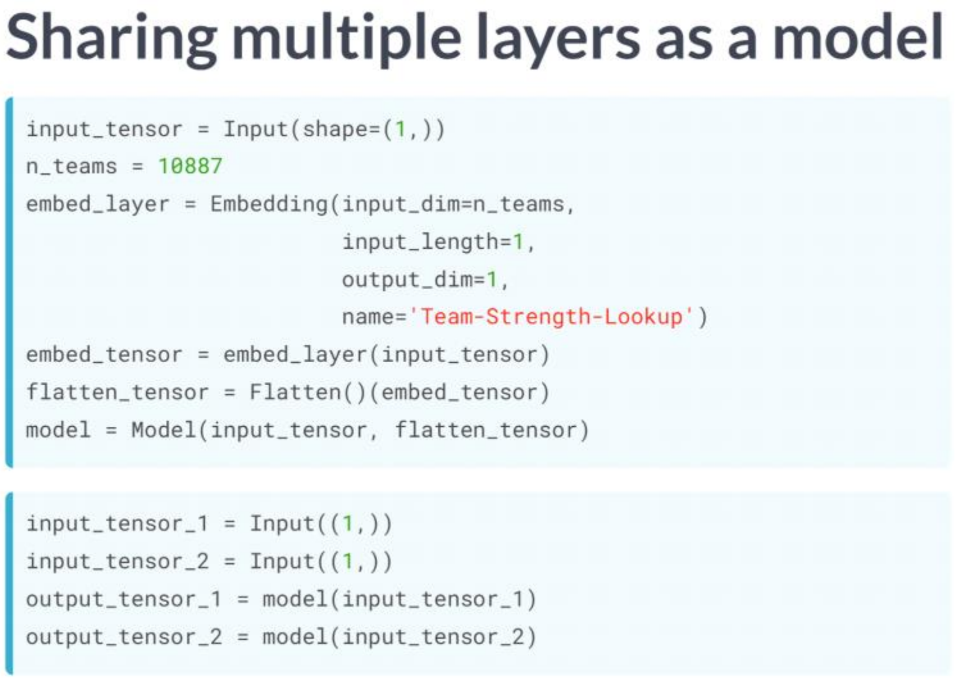
the Dense() function returns a function as its output



This function, which Dense() outputs, takes a tensor as input and produces a tensor as output.



You can use the same Dense() function to create a shared layer!

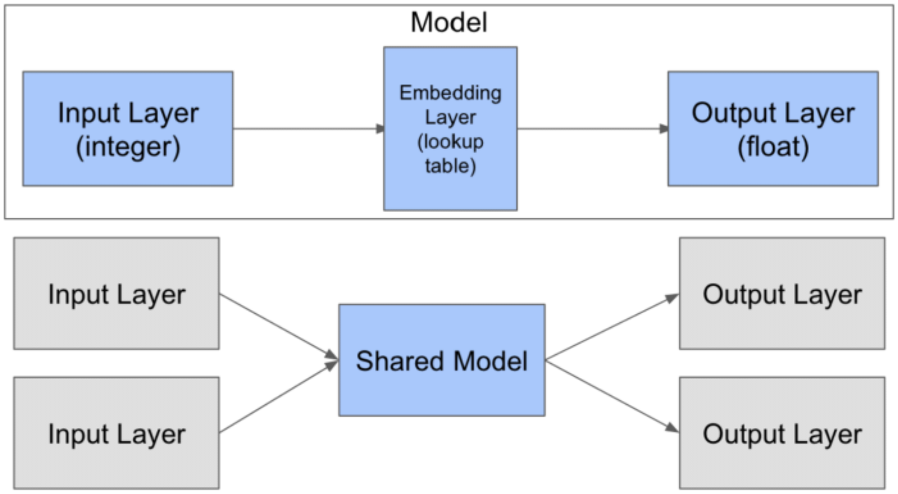


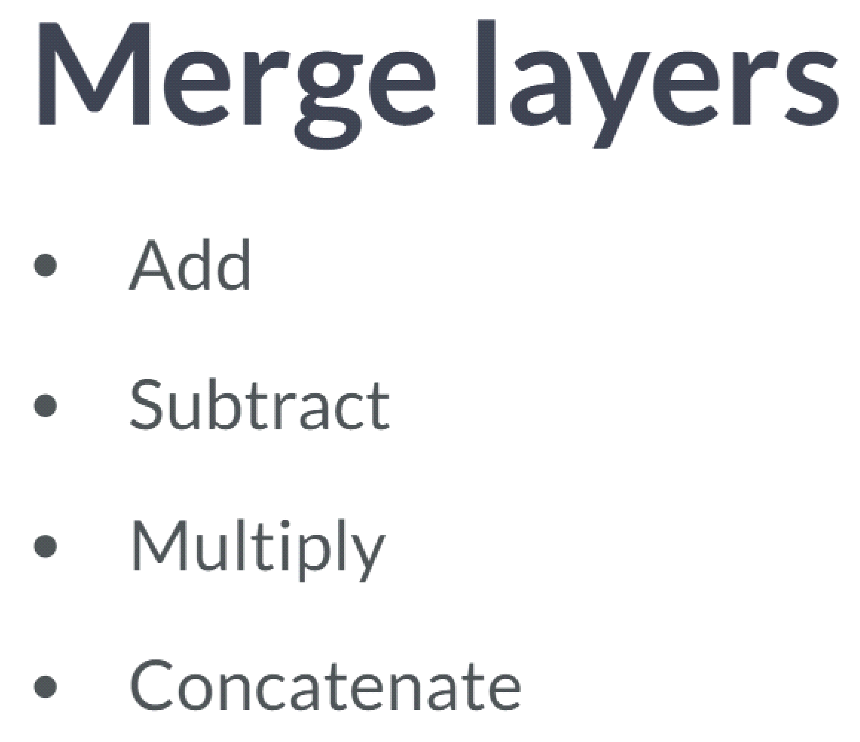
Sharing multiple layers as a model

```
input_tensor = Input(shape=(1,))
n_teams = 10887
embed_layer = Embedding(input_dim=n_teams,
                        input_length=1,
                        output_dim=1,
                        name='Team-Strength-Lookup')
embed_tensor = embed_layer(input_tensor)
flatten_tensor = Flatten()(embed_tensor)
model = Model(input_tensor, flatten_tensor)
```

```
input_tensor_1 = Input((1,))
input_tensor_2 = Input((1,))
output_tensor_1 = model(input_tensor_1)
output_tensor_2 = model(input_tensor_2)
```

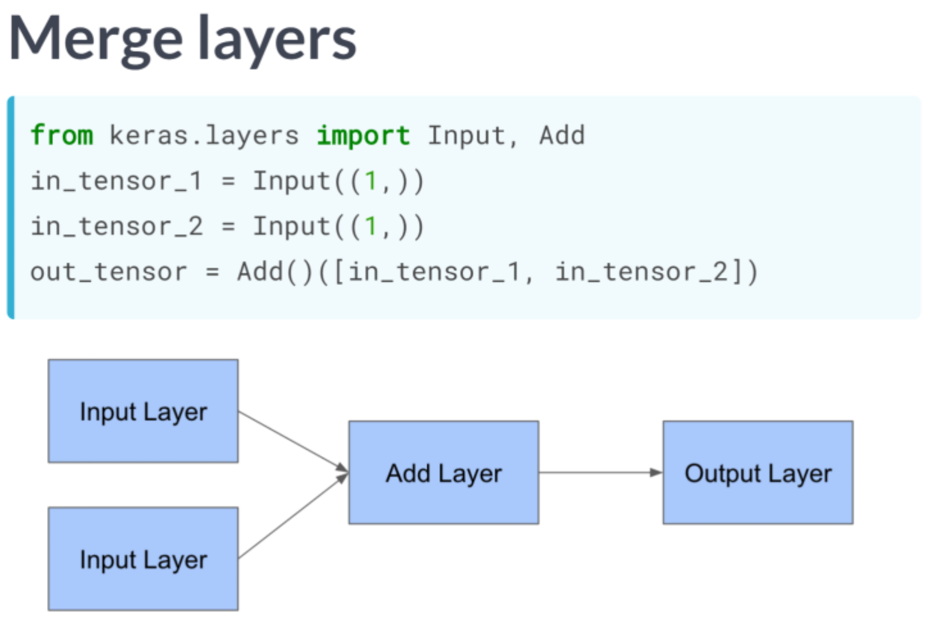
Sharing multiple layers as a model





Merge layers

- Add
- Subtract
- Multiply
- Concatenate

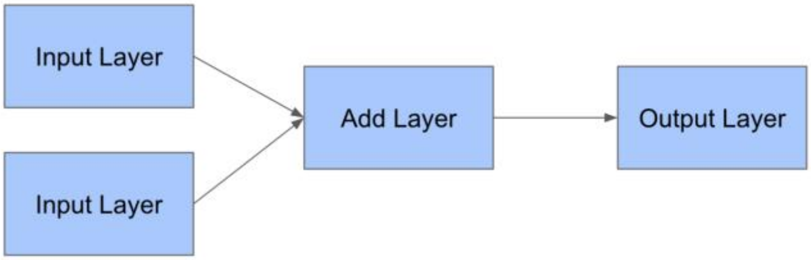


Merge layers

```
from keras.layers import Input, Add
in_tensor_1 = Input((1,))
in_tensor_2 = Input((1,))
out_tensor = Add()([in_tensor_1, in_tensor_2])
```

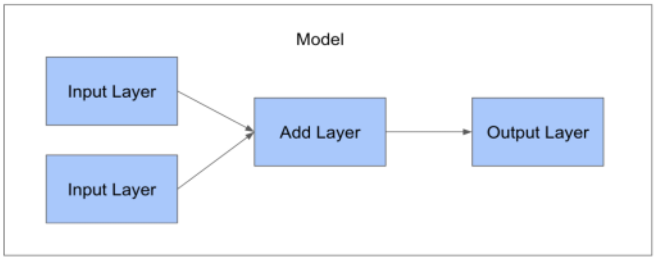
Merge layers

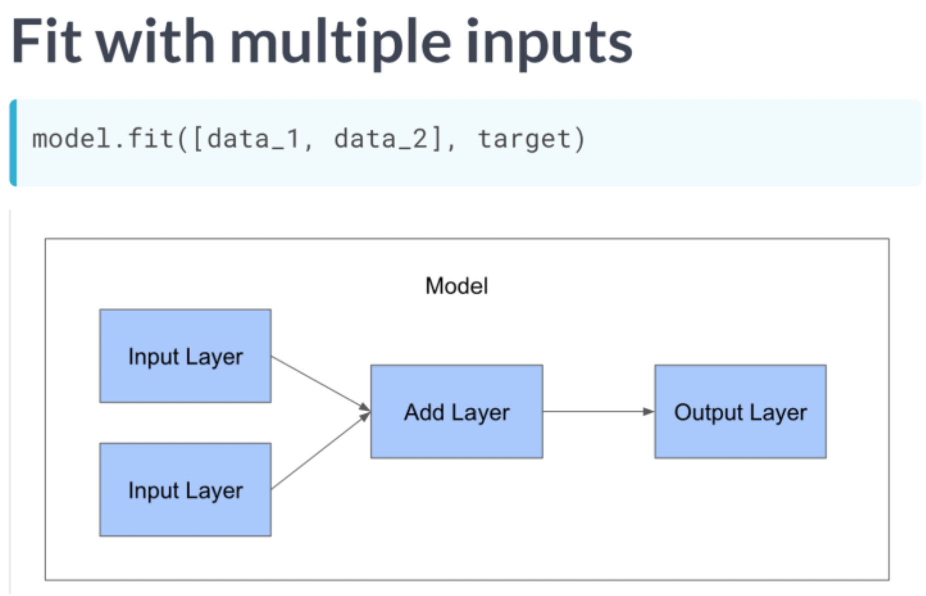
```
in_tensor_3 = Input((1,))
out_tensor = Add()([in_tensor_1, in_tensor_2, in_tensor_3])
```



Create the model

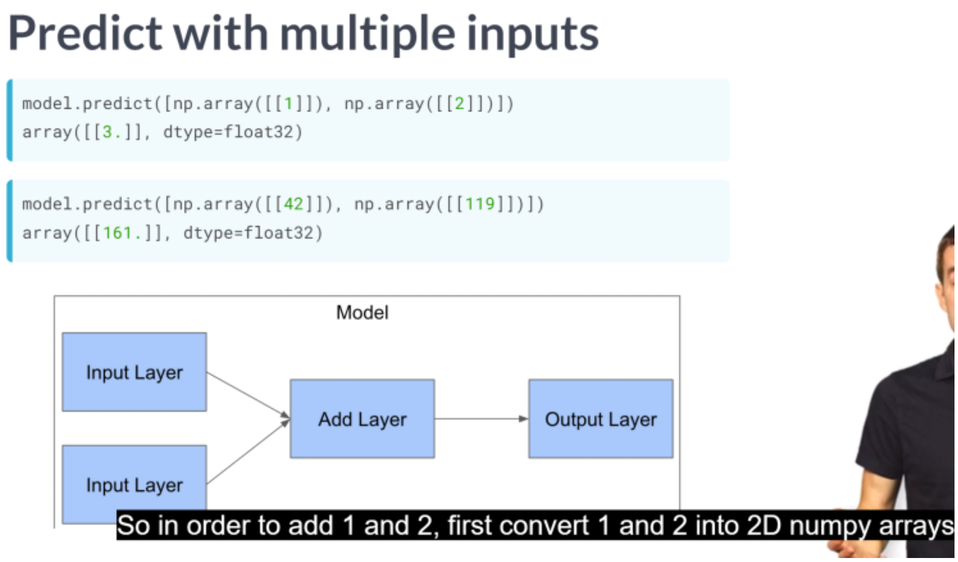
```
from keras.models import Model
model = Model([in_tensor_1, in_tensor_2], out_tensor)
```





Fit with multiple inputs

```
model.fit([data_1, data_2], target)
```

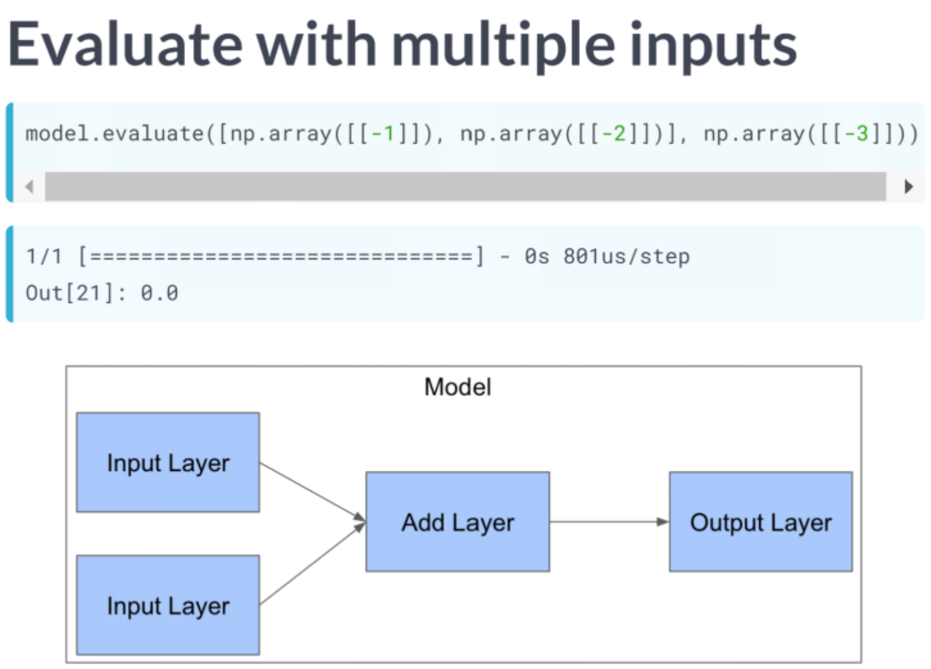


Predict with multiple inputs

```
model.predict([np.array([[1]]), np.array([[2]])])
array([[3.]], dtype=float32)
```

```
model.predict([np.array([[42]]), np.array([[119]])])
array([[161.]], dtype=float32)
```

So in order to add 1 and 2, first convert 1 and 2 into 2D numpy arrays

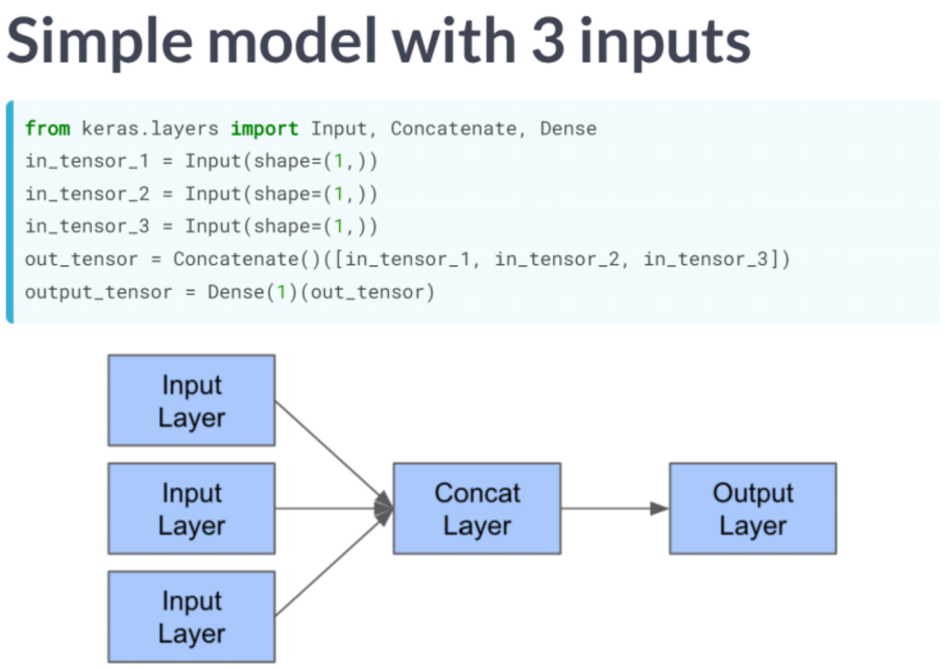


Evaluate with multiple inputs

```
model.evaluate([np.array([[-1]]), np.array([[-2]])], np.array([[-3]]))
```

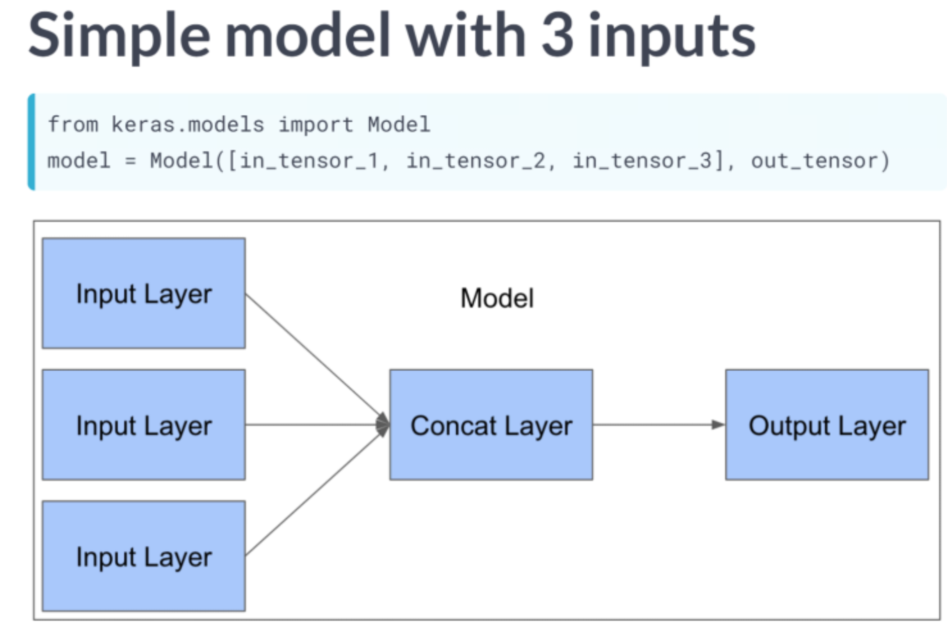
```
1/1 [==============================] - 0s 801us/step
Out[21]: 0.0
```

Multiple Inputs: 3 Inputs (and Beyond!)



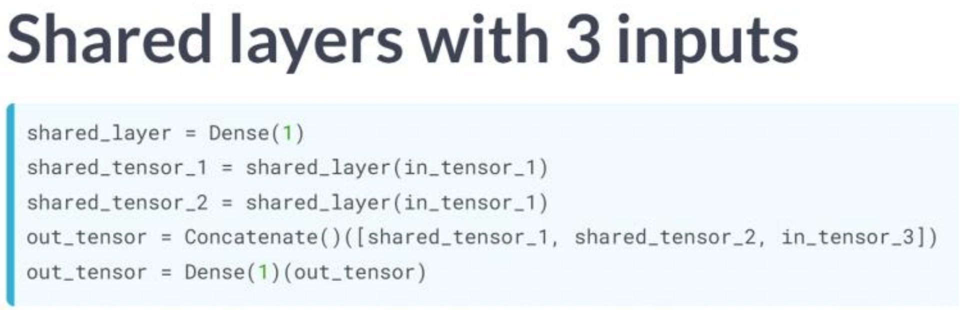
Simple model with 3 inputs

```
from keras.layers import Input, Concatenate, Dense
in_tensor_1 = Input(shape=(1,))
in_tensor_2 = Input(shape=(1,))
in_tensor_3 = Input(shape=(1,))
out_tensor = Concatenate()([in_tensor_1, in_tensor_2, in_tensor_3])
output_tensor = Dense(1)(out_tensor)
```



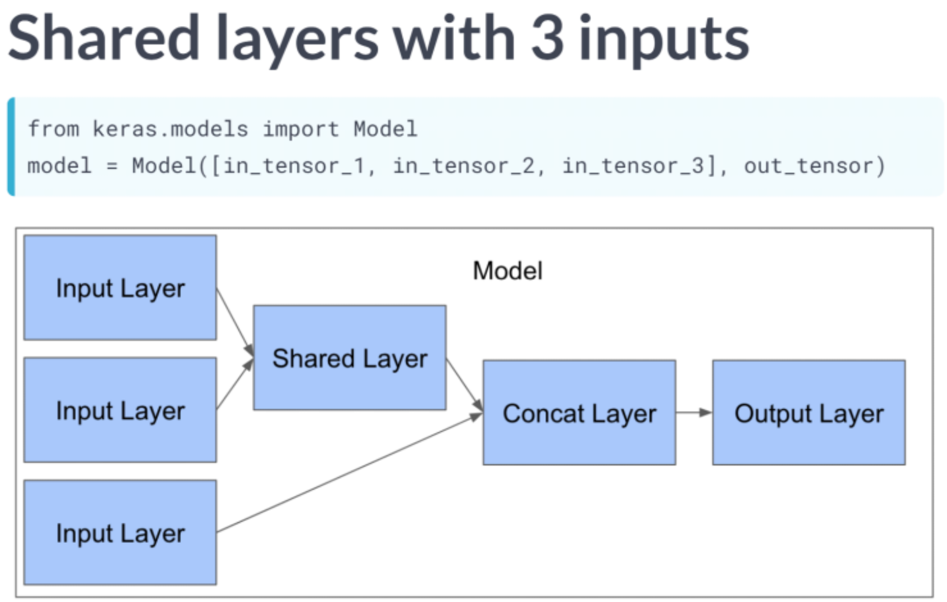
Simple model with 3 inputs

```
from keras.models import Model
model = Model([in_tensor_1, in_tensor_2, in_tensor_3], out_tensor)
```



Shared layers with 3 inputs

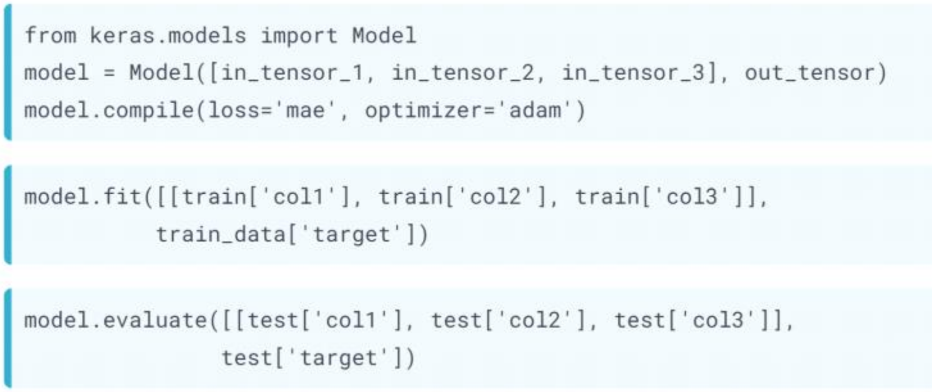
```
shared_layer = Dense(1)
shared_tensor_1 = shared_layer(in_tensor_1)
shared_tensor_2 = shared_layer(in_tensor_1)
out_tensor = Concatenate()([shared_tensor_1, shared_tensor_2, in_tensor_3])
out_tensor = Dense(1)(out_tensor)
```



Shared layers with 3 inputs

```
from keras.models import Model
model = Model([in_tensor_1, in_tensor_2, in_tensor_3], out_tensor)
```

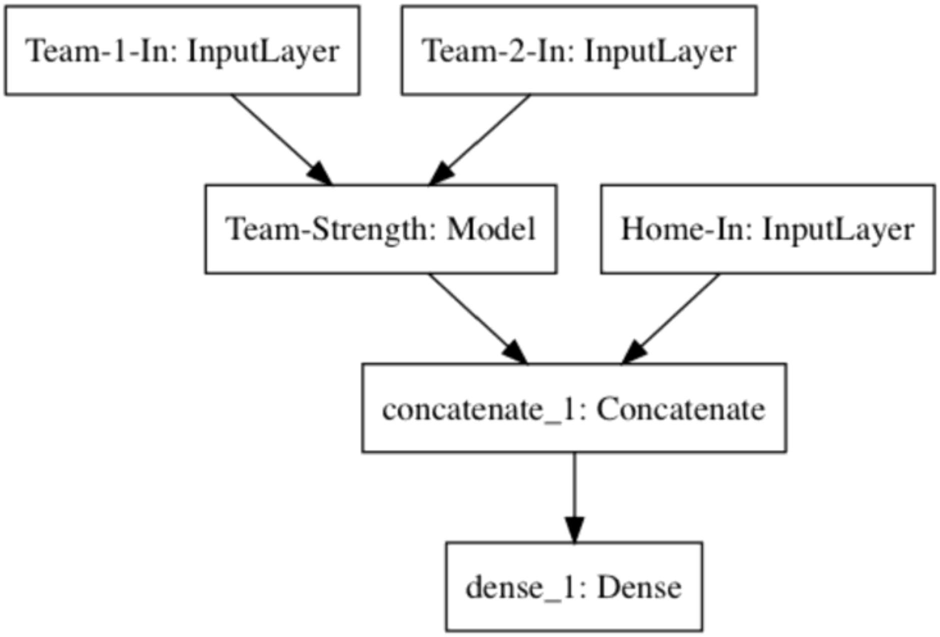
Fitting a 3 input model



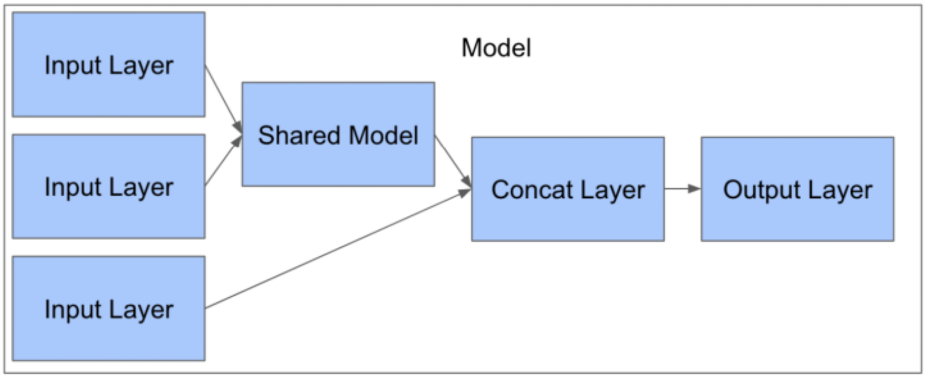
```
from keras.models import Model
model = Model([in_tensor_1, in_tensor_2, in_tensor_3], out_tensor)
model.compile(loss='mae', optimizer='adam')
```

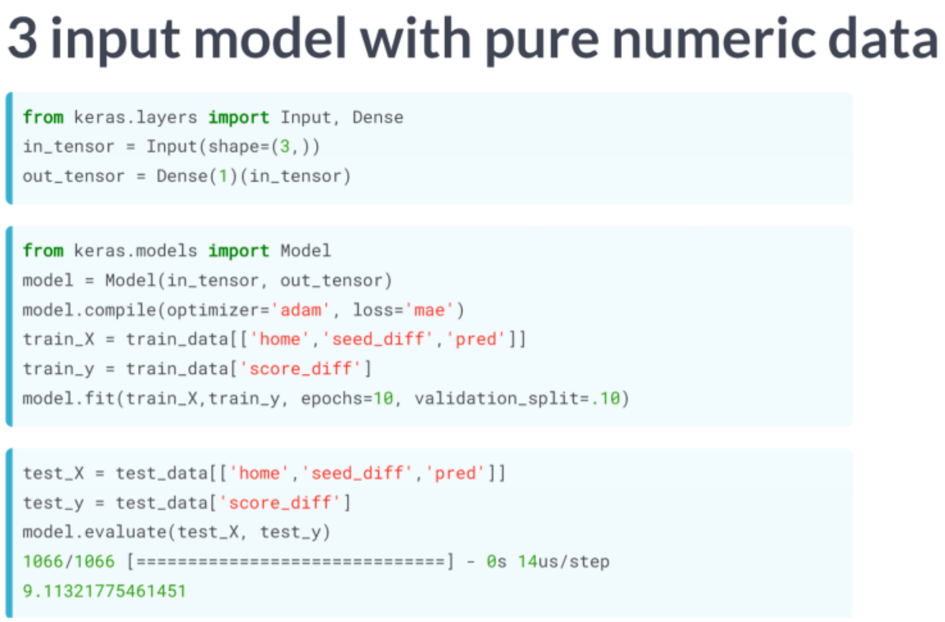
```
model.fit([[train['col1'], train['col2'], train['col3']],
          train_data['target'])
```

```
model.evaluate([[test['col1'], test['col2'], test['col3']],
               test['target'])
```



Understanding a model plot!





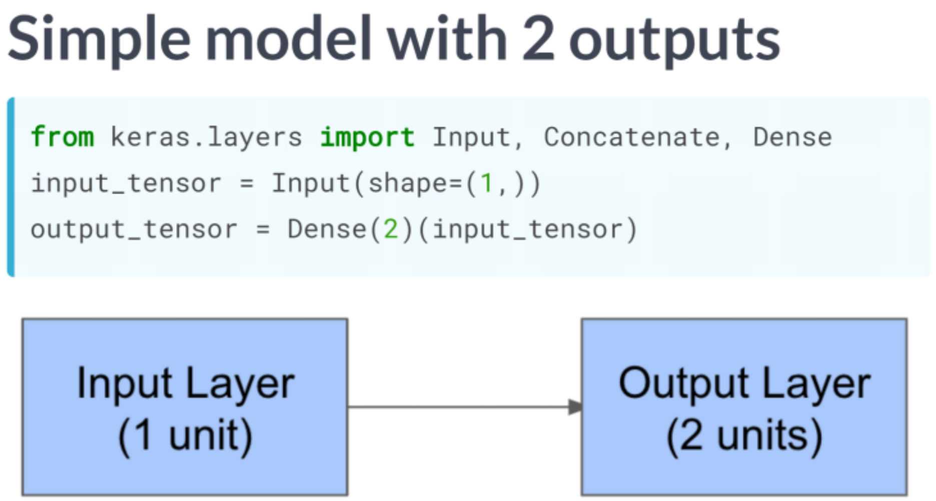
3 input model with pure numeric data

```
from keras.layers import Input, Dense
in_tensor = Input(shape=(3,))
out_tensor = Dense(1)(in_tensor)
```

```
from keras.models import Model
model = Model(in_tensor, out_tensor)
model.compile(optimizer='adam', loss='mae')
train_X = train_data[['home','seed_diff','pred']]
train_y = train_data['score_diff']
model.fit(train_X,train_y, epochs=10, validation_split=.10)
```

```
test_X = test_data[['home','seed_diff','pred']]
test_y = test_data['score_diff']
model.evaluate(test_X, test_y)
1066/1066 [==============================] - 0s 14us/step
9.113217754614 51
```

Multiple Outputs

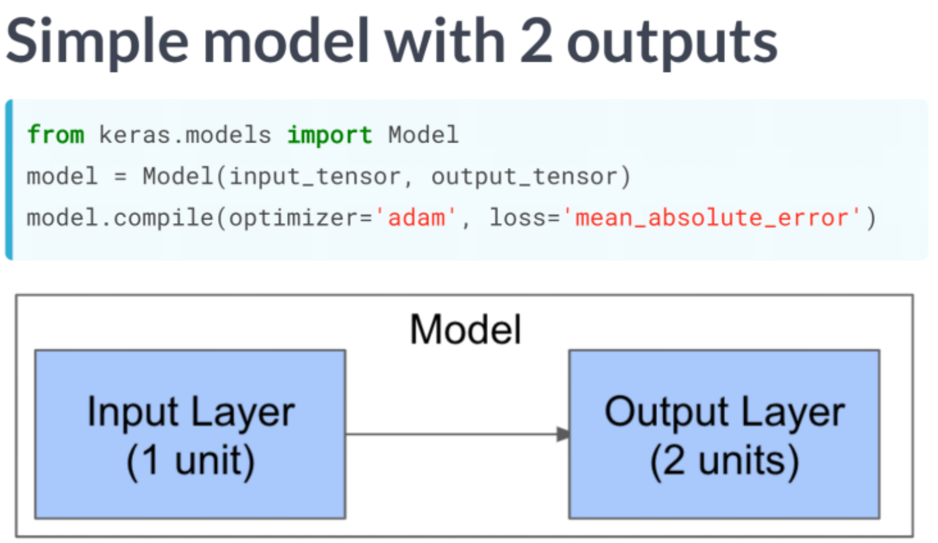


Simple model with 2 outputs

```
from keras.layers import Input, Concatenate, Dense
input_tensor = Input(shape=(1,))
output_tensor = Dense(2)(input_tensor)
```

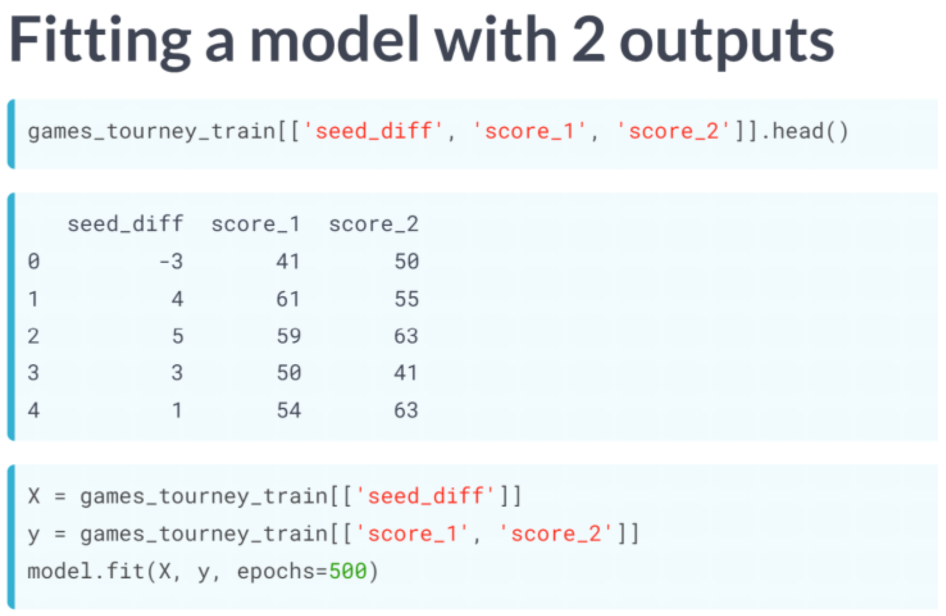
Input Layer
(1 unit)

Output Layer
(2 units)



Simple model with 2 outputs

```
from keras.models import Model
model = Model(input_tensor, output_tensor)
model.compile(optimizer='adam', loss='mean_absolute_error')
```

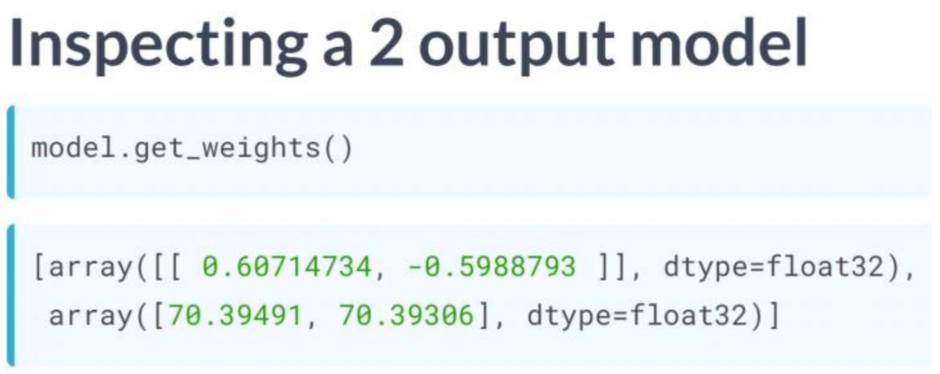


Fitting a model with 2 outputs

```
games_tourney_train[['seed_diff', 'score_1', 'score_2']].head()
```

```
   seed_diff  score_1  score_2
0         -3       41       50
1          4       61       55
2          5       59       63
3          3       50       41
4          1       54       63
```

```
X = games_tourney_train[['seed_diff']]
y = games_tourney_train[['score_1', 'score_2']]
model.fit(X, y, epochs=500)
```

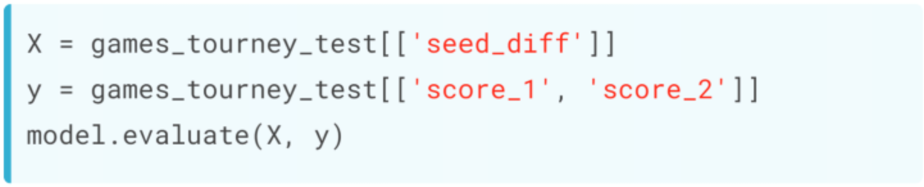


Inspecting a 2 output model

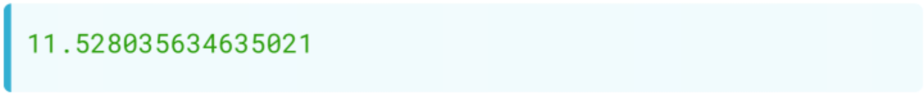
```
model.get_weights()
```

```
[array([[ 0.60714734, -0.5988793 ]], dtype=float32),
 array([70.39491, 70.39306], dtype=float32)]
```

Evaluating a model with 2 outputs

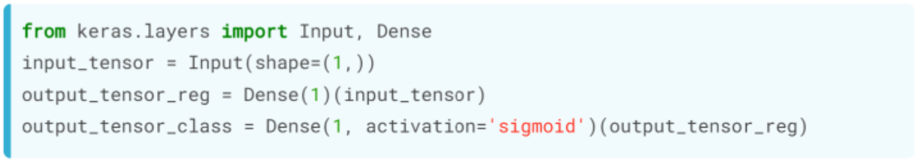


```
X = games_tourney_test[['seed_diff']]
y = games_tourney_test[['score_1', 'score_2']]
model.evaluate(X, y)
```

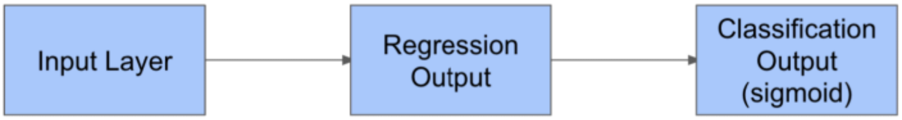


```
11.528035634635021
```

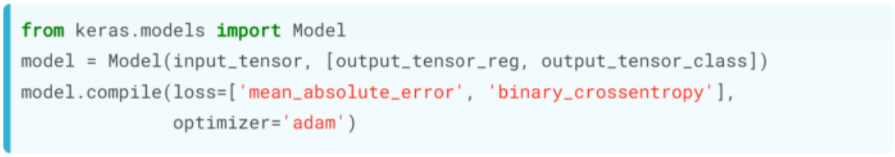
Build a simple regressor/classifier



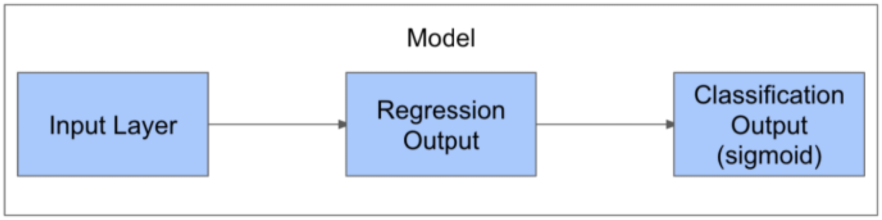
```
from keras.layers import Input, Dense
input_tensor = Input(shape=(1,))
output_tensor_reg = Dense(1)(input_tensor)
output_tensor_class = Dense(1, activation='sigmoid')(output_tensor_reg)
```



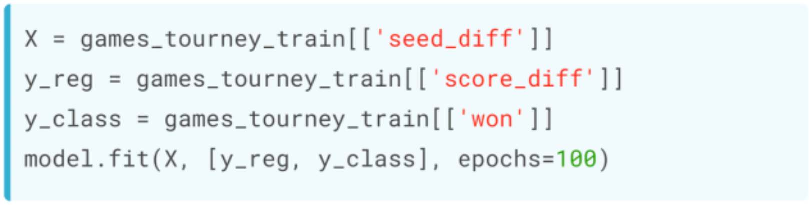
Make a regressor/classifier model



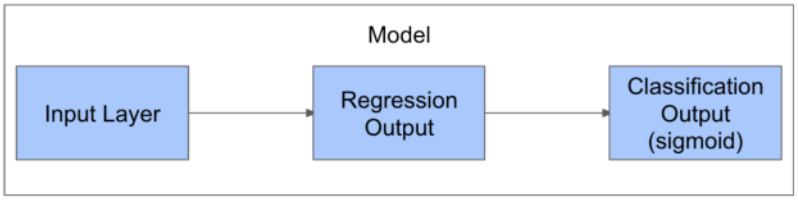
```
from keras.models import Model
model = Model(input_tensor, [output_tensor_reg, output_tensor_class])
model.compile(loss=['mean_absolute_error', 'binary_crossentropy'],
              optimizer='adam')
```



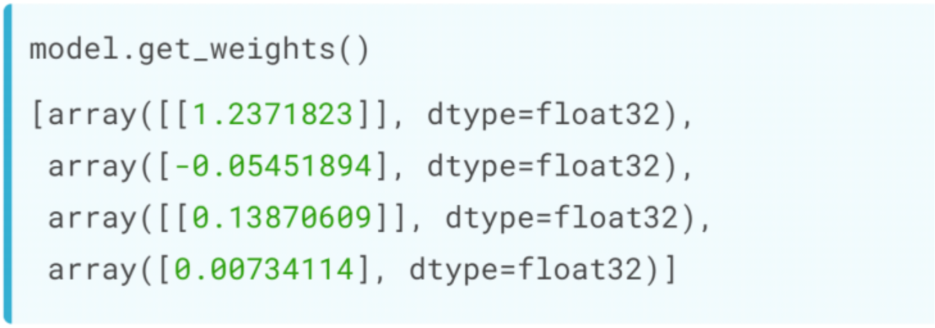
Fit the combination classifier/regressor



```
X = games_tourney_train[['seed_diff']]
y_reg = games_tourney_train[['score_diff']]
y_class = games_tourney_train[['won']]
model.fit(X, [y_reg, y_class], epochs=100)
```

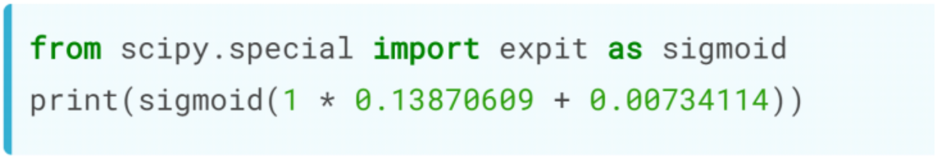


Look at the model's weights

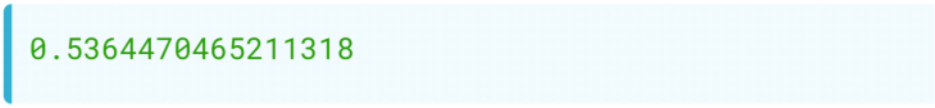


```
model.get_weights()
```

```
[array([[1.2371823]], dtype=float32),
 array([-0.05451894], dtype=float32),
 array([[0.13870609]], dtype=float32),
 array([0.00734114], dtype=float32)]
```

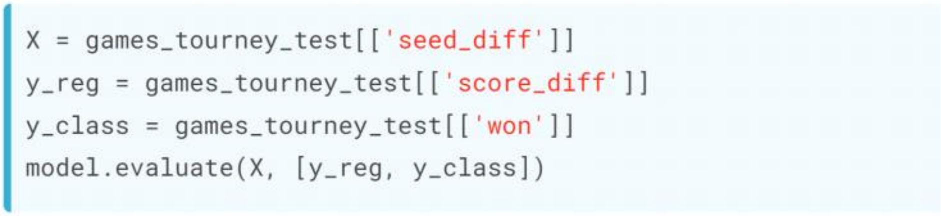


```
from scipy.special import expit as sigmoid
print(sigmoid(1 * 0.13870609 + 0.00734114))
```

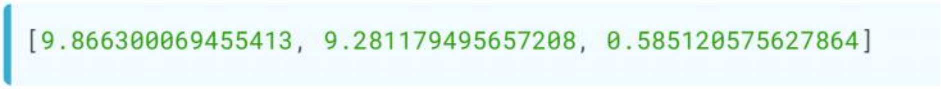


```
0.5364470465211318
```

Evaluate the model on new data



```
X = games_tourney_test[['seed_diff']]
y_reg = games_tourney_test[['score_diff']]
y_class = games_tourney_test[['won']]
model.evaluate(X, [y_reg, y_class])
```



```
[9.866300069455413, 9.281179495657208, 0.585120575627864]
```