Introduction

In this exercise, you'll build a model to predict hotel cancellations with a binary classifier.

In [1]:
```
# Setup plotting
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
# Set Matplotlib defaults
plt.rc('figure', autolayout=True)
plt.rc('axes', labelweight='bold', labelsize='large',
       titleweight='bold', titlesize=18, titlepad=10)
plt.rc('animation', html='html5')

# Setup feedback system
from learntools.core import binder
binder.bind(globals())
from learntools.deep_learning_intro.ex6 import *
```

First, load the *Hotel Cancellations* dataset.

In [2]:
```
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer

hotel = pd.read_csv('../input/dl-course-data/hotel.csv')

X = hotel.copy()
y = X.pop('is_canceled')

X['arrival_date_month'] = \
    X['arrival_date_month'].map(
        {'January':1, 'February': 2, 'March':3,
         'April':4, 'May':5, 'June':6, 'July':7,
         'August':8, 'September':9, 'October':10,
         'November':11, 'December':12}
    )

features_num = [
    "lead_time", "arrival_date_week_number",
    "arrival_date_day_of_month", "stays_in_weekend_nights",
    "stays_in_week_nights", "adults", "children", "babies",
    "is_repeated_guest", "previous_cancellations",
    "previous_bookings_not_canceled", "required_car_parking_spaces",
    "total_of_special_requests", "adr",
]
features_cat = [
    "hotel", "arrival_date_month", "meal",
    "market_segment", "distribution_channel",
    "reserved_room_type", "deposit_type", "customer_type",
]
```

```
transformer_num = make_pipeline(
    SimpleImputer(strategy="constant"), # there are a few missing values
    StandardScaler(),
)
transformer_cat = make_pipeline(
    SimpleImputer(strategy="constant", fill_value="NA"),
    OneHotEncoder(handle_unknown='ignore'),
)

preprocessor = make_column_transformer(
    (transformer_num, features_num),
    (transformer_cat, features_cat),
)

# stratify - make sure classes are evenlly represented across splits
X_train, X_valid, y_train, y_valid = \
    train_test_split(X, y, stratify=y, train_size=0.75)

X_train = preprocessor.fit_transform(X_train)
X_valid = preprocessor.transform(X_valid)

input_shape = [X_train.shape[1]]
```

1) Define Model

The model we'll use this time will have both batch normalization and dropout layers. To ease reading we've broken the diagram into blocks, but you can define it layer by layer as usual.

Define a model with an architecture given by this diagram:

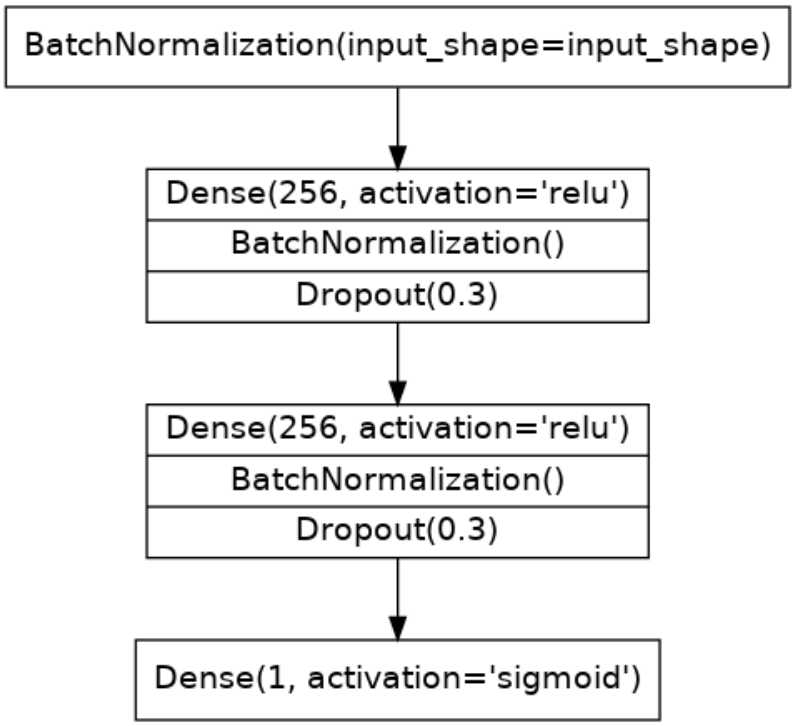


Diagram of a binary classifier.

In [3]:

```
from tensorflow import keras
from tensorflow.keras import layers

# YOUR CODE HERE: define the model given in the diagram
model = keras.Sequential([
    layers.BatchNormalization(input_shape=input_shape),
    layers.Dense(256, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(rate=0.3),
    layers.Dense(256, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(rate=0.3),
    layers.Dense(1, activation='sigmoid'),
])

# Check your answer
q_1.check()
```

```
2022-12-18 05:45:58.811266: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
successful NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero
2022-12-18 05:45:58.910418: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
successful NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero
2022-12-18 05:45:58.911170: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
successful NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero
2022-12-18 05:45:58.912887: I tensorflow/core/platform/cpu_feature_guard.cc:142] This Te
nsorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the
following CPU instructions in performance-critical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler fla
gs.
2022-12-18 05:45:58.915558: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
successful NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero
2022-12-18 05:45:58.916259: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
successful NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero
2022-12-18 05:45:58.916974: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
successful NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero
2022-12-18 05:46:01.175037: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
successful NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero
2022-12-18 05:46:01.175936: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
successful NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero
2022-12-18 05:46:01.176612: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
successful NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero
2022-12-18 05:46:01.177187: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Cre
ated device /job:localhost/replica:0/task:0/device:GPU:0 with 15401 MB memory:  -> devic
e: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0
```

Correct

2) Add Optimizer, Loss, and Metric

Now compile the model with the Adam optimizer and binary versions of the cross-entropy loss and accuracy metric.

In [4]:
```
# YOUR CODE HERE
model.compile(optimizer='adam',loss='binary_crossentropy',
    metrics=['binary_accuracy'],)

# Check your answer
q_2.check()
```

Correct

In [5]:
```
# Lines below will give you a hint or solution code
#q_2.hint()
#q_2.solution()
```

Finally, run this cell to train the model and view the learning curves. It may run for around 60 to 70 epochs, which could take a minute or two.

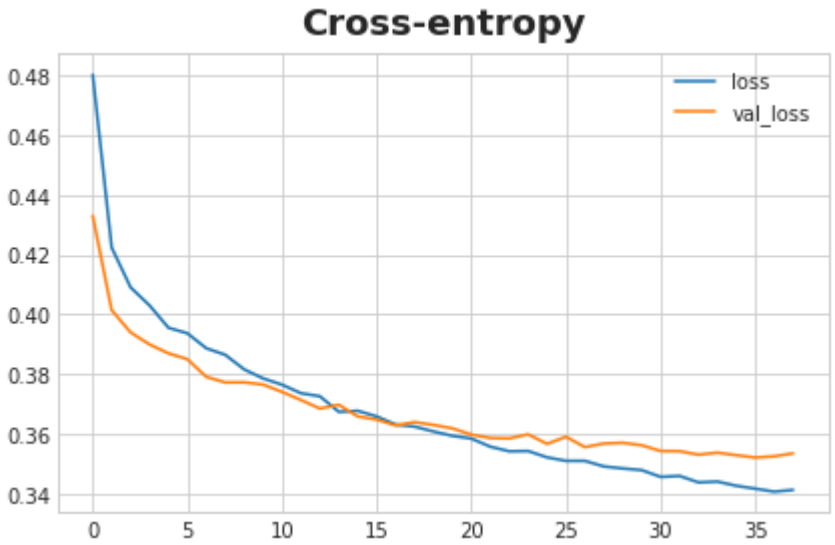
In [6]:
```
early_stopping = keras.callbacks.EarlyStopping(
    patience=5,
    min_delta=0.001,
    restore_best_weights=True,
)
history = model.fit(
    X_train, y_train,
    validation_data=(X_valid, y_valid),
    batch_size=512,
    epochs=200,
    callbacks=[early_stopping],
)

history_df = pd.DataFrame(history.history)
history_df.loc[:, ['loss', 'val_loss']].plot(title="Cross-entropy")
history_df.loc[:, ['binary_accuracy', 'val_binary_accuracy']].plot(title="Accuracy")
```

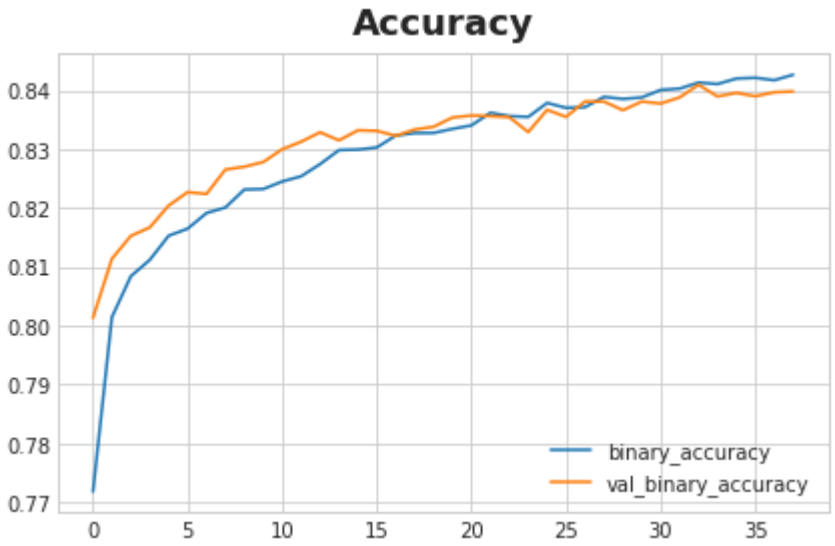
```
2022-12-18 05:46:01.804425: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:1
85] None of the MLIR Optimization Passes are enabled (registered 2)
Epoch 1/200
175/175 [==============================] - 3s 7ms/step - loss: 0.4803 - binary_accuracy:
0.7718 - val_loss: 0.4330 - val_binary_accuracy: 0.8013
Epoch 2/200
175/175 [==============================] - 1s 4ms/step - loss: 0.4224 - binary_accuracy:
0.8014 - val_loss: 0.4014 - val_binary_accuracy: 0.8114
Epoch 3/200
175/175 [==============================] - 1s 4ms/step - loss: 0.4090 - binary_accuracy:
0.8084 - val_loss: 0.3939 - val_binary_accuracy: 0.8153
Epoch 4/200
175/175 [==============================] - 1s 5ms/step - loss: 0.4030 - binary_accuracy:
0.8112 - val_loss: 0.3899 - val_binary_accuracy: 0.8167
```

```
Epoch 5/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3955 - binary_accuracy:
0.8153 - val_loss: 0.3869 - val_binary_accuracy: 0.8204
Epoch 6/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3936 - binary_accuracy:
0.8165 - val_loss: 0.3850 - val_binary_accuracy: 0.8227
Epoch 7/200
175/175 [==============================] - 1s 5ms/step - loss: 0.3886 - binary_accuracy:
0.8192 - val_loss: 0.3790 - val_binary_accuracy: 0.8224
Epoch 8/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3864 - binary_accuracy:
0.8201 - val_loss: 0.3772 - val_binary_accuracy: 0.8266
Epoch 9/200
175/175 [==============================] - 1s 5ms/step - loss: 0.3816 - binary_accuracy:
0.8232 - val_loss: 0.3772 - val_binary_accuracy: 0.8270
Epoch 10/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3785 - binary_accuracy:
0.8232 - val_loss: 0.3765 - val_binary_accuracy: 0.8278
Epoch 11/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3764 - binary_accuracy:
0.8245 - val_loss: 0.3740 - val_binary_accuracy: 0.8300
Epoch 12/200
175/175 [==============================] - 1s 5ms/step - loss: 0.3736 - binary_accuracy:
0.8254 - val_loss: 0.3713 - val_binary_accuracy: 0.8313
Epoch 13/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3725 - binary_accuracy:
0.8275 - val_loss: 0.3684 - val_binary_accuracy: 0.8329
Epoch 14/200
175/175 [==============================] - 1s 5ms/step - loss: 0.3673 - binary_accuracy:
0.8299 - val_loss: 0.3697 - val_binary_accuracy: 0.8315
Epoch 15/200
175/175 [==============================] - 1s 5ms/step - loss: 0.3677 - binary_accuracy:
0.8299 - val_loss: 0.3658 - val_binary_accuracy: 0.8332
Epoch 16/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3658 - binary_accuracy:
0.8303 - val_loss: 0.3647 - val_binary_accuracy: 0.8331
Epoch 17/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3631 - binary_accuracy:
0.8323 - val_loss: 0.3628 - val_binary_accuracy: 0.8323
Epoch 18/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3624 - binary_accuracy:
0.8327 - val_loss: 0.3639 - val_binary_accuracy: 0.8334
Epoch 19/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3607 - binary_accuracy:
0.8328 - val_loss: 0.3629 - val_binary_accuracy: 0.8338
Epoch 20/200
175/175 [==============================] - 1s 5ms/step - loss: 0.3593 - binary_accuracy:
0.8335 - val_loss: 0.3618 - val_binary_accuracy: 0.8354
Epoch 21/200
175/175 [==============================] - 1s 6ms/step - loss: 0.3584 - binary_accuracy:
0.8341 - val_loss: 0.3597 - val_binary_accuracy: 0.8357
Epoch 22/200
175/175 [==============================] - 1s 6ms/step - loss: 0.3557 - binary_accuracy:
0.8362 - val_loss: 0.3586 - val_binary_accuracy: 0.8356
Epoch 23/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3541 - binary_accuracy:
0.8356 - val_loss: 0.3584 - val_binary_accuracy: 0.8355
Epoch 24/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3542 - binary_accuracy:
0.8355 - val_loss: 0.3598 - val_binary_accuracy: 0.8329
```

```
Epoch 25/200
175/175 [==============================] - 1s 5ms/step - loss: 0.3521 - binary_accuracy:
0.8379 - val_loss: 0.3566 - val_binary_accuracy: 0.8367
Epoch 26/200
175/175 [==============================] - 1s 5ms/step - loss: 0.3509 - binary_accuracy:
0.8370 - val_loss: 0.3590 - val_binary_accuracy: 0.8355
Epoch 27/200
175/175 [==============================] - 1s 5ms/step - loss: 0.3509 - binary_accuracy:
0.8371 - val_loss: 0.3555 - val_binary_accuracy: 0.8381
Epoch 28/200
175/175 [==============================] - 1s 5ms/step - loss: 0.3490 - binary_accuracy:
0.8389 - val_loss: 0.3567 - val_binary_accuracy: 0.8381
Epoch 29/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3484 - binary_accuracy:
0.8386 - val_loss: 0.3570 - val_binary_accuracy: 0.8366
Epoch 30/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3478 - binary_accuracy:
0.8388 - val_loss: 0.3561 - val_binary_accuracy: 0.8381
Epoch 31/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3455 - binary_accuracy:
0.8401 - val_loss: 0.3542 - val_binary_accuracy: 0.8378
Epoch 32/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3459 - binary_accuracy:
0.8403 - val_loss: 0.3541 - val_binary_accuracy: 0.8388
Epoch 33/200
175/175 [==============================] - 1s 5ms/step - loss: 0.3437 - binary_accuracy:
0.8413 - val_loss: 0.3530 - val_binary_accuracy: 0.8410
Epoch 34/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3440 - binary_accuracy:
0.8411 - val_loss: 0.3537 - val_binary_accuracy: 0.8390
Epoch 35/200
175/175 [==============================] - 1s 5ms/step - loss: 0.3426 - binary_accuracy:
0.8420 - val_loss: 0.3528 - val_binary_accuracy: 0.8396
Epoch 36/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3416 - binary_accuracy:
0.8422 - val_loss: 0.3520 - val_binary_accuracy: 0.8390
Epoch 37/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3406 - binary_accuracy:
0.8417 - val_loss: 0.3524 - val_binary_accuracy: 0.8397
Epoch 38/200
175/175 [==============================] - 1s 4ms/step - loss: 0.3412 - binary_accuracy:
0.8427 - val_loss: 0.3534 - val_binary_accuracy: 0.8398
```

Out[6]: <AxesSubplot:title={'center':'Accuracy'}>





3) Train and Evaluate

What do you think about the learning curves? Does it look like the model underfit or overfit? Was the cross-entropy loss a good stand-in for accuracy?

In [7]:
```
# View the solution (Run this cell to receive credit!)
q_3.check()
```

Correct:

Though we can see the training loss continuing to fall, the early stopping callback prevented any overfitting. Moreover, the accuracy rose at the same rate as the cross-entropy fell, so it appears that minimizing cross-entropy was a good stand-in. All in all, it looks like this training was a success!

Conclusion

Congratulations! You've completed Kaggle's *Introduction to Deep Learning* course!

With your new skills you're ready to take on more advanced applications like computer vision and sentiment classification. What would you like to do next?

Why not try one of our *Getting Started* competitions?

- Classify images with TPUs in **Petals to the Metal**
- Create art with GANs in **I'm Something of a Painter Myself**
- Classify Tweets in **Real or Not? NLP with Disaster Tweets**
- Detect contradiction and entailment in **Contradictory, My Dear Watson**

Until next time, Kagglers!

Have questions or comments? Visit the course discussion forum to chat with other learners.