hotel

March 26, 2021

0.1 Hotel Cancellations (Originally from Kaggle Courses)

We will build a model to predict hotel cancellations with a binary classifier.

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

[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('./hotel.csv')

X = hotel.copy() # Deep Copy
y = X.pop('is_canceled')
```

[3]:
```
X.head()
```

[3]:

	hotel	lead_time	arrival_date_year	arrival_date_month	\
0	Resort Hotel	342	2015	July	
1	Resort Hotel	737	2015	July	
2	Resort Hotel	7	2015	July	
3	Resort Hotel	13	2015	July	
4	Resort Hotel	14	2015	July	

	arrival_date_week_number	arrival_date_day_of_month	\
0	27	1	
1	27	1	
2	27	1	
3	27	1	

```
4                        27                            1

   stays_in_weekend_nights  stays_in_week_nights  adults  children  …  \
0                        0                     0       2       0.0  …
1                        0                     0       2       0.0  …
2                        0                     1       1       0.0  …
3                        0                     1       1       0.0  …
4                        0                     2       2       0.0  …

   deposit_type  agent company days_in_waiting_list customer_type   adr  \
0    No Deposit    NaN     NaN                    0     Transient   0.0
1    No Deposit    NaN     NaN                    0     Transient   0.0
2    No Deposit    NaN     NaN                    0     Transient  75.0
3    No Deposit  304.0     NaN                    0     Transient  75.0
4    No Deposit  240.0     NaN                    0     Transient  98.0

   required_car_parking_spaces  total_of_special_requests reservation_status  \
0                            0                          0          Check-Out
1                            0                          0          Check-Out
2                            0                          0          Check-Out
3                            0                          0          Check-Out
4                            0                          1          Check-Out

  reservation_status_date
0              2015-07-01
1              2015-07-01
2              2015-07-02
3              2015-07-02
4              2015-07-03

[5 rows x 31 columns]
```

[4]:
```
y.head()
```

[4]:
```
0    0
1    0
2    0
3    0
4    0
Name: is_canceled, dtype: int64
```

[5]:
```
# Mapping dates to numbers

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})
```

[6]:
```
# Defining features

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"]
```

[7]:
```
# Defining categorical features

features_cat = ["hotel", "arrival_date_month", "meal",
                "market_segment", "distribution_channel",
                "reserved_room_type", "deposit_type", "customer_type"]
```

[8]:
```
# Transforming numerical and categorical features

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))
```

[9]:
```
# 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]]
print(input_shape)
```

```
[63]
```

0.2 Defining a Model

The model will have both batch normalization and dropout layers.

[10]:
```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([layers.BatchNormalization(),
                          layers.Dense(256, activation='relu',␣
 ↪input_shape=input_shape),
                          layers.BatchNormalization(),
                          layers.Dropout(rate=0.3),
                          layers.Dense(256, activation='relu'),
                          layers.BatchNormalization(),
                          layers.Dropout(rate=0.3),
                          layers.Dense(1, activation='sigmoid')])
```

0.3 Optimizer, Loss Function, and Accuracy Metric

We use Adam optimizer and binary versions of the cross-entropy loss and accuracy metric.

[11]:
```
model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['binary_accuracy'])
```

0.4 Early stopping to prevent overfitting

[12]:
```
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])
```

```
Epoch 1/200
175/175 [==============================] - 3s 10ms/step - loss: 0.5533 -
binary_accuracy: 0.7312 - val_loss: 0.4381 - val_binary_accuracy: 0.8044
Epoch 2/200
175/175 [==============================] - 1s 8ms/step - loss: 0.4311 -
binary_accuracy: 0.7976 - val_loss: 0.4026 - val_binary_accuracy: 0.8138
Epoch 3/200
175/175 [==============================] - 1s 8ms/step - loss: 0.4133 -
binary_accuracy: 0.8075 - val_loss: 0.3988 - val_binary_accuracy: 0.8138
Epoch 4/200
175/175 [==============================] - 1s 8ms/step - loss: 0.4009 -
binary_accuracy: 0.8117 - val_loss: 0.3913 - val_binary_accuracy: 0.8175
Epoch 5/200
```

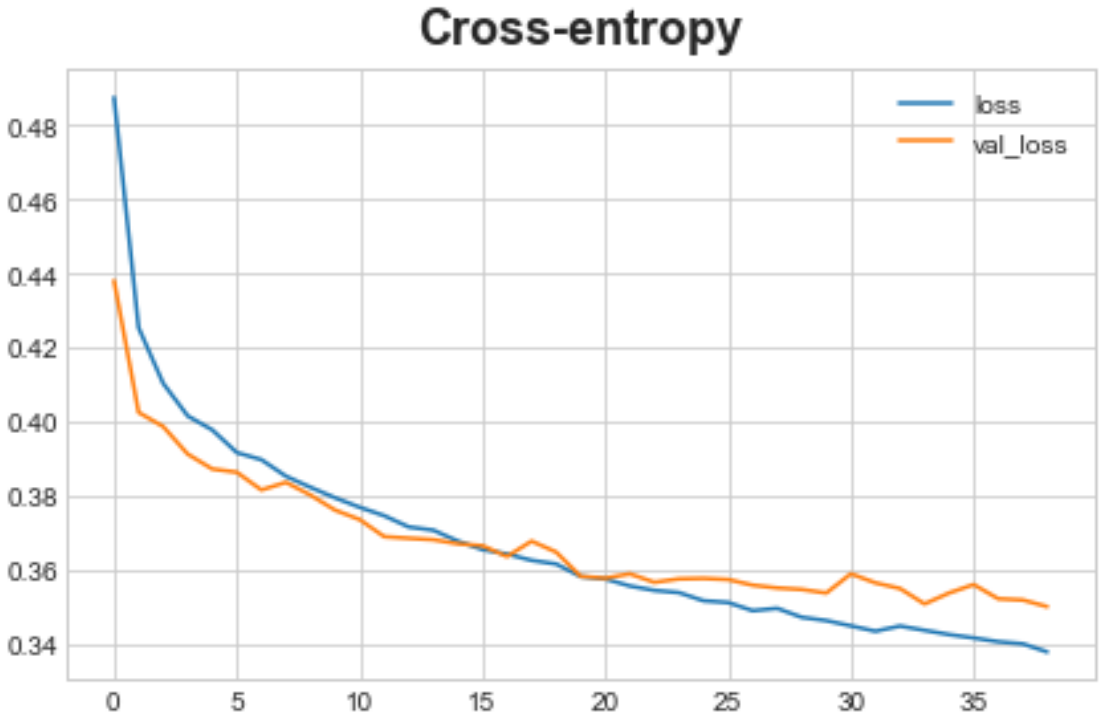
```
175/175 [==============================] - 1s 8ms/step - loss: 0.3978 -
binary_accuracy: 0.8147 - val_loss: 0.3873 - val_binary_accuracy: 0.8209
Epoch 6/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3906 -
binary_accuracy: 0.8185 - val_loss: 0.3864 - val_binary_accuracy: 0.8213
Epoch 7/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3900 -
binary_accuracy: 0.8184 - val_loss: 0.3817 - val_binary_accuracy: 0.8247
Epoch 8/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3834 -
binary_accuracy: 0.8215 - val_loss: 0.3837 - val_binary_accuracy: 0.8239
Epoch 9/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3818 -
binary_accuracy: 0.8211 - val_loss: 0.3803 - val_binary_accuracy: 0.8265
Epoch 10/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3794 -
binary_accuracy: 0.8224 - val_loss: 0.3762 - val_binary_accuracy: 0.8300
Epoch 11/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3764 -
binary_accuracy: 0.8249 - val_loss: 0.3737 - val_binary_accuracy: 0.8328
Epoch 12/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3735 -
binary_accuracy: 0.8267 - val_loss: 0.3691 - val_binary_accuracy: 0.8318
Epoch 13/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3715 -
binary_accuracy: 0.8279 - val_loss: 0.3686 - val_binary_accuracy: 0.8322
Epoch 14/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3703 -
binary_accuracy: 0.8279 - val_loss: 0.3683 - val_binary_accuracy: 0.8324
Epoch 15/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3670 -
binary_accuracy: 0.8296 - val_loss: 0.3672 - val_binary_accuracy: 0.8328
Epoch 16/200
175/175 [==============================] - 2s 9ms/step - loss: 0.3632 -
binary_accuracy: 0.8324 - val_loss: 0.3666 - val_binary_accuracy: 0.8347
Epoch 17/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3638 -
binary_accuracy: 0.8301 - val_loss: 0.3637 - val_binary_accuracy: 0.8346
Epoch 18/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3639 -
binary_accuracy: 0.8319 - val_loss: 0.3679 - val_binary_accuracy: 0.8341
Epoch 19/200
175/175 [==============================] - 1s 8ms/step - loss: 0.3591 -
binary_accuracy: 0.8340 - val_loss: 0.3649 - val_binary_accuracy: 0.8345
Epoch 20/200
175/175 [==============================] - 2s 9ms/step - loss: 0.3568 -
binary_accuracy: 0.8338 - val_loss: 0.3584 - val_binary_accuracy: 0.8375
Epoch 21/200
```

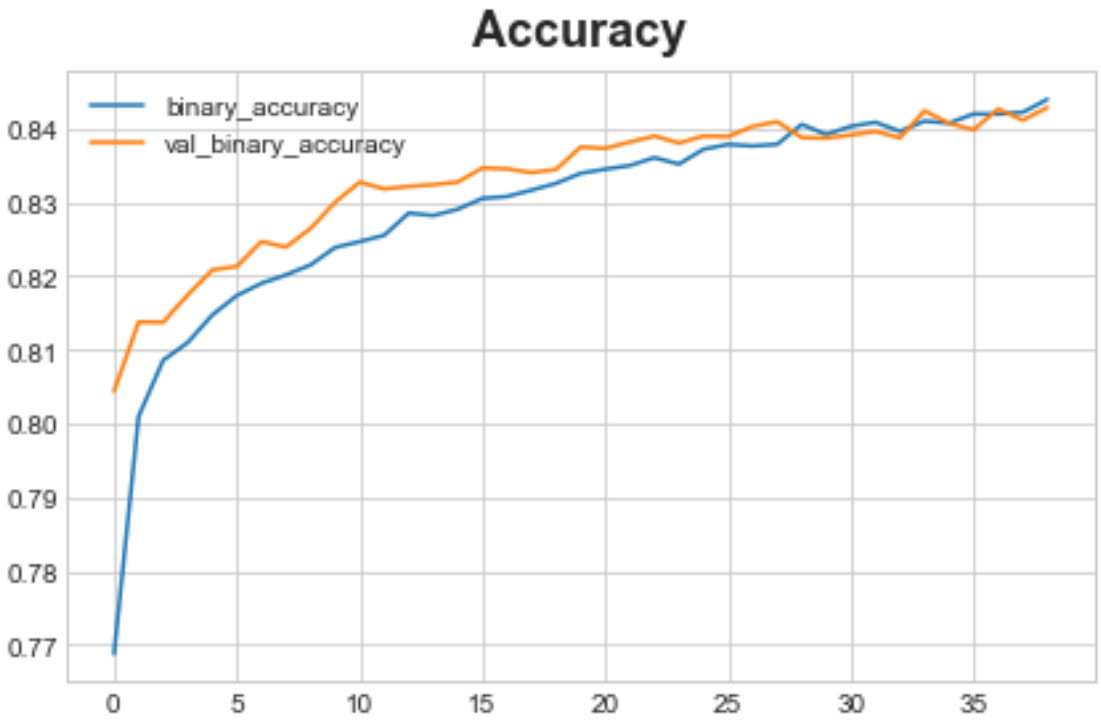
```
175/175 [==============================] - 2s 9ms/step - loss: 0.3575 -
binary_accuracy: 0.8347 - val_loss: 0.3579 - val_binary_accuracy: 0.8373
Epoch 22/200
175/175 [==============================] - 2s 10ms/step - loss: 0.3543 -
binary_accuracy: 0.8354 - val_loss: 0.3591 - val_binary_accuracy: 0.8382
Epoch 23/200
175/175 [==============================] - 2s 10ms/step - loss: 0.3503 -
binary_accuracy: 0.8385 - val_loss: 0.3568 - val_binary_accuracy: 0.8390
Epoch 24/200
175/175 [==============================] - 2s 9ms/step - loss: 0.3513 -
binary_accuracy: 0.8373 - val_loss: 0.3577 - val_binary_accuracy: 0.8381
Epoch 25/200
175/175 [==============================] - 2s 10ms/step - loss: 0.3498 -
binary_accuracy: 0.8383 - val_loss: 0.3578 - val_binary_accuracy: 0.8390
Epoch 26/200
175/175 [==============================] - 2s 10ms/step - loss: 0.3501 -
binary_accuracy: 0.8385 - val_loss: 0.3575 - val_binary_accuracy: 0.8390
Epoch 27/200
175/175 [==============================] - 2s 10ms/step - loss: 0.3453 -
binary_accuracy: 0.8392 - val_loss: 0.3560 - val_binary_accuracy: 0.8403
Epoch 28/200
175/175 [==============================] - 2s 10ms/step - loss: 0.3490 -
binary_accuracy: 0.8372 - val_loss: 0.3552 - val_binary_accuracy: 0.8410
Epoch 29/200
175/175 [==============================] - 2s 10ms/step - loss: 0.3460 -
binary_accuracy: 0.8402 - val_loss: 0.3548 - val_binary_accuracy: 0.8388
Epoch 30/200
175/175 [==============================] - 2s 10ms/step - loss: 0.3446 -
binary_accuracy: 0.8395 - val_loss: 0.3539 - val_binary_accuracy: 0.8387
Epoch 31/200
175/175 [==============================] - 2s 10ms/step - loss: 0.3419 -
binary_accuracy: 0.8416 - val_loss: 0.3591 - val_binary_accuracy: 0.8392
Epoch 32/200
175/175 [==============================] - 2s 10ms/step - loss: 0.3416 -
binary_accuracy: 0.8418 - val_loss: 0.3566 - val_binary_accuracy: 0.8397
Epoch 33/200
175/175 [==============================] - 2s 10ms/step - loss: 0.3424 -
binary_accuracy: 0.8395 - val_loss: 0.3551 - val_binary_accuracy: 0.8388
Epoch 34/200
175/175 [==============================] - 2s 10ms/step - loss: 0.3428 -
binary_accuracy: 0.8408 - val_loss: 0.3509 - val_binary_accuracy: 0.8424
Epoch 35/200
175/175 [==============================] - 2s 11ms/step - loss: 0.3419 -
binary_accuracy: 0.8417 - val_loss: 0.3539 - val_binary_accuracy: 0.8408
Epoch 36/200
175/175 [==============================] - 2s 11ms/step - loss: 0.3412 -
binary_accuracy: 0.8418 - val_loss: 0.3562 - val_binary_accuracy: 0.8399
Epoch 37/200
```

```
175/175 [==============================] - 2s 10ms/step - loss: 0.3397 -
binary_accuracy: 0.8425 - val_loss: 0.3523 - val_binary_accuracy: 0.8427
Epoch 38/200
175/175 [==============================] - 2s 11ms/step - loss: 0.3399 -
binary_accuracy: 0.8423 - val_loss: 0.3520 - val_binary_accuracy: 0.8412
Epoch 39/200
175/175 [==============================] - 2s 11ms/step - loss: 0.3358 -
binary_accuracy: 0.8450 - val_loss: 0.3502 - val_binary_accuracy: 0.8429
```

[13]:
```
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")
```

[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f965e6e1f40>




Accuracy
binary_accuracy
val_binary_accuracy
0.84
0.83
0.82
0.81
0.80
0.79
0.78
0.77
0
5
10
15
20
25
30
35

[ ]: