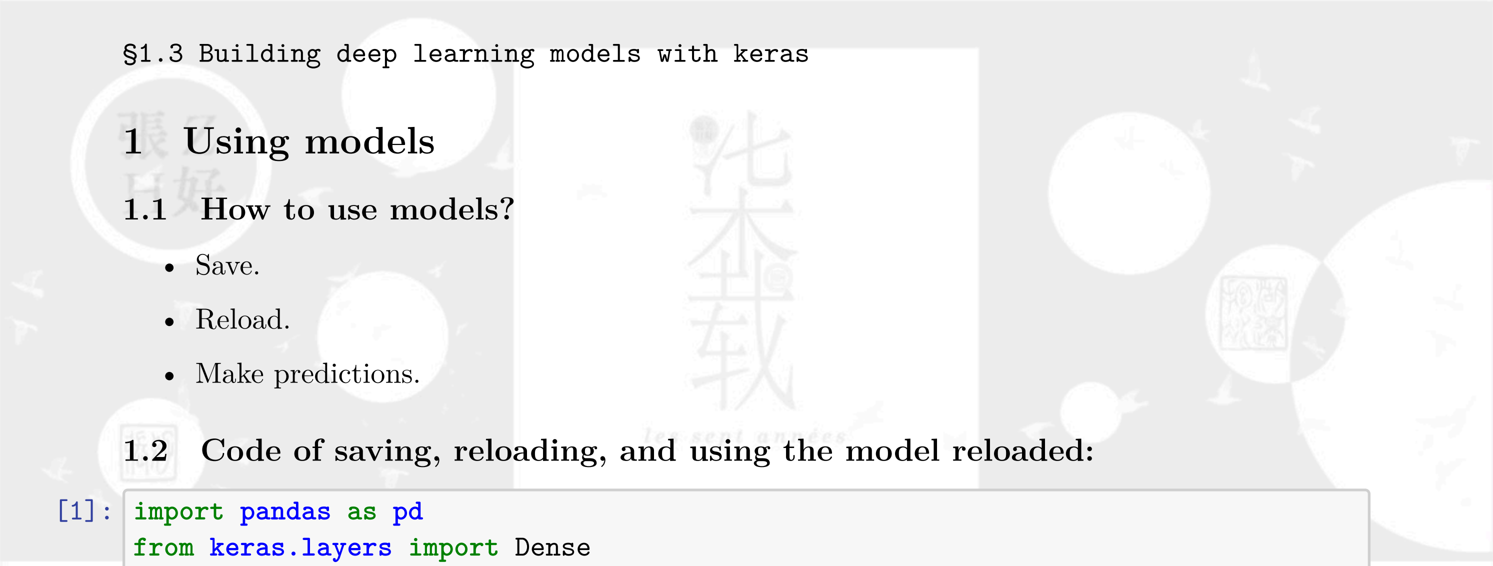
# Using models

Puteaux, Fall/Winter 2020-2021

```
################################
##                            ##
##  Deep Learning in Python   ##
##                            ##
################################
```

```
§1 Introduction to Deep Learning in Python
```



```
§1.3 Building deep learning models with keras
```

# 1 Using models

## 1.1 How to use models?

- Save.
- Reload.
- Make predictions.

## 1.2 Code of saving, reloading, and using the model reloaded:

[1]:
```python
import pandas as pd
from keras.layers import Dense
from keras.models import Sequential
from keras.utils.np_utils import to_categorical

data = pd.read_csv('ref5. Basketball shot log.csv')


def data_preparation(df):
    df = df.reindex(columns=[
        'SHOT_CLOCK', 'DRIBBLES', 'TOUCH_TIME', 'SHOT_DIST', 'CLOSE_DEF_DIST',
        'SHOT_RESULT'
    ])
    df['SHOT_CLOCK'] = df['SHOT_CLOCK'].fillna(0)
    df['SHOT_RESULT'].replace('missed', 0, inplace=True)
    df['SHOT_RESULT'].replace('made', 1, inplace=True)
    df.columns = df.columns.str.lower()
    return df


df = data_preparation(data)
predictors = df.drop(['shot_result'], axis=1).to_numpy()
n_cols = predictors.shape[1]
target = to_categorical(df.shot_result)

model = Sequential()
model.add(Dense(100, activation='relu', input_shape=(n_cols, )))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(predictors, target)
```

```
4003/4003 [==============================] - 4s 988us/step - loss: 0.6641 -
accuracy: 0.6055
```

```
[1]: <tensorflow.python.keras.callbacks.History at 0x7f9086403310>
```

```
[2]: from keras.models import load_model

     model.save('ref7. Model file.h5')
     my_model = load_model('ref7. Model file.h5')

     predictions = my_model.predict(predictors)
     probability_true = predictions[:, 1]
     probability_true
```

```
[2]: array([0.43192425, 0.3401706 , 0.35761935, …, 0.40975374, 0.40411907,
            0.47777078], dtype=float32)
```

```
[3]: my_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 100)               600
_________________________________________________________________
dense_1 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_2 (Dense)              (None, 100)               10100
_________________________________________________________________
```

```
dense_3 (Dense)              (None, 2)                 202
=================================================================
Total params: 21,002
Trainable params: 21,002
Non-trainable params: 0
_________________________________________________________________
```

## 1.3 Practice exercises for using models:

▶ **Package pre-loading:**

[4]:

```
import pandas as pd
from keras.layers import Dense
from keras.models import Sequential
from keras.utils import to_categorical
```

▶ **Data pre-loading:**

[5]:

```
df = pd.read_csv('ref6. Titanic.csv')

df.replace(False, 0, inplace=True)
df.replace(True, 1, inplace=True)

predictors = df.drop(['survived'], axis=1).to_numpy()
n_cols = predictors.shape[1]
target = to_categorical(df.survived)

pred_data = pd.read_csv('ref8. Titanic predictors data.csv')
pred_data.replace(False, 0, inplace=True)
pred_data.replace(True, 1, inplace=True)
```

▶ **Making predictions practice:**

[6]:

```
# Specify, compile, and fit the model
model = Sequential()
model.add(Dense(32, activation='relu', input_shape=(n_cols, )))
model.add(Dense(2, activation='softmax'))
model.compile(optimizer='sgd',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(predictors, target)

# Calculate predictions: predictions
predictions = model.predict(pred_data)

# Calculate predicted probability of survival: predicted_prob_true
predicted_prob_true = predictions[:, 1]
```

```
# print predicted_prob_true
print(predicted_prob_true)
```

```
28/28 [==============================] - 0s 8ms/step - loss: 4.6189 - accuracy:
0.5947
[0.33328182 0.46244395 0.98904455 0.45726845 0.23479225 0.23322007
 0.19821607 0.3631005  0.2882916  0.43549198 0.2729657  0.42995116
 0.2880962  0.7672388  0.23744662 0.21667662 0.32192716 0.4371153
 0.17939065 0.6097299  0.35813093 0.2831361  0.20222539 0.385043
 0.8812879  0.23198238 0.3695973  0.8302557  0.23997924 0.32296962
 0.4527913  0.6848475  0.21921225 0.29762074 0.34601828 0.37780848
 0.32957593 0.20942448 0.32904848 0.45084354 0.32383007 0.4500415
 0.42743757 0.2130326  0.36068285 0.19255604 0.80995643 0.22487696
 0.42362073 0.7379827  0.8083559  0.10151579 0.45197144 0.49407145
 0.47167435 0.39426166 0.922086   0.43898413 0.49883062 0.21921225
 0.21931268 0.44096443 0.46569982 0.80711687 0.43567106 0.35603607
 0.4742675  0.388488   0.2390165  0.4645945  0.27332    0.4191591
 0.2877974  0.18710254 0.45809928 0.41166952 0.35927066 0.33088225
 0.20766094 0.3508171  0.44333902 0.22947581 0.3878728  0.34814742
 0.25498232 0.4627092  0.40392458 0.45070586 0.47427937 0.4299565
 0.25209993]
```