LELEC2870: Exercise Session 2
The Scikit learn Pipeline

1 Objectives

The `sklearn` package comes packed with a very powerful API. Once you understand how it functions, you'll be able to write your training pipeline in a single line of code. This increases readability and will be an enormous time gain.

2 The Pipeline

2.1 Transforming the data

The first step for a lot of models is to preprocess the data. Some preprocessing (like normalization for ex.) needs to be *fitted* to the training data. Afterwards the data can be *transformed* with these computed parameters. The API is as follows:

```
class MyTransformer():
    def fit(self, X, y=None):
        # Code to fit the data
        # Return the object itself!
        return self

    def transform(self, X):
        # Transform X based on the fitted data
        # Return X-transformed
        return X_tranformed

    def fit_transform(self, X, y=None):
        # This is a common shortcut to directly
        #   transform the data after fitting
        return self.fit(X, y).transform(X)
```

2.2 Learning a model

Now that we're able to transform the data we want it to be *fitted* by a model. Afterwards we want the model to *predict* new values for (un)seen data. And to *score* those predictions compared to the ground truth.

```
class MyModel():
    def fit(self, X, y):
        # Code to fit the data
        # Return the object itself!
        return self

    def predict(self, X):
        # Predict y based on the fitted parameters and X
        # Return the predicted y
        return y_predict

    def fit_predict(self, X, y):
        return self.fit(X, y).predict(X)

    def score(self, X, y_true):
        y = self.predict(X)
        # Predict y
        # Return the score: a metric defining a 'distance'
        #   between the true solution and the predicted one
        return my_score(y, y_true)
```

2.3 Putting it all together

Using the `Pipeline` object inside `sklearn`, we can finally declare a **sequential** list of the transformations and models we want to apply on our data:

```
my_pipe = Pipeline([
    ('transform', MyTransformer()),
    ('model', MyModel())
])
y_predict = my_pipe.fit_predict(X, y_true)
```

In the example above you can see that we don't need to call any `transform` function because calling `fit_predict` automatically calls `transform` if `predict` isn't available.

You can use this in combination with tools for cross-validation (function `cross_val_score` in the module `sklearn.model_selection`) and many others. Do not forget to read the API carefully every time you use a new function and don't waste your time rewriting functions that already exist (unless stated explicitly).