SGN-41007 Assignment Task 1

*Predict competition classes using **scikit-learn** tools.*

- ▷ The first assignment task uses scikit-learn tools with manual feature extraction.
- ▷ Here, we download the data, fit and predict in Python and write the results in a submission file.
- ▷ A written report of the group solution is required; due date Monday 11.2.2019 at 23:59. Return to Moodle.

1. *Load data.*

   When you have downloaded the data from Kaggle, open the three files in your Python script using `numpy.load()` and `numpy.loadtxt()`. Name the loaded variables as `X_train` and `y_train`, which are directly usable for `model.fit()`; and `X_test` which contains the data for the secret test part.

2. *Create an index of class names.*

   The class names in the `y_train` and `y_test` are strings. Scikit-learn does accept labels as strings, but we will convert them to integers for later use (Keras needs one-hot-encoded classes). To this aim, you need to apply `sklearn.preprocessing.LabelEncoder()` to swap between text labels and numerical labels.

3. *Split to training and testing.*

   Instead of simply fitting and submitting the predictions, you are required to compare different algorithms locally. This enables more experiments, as you are only allowed two submission per day. Therefore, you want to split your training data to train/test sets locally, *e.g.* in proportion 80/20, *i.e.*, 80% for training and 20% for local evaluation.

   Since the samples are extracted from sequences, you need to avoid having samples close in time in your local training and test sets. Otherwise, you are showing the sample $x_[t]$ at training and predicting the class for sample $x[t+1]$ resulting in unrealistically high CV scores. **shuffle**

   When preparing the data for the competition, we split first the full sequences to longer blocks, assigned each block to train/public/private sets, and finally split the blocks to 128-sample long pieces. Thus, all Kaggle LB data is computed from sample not near in time with your training data. More specifically, our procedure while creating the competition data was the following:

   a) Split the sequences into blocks.

   b) Assign all samples of each block to train, validation or test set.

   c) Split each block to 128-timestep long samples.

The block structure is available at Kaggle competition website (`groups.csv`), and it defines the block ID for each sample. A reliable CV procedure would put all samples from one block to either training or testing. Generate a train/test split using your training data with `sklearn.model_selection.GroupShuffleSplit`. You can give the group information to this function directly. Note that this is a generator, so you may need to transform the result into a list or numpy array before indexing.

4. *Extract features*.

   The data in `X_train` and `X_test` are multidimensional arrays. For example, the shape of `X_train` is `(1703, 10, 128)`, which means that there are 1703 samples, each representing a time series 128 timesteps from 10 sensors. However, scikit-learn classifiers only accept vectors, not matrices. Thus, try three vectorization approaches:

   a) Straightforward reshape using `numpy.ravel()` or `numpy.resize()` to make 1280-dimensional vector from each sample.

   b) Compute the average over the time axis using `numpy.mean()` with appropriate `axis` argument. This should make each sample 10-dimensional.

   c) Compute the average and standard deviation over the time axis and concatenate these to 20-dimensional feature vectors.

   d) Optional: Choose which sensors you wish to use, and possibly transform their readings into a more suitable format (e.g., quaternions into orientation angles, accelerations and velocities into absolute values, etc.).

   Fit a Linear discriminant analysis classifier with each variant and test with the 20% of validation data. Report the accuracies (`sklearn.metrics.accuracy_score()`) for each method in a table in your report.

5. *Try different models*.

   After choosing the best feature extraction procedure, experiment with the following classifiers (additional ones are welcome, too); measure the accuracy on the 20% validation set and report.

   a) Linear discriminant analysis classifier.

   b) Support vector machine (linear kernel).

   c) Support vector machine (RBF kernel).

   d) Logistic regression.

   e) Random forest.

6. *Create submission file*.

Select the best of the above classifiers, fit that with all training data (80% + 20%), and predict classes for the test data.

Create a submission file like in the following template.

```
# Assume that your LabelEncoder is called le.

y_pred = model.predict(X_train)
labels = list(le.inverse_transform(y_pred))

with open("submission.csv", "w") as fp:
   fp.write("Id,Scene_label\n")

   for i, label in enumerate(labels):
      fp.write("%d,%s\n" % (i, label))
```

7. *Submit.*

   Log in to Kaggle and submit your solution. Describe in your report what was your Kaggle score and how it differs from what you estimated locally using the 20% validation set.