**Documentation of presented project**

**Data used for classification**

Each .pickle file contains one array with a single dimension signal that was generated. Every signal has 5120000 data points and is labelled. Label can be found in file name of the file. There are only two class labels: 0 and 1. None of the arrays has any missing values.

**Necessary libraries**

Libraries used in the project are as follows: pandas, numpy, oc, gc, sklearn, matplotlib, pickle, keras.

**Methods created in the project**

* loadDataFromDirectorySplit(directory, class\_index)

Method loads each .pickle file from given directory and splits each signal into 128 parts. Labels are taken from file name and assigned to each signal part.

Inputs:

directory – path to directory containing data to be loaded

class\_index – index of class label in the string containing path of the file

Return value:

Method returns two lists, one containing arrays of lists of parts of loaded signal and second containing class labels of each signal part.

* loadDataFromDirectory(directory, class\_index)

Method loads each .pickle file from given directory into a list. Labels are taken from file name and assigned to each signal part.

Inputs:

directory – path to directory containing data to be loaded

class\_index – index of class label in the string containing path of the file

Return value:

Method returns two lists, one containing signals and second containing class labels of each signal.

* flatten(lst)

Method unnests lists from numpy.arrays. It changes list from [np.arr([...], ..., [...]), ..., np.arr([...], ..., [...])] to [[...], ..., [...]].

Inputs:

lst – list with architecture [np.arr([...], ..., [...]), ..., np.arr([...], ..., [...])]

Return value:

Method returns list with architecture [[...], ..., [...]].

* makeModel(input\_shape)

Method creates CNN for binary classification with four convolutional layers and three fully connected layers.

Inputs:

Input\_shape – shape of data used for training the model

Return value:

An instance of the Sequential class containing convolutional neural network.

* makePrediction(data, classes, path\_to\_model)

Method splits each inputted signal into 128 parts, multiplies instances of classes by 128, standardizes split signals, loads model form a file and creates prediction for inputted signals.

Inputs:

data – list of signals

classes – list of labels corresponding to data inputted earlier

path\_to\_model – path to .h5 file with the trained model

Return value:

List containing prediction for signals in order of inputted data.

**Preferable environment for project functionality**

Solution presented in jupyter notebook works the best in the following directory tree:

RecruitmentResistell/

├── best\_model.h5

├── Data/

│ └── all files with signals

└── Jan\_Winnicki\_Recruitment\_Project.ipynb

While directory tree remains the same there should be no need for changing anything in the notebook.