# Code documentation

## Abstract

The python code implements full automation in preparing the features, loading them into a dataset, performing models evaluation, choosing the best model according to the user-defined metrics, and making a classification on the desired test dataset and other unseen datasets.

The user must define:

* The features extractions dataset
* The tasks
* The models to test
* Validation metrics

The code includes explicit annotations and comments for better understanding

## The general process

### Features extraction

The pipeline from the HW no. 2 was used for this purpose. Additional option to use different pipelines. As explained in HW2, the pipeline is made of stages, each performs a certain operation on some feature/features in the dataframe, and returns the dataframe with modified features. This way, the dataframe at the output from the pipeline contains dataframe, which is ready to be input to the classifier.

### Defining the tasks

Since we have multiple tasks, we use the Class to define those. Each task definition required target specification (column name), and the metrics, upon which the best classifier should be chosen.

### Defining the models

User defines the list of models and dictionary of parameters he wishes to test. If no parameters are passed, the single model is used. If parameters dictionary includes numerous parameters, the GridSearch Cross Validation is used to find the best parameters for this model (according to the metrics for a specific task that the learning is made upon).

### Defining the validation metrics

User defines the metrics, which he wants to be tested on the validation dataset. For each task, all the models (all the best models after the CV process, if numerous parameters were specified) are later tested on all those metrics and the information is displayed in a table.

### Choosing the best model

For each task, the loop over all models occurs. If numerous parameters were specified, the GridSearchCV from sklearn is used, according to the metrics specified for the specific task. The scored are saved for each model, and the best is chosen.

### Evaluating on the test dataset

After the best models were chosen (they may be different for each task), they are evaluated on the given dataset. The results are then printed in the .csv file with predictions.

### Logging

The code contains explicit print statements, which are saved in the log file for further examination. It is possible to run the whole script in a shell mode without losing valuable information.

# Chosen models for the Virus task

## Spreader

### Process

### Result

The chosen model for this task is …..

## Risk

### Process

Use Sequential Forward Selection to find best features.

### Metric

Low penalty for assessing someone who isn’t at risk for disease as someone who is at risk (False Positive). On the other hand, high penalty for assessing someone who is at risk as being safe (False Negative). This implies high recall is crucial. Yet, the viruses aren’t life-threatening in nearly all cases, then accuracy should be taken into account as well, because wrongly thinking he/she is at risk can still be a burden on someone’s life.

Therefore, F1 score is ideal, because it can be thought of as a weighted average of precision and recall.

### 2.2.3 Result

The chosen model for this task is …..

## Disease

### Process

### Result

The chosen model for this task is …..

# Conclusion

Was cute.