Standard Operating Procedure

TU/e

SOP details

Title	Training a machine learning model from the TopoChip ranking data
Description	This SOP describes the typical steps that are required to train and explain machine learning model from the TopoChip ranking data.
Author	Aliaksei Vasilevich; Tim Kuijpers
SOP number	4.6
Version number	5

1 Purpose

To train a classification model (Gradient Boosting Classifier) on the TopoChip ranking data.

2 Principle

This SOP guides you through a data analysis template that contains functions for training and explaining the machine learning model from the TopoChip ranking data. It explains the key components, input, and output for the functions. The SOP also provides references to data analysis methods used in the tutorial. Steps that are covered in this tutorial are:

- Observe black and white images that represent the design of the topographies.
- Selection of the topography design descriptors
- Preparing data for the model
- Finding the best parameters for the machine learning model
- Training the model
- Assess the accuracy of the trained model
- Check Topo descriptors (parameters describing one unique topographical design) that are important for the model
- Explain the model by using the SHAP visualization algorithm.

3 Important to know before starting

1. Before training a Machine Learning (ML) model it is important to observe the design of topographies that belong to a certain class, which is defined by selecting the top and bottom performing surfaces from the rank. Images will provide an idea of what type of descriptors are suitable for this analysis.
2. There are three different sets of Topo descriptors that are combined in one set, and can be distinguishes based on the Descriptor name, they are:
    - **1. CellProfiler based Features**, these features were calculated by CellProfiler from the black and white images of the topographies. The description of these features can be obtained from the CellProfiler Documentation https://cellprofiler.org/manuals/. After CellProfiler the descriptive statistics were calculated on the features in R. The descriptor name contains the type of images that were used, CellProfiler Feature, and type of descriptive statistics, as shown below. Three subtypes of images were used as input into CellProfiler, which are:
        - *1.1. Pattern:* Black and white images with pillars in white and the bottom in black.

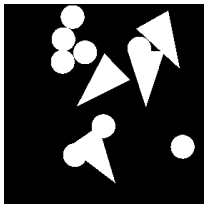

Figure 1. Pattern from *FeatureIdx* 14.

The typical descriptor name for this type is “Pattern_AreaShape_Area_sd”. Where “Pattern” indicates that a pattern image was used, “AreaShape_Area” is a CellProfiler Feature, which according to the documentation, is the number of pixels in the objects. As part of the pipeline, CellProfiler recognized all separated pillars (white areas, separated by black) as separate objects and reported area for each of these objects. The number of pillars in this example is 8 (Figure 1). In this particular case we found a Standard Deviation (SD) of separated pillars areas, which is reflected in a column name suffix - “_sd”.

- *1.2. Space:* Images of space were obtained by merging inverted Pattern images in 3 different combinations (Figure 2), to represent gaps between pillars as white and processed in CellProfiler and R in the same way as described for pillars.

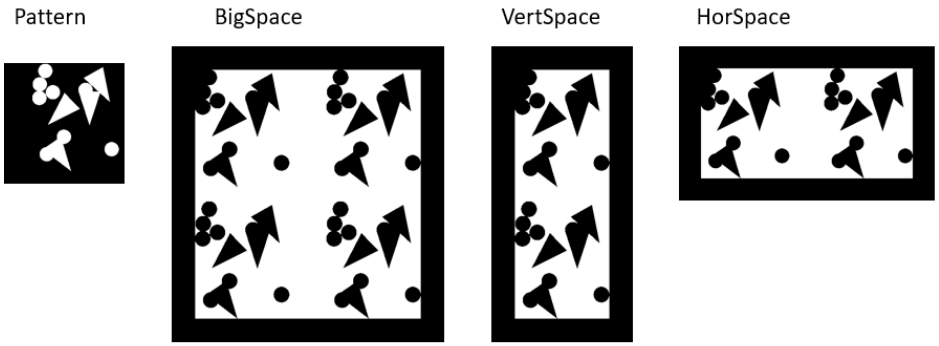

Figure 2. Original image of the pattern, and three types of spaces generated from it.

- 1.3. Texture. Here we loaded to CellProfiler images of the whole TopoUnit and measured “Texture”. Example of loaded images shown in Image 3.

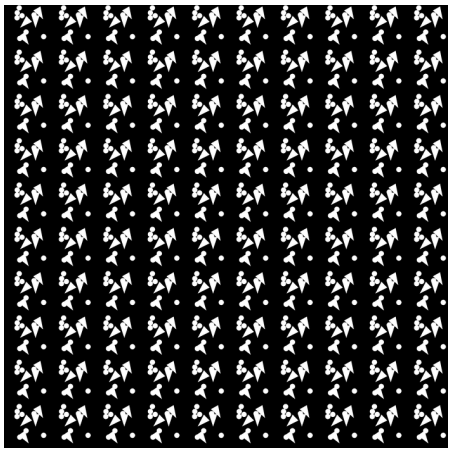

Figure 3. Type of images that were used to calculate Texture parameters.

- **2. Features using inscribed Circles** (Figure 4). For quantification of the spacing between pillars, binary images (Figure 1) were inverted and replicated across a larger area. We further employed the MaxInscribed Circles algorithm as described here https://imagej.net/Max_Inscribed_Circles to identify the size and number of circles that can be fitted in the gap between pillars. The algorithm was looped until a circle diameter smaller than 3 pixels is found. The typical column name for these features: "InsCirclRadius_sd"

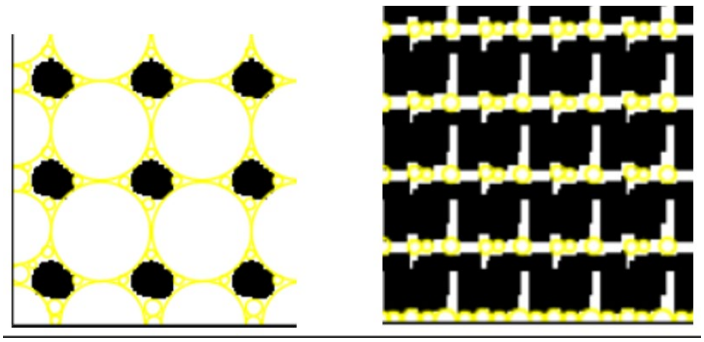

Figure 4. Example of the Inscribed Circles, in a gap between pillars.

- **3. Kamiel Features:** these features were developed by Kamiel Cornelissen and described in his Master Thesis. The typical name of these features is Kamiel_FCP. The master thesis is available upon request.

3. The model is not aware of the nature of descriptors, so you need to remove descriptors that do not make sense, for example, the filename of images (which was removed from the data in the template).
4. Features that the model used to separate two classes, top and bottom, are not necessarily useful to the interpretation of the phenomenon. An excellent resource to learn more about interpretability of the machine learning models https://christophm.github.io/interpretable-ml-book/
5. Inspiration for the Machine Learning analysis can be found on the Kaggle website: https://www.kaggle.com/notebooks
6. In this tutorial, we have used the SHAP package for model interpretability. The excellent primer on the application of this package can be found here https://evgenypogorelov.com/multiclass-xgb-shap.html
7. In this tutorial, we extensively use functions from Sklearn, which is a current standard for machine learning tasks in Python. Its website contains a lot of educational materials about ML https://scikit-learn.org/stable/

4 Required materials

4.1 Workplace

This SOP can be performed in the office or home on your Laptop/Desktop.

4.2 Requirements

1. Completion SOPs 4.1 to 4.5
2. Understand the following concepts:

- Binary Classification, Machine Learning, Supervised Learning https://europepmc.org/article/med/28693857
- Classification with trees https://scikit-learn.org/stable/modules/tree.html#classification
- Introduction to SHAP algorithm https://github.com/slundberg/shap
- Train test split https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation
- AUC curve https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics

5 Procedure

5.1 Load the required modules

1. **When you completed SOP 4.1 all modules should load without raising an error. If a module cannot be loaded, please go back to SOP 4.1 and make sure you installed all modules.**

5.2 Load the input data

1. Open the Jupyter notebook named "4_Train_MLModel.ipynb" to start and train the machine learning (ML) model.
2. The necessary data files are loaded in *Step 2* of the notebook, including the feature images and the *TopoDescriptors*. The feature images will be used to get a visual impression of the top and bottom ranked surfaces, whereas the *TopoDescriptors* are used in the machine learning model.
3. When not all files can be located, a warning will be displayed. There are two possible solutions:
    a. You did not rank the features (SOP 4.5) or moved the file from its original location. It is important to keep the file *"Ranking.csv"* in the folder *"/DataAnalysis/"*.
    b. You did not remove the outliers (SOP 4.4) and created the file "*imageOutliersRemoved.csv"* or removed it from its original location. Again, it is important to keep this file in the folder "/DataAnalysis/".

5.3 Select the feature of interest and visualize the topographies

1. When *Step 2* did not return any errors, you can move on to *Step 3 "Select Features for the machine learning "*.
2. In *Step 3*, we will have a visual inspection of the top and bottom ranked topographies. This way, we already get an idea of what type of topographies might play an important role. The function *surfaceDesignTopAndBottom* uses the following output:
    a. *sortedRanking* - The data variable with results of the ranking in sorted order, from lowest to highest.
    b. *featureOfInterest* - Measurement that is used for ranking, that contains, for example, number of cells or the expression of the protein of interest.
    c. *number*, which is a number of surfaces per class to show.
3. The output of this function is a plot, with on the left side the top ranked surfaces and on the right side the bottom ranked surfaces.
4. Every image shows an area of the surface that corresponds to 56x56 microns, where the total area of every surface is 280 x 280 microns.

5. On top of every image, the *FeatureIdx* is specified and the value of the *featureOfInterest.*

5.4 Select the descriptors

1. In *Step 4 "load and visualize descriptors data ",* you are asked to specify the descriptors you want to consider while training and testing the machine learning model. Here you can choose a smaller selection based on a specific research question (For instance you have an interest in cell shape features, or actin organization) or include all descriptors, or anything in between. The data from CellProfiler contains some metadata information, such as height and width of images or Group index, which are not meaningful for our application and should always be excluded.
2. Texture parameters, even though they can produce a more accurate model, are hard to interpretate and you should decide if you want to include them.
3. The function to select the descriptors takes three inputs:
    a. *rank* - a calculated rank data.
    b. *filterMetric* - refer to a column name that contains either p-value or SNR.
    c. *valuesRangeToInclude* - specify the inclusive range of values that should be preserved during the filtering step if required.

5.5 Training of the Machine Learning (ML) model

1. Based on the ranked surfaces and the descriptors you have selected in *step 4*; you can train a model to predict the most important descriptors (*Step 5: train machine learning models)*.
2. First, in *Step 5.1*, you will combine the results of the ranking and the selected descriptors to create two output variables X and Y. These variables will be used as the input for your ML model.
3. X contains the scaled descriptor information
4. Y contains the class information (binary), where 1 corresponds to the top ranked surfaces and 0 to the bottom ranked surfaces.
5. To train and test the model, split the data into a training and testing set (*Step 5.2)*. By default, the function will split the data into 25% test data and 75% train data.
6. The classification model is the *XGBClassifier().* This is an eXtreme Gradient Boosting algorithm based on a gradient boosted decision trees algorithm.
7. By default, the XGBClassifier() has a set of predefined parameters but optimization of those parameters is needed based on our dataset characteristics. This is performed by using a *RandomizedSearchCV* (*Step 5.4*).
8. You can print the most optimal set of parameters in *step 5.5* and compare them to the default parameters if you like (see manual of the *XBGBoost* module).
9. Train and test the ML model with the optimized set of parameters (*Step 5.6).*
10. Print the accuracy values calculated for the trained and tested data. Print out accuracy values calculated on trained and tested data. If the accuracy is high (towards 1), it suggests that the trained model describes the data well. If the accuracy metric is around 0.5, it means that model is not better than a random guess for binary classification.
11. Plot the AUC-ROC curve (*Step 5.8)* to visualize the performance of your model. ROC is a probability curve and AUC represent the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. The higher the AUC, the better the model is at predicting the data.

12. Step 5.9 is used to visualize the 10 most important features for the model. This will create a plot with on the y-axis the top 10 features and on the x-axis the feature weight.

    Warning: the choices you have made during the ML testing and training phase do influence the model's outcome. When you are unfamiliar with overfitting or underfitting, read upon those issues and adjust the workflow where needed.

5.6 Use SHAP module to explain predicted model

1. To understand the model, the Shapely values are calculated. These values express the contribution that each feature has on the output of the model.
2. First, define the *explainerDefinition* to calculate the Shapely values. This function performs two tasks:
    i. It will calculate the Shapely values based on a *TreeExplainer* model.
    ii. It will create an explanation force plot, showing the positive Shapley values in red and the negative ones in blue for the selected surfaces.
3. A summary plot can be generated in *Step 6.2* (Figure 5). The summary plot combines feature importance with feature effects. Each point on the summary plot is a Shapley value for a feature and an instance. The position on the y-axis is determined by the feature and on the x-axis by the Shapley value. The colour represents the value of the feature from low to high. Overlapping points are jittered in y-axis direction, so we get a sense of the distribution of the Shapley values per feature. The features are ordered according to their importance.

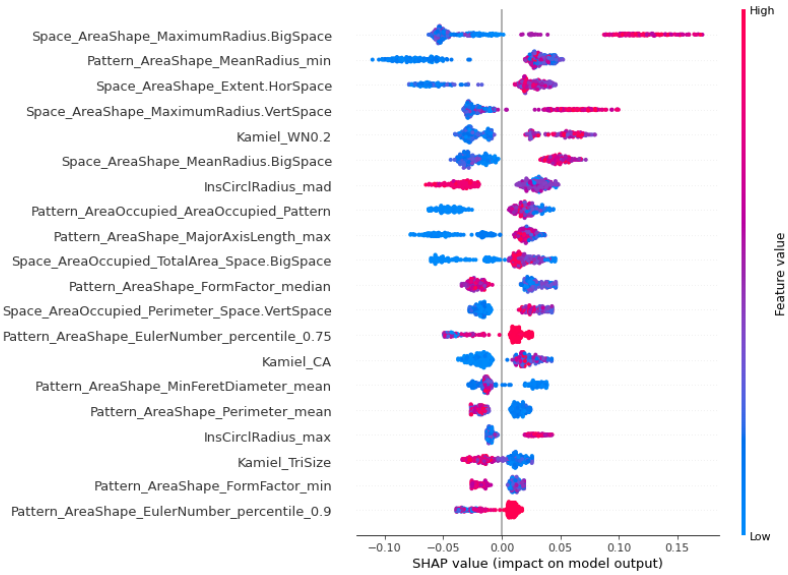

Figure 5 SHAP summary plot

4. Finally, a scatter plot can be generated to plot two features of interest against each other. If two features are important to the model, you would expect to see a separation based on the 2 classes.

5.7 Save the notebook

1. Once you finished your analysis, you can save the notebook by clicking on the *Save* icon. This way, all results and figures are stored to be used in the future