Machine learning for the meta-analyses of microbial pathogens' volatile signatures

by Susana I.C.J. Palma *et al.*

(doi:10.1038/s41598-018-21544-1)

Supporting Tutorial: machine learning scripts to automatically classify pathogens in a pathogen-VOC database

Requirements

Pathogen-VOC database

An excel file with the same structure as the supporting dataset that accompanies the paper .

Script files

- `voc_proc_complete2017.py`
- `Identification_with_LOCV_to_share.ipynb`
- `Verification_to_share.ipynb`

Download these files to a folder in the desktop. Add your Pathogen-VOC database (excel file) to this folder.

Software

"Anaconda 5.1" is a free and open source distribution of Python programming language, which includes:

- "**Spyder**", an interactive environment for Python programming
- **"Jupyter notebook"**, a web application to share documents that contain live code
- the data science python libraries necessary to run the scripts that we built.

Make sure you install the **Anaconda 5.1** package with **Python 3.6** version. (download link: https://www.anaconda.com/download/)

Procedures

A. Structuring input data in the form of a binary pathogen-VOC matrix.

1. Organize your data in an excel table structured as the supporting dataset of our paper (available online - https://www.nature.com/articles/s41598-018-21544-1#Sec15).
2. Save the table as text file ("**.txt**" – text separated by tabs) in the same folder of the downloaded scripts.
3. Replace the "**.txt**" termination by "**.tsv**".
4. Open the script `voc_proc_complete2017.py` in **Spyder**, write the name of your table file in the indicated line and run the script (Figure 1).
5. The result of the script is a **.tsv** file called **"input_for_classifiers.tsv"**, organised as showed in Figure 2.

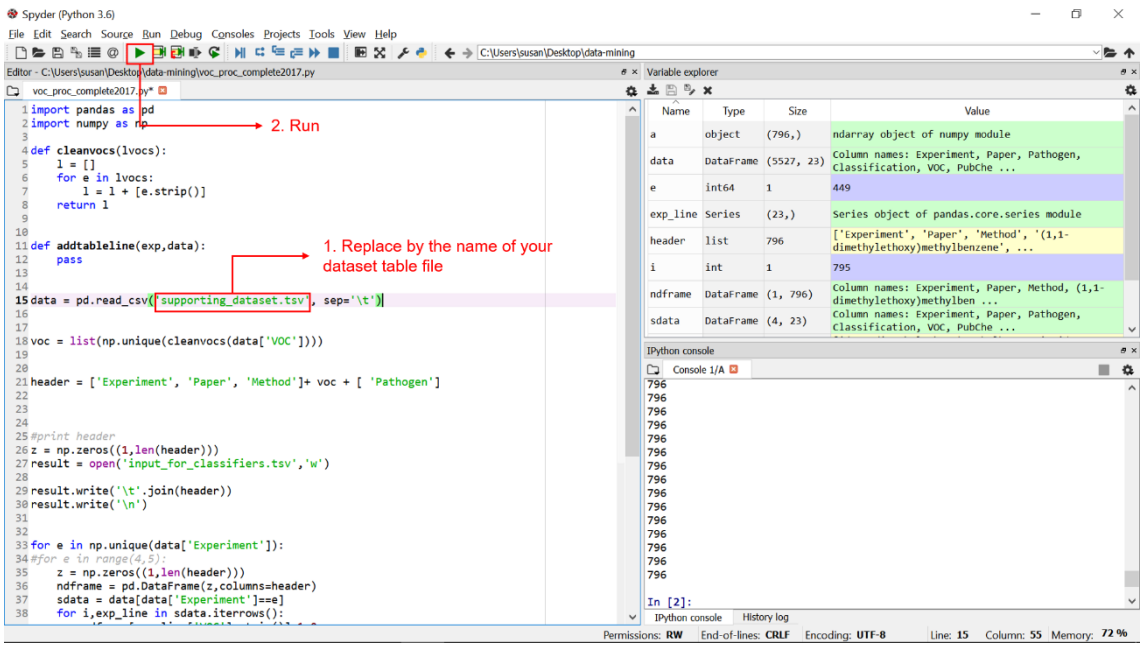

Figure 1. Spyder environment with the script `voc_proc_complete2017.py` open. Follow the instructions in red to run the script.

Experiment	Paper	Method	VOC 1	VOC 2	VOC 3	⋯	VOC 791	VOC 792	Pathogen
1	1	SIFT – MS	1	0	1	…	0	1	*P. aeruginosa*
2	1	SIFT – MS	0	0	1	…	0	0	*S. aureus*
⋮	…	…	…	…	…	…	…	…	⋮
449	62	GC – MS	1	1	0	⋯	1	0	*E. coli*

Figure 2. Schematic representation of the layout of the file "**input_for_classifiers.tsv".** This is the binary pathogen-VOC matrix that will be used as input for the machine learning classification scripts. Each line is one example to train the classifiers.

B. Running the classification in multiclass (identification) mode, with "leave-one-out" cross-validation.

1. Open **Jupyter notebook**. A Firefox tab opens automatically. In the open tab, select the folder "desktop" and then the folder where the downloaded scripts are located.
2. Open the script "`Identification_with_LOCV_to_share.py`".
3. Run the cells sequentially:
   - **3.1.** Cell 1: selects the data that will be the input for the Support Vector Machines (SVM) classifier (selects only the pathogens for with there are more than 4 examples in the table "input_for_classifiers.tsv"). Returns the names of the selected pathogens.

   - **3.2.** Cell 2: selects the best set of VOCs to classify the pathogens in the multiclass mode (identification) by applying a SVM-based classifier iteratively and a sequential forward feature selection process. In the end of each iteration, this cell returns:
     - Number of the iteration
     - Current classification accuracy
     - List with the numbers of the VOCs that were selected until the current iteration, which are responsible by the current classification accuracy.

     The algorithm in cell 2 is pre-set to perform 35 iterations. Note that the classification accuracy may stabilise earlier, but the algorithm continues to add VOCs to the list until the 35th iteration. At the 35th iteration it stops. The best VOCs (those that better contribute do discriminate between pathogens) are those added until the accuracy stabilises (**Figure 3**).

```
15
0.758928571429
[53, 332, 572, 32, 333, 618, 629, 102, 234, 601, 393, 600, 531, 676,
523, 392]
o #############
16
0.761904761905
[53, 332, 572, 32, 333, 618, 629, 102, 234, 601, 393, 600, 531, 676,
523, 392, 699]
o #############
17
0.764880952381
[53, 332, 572, 32, 333, 618, 629, 102, 234, 601, 393, 600, 531, 676,
523, 392, 699, 48]
o #############
18
0.764880952381
[53, 332, 572, 32, 333, 618, 629, 102, 234, 601, 393, 600, 531, 676,
523, 392, 699, 48, 0]

(…)

34
0.764880952381
[53, 332, 572, 32, 333, 618, 629, 102, 234, 601, 393, 600, 531, 676,
523, 392, 699, 48, 0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17]
```

Figure 3. Example output of the identification mode classification (cell 3 of the script `Identification_with_LOCV_to_share.py`). Iteration 17 is highlighted to show that a classification accuracy of 76.5% was achieved with the 18 VOCs listed between brackets. From iteration 17 until iteration 34 the addition of more VOCs (0, 1, 2, 3, …, 17) did not improve the classification accuracy. Therefore, the best set of VOCs for the multiclass classification is that containing the 18 VOCs highlighted in green.

NOTE: the execution of cell 2 is computationally heavy, specially if your input matrix is big. For example, to obtain the results in Table 1 of the paper, it took approximately 24h (processor: intel core i7-5500 CPU, 2.40GHz, 8.0 GB RAM).

3.3. Cell 3: returns the names of the VOCs selected in cell 2 (**Figure 4**).

3.4. Cell 4: calculates the confusion matrix (Table 2 of the paper), which allows to assess the quality of the predictions made by the classifier using the best VOCs.
NOTE: you need to manually write the list of best VOC numbers in the proper line of cell 4 for the confusion matrix to be properly calculated (**Figure 5**).

3.5. Cell 5: calculates the performance (sensitivity and precision) of the classifier for each pathogen, as well as the respective average and standard deviation values (**Figure 6**).

```
In [8]: print (sX.columns[features])

        Index(['1-decanol', '3-methylbutanal', 'ethyl acetate',
               '1,3,5-trimethylbenzene', '3-methylbutanoic acid', 'indole',
               'isopentanol', '1-undecene', '2-methylbutanal', 'gama-butyrolactone',
               '4-methylphenol', 'furan', 'cymol', 'methyl nicotinate',
               'cyclohexanone', '4-methylpentanoic acid', 'n-butyl acetate',
               '1-butanethiol',
               '(1-methylethenyl)benzene', '(E)-2-butene', '(E)-2-methyl-2-butenal',
               '(E)-3-nonen-2-one', '(E)-3-octenal', '(E)-3-penten-2-one',
               '(E)-methylthio-1-propene', '(E,E)-2,4-decadienal',
               '(Z)-1-methylthio-1-propene', '(Z)-2-butene', '(Z)-2-heptenal',
               '(Z)-2-pentenol', '(Z)-7-tetradecen-1-ol',
               '(dimethylamino)acetonitrile', '(methylthio)methylbenzene',
               '1,1,2,2-tetrachloroethane'],
              dtype='object')
```

Figure 4. Names of the VOCs selected in cell 2. The best VOCs to classify the pathogens in the input matrix are those highlighted in green (corresponding to the VOC numbers returned by iteration 17, shown in Figure 3).

```
In [12]: ###### Cell 4 ########


from sklearn import svm, cross_validation
from sklearn.svm import SVC
from sklearn.cross_validation import LeaveOneOut, StratifiedKFold


# The following list of features numbers corresponds to best VOCs reported in the paper. In the case of a new input
# data matrix, run the previous cells first and, then, replace this list with the list of best features numbers
#returned by cell 2

f = [53, 332, 572, 32, 333, 618, 629, 102, 234, 601, 393, 600, 531, 676, 523, 392, 699, 48]

from sklearn.metrics import confusion_matrix

clf.fit(sX[sX.columns[f]],sY) #trains the classifier using only the best features.
pY=clf.predict(sX[sX.columns[f]]) #tests the classifier using only the best features
mc = confusion_matrix(sY,pY)

print('confusion matrix:', '\n')
print (mc) #presents the confusion natrix


confusion matrix:

[[ 8  0  0  0  0  0  0  0  0  0  0]
 [ 0  6  0  0  0  0  0  0  0  0  0]
 [ 0  0 60  0  2  0  2 19  1  0  0]
 [ 0  0  1  5  0  0  0  0  1  0  0]
 [ 0  0  0  0 16  0  0  6 11  0  0]
 [ 0  0  0  0  0  9  0  0  0  0  0]
 [ 0  0  0  0  1  0  6  5  2  0  0]
 [ 0  0  2  0  0  0  0 114  2  0  0]
 [ 0  0  2  0  0  0  0 11 29  0  0]
 [ 0  0  0  0  0  0  0  1  1  3  0]
 [ 0  0  1  0  0  0  0  4  0  0  5]]
```

Figure 5. To calculate the confusion matrix, you need to indicate which features (VOCs) the classifier shall use to make the predictions. Write the list of best features (those selected in cell 3) in the line highlighted in red.

Figure 6. Classification Performance matrix, with the Sensitivity (left column) and precision (right column) of the classifier to predict the identity of the pathogens from information on the presence of the best VOCs. Each line of the matrix corresponds to the performance of the classifier regarding one pathogen (pathogen list returned by cell 1).

C. Running the classification in dual class (verification) mode, with "leave-one-out" cross-validation.

1. Open **Jupyter notebook**. A Firefox tab opens automatically. In the open tab, select the folder "desktop" and then the folder where the downloaded scripts are located.
2. Open the script "`Verification_to_share.py`".
3. Run the cells sequentially:
   - **3.1.** Cell 1: selects the data that will be the input for the Support Vector Machines (SVM) classifier (selects only the pathogens for with there are more than 4 examples in the table "input_for_classifiers.tsv"). Returns the names of the selected pathogens and dimensions of the input pathogen-voc matrix.

     NOTE that cell 1 does the same selection as cell 1 from the multiclass script, described in **B.**
   - **3.2.** Cell 2: selects the best set of VOCs to classify the pathogens in the dual class mode (verification) by applying a SVM-based classifier and a sequential forward feature selection process for each "pathogen *vs.* other pathogens" case. For each pathogen, it returns:
     - Pathogen name

- Number of experiments involving the pathogen (inputs to train and test the classifier)
- Base and computed classification accuracy, computed classification sensitivity and precision. (base accuracy corresponds to the non-informed/random classification result)
- Set of VOCs that separate the pathogen from the others with the reported classification performance (Figure 7).

```
########
Pseudomonas aeruginosa
Examples:
118
Base Accuracy:
0.64880952381
Computed Accuracy:
0.928571428571
Computed Sensitivity:
0.872881355932
Computed Precision:
0.919642857143
selected Features:
['hydrogen cyanide' '2-aminoacetophenone' 'ammonia (dimer)' '1-pentanol'
 '1-undecene' '2,4-dimethyl-1-heptene' '1-decanol' '2-propanol']
```

Figure 7. Example a verification mode classification result.