Tutorial for „Using Jupyter Notebooks for Re-training Machine Learning Models"

A Jupyter Notebook is provided to generate/retrain classification models for six transporter proteins (BCRP, BSEP, OATP1B1, OATP1B3, MRP3, P-gp). Four different classifiers can be selected without an extensive descriptor selection and hyperparameter search as they have been pre-selected. An in-house data set (provided by the user) can be combined with the available UNIVIE data set(s) to extend the chemical space of the model.

Getting Started

- Download files from the repository

File name	**Description**
standardise.py	Standardizer
retraining_env.yml	Virtual Environment
RDKIT_Descriptors.txt	List of selected descriptors
Data folder:	
*NameOfTransporter*_Univie.sdf	Training set provided by the University of Vienna
*NameOfTransporter*_ChEMBL28.sdf	Test set provided by the University of Vienna

- Install Anaconda on your device
- Install/activate virtual environment (retraining_env.yml)

  conda env create -f retraining_env.yml

  conda activate retraining_env
- Start the Jupyter Notebook (If you don't have any experience with using Jupyter Notebook, please have a look at "Installation Guides/Tutorials" before starting.)

Installation Guides/Tutorials

Download Repository:
https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository

Anaconda: https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html

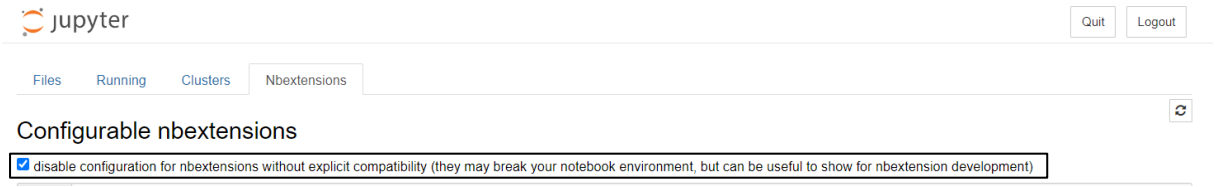
Virtual Environment:
https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

How to use Jupyter Notebooks: https://jupyter-notebook.readthedocs.io/

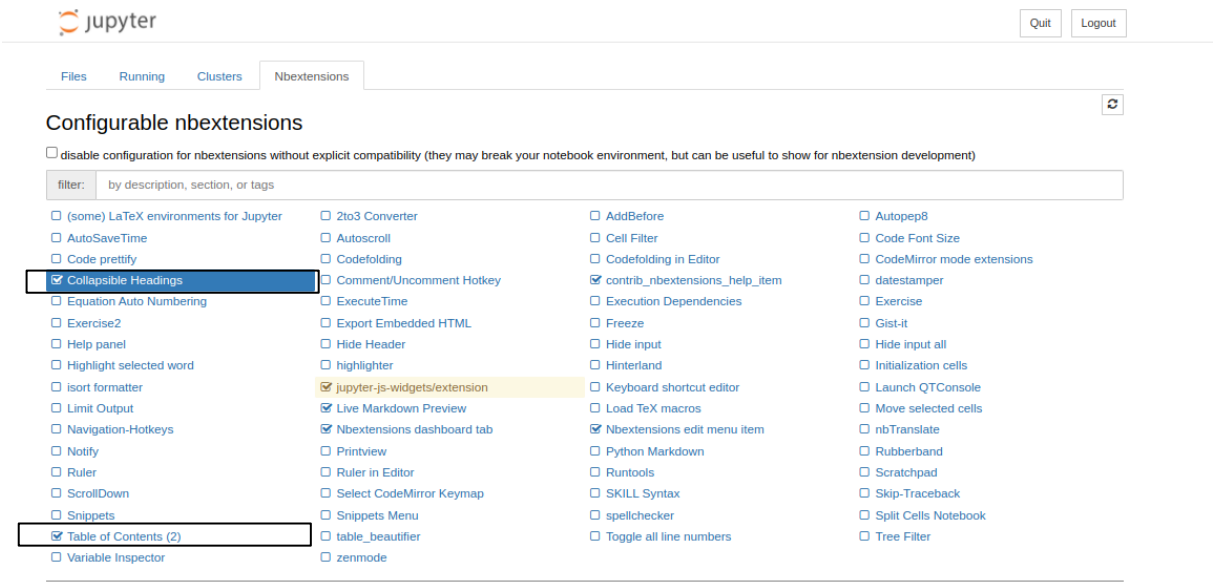
Jupyter Notebook

Please ensure, before you open the Jupyter Notebook within “Files”, that the “nbextensions” named Collapsible Headings and Table of Contents are checked. This allows to collapse the cell code based on the heading as well as a representation of a table of content for a better depiction and better handling of the JN.

To do so, uncheck the configuration for nbextensions without explicit compatibility.



This will lead you to the the nbextensions menu where you will be able to put a check on the collapsible headings and the table of content.



Once you set the tags, go back to “Files” and open the the Jupyter Notebook.

There are four sections available in the code:

1. Data Upload & Data Addition

2. Data Set Preparation for ML Task

3. Applicability Domain

4. Model Generation & Evaluation

Please follow the guide below as described, before running the notebook. A star next to the title/header indicates that an action from your side is required.

Procedure:

“Step 1: Data Upload & Data Addition”

The “Data Collection”- step includes some code cells to extend the provided data set as well as prepare it prior to model building. First, you have to specify which UNIVIE transporter dataset should be used. This can be done by changing the file name at the variable Univie_Data. If you want to add additional data, you have to change the file name at the variable “Intern_Data”. Please, be aware that only the SDF format can be used.

UNIVIE Data*

UNIVIE data refers to collected, curated data that has been provided by the University of Vienna for the purpose of training/retraining ML models for six different endpoints:

- BCRP
- BSEP
- OATP1B1
- OATP1B3
- MRP3
- P-gp

Datasets that will be used within this JN such as the training set and the test set have to saved within the folder /data.

You can select the dataset for the endpoint by changing the file name at the variable “UNIVIE_Data”: e.g.: "data/BSEP_Univie.sdf" --> "data/BCRP_Univie.sdf"

```
# Please add the name of the UNIVIE file
Univie_Data = "data/BSEP_Univie.sdf"
```

Intern Data*

You can include your own in-house dataset for the previous selected endpoint by changing the file name at the variable “Intern_Data” to the name of your dataset.

Keep in mind that the path to your dataset has to be properly defined.

```
# Please add the name of your file
Intern_Data = "data/BSEP_ChEMBL28.sdf"
```

!!! If you don't want to add additional data, you can skip Step 1!!!

You can modify the code to customize Step 1, but it is not mandatory. It automatically, calculates important parameters such as SMILES and InChIs as well as removes stereoisomers and duplicates. At the end of this step an SDF with the training set will be generated. Each code cell in the Jupyter Notebook is annotated with a brief explanation.

"Step 2: Data Set Preparation for ML Task"

At cell "Read SDF":

Add the name of your training set under "Training set*".

Training set*

The training set is uploaded into the JN at the variable "molecules" via the RDKit functionality for working with molecular file format.

You can select the training set for the endpoint by changing the file name at the variable "molecules".

```
# Please add the name of your training set
molecules = Chem.ForwardSDMolSupplier("data/BSEP_Univie.sdf", sanitize=False)
```

Add the name of your test set under "Test set*".

Test set*

The test set is uploaded into the JN at the variable "test_molecules" via the RDKit functionality for working with molecular file format.

You can select the test set for the endpoint by changing the file name at the variable "test_molecules".

```
# Please add the name of your test set
test_molecules = Chem.ForwardSDMolSupplier("data/BSEP_ChEMBL28.sdf", sanitize=False)
```

Once the training and test set are chosen, code cells within "Step 2: Data Set Preparation for ML Task" can be executed. Each compound within the training and test set is standardized and checked if the standardization was successful. 70 RDKit Descriptors are then loaded from the provided text file into the Jupyter Notebook. The descriptor calculation for each compound within the training and test set is then performed. The number of compounds with successfully calculated RDKit descriptors as well as the corresponding activity values are displayed. Additionally, the number of inhibitors/non-inhibitors is checked as well as the total amount of compounds. NaN values are replaced with zeros to avoid potential errors.

Further code cells in this section are available for customization such as the standardization step as well as the selection of the descriptors. But it is not mandatory to modify them.

“Step 3: Applicability Domain”

The Applicability Domain is assessed to check if the test set is within the chemical space of the model. If this is the case, the compound is marked as in domain, otherwise it is marked out of domain. A file is generated namely:” Outlier_Compounds.sdf” which includes all compounds which are out of domain.

At “Verify that the test set is within the chemical space of the model” fill in the name of your test set file.

Verify that the test set is within the chemical space of the model*

You can select the test set for the AD by changing the file name at the variable “Test_Data".

```
Test_Data = "data/BSEP_ChEMBL28.sdf"
```

“Step 4: Model Generation & Evaluation”

Add the name of the standardized training data set and the name of the standardized test set file which is created in “Step 2: Data Set Preparation for ML Task”.

Please add the name of the used training set*

You can select the dataset for the creation of the ML models by changing the file name at the variable “Train_Stand_SDF”.

```
Train_Stand_SDF = "data/BSEP_standardised_train_molecules.sdf"
df_Data = PandasTools.LoadSDF(Train_Stand_SDF)
```

Please add the name of the used test set*

You can select the dataset for testing the ML models by changing the file name at the variable “Test_Stand_SDF”.

```
Test_Stand_SDF = "data/BSEP_standardised_test_molecules.sdf"
df_Test_Data = PandasTools.LoadSDF(Test_Stand_SDF)
```

Since some of the compounds could be discarded during the standardization process, a new sdf file with the standardized test set is created after the standardisation step. This file is used for the comparison between classification values and predicted values.

"Step 4: Model Generation & Evaluation" provides the different machine learning approaches, namely Logistic Regression, Support Vector Machine, Random Forest as well as k-nearest neighbor. No customization is necessary at that step.

Each classifier is following the same pattern:

- Creation of the classifier with its optimized hyperparameters.
- Save model as a pkl file.
- 10-fold cross validation.
- Calculation of the most important metrics
- Interactive view of results
- Identification of incorrectly predicted test compounds.
- Save wrongly predicted compounds.
- Evaluation of the test set.
- Comparison of predicted classification values with actual classification values.

Use Case for Bile Salt Export Pump (BSEP):

This section allows a better understanding of how to use the JN for the prediction of inhibitory activity.

Use case name:	Retraining BSEP ML models & prediction of inhibitory activity towards BSEP
Univie data:	BSEP_Univie.sdf
Intern data:	BSEP_ChEMBL28.sdf
Training set:	BSEP_Univie.sdf
Test set:	BSEP_ChEMBL28.sdf
Description:	
Step 1: Data Upload & Data Addition	In this use case, BSEP data provided by the UNIVIE can be combined with own in-house data. The addition of new compounds allows the creation of a new training set that can be used for the retraining of ML models within the JN. ChEMBL28 data is used for the simulation of in-house data which is as well provided by UNIVIE. However, if an in-house BSEP dataset in form of an SDF-file exists, you can change the name of your file in the

	cell “Intern Data*” at the variable “Intern_Data”. Your dataset must be added to the folder *data* to work. Importantly, the star next to the header indicates that data can be added/changed. Once data is added InChIs, SMILES and InChIKeys are calculated for additional information about the molecules followed by duplicate check, comparison of classification values of removed duplicates, actual duplicate removal and the selection of important columns. A new training set is created as an SDF-file. If you don’t want to add additional data, you can skip this step.
Step2: Data Set Preparation for ML Task	A BSEP training set and a BSEP test set are needed for the creation of the ML models and the evaluation of the models. Both datasets must be prepared prior to model training and model evaluation. For doing so, the training set and the test set can be added/changed at the cell “Training set” and cell “Test set” within “Step 2: Data Set Preparation for ML Task”. Once the datasets for BSEP are added the standardization can be performed. The standardized training set and standardized test set are saved as SDF-files in the folder *data*. The number of compounds that were able to be standardized is displayed for both, the training and test set. Next, the 70 descriptors that have been found to be important for describing the six endpoints such as BSEP are checked and calculated for each compound within both standardized datasets. The results of the descriptor calculation and the classification value are displayed for both datasets as well as the number of actives and inactive compounds within both data sets. Additionally, NaN values are replaced with zeros for avoiding potential errors.
Step 3: Applicability Domain	The JN offers a verification if the BSEP test set is within the chemical space of the model. The test set for the AD can be added/changed at the cell “Verify that the

	test set is within the chemical space of the model*" at the variable "Test_Data". The AD is calculated and displayed as a depiction for a general representation if the BSEP test set is inside or outside the AD. In addition, the BSEP test set is checked if it is within the chemical space of the model and an SDF-file is generated which includes all compounds that are out of domain.
Step 4: Model Generation & Evaluation	In the cell "Step 4: Model Generation & Evaluation" four different classifiers are used for model generation: • Logistic Regression (LR) • Support Vector Machine (SVM) • Random Forest (RF) • k-nearest neighbor (KNN) The scikit-learn Python library (version 0.24.2) implementations are used to train binary classification models for BSEP. The used BSEP training set and the used BSEP test set can be added/changed at the cell "Please add the name of the used training set*" and the cell "Please add the name of the used test set*" at the variable "Train_Stand_SDF" and "Test_Stand_SDF". The standardized training set and the standardized test set are important for the comparison of resulted predictions and classification values of the test set as well as if the compounds within the test set are within the AD. The performance of the models is tested for each classifier, and following statistical metrics are used: • Accuracy Score • Sensitivity Score • Specificity Score • Balanced Accuracy (BA) Score • F1 Score • Area Under the Curve (AUC) Score • Precision Score

	• Matthews Correlation Coefficient (MCC) Score

What will be saved and what can you analyze:

If you provide additional data and use it to combine it with the UNIVIE data set, a new file will be created, namely "*NameOfTransporter*_Training_Set.sdf". Containing the new compounds as well as the UNIVIE data. The file is stored in the folder *data*.

Generated files including results are stored in the folder *results*.

At section "Applicability Domain", all compounds which are out of domain will be saved in the file "*NameOfTransporter*_Outlier_Compounds.sdf" and in the folder *results*. The structure of the compounds will be also visualized in the notebook.

For each classifier, you will receive statistical performance metrics for the evaluation of the models and the models will be saved as pkl-files in the folder *results*:

- "*NameOfTransporter*_model_LR.pkl"
- "*NameOfTransporter*_model_SVM.pkl"
- "*NameOfTransporter*_model_RF.pkl"
- "*NameOfTransporter*_model_KNN.pkl"

Incorrectly predicted compounds will be saved as SDF_files in the folder *data*:

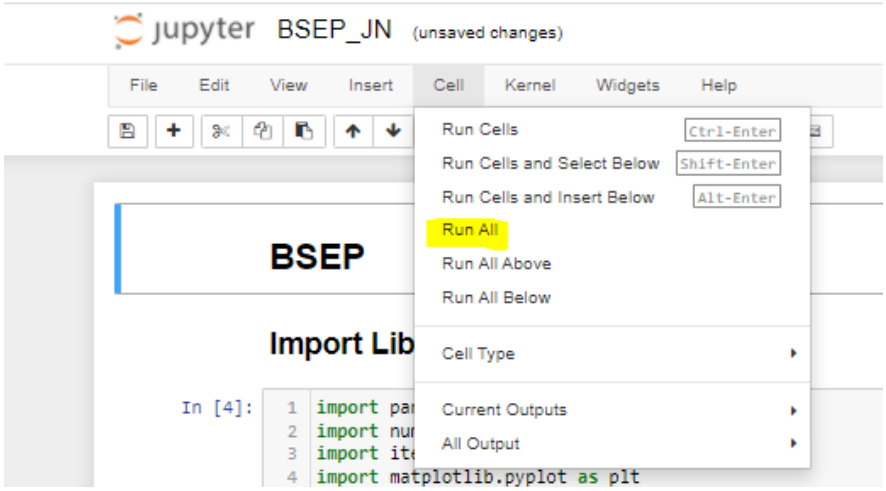
- "*NameOfTransporter*_WP_LR_Compounds.sdf"
- "*NameOfTransporter*_WP_SVM_Compounds.sdf"
- "*NameOfTransporter*_WP_RF_Compounds.sdf"
- "*NameOfTransporter*_WP_KNN_Compounds.sdf"

These compounds can also be visualized within the Jupyter Notebook.
Statistical metrics from the cross validation and the external test set validation can be selected separately.

Additionally, an SDF-file and a CSV-file is generated for each classifier including the information from the prediction and the applicability domain run in the folder *results:*

- "*NameOfTransporter*_LR_Test_Prediction.sdf"
- "*NameOfTransporter*_LR_Test_Prediction.csv"
- "*NameOfTransporter*_SVM_Test_Prediction.sdf"
- "*NameOfTransporter*_SVM_Test_Prediction.csv"
- "*NameOfTransporter*_RF_Test_Prediction.sdf"
- "*NameOfTransporter*_RF_Test_Prediction.csv"
- "*NameOfTransporter*_KNN_Test_Prediction.sdf"
- "*NameOfTransporter*_KNN_Test_Prediction.csv"

You can run the whole Jupyter Notebook now and analyze the results in the current dropdown fields as in the depiction below:


jupyter BSEP_JN (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run Cells Ctrl-Enter
Run Cells and Select Below Shift-Enter
Run Cells and Insert Below Alt-Enter
Run All
Run All Above
Run All Below
Cell Type
Current Outputs
All Output
BSEP
Import Lib
In [4]:
1 import pa
2 import nu
3 import it
4 import matplotlib.pyplot as plt