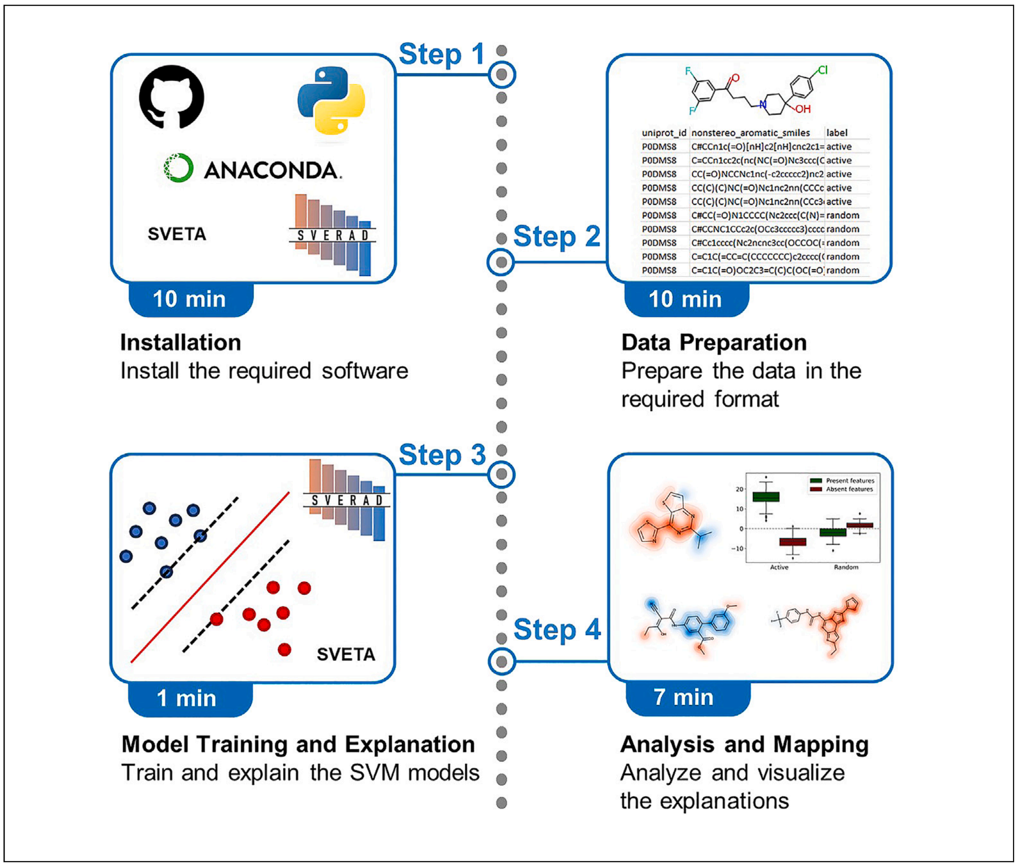
Protocol

Protocol to explain support vector machine predictions via exact Shapley value computation


Step 1
ANACONDA.
SVETA
SVERAD
10 min
Installation
Install the required software
Step 2
10 min
Data Preparation
Prepare the data in the required format
Step 3
SVERAD
SVETA
1 min
Model Training and Explanation
Train and explain the SVM models
Step 4
Present features
Absent features
Active
Random
7 min
Analysis and Mapping
Analyze and visualize the explanations

uniprot_id	nonstereo_aromatic_smiles	label
P0DMS8	C#CCn1c(=O)[nH]c2[nH]cnc2c1=	active
P0DMS8	C=CCn1cc2c(nc(NC(=O)Nc3ccc(C	active
P0DMS8	CC(=O)NCCNc1nc(-c2ccccc2)nc2	active
P0DMS8	CC(C)(C)NC(=O)Nc1nc2nn(CCCc	active
P0DMS8	CC(C)(C)NC(=O)Nc1nc2nn(CCc3	active
P0DMS8	C#CC(=O)N1CCCC(Nc2ccc(C(N)=	random
P0DMS8	C#CCNC1CCc2c(OCc3ccccc3)cccc	random
P0DMS8	C#Cc1cccc(Nc2ncnc3cc(OCCOC(=	random
P0DMS8	C=C1C(=CC=C(CCCCCCC)c2cccc(	random
P0DMS8	C=C1C(=O)OC2C3=C(C)C(OC(=O	random

Shapley values from cooperative game theory are adapted for explaining machine learning predictions. For large feature sets used in machine learning, Shapley values are approximated. We present a protocol for two techniques for explaining support vector machine predictions with exact Shapley value computation. We detail the application of these algorithms and provide ready-to-use Python scripts and custom code. The final output of the protocol includes quantitative feature analysis and mapping of important features for visualization.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Andrea Mastropietro, Jürgen Bajorath

mastropietro@diag.uniroma1.it (A.M.)
bajorath@bit.uni-bonn.de (J.B.)

Highlights

Guidance on calculating exact Shapley values for SVMs using the Tanimoto kernel

Instructions for calculating exact Shapley values for SVMs using RBF kernels

Steps for using Shapley values to quantify feature importance and explain predictions

Details provided on the practical use of SVETA and SVERAD

Mastropietro & Bajorath, STAR Protocols 5, 103010
June 21, 2024 © 2024 The Author(s). Published by Elsevier Inc.
https://doi.org/10.1016/j.xpro.2024.103010

Protocol

Protocol to explain support vector machine predictions via exact Shapley value computation

Andrea Mastropietro[1,4,*] and Jürgen Bajorath[2,3,5,*]

[1]Deparment of Computer, Control and Management Engineering "Antonio Ruberti", Sapienza University of Rome, Via Ariosto 25, 00185 Rome, Italy

[2]Department of Life Science Informatics and Data Science, B-IT, LIMES Program Unit Chemical Biology and Medicinal Chemistry, Rheinische Friedrich-Wilhelms-Universität, Friedrich-Hirzebruch-Allee 5/6, 53115 Bonn, Germany

[3]Lamarr Institute for Machine Learning and Artificial Intelligence, Friedrich-Hirzebruch-Allee 5/6, 53115 Bonn, Germany

[4]Technical contact

[5]Lead contact

*Correspondence: mastropietro@diag.uniroma1.it (A.M.), bajorath@bit.uni-bonn.de (J.B.)
https://doi.org/10.1016/j.xpro.2024.103010

SUMMARY

Shapley values from cooperative game theory are adapted for explaining machine learning predictions. For large feature sets used in machine learning, Shapley values are approximated. We present a protocol for two techniques for explaining support vector machine predictions with exact Shapley value computation. We detail the application of these algorithms and provide ready-to-use Python scripts and custom code. The final output of the protocol includes quantitative feature analysis and mapping of important features for visualization. For complete details on the use and execution of this protocol, please refer to Feldmann and Bajorath[1] and Mastropietro et al.[2]

BEFORE YOU BEGIN

This protocol details the use of the *Shapley value-expressed Tanimoto similarity* (SVETA) and the *Shapley value-expressed radial basis function* (SVERAD) algorithms for the exact computation of Shapley values to explain support vector machine (SVM) predictions in chemistry. The SVETA approach was specifically designed for SVM models relying on Tanimoto similarity[3] (which is typically calculated in compound comparison) and the corresponding Tanimoto kernel, while SVERAD was developed for SVM models using the more generally applicable family of radial basis function (RBF) kernels (including the popular Gaussian kernel). In SVM modeling, object similarity relationships are determined by chosen kernel functions. The Tanimoto kernel was specifically introduced in cheminformatics for quantifying Tanimoto similarity of molecular fingerprint representations. Both SVETA and SVERAD employ binary molecular fingerprints as features. Any binary fingerprint representation can be used. The software tools were developed on a Linux-based system (Ubuntu 22.04) but are also usable with different operating systems. Detailed below are the steps required to install the environment and packages for executing SVETA and SVERAD and the workflow for applying the protocols on exemplary (freely available) compound data with ready-to-use Python scripts and custom code. Computational times are reported for a machine with capacity specified in the materials and equipment section. Using a different system configuration will lead to different execution times.

Installation

Timing: 10 min

1. Install Python 3 (versions 3.10.5 and 3.11.0 were tested, but different versions are compatible as well) and the libraries required to run the code. We suggest creating and using a conda virtual environment:
   a. Install Anaconda from https://www.anaconda.com/download.
   b. Download or clone the SVERAD repository from https://github.com/AndMastro/SVERAD (which also contains the code required to execute SVETA):

```
>git clone https://github.com/AndMastro/SVERAD.git
```

Note: Git should be installed to run the command. If it is not installed, follow the instructions from https://git-scm.com.

   c. Create a conda environment using the environment.yml file provided in the cloned repository:
      i. Open environment.yml and edit the parameter prefix to match your conda environment folder.
      ii. Open a terminal window in the repository root folder.
      iii. Run the command:

```
>conda env create -f environment.yml
```

Troubleshooting 1

Note: Alternatively, if you do not wish to use a conda environment and prefer a local Python installation instead, the required Python packages (found in the key resources table) can be installed using pip:

```
>pip install package_name
```

If you prefer using a conda environment but not the provided one, manually install alternative packages using conda:

```
>conda install package_name
```

Troubleshooting 2

2. Install the SVETA module:
   a. In the repository, move to the folder *src/sveta.*
   b. Run the command:

```
>pip install .
```

3. Install the SVERAD module:
   a. In the repository, move to the folder *src/sverad.*
   b. Run the command:

```
>pip install .
```

Note: Installing the SVETA and SVERAD modules will also cover additional dependencies, if not previously installed.

Note: Alternatively, for enhanced customization, SVETA and SVERAD can also be installed in development mode such that any modification made to the Python source files is immediately reflected in the code:

```
>pip install -e .
```

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Compound activity data	ChEMBL 30	https://doi.org/10.6019/CHEMBL.database.30
Confirmed aggregators	Aggregator advisor	https://advisor.docking.org/
Adenosine receptor 3 ligand dataset	This paper Feldmann and Bajorath[1] Mastropietro et al.[2]	https://doi.org/10.17632/hz3pjthz2t.1
Software and algorithms		
SVETA v2	This paper Feldmann and Bajorath[1] Zenodo[4]	Zenodo: https://doi.org/10.5281/zenodo.6792073
SVERAD v1.0.1	This paper Mastropietro et al.[2] Zenodo[5]	GitHub: https://github.com/AndMastro/SVERAD Zenodo: https://doi.org/10.5281/zenodo.10803755
RDKit 2023.09.6	Zenodo	Zenodo: https://doi.org/10.5281/zenodo.10793672
Lilly Medchem Rules	GitHub	https://github.com/IanAWatson/Lilly-Medchem-Rules
scikit-learn 1.4.1.post1	GitHub	https://github.com/scikit-learn/scikit-learn
Matplotlib 3.8.0	GitHub	https://github.com/matplotlib/matplotlib
NumPy 1.26.4	GitHub	https://github.com/numpy/numpy
SciPy 1.12.0	GitHub	https://github.com/scipy/scipy
tqdm 4.66.2	GitHub	https://github.com/tqdm/tqdm
Other		
Intel Core i7-12700H @ max 4.70 GHz CPU	N/A	N/A
16 GB RAM	N/A	N/A
Windows/Linux/macOS operating system	N/A	N/A

MATERIALS AND EQUIPMENT

Computational resources		
Component	Brand	Model/Capabilities/Version
CPU	Intel	Core i7-12700H @ max 4.70 GHz
RAM	Any	16 GB
Operating System	Linux/Windows/macOS	Ubuntu 22.04/11/Catalina

STEP-BY-STEP METHOD DETAILS

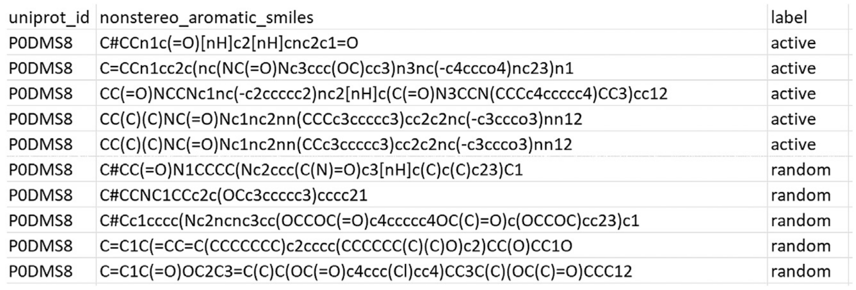
The use of SVETA and SVERAD for explaining predictions of SVM models using the Tanimoto and RBF kernels, respectively, is described. We show how to properly encode the data for use with the scripts provided in the repository and detail how to apply SVETA and SVERAD on the provided compound data set (encoded as SMILES[6] strings) and how to use the modules with custom code.

Data preparation

Timing: 10 min

This step generates the data in the required format and should be performed manually by the user.

1. Format the data as a tab-separated value file (.tsv). The first column (uniprot_id) should contain the name of the target, the second column (nonstereo_aromatic_smiles) the SMILES string of each compound, and the third column (label) should state whether the compound is active or not (using the labels *active* or *random*). An example of a suitable file is provided in Figure 1.



uniprot_id	nonstereo_aromatic_smiles	label
P0DMS8	C#CCn1c(=O)[nH]c2[nH]cnc2c1=O	active
P0DMS8	C=CCn1cc2c(nc(NC(=O)Nc3ccc(OC)cc3)n3nc(-c4ccco4)nc23)n1	active
P0DMS8	CC(=O)NCCNc1nc(-c2ccccc2)nc2[nH]c(C(=O)N3CCN(CCCc4ccccc4)CC3)cc12	active
P0DMS8	CC(C)(C)NC(=O)Nc1nc2nn(CCCc3ccccc3)cc2c2nc(-c3ccco3)nn12	active
P0DMS8	CC(C)(C)NC(=O)Nc1nc2nn(CCc3ccccc3)cc2c2nc(-c3ccco3)nn12	active
P0DMS8	C#CC(=O)N1CCCC(Nc2ccc(C(N)=O)c3[nH]c(C)c(C)c23)C1	random
P0DMS8	C#CCNC1CCc2c(OCc3ccccc3)cccc21	random
P0DMS8	C#Cc1cccc(Nc2ncnc3cc(OCCOC(=O)c4ccccc4OC(C)=O)c(OCCOC)cc23)c1	random
P0DMS8	C=C1C(=CC=C(CCCCCCC)c2cccc(CCCCCC(C)(C)O)c2)CC(O)CC1O	random
P0DMS8	C=C1C(=O)OC2C3=C(C)C(OC(=O)c4ccc(Cl)cc4)CC3C(C)(OC(C)=O)CCC12	random

Figure 1. Example of a suitable input file containing the name of the target (uniport_id), SMILES strings, and labels (*active/random*)
An exemplary compound data file is provided in the repository.

   a. Create a .tsv containing the desired compounds.
   b. Place the generated file in a folder of interest.
2. Open the parameters.yml file provided in the repository. It contains customizable parameters for the scripts. Edit the DATASET_PATH field with the location of your generated data file and also edit the remaining fields as needed.

   Note: The repository contains data from the original publications[1,2] that are used by default when executing the provided scripts. The data set consists of 287 adenosine receptor 3 ligands used as positive training and test instances and the same number of other compounds randomly selected from ChEMBL[7,8] used as negative instances.

SVM model training and explanation

Timing: 1 min

This is the main step performing both the training of the SVM models (via grid-search optimization of hyperparameters) and the subsequent Shapley value analysis. We show how to train and explain SVM models using the provided ready-to-use scripts and alternatives for adding SVETA and SVERAD to custom code.

3. Run the script for model training and explanation:
   a. Open a terminal in the repository root folder.
   b. Activate the conda environment:

```
>conda activate sverad_env
```

   Troubleshooting 3
   c. Run the script:

```
>python trainer_explainer_script.py
```

   Troubleshooting 4

   Note: The script will load arguments from the parameters.yml file. Since the repository is actively maintained and updated to ensure up-to-date content, this file may vary (refer to the GitHub repository). At the time of writing, the file contains the following parameters:

   i. DATASET_PATH: location of the dataset .tsv file.

ii. DATASET_PICKLE_PATH: location for saving/loading the data set in pickle format used by the code.
iii. FINGERPRINTS_PICKLE_PATH: location for saving/loading the molecular fingerprint generated by the script, used as molecular representation/input features for the SVM models.
iv. MODEL_PATH: location for saving the generated models.
v. EXPLANATION_PATH: location for saving the generated explanations.
vi. PREDICTION_PATH: location for saving the model predictions.
vii. TARGET_UNIPROT_ID: name of the target (first column in the .tsv file).
viii. SAVE_DATASET_PICKLE: binary label specifying whether or not to save the data set and the fingerprints as pickle files.
ix. LOAD_DATASET_PICKLE: binary label specifying whether or not to load precomputed data set and fingerprints as pickle files.
x. EMPTY_SET_VALUE: value to assign to the empty coalition for the computation of the Shapley values (defaults to 0).
xi. SAVE_EXPLANATIONS: binary label specifying whether or not to save the generated explanations and predictions.
xii. SAVE_MODELS: binary label specifying whether or not to save the generated models.
xiii. SEED: seed used by the random number generators (defaults to 42).

Note: Instead of relying on the provided scripts, it is also possible to import SVETA and SVERAD in custom code, as illustrated below:

```
from sverad.sverad_svm import ExplainingSVC as SVERADExplainingSVC

from sveta.svm import ExplainingSVC as SVETAExplainingSVC

C = 1.0

GAMMA = 1.0

SEED = 42

EMPTY_SET_VALUE = 0.0

sverad_model = SVERADExplainingSVC(C = C, gamma_val = GAMMA, random_state=SEED, empty_set_
value=EMPTY_SET_VALUE)

sveta_model = SVETAExplainingSVC(C = C, random_state=SEED, no_player_value=EMPTY_SET_VALUE)
```

Note: First, the required modules should be imported. Then, one can define the models. The classes SVERADExplainingSVC and SVETAExplainingSVC provide SVM classification models with training and explanation procedures. The parameter C is used to control the applied regularization in the SVM, random_state defines the seed for the random number generator internally used by the functions, and empty_set_value (termed no_player_value in SVETA) indicates the value for the empty coalition. The parameter gamma_val is specific for the RBF kernel and used to modulate the decision boundary of the model. Larger values determine a more complex boundary, while smaller values are used to make the boundary smoother. For more details on additional optional parameters, please, refer to the GitHub repository.

Note: The Morgan fingerprint[9] (with bond radius 2) is used here, which has been the prototype for the current state-of-the-art class of (extended connectivity) fingerprints calculated from molecular graphs. These fingerprints capture topological atom (environment) patterns that can be used as molecule-dependent feature sets (“unfolded”) or hashed (“folded”) into a bit string representation of constant length (e.g., 2048 bits).

Then, one can train and test the models and generate Shapley value-based explanations:

```
X_train = ... #your training data samples

y_train = ... #your training data labels

X_test = ... #your test data samples

sverad_model.fit(X_train, y_train)

sveta_model.fit(X_train, y_train)

sverad_preds = model.predict(X_test)

sveta_preds = model.predict(X_test)

sverad_shapley_values = sverad_model.feature_weights(X_test)

sveta_shapley_values = sveta_model.feature_weights(X_test)
```

Note: Once the models are trained using the fit() method, they can be used to predict new samples with predict(). Finally, the feature_weights() method generates exact Shapley values for input features and the predictions of the samples passed to the function.

Note: Shapley values can be calculated for any defined molecular features. However, fingerprints capturing structural patterns or fragments are generally preferred for applications in chemistry because structural features can be mapped on predicted compounds, providing intuitive access to molecular regions determining predictions.

Explanation analysis and feature mapping

Timing: 7 min

After having trained and tested the models and computed the Shapley values, the importance of features present or absent in correctly predicted test compounds can be quantified and the features can be mapped on compound structures.

4. Run the analysis script:
   a. With a terminal open in the repository root folder, run:

```
>python explanation_analyzer_script.py
```

Troubleshooting 5

The script loads arguments from the parameters.yml file including:

i. TARGET_UNIPROT_ID: name of the target (first column in the .tsv file).
ii. DATASET_PATH: location of the dataset .tsv file.
iii. SAVE_DATA_PATH: location for saving the generated plots and images.
iv. DATASET_PICKLE_PATH: location of the data set in pickle format.
v. FINGERPRINTS_PICKLE_PATH: location of the fingerprints in pickle format.
vi. MODEL_PATH: location of the models in pickle format.
vii. EXPLANATION_PATH: location of the explanations in pickle format.
viii. PREDICTION_PATH: location of the .tsv containing prediction information.
ix. SAVE_PLOTS: binary label indicating whether or not to save the plots with feature contributions.
x. COLOR_PRESENT_FEATURES: string defining the color of present features in the plot.

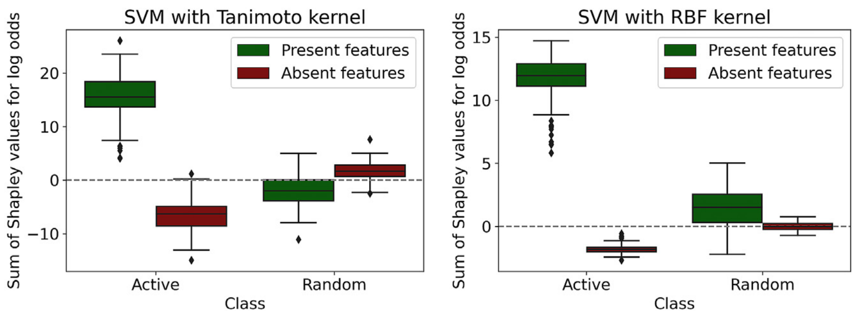


Figure 2. Contribution of present and absent features to the correct prediction of active and randomly selected compounds quantified using SVETA (left) and SVERAD (right)

xi. COLOR_ABSENT_FEATURES: string defining the color of absent features in the plot.
xii. SAVE_MAPPINGS: binary label indicating whether or not to save the feature mappings.
xiii. FIGURE_FORMAT: format of the saved plots.
xiv. SEED: seed used by the random number generators (defaults to 42).

EXPECTED OUTCOMES

The final results of the protocol, generated by the script explanation_analyzer_script.py, are represented boxplots reporting contributions of features present or absent in correctly predicted test compounds. Present features are then mapped to the corresponding compounds, highlighting their relevance for the prediction. An exemplary plot reporting feature contribution is given in Figure 2.

The script also outputs additional statistics and information including the parameters of the best models obtained via grid search, average numbers of intersecting features, symmetric difference, and union features between the input samples and the SVM support vectors, training and test set accuracy, the models' expected values, and contributions of present and absent features for active and inactive compounds.

Feature mapping is generated for any correctly predicted compound. Figures 3 and 4 show the mapping for exemplary active and random compounds, respectively, using SVETA and SVERAD. The mappings delineate molecular substructures supporting correct predictions (red for active and blue for random compounds) or opposing them (blue for active compounds and red for random compounds).

QUANTIFICATION AND STATISTICAL ANALYSIS

Both SVETA and SVERAD were thoroughly assessed to ensure the correct and exact computation of Shapley values. We generated 20 binary feature vectors containing a small number of features (15), for which exact Shapley value computation via exhaustive enumeration of all feature coalitions was feasible. We then computed Shapley values for the Tanimoto and RBF kernels and all pairs of vectors using SVETA and SVERAD, respectively, obtaining the same values as produced by the explicit enumeration, thus demonstrating the validity of the protocol. Moreover, we compared the Shapley values with the widely adopted Shapley Additive Explanations (SHAP)[10,11] approximation. The Fisher-transformed Pearson's *r* correlation coefficient between SHAP and SVETA was 0.82 ± 0.25, and between SHAP and SVERAD was 0.72 ± 0.43, reflecting the underlying local approximation of the SHAP values, as also reported in the original publications.[1,2]

The same analysis was performed for the Shapley values computed for the logits of the SVM predictions. In this case, the SHAP approximation also displayed limited or no correlation with exact Shapley values (with median correlation coefficients of 0.682 for SHAP vs. SVETA and 0.120 for SHAP

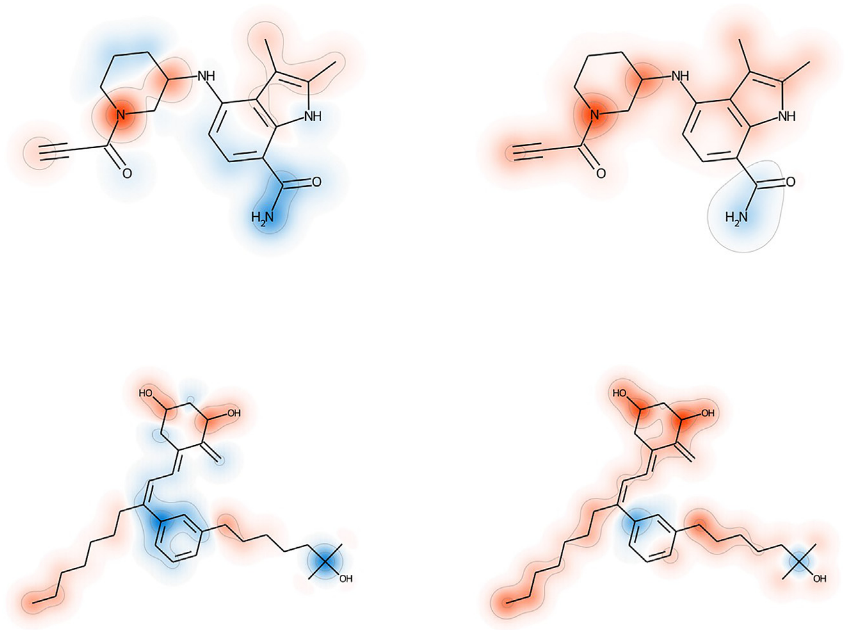


Figure 4. Feature mapping on correctly predicted random compounds using SVETA (left) and SVERAD (right)

vs. SVERAD), assigning preference to the exact Shapley value computation for assessing feature contributions of SVM models.[1,2]

LIMITATIONS

SVETA and SVERAD are usable only with SVM models relying on the Tanimoto and the RBF kernels, respectively, and with binary fingerprint descriptors.

TROUBLESHOOTING

Problem 1

Related to installation. If working in a Windows PowerShell, the user may encounter the error "conda is not recognized as an internal or external command." This is then due to the fact that conda was not initialized in the PowerShell.

Potential solution

Initialize conda with the command:

```
>conda init
```

and then restart the terminal.

Problem 2

Related to installation. The environment.yml file was generated under a Linux-based system. Even if it was generated for cross-platform use, it might fail to install the packages under a different system.

Potential solution

Manually install the packages using pip or conda.

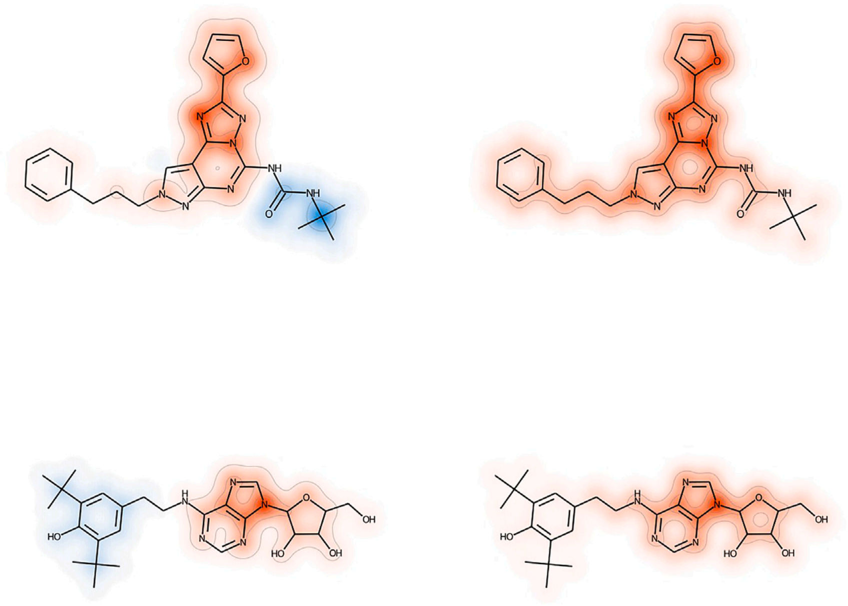


Figure 3. Feature mapping on correctly predicted active compounds using SVETA (left) and SVERAD (right)

Problem 3

Related to SVM model training and explanation. The environment name *sverad_env* is the default found in the provided environment.yml file. If upon activation the error "EnvironmentNameNot-Found" occurs, it means that the environment was installed under a different name.

Potential solution

If the environment.yml file was edited and the *name* field changed, be sure to use that name when activating the environment. If, instead, one uses a custom environment, activate it by providing the correct name. It is possible to see a list of the installed environments using the command:

```
>conda env list
```

Problem 4

Related to SVM model training and explanation. Depending on the environment variables, when running the python command in the terminal, one might encounter an error message like "Command not found".

Potential solution

Try typing python3 instead of python when launching Python scripts from terminal. If there still is an error, ensure that Python is correctly installed (either in the conda environment or standalone if one has opted for a local Python installation).

Problem 5

Related to explanation analysis and feature mapping. If running the script explanation_analyzer_script.py fails, returning the message "No such file or directory: [file_name]", it means that input files needed by the script are not placed in the correct folders.

Potential solution

Check the following:

- When running trainer_explainer_script.py, depending on the file(s) not found, ensure that the fields SAVE_DATASET_PICKLE, SAVE_EXPLANATIONS, and SAVE_MODELS in parameters.yml are set to True.
- When running explanation_analyzer_script.py, check parameters.yml to ensure that the location of the files to be loaded matches their actual location.

RESOURCE AVAILABILITY

Lead contact

Requests for further information, resources, and software should be directed to (and will be answered by) the lead contact, Jürgen Bajorath (bajorath@bit.uni-bonn.de).

Technical contact

Questions about the technical specifics of performing the protocol should be directed to (and will be answered by) the technical contact, Andrea Mastropietro (mastropietro@diag.uniroma1.it).

Materials availability

Not applicable.

Data and code availability

The compound data used in this protocol are available as Mendeley data at https://doi.org/10.17632/hz3pjthz2t.1. The SVETA source code is available in an open access deposition at Zenodo (https://doi.org/10.5281/zenodo.6792073). The SVERAD source code is available both on GitHub (https://github.com/AndMastro/SVERAD) and on Zenodo (https://doi.org/10.5281/zenodo.10803755).

ACKNOWLEDGMENTS

The authors would like to thank Christian Feldmann for his contribution to the original SVETA and SVERAD papers and Jannik P. Roth, Tiago Janela, and Alec Lamens for helpful discussions.

AUTHOR CONTRIBUTIONS

Conceptualization, A.M. and J.B.; software, A.M.; investigation, A.M.; writing – original draft, A.M. and J.B.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Feldmann, C., and Bajorath, J. (2022). Calculation of exact Shapley values for support vector machines with Tanimoto kernel enables model interpretation. iScience *25*, 105023. https://doi.org/10.1016/j.isci.2022.105023.
2. Mastropietro, A., Feldmann, C., and Bajorath, J. (2023). Calculation of exact Shapley values for explaining support vector machine models using the radial basis function kernel. Sci. Rep. *13*, 19561. https://doi.org/10.1038/s41598-023-46930-2.
3. Tanimoto, T.T. (1958). Elementary Mathematical Theory of Classification and Prediction (IBM Internal Report).
4. Feldmann, C., and Bajorath, J. (2022). Calculation of Exact Shapley Values for Support Vector Machines with Tanimoto Kernel Enables Model Interpretation. Zenodo. https://doi.org/10.5281/zenodo.6792073.
5. Mastropietro, A. (2024). SVERAD (v1.0.1) (Zenodo). https://doi.org/10.5281/zenodo.10803755.
6. Weininger, D. (1988). SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. J. Chem. Inf. Comput. Sci. *28*, 31–36. https://doi.org/10.1021/ci00057a005.
7. Gaulton, A., Bellis, L.J., Bento, A.P., Chambers, J., Davies, M., Hersey, A., Light, Y., McGlinchey, S., Michalovich, D., Al-Lazikani, B., and Overington, J.P. (2012). ChEMBL: a large-scale bioactivity database for drug discovery. Nucleic Acids Res. *40*, D1100–D1107. https://doi.org/10.1093/nar/gkr777.
8. Bento, A.P., Gaulton, A., Hersey, A., Bellis, L.J., Chambers, J., Davies, M., Krüger, F.A., Light, Y., Mak, L., McGlinchey, S., et al. (2014). The ChEMBL bioactivity database: an update. Nucleic Acids Res. *42*, D1083–D1090. https://doi.org/10.1093/nar/gkt1031.
9. Morgan, H.L. (1965). The generation of a unique machine description for chemical structures - a technique developed at chemical abstracts service. J. Chem. Doc. *5*, 107–113.
10. Lundberg, S.M., and Lee, S.-I. (2017). A unified approach to interpreting model predictions. Adv. Neural Inf. Process. Syst. *30*. https://doi.org/10.48550/arXiv.1705.07874.
11. Chen, H., Covert, I.C., Lundberg, S.M., and Lee, S.-I. (2023). Algorithms to estimate Shapley value feature attributions. Nat. Mach. Intell. *5*, 590–601. https://doi.org/10.1038/s42256-023-00657-x.