Scikit-Mol – RDKit integration in Scikit-Learn: Updates and New Features

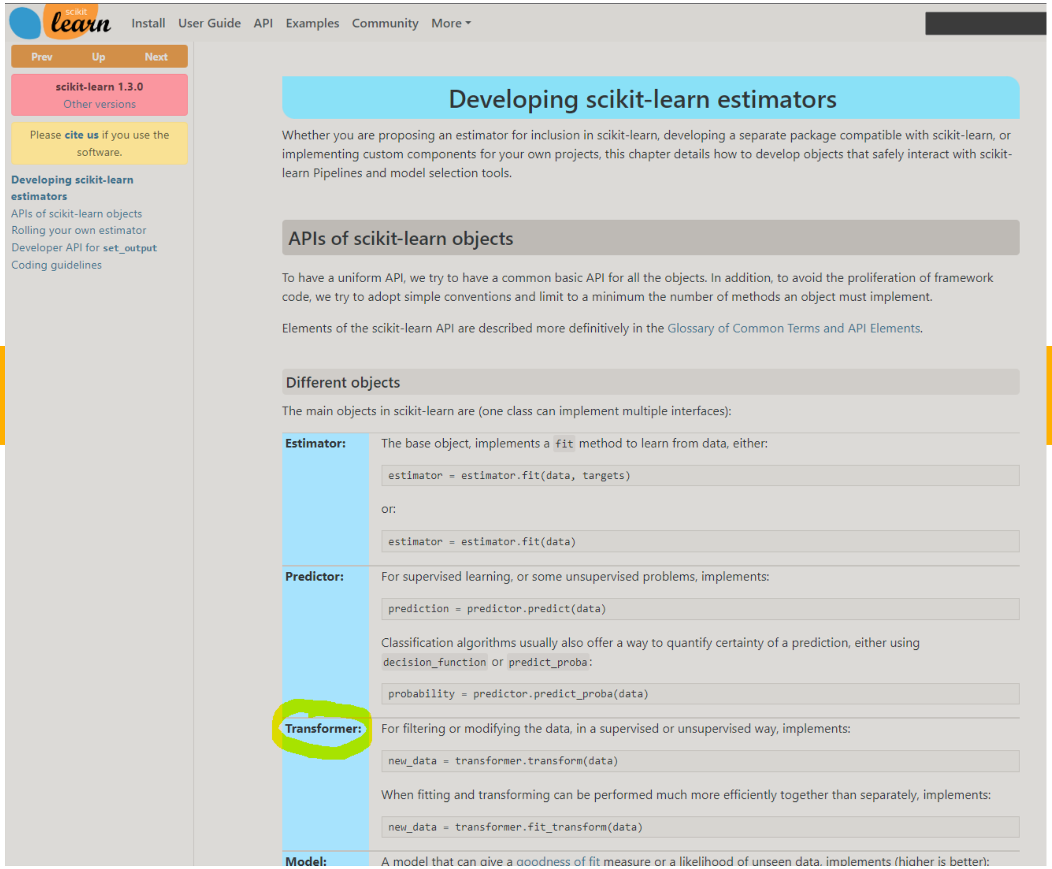
Some images from Pixabay/Generated

Brief Summary

Project started at **2022 RDKit UGM Hackathon**

+

Open-Source Cheminformatics
and Machine Learning



scikit-learn 1.3.0
Other versions

Please **cite us** if you use the software.

Developing scikit-learn estimators
APIs of scikit-learn objects
Rolling your own estimator
Developer API for `set_output`
Coding guidelines

Developing scikit-learn estimators

Whether you are proposing an estimator for inclusion in scikit-learn, developing a separate package compatible with scikit-learn, or implementing custom components for your own projects, this chapter details how to develop objects that safely interact with scikit-learn Pipelines and model selection tools.

APIs of scikit-learn objects

To have a uniform API, we try to have a common basic API for all the objects. In addition, to avoid the proliferation of framework code, we try to adopt simple conventions and limit to a minimum the number of methods an object must implement.

Elements of the scikit-learn API are described more definitively in the Glossary of Common Terms and API Elements.

Different objects

The main objects in scikit-learn are (one class can implement multiple interfaces):

Estimator: The base object, implements a `fit` method to learn from data, either:

```
estimator = estimator.fit(data, targets)
```

or:

```
estimator = estimator.fit(data)
```

Predictor: For supervised learning, or some unsupervised problems, implements:

```
prediction = predictor.predict(data)
```

Classification algorithms usually also offer a way to quantify certainty of a prediction, either using `decision_function` or `predict_proba`:

```
probability = predictor.predict_proba(data)
```

Transformer: For filtering or modifying the data, in a supervised or unsupervised way, implements:

```
new_data = transformer.transform(data)
```

When fitting and transforming can be performed much more efficiently together than separately, implements:

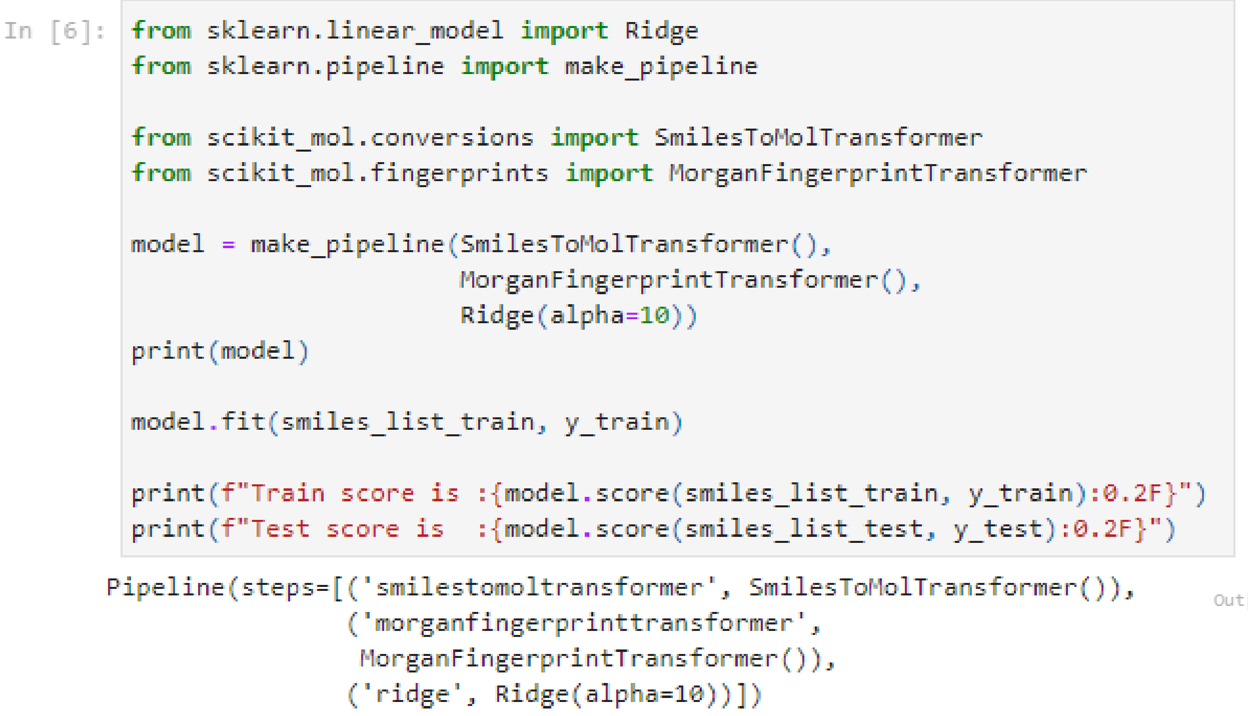
```
new_data = transformer.fit_transform(data)
```

Model: A model that can give a goodness of fit measure or a likelihood of unseen data, implements (higher is better):

https://scikit-learn.org/stable/developers/develop.html

https://github.com/EBjerrum/scikit-mol

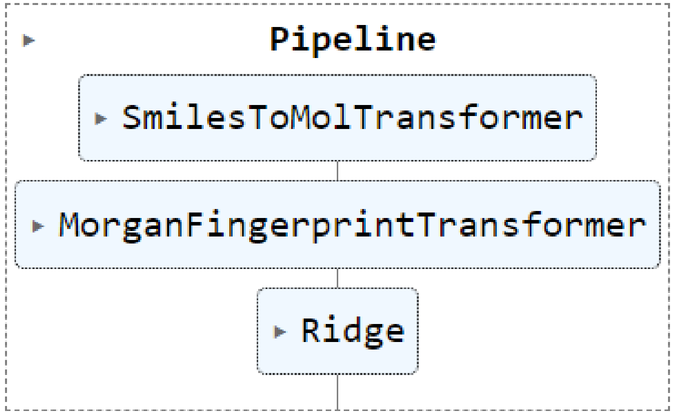
Quick RDKit Integration into Scikit-Learn Pipelines



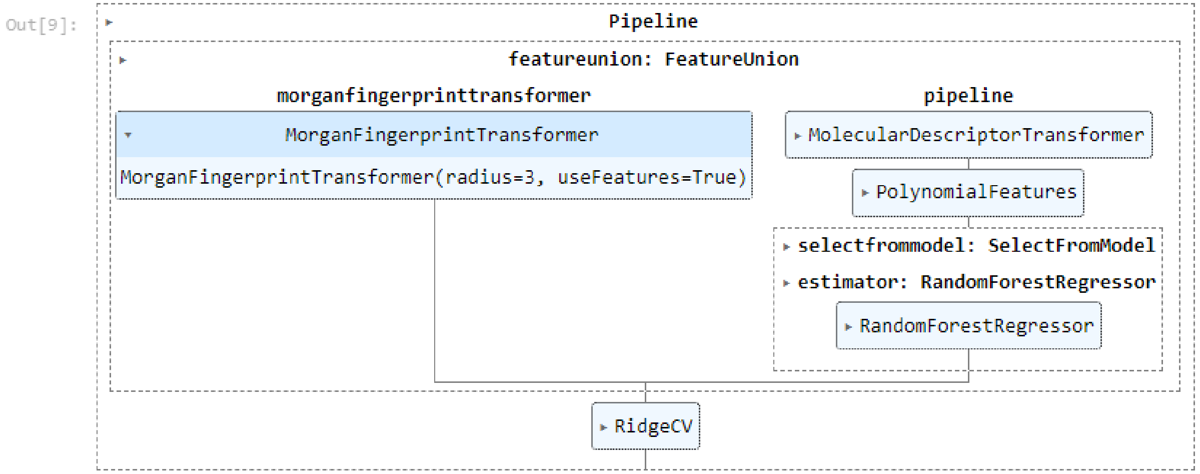
In [6]:

```python
from sklearn.linear_model import Ridge
from sklearn.pipeline import make_pipeline

from scikit_mol.conversions import SmilesToMolTransformer
from scikit_mol.fingerprints import MorganFingerprintTransformer

model = make_pipeline(SmilesToMolTransformer(),
                      MorganFingerprintTransformer(),
                      Ridge(alpha=10))
print(model)

model.fit(smiles_list_train, y_train)

print(f"Train score is :{model.score(smiles_list_train, y_train):0.2F}")
print(f"Test score is  :{model.score(smiles_list_test, y_test):0.2F}")
```

```
Pipeline(steps=[('smilestomoltransformer', SmilesToMolTransformer()),
                ('morganfingerprinttransformer',
                 MorganFingerprintTransformer()),
                ('ridge', Ridge(alpha=10))])
Train score is :0.74
Test score is  :0.63
```



- Featureization from RDKit molecules or SMILES strings
- Integrate into Scikit-Learn models and pipelines
- Fit Scikit-Learn models directly on molecules or SMILES
- Easily get hyperparameter tuning of fingerprinting as compatible with e.g. SKOpt
- Self-documenting and self-contained models
- Use standard scikit-learn classes to build more complex models



Out[9]:
Pipeline
featureunion: FeatureUnion
morganfingerprinttransformer
MorganFingerprintTransformer
MorganFingerprintTransformer(radius=3, useFeatures=True)
pipeline
MolecularDescriptorTransformer
PolynomialFeatures
selectfrommodel: SelectFromModel
estimator: RandomForestRegressor
RandomForestRegressor
RidgeCV

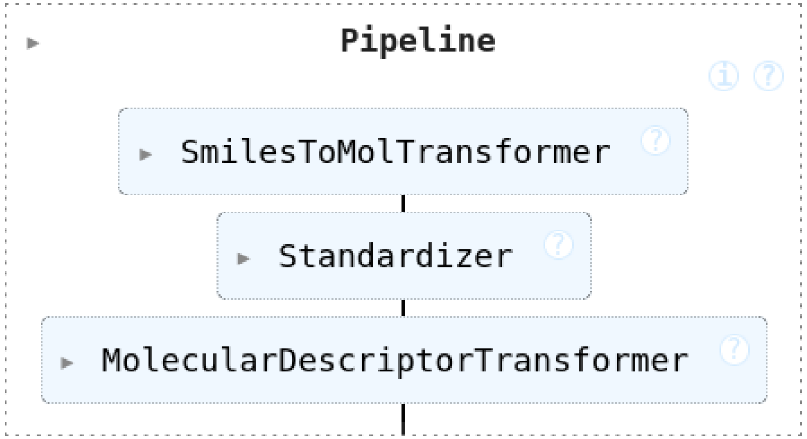
Maintenance and Project Improvements

- Online documentationhttps://scikit-mol.readthedocs.io/en/latest/
- Switch to RDKit generators
- Conda package:https://anaconda.org/conda-forge/scikit-mol
- uv build framework

- Improved parallelism using joblib
- Improved handling of parallelism for ad-hoc custom feature transformers

Pandas Input and Output

- `.set_output(transform="pandas")`
- Very handy when doing feature importance analysis!
- Column names are checked for consistency when predicting
- `model.predict(features[model.feature_names_in_])`

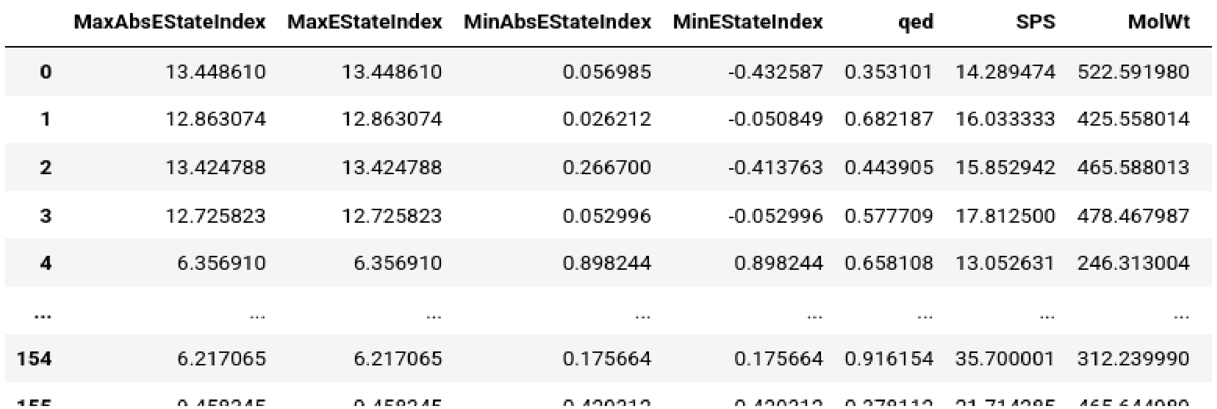


Pipeline

SmilesToMolTransformer

Standardizer

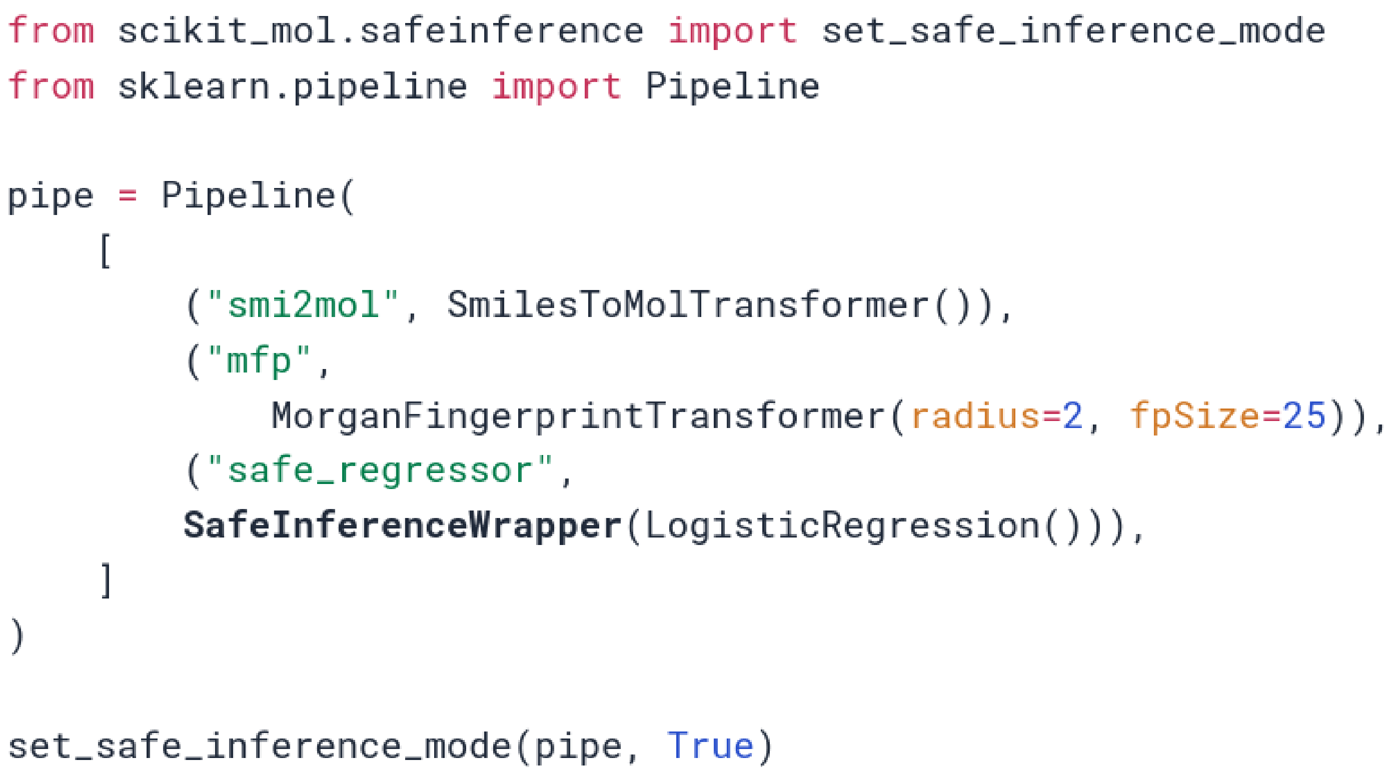
MolecularDescriptorTransformer



	MaxAbsEStateIndex	MaxEStateIndex	MinAbsEStateIndex	MinEStateIndex	qed	SPS	MolWt
0	13.448610	13.448610	0.056985	-0.432587	0.353101	14.289474	522.591980
1	12.863074	12.863074	0.026212	-0.050849	0.682187	16.033333	425.558014
2	13.424788	13.424788	0.266700	-0.413763	0.443905	15.852942	465.588013
3	12.725823	12.725823	0.052996	-0.052996	0.577709	17.812500	478.467987
4	6.356910	6.356910	0.898244	0.898244	0.658108	13.052631	246.313004
...	...	...	...	...	...	...	...
154	6.217065	6.217065	0.175664	0.175664	0.916154	35.700001	312.239990
155	0.458245	0.458245	0.420213	0.420213	0.378112	21.714285	465.644080

Safe Inference Mode

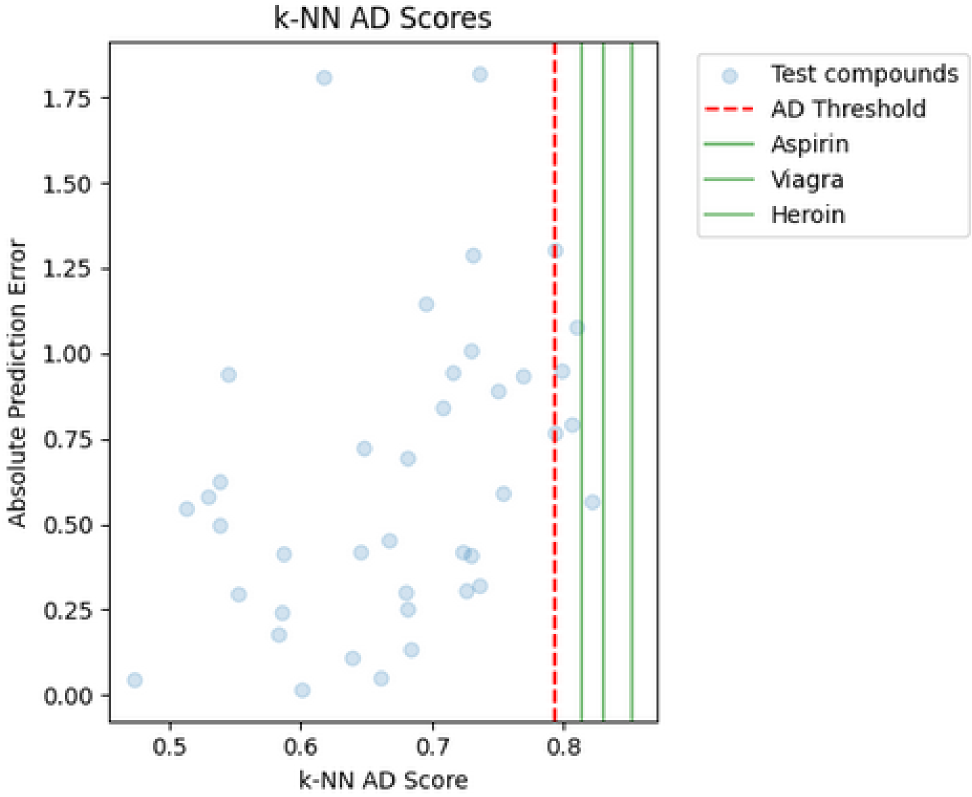
- Problematic molecules or SMILES ~~can~~*will* be sent for models by ~~users~~GenAI
- **safe_inference** mode ensures that the whole batch doesn’t fail
- Scikit-Mol transformers handle **“False” objects** (like False, 0, InvalidMol) gracefully
- Scikit-learn estimators should be wrapped in **SafeInferenceWrapper**



```python
from scikit_mol.safeinference import set_safe_inference_mode
from sklearn.pipeline import Pipeline

pipe = Pipeline(
    [
        ("smi2mol", SmilesToMolTransformer()),
        ("mfp",
            MorganFingerprintTransformer(radius=2, fpSize=25)),
        ("safe_regressor",
        SafeInferenceWrapper(LogisticRegression())),
    ]
)

set_safe_inference_mode(pipe, True)
```

Feature Based Applicability Domain Estimators

- **Applicability domain estimators** now available
- Can be fit to molecular set with a **percentile cutoff** (i.e. validation set)
- Slight deviation from sklearn api
  - `.transform()` → returns score
  - `.predict()` → returns 0 or 1
  - `.score_transform()` → returns 0 to 1 (soft boundary)
- bounding box, convex hull, hotelling, isolation forest, kernel density, kNN, leverage, local outlier, mahalanobis, topkat



Code adapted from MLChemAD (Olivier J. M. Béquignon) Thanks!

Conclusion and Acknowledgements

- Scikit-Mol is the original RDKit hackathon initiated project
- New features still being added as contributions from individuals, academia and industry
- Plenty of notebooks and examples at https://scikit-mol.readthedocs.io/en/latest/
- **pip/conda install scikit-mol**

- **Thanks for your attention!**

- **Acknowledgements**

  Carmen Esposito @cespos, Son Ha @son-ha-264, Oh-hyeon Choung @Ohyeon5, Andreas Poehlmann @ap--, Ya Chen @anya-chen, Anton Siomchen @asiomchen, Rafał Bachorz @rafalbachorz, Adrien Chaton @adrienchaton, @VincentAlexanderScholz, Benjamin Ries @RiesBen, @enricogandini, @mikemhenry, Christian Feldmann @c-feldmann, Mieczyslaw Torchala @mieczyslaw, Kyle Barbary @kbarbary, Katja Bjerrum @productivista …

- Cheminformania Consulting, Aptuit, BASF, Bayer AG, Boehringer Ingelheim, Chodera Lab (MSKCC), EPAM Systems,ETH Zürich, Evotec, Johannes Gutenberg University, Martin Luther University, Odyssey Therapeutics, Open Molecular Software Foundation, Openfree.energy, Polish Academy of Sciences, Productivista, Simulations-Plus Inc., University of Vienna

E. J. Bjerrum et al., “Scikit-Mol brings cheminformatics to Scikit-Learn,” ChemRxiv, Dec. 2023, doi: 10.26434/chemrxiv-2023-fzqwd.