# Supporting information for "On failure modes in molecule generation and optimization"

Philipp Renz[1], Dries Van Rompaey[2], Jörg Kurt Wegner[2], Sepp Hochreiter[1], and Günter Klambauer[1]

[1]LIT AI Lab & Institute for Machine Learning, Johannes Kepler University Linz, Altenberger Strasse 69, A-4040 Linz, Austria
[2]High Dimensional Biology and Discovery Data Sciences, Janssen Research & Development, Janssen Pharmaceutica N.V., Turnhoutseweg 30, Beerse B-2340, Belgium

## Contents



## List of Figures



## List of Tables

| Target | ChEMBL ID | Active | Inactive | ROC AUC | AP | Fraction pos. |
|---|---|---|---|---|---|---|
| JAK2 | CHEMBL3888429 | 140 | 527 | 0.78±0.03 | 0.44±0.05 | 0.21 |
| EGFR | CHEMBL1909203 | 40 | 802 | 0.72±0.08 | 0.15±0.05 | 0.05 |
| DRD2 | CHEMBL1909140 | 59 | 783 | 0.85±0.03 | 0.53±0.09 | 0.07 |

Table S1: Information on the data sets. Area under ROC curve (ROC AUC) and average precision (AP) show the performance of the trained classifiers on hold out test data. The last column denotes the fraction of positive samples which serves as a random baseline for AP.

# S1 Method details

## S1.1 AddCarbon model

Sampling from the AddCarbon model is done in the following way. A random compound is drawn from the training set and its canonical SMILES [1] is calculated using rdkit [2]. Then a 'C' is added to a uniformly random position in this canonical SMILES. If the resulting string is not a valid SMILES another position is tested. If it is valid and the corresponding canonical SMILES has a string edit distance of 1 to the original canonical SMILES and is not already in the training set it is returned. If no position results in a valid SMILES a new compound is chosen from the training set and the same procedure is applied.

## S1.2 Preprocessing of assay data

We downloaded the data for all three optimization tasks from ChEMBL. The identifiers for the assays are listed in Table S1. To obtain binary labels for the JAK2 task we labelled all compounds with a $pIC_{50}$ greater than 8 as active. For the EGFR and DRD2 tasks we extracted the label from the "Comment" column.

## S1.3 Molecular optimizers

We used the implementation of both the graph-based genetic algorithm and the hill-climbing SMILES-LSTM provided by [3]. For the genetic algorithm we used following settings:

- `population_size`: 100 (Population size)
- `offspring_size`: 200 (size of offspring)
- `generations`: 150 (Number of optimization generations)
- `mutation_rate`: 0.01 (Rate of mutation)
- `random_start`: True (Whether to use a random starting population)
- `patience`: 5 (How long to be patient without improvements)
- `smi_file`: described below (List of molecules to draw initial population from)

The starting population was drawn from the distribution-learning training set also provided by [3]. For the hill-climbing SMILES-LSTM we used these settings:

- `n_epochs`: 151 (Number of iterations)
- `mols_to_sample`: 1028 (Molecules to sample in each step)
- `keep_top`: 512 (Number of molecules to keep in each iteration)

- `optimize_n_epochs`: 1 (Number of fine-tuning epochs per iteration)
- `max_len`: 100 (Maximum SMILES length)
- `optimize_batch_size`: 64 (Finetune optimization batch size)
- `number_final_samples`: 1028 (Number of final samples returned)
- `sample_final_model_only`: False (Whether intermediate samples are returned among results)
- `random_start`: True (Whether to start randomly or pretrain on best compounds of a starting population)

For the particle swarm optimization we used the implementation provided by the authors and the following settings:

- `init_smiles`: Randomly drawn list of molecules from the same set as used for GA (Molecules from which to start optimization)
- `num_part`: 200 (Number of particles)
- `num_swarms`: 1 (Number of swarms)

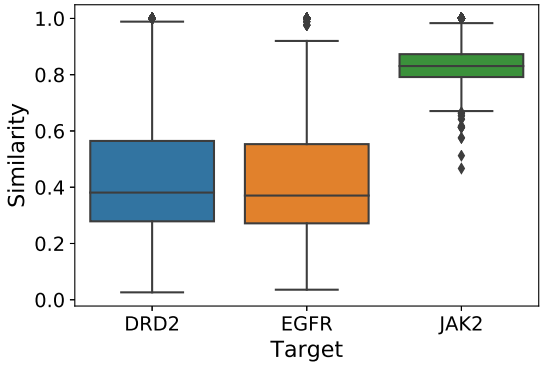



Figure S1: Distribution of similarities to the nearest neighbour in split 1 of compounds in split 2.

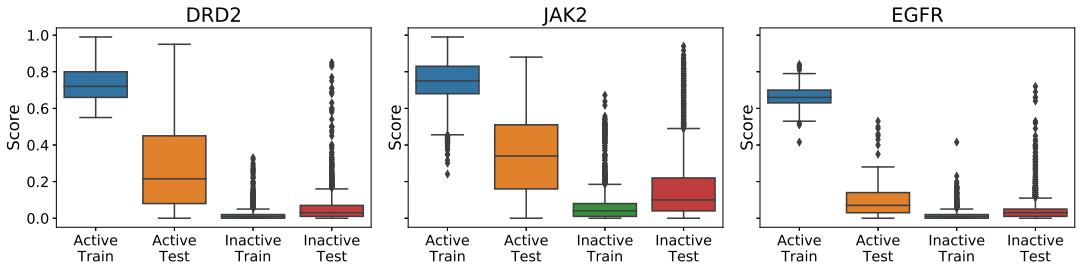



Figure S2: Distribution of scores for different groups of compounds. The distribution of scores of actives from the training set is shifted positively with respect to those from the test set, and vice versa for inactives.

# S2 Additional Results

## S2.1 Additional information on data splits

Fig. S1 shows the distribution of similarities of compounds in split 2 to the nearest neighbour in split 1. For the JAK2 task many compounds have a very similar neighbour in the other split.

## S2.2 Training data bias of classifiers

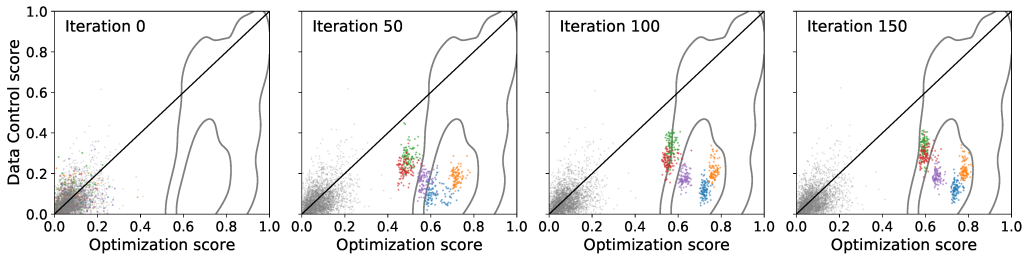
Fig. S2 shows the distribution of scores for different groups of compounds. Actives from the training set obtain a higher median score than those from the test set by the trained classifier. Conversely, inactives from the training set obtain a lower median score than those from the test set. These results are aggregated using ten different random seeds for splitting the data and for fitting the random forest.

## S2.3 Complementary results

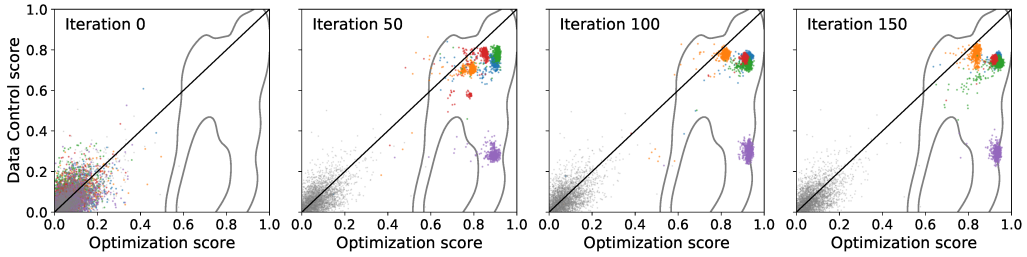
This section contains complementary results for the optimizers and tasks not shown in Section "Failure mechanisms in goal-directed generation" and includes Figs. S3, S4, S5 and S6.

## S2.4 Additional information on optimized molecules

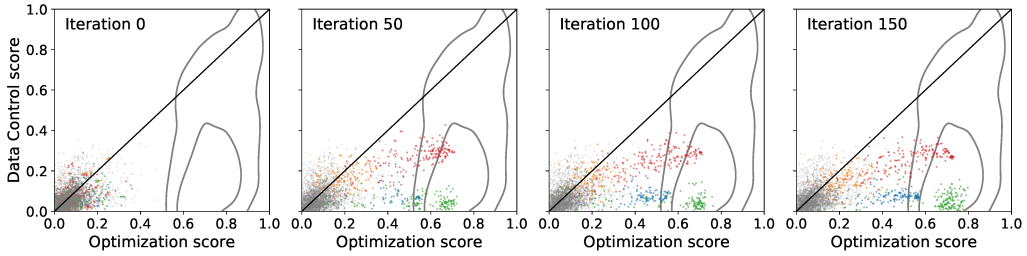
Fig. S9 shows additional metrics over the course of optimization. These metrics include:



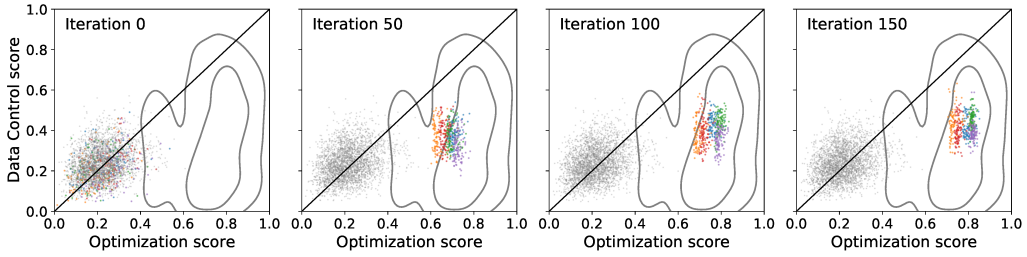

(a) GA on DRD2




(b) LSTM on DRD2



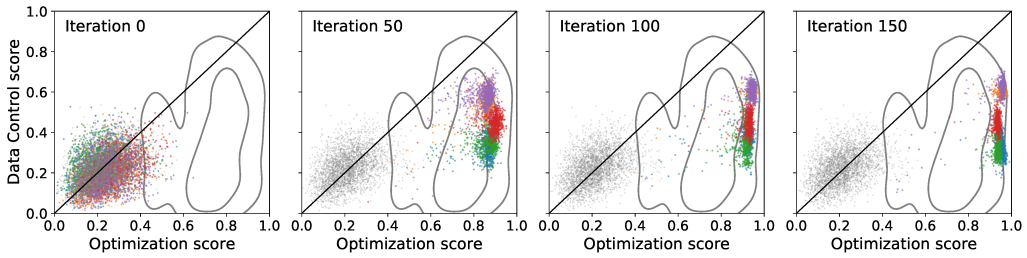

(c) PS on DRD2



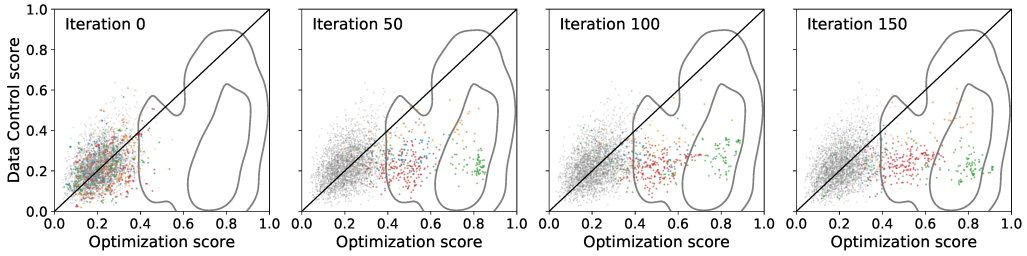

(d) GA on JAK2

| # | SMARTS | Counts | rel. freq. |
|---|---|---|---|
| 1 | O-O-O | 1 | 6.3e-07 |
| 2 | ON([H])O | 4 | 2.5e-06 |
| 3 | N-F | 2 | 1.3e-06 |
| 4 | S-O-O | 12 | 7.5e-06 |

Table S2: Uncommon small substructures contained in Fig. 1 and how often they occur in 1,591,378 ChEMBL compounds. The columns provide the substructure as SMARTS pattern, the absolute ("Counts") and the relative frequency ("rel. freq.") of occurrences of this pattern.

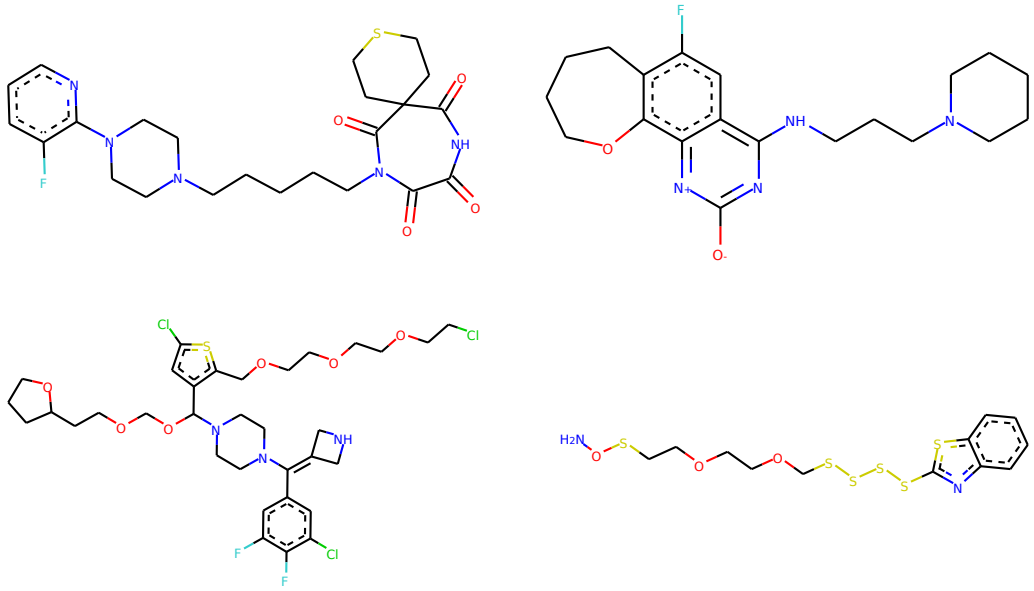



(e) LSTM on JAK2

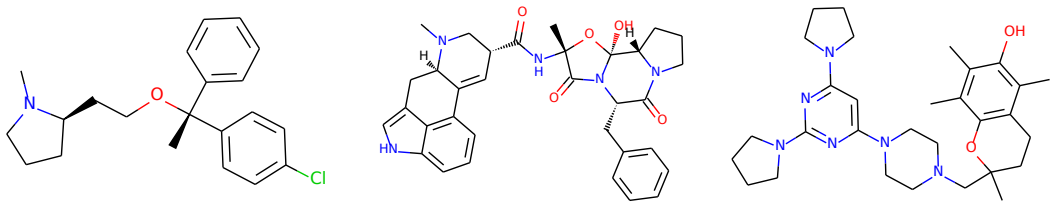



(f) PS on JAK2

Figure S3: Scatter plots of OS vs. DCS on molecules during the course of training. Different colors correspond to different runs. Grey points are random compounds from ChEMBL. The contour lines show the region where split 1 actives lie. Due to symmetry the region of split 2 actives could be obtained by mirroring the contour lines across the diagonal. During the course of training the optimized molecules move towards the region of split 1 actives.

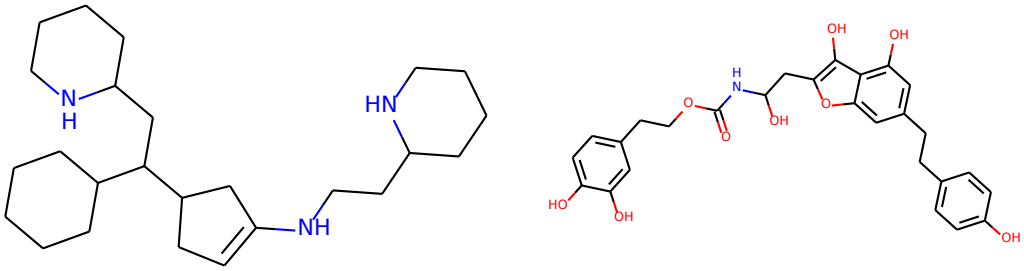


(a) Compounds generated by PS

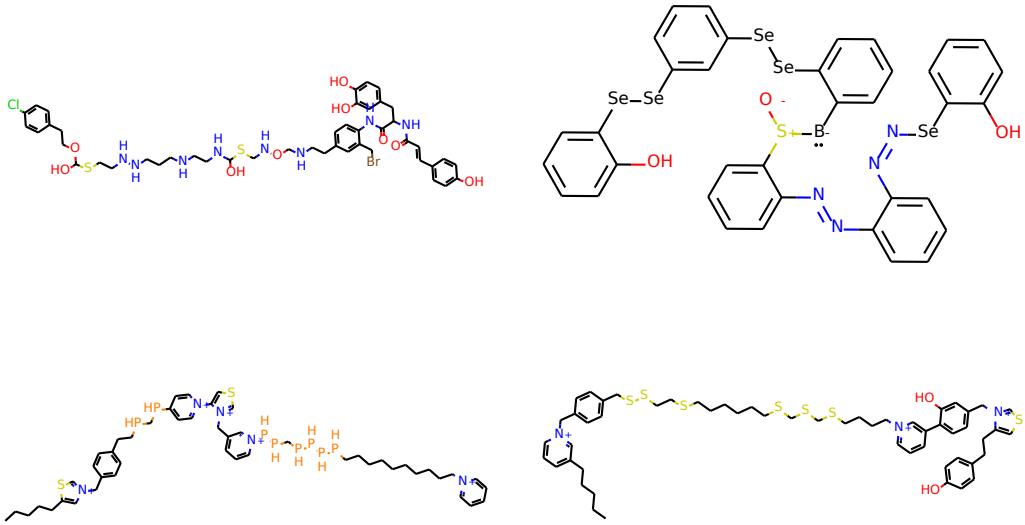


(b) Reference compounds from the training set

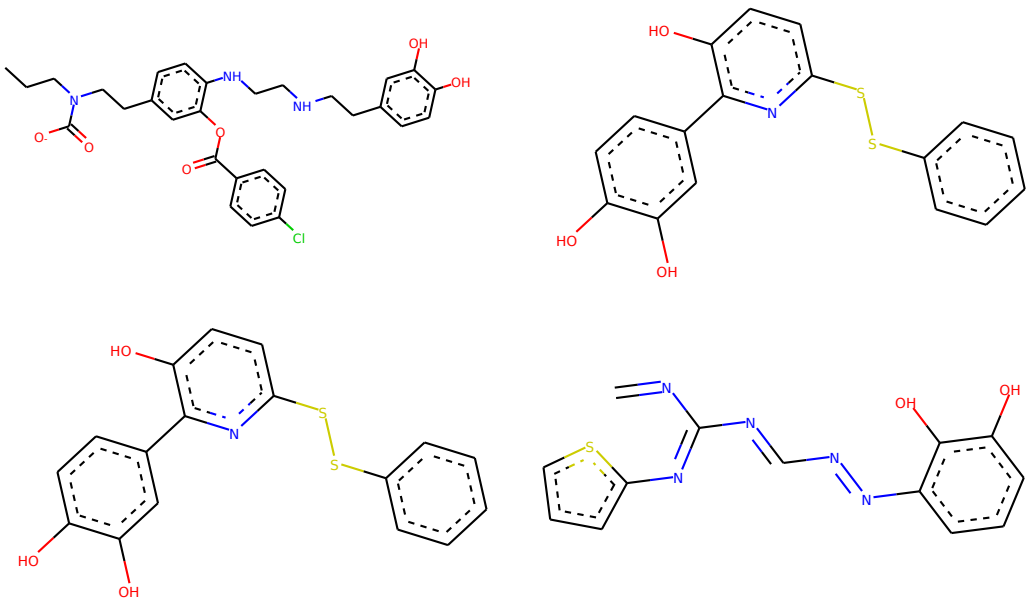
Figure S4: High scoring compounds generated for the DRD2 task by PS, with actives from the training set for comparison.



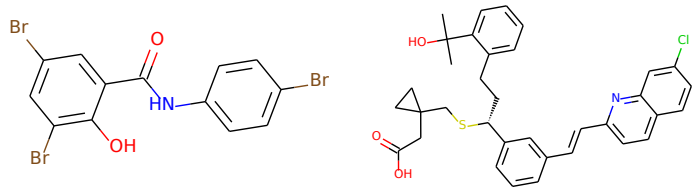
(a) Compounds generated by a graph-based GA



(b) Compounds generated by a SMILES-based LSTM

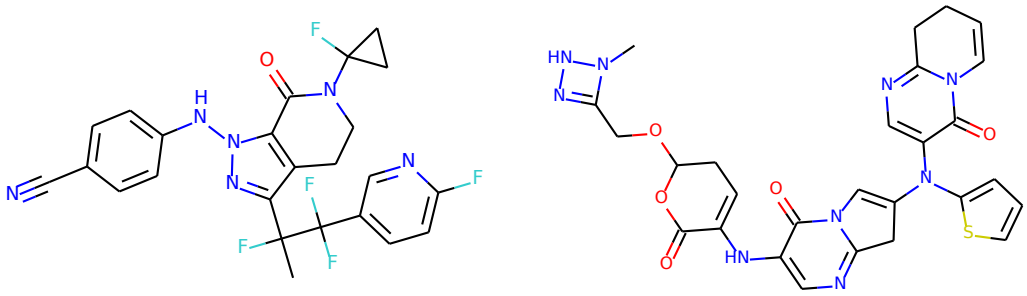


(c) Compounds generated by PS

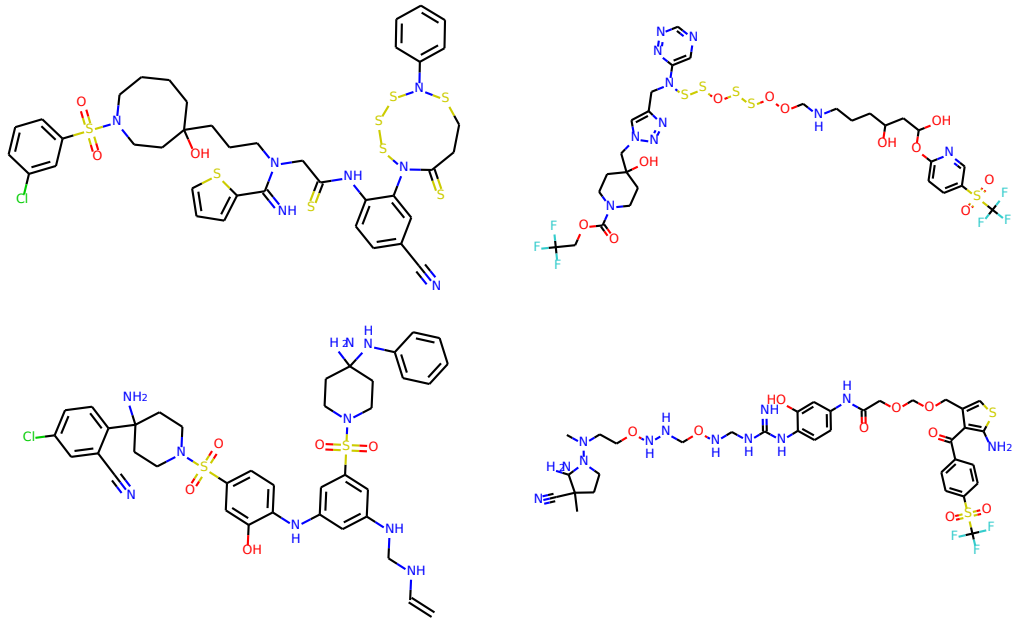


(d) Reference compounds from the training set

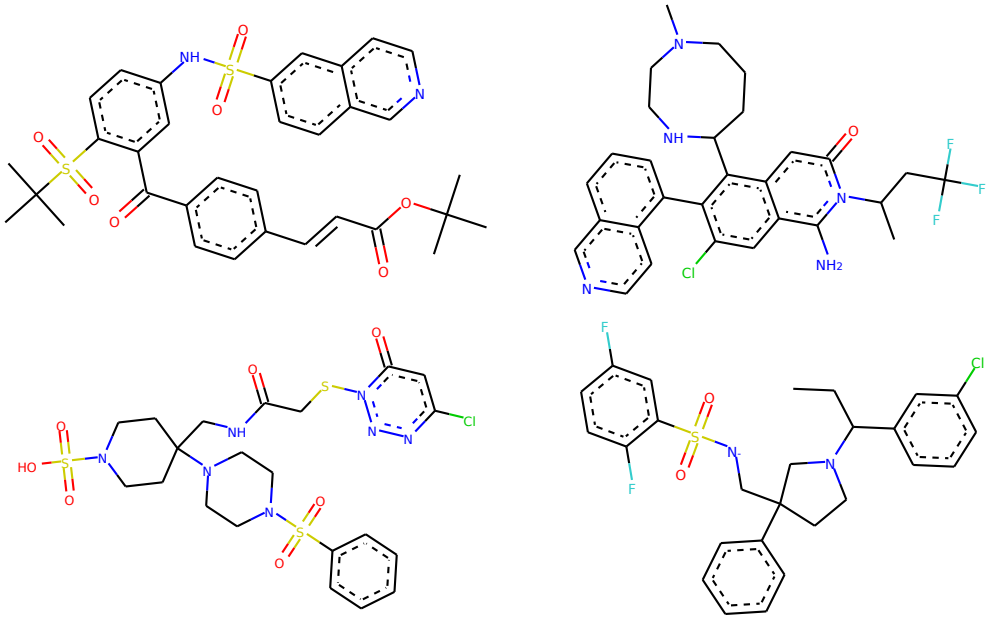
Figure S5: Random samples generated for the EGFR task, with actives from the training set for comparison.



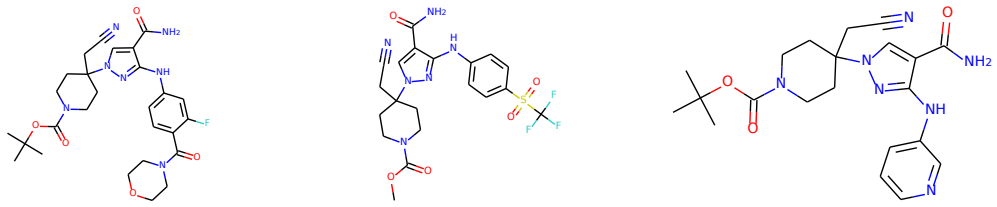
(a) Compounds generated by a graph-based GA



(b) Compounds generated by a SMILES-based LSTM

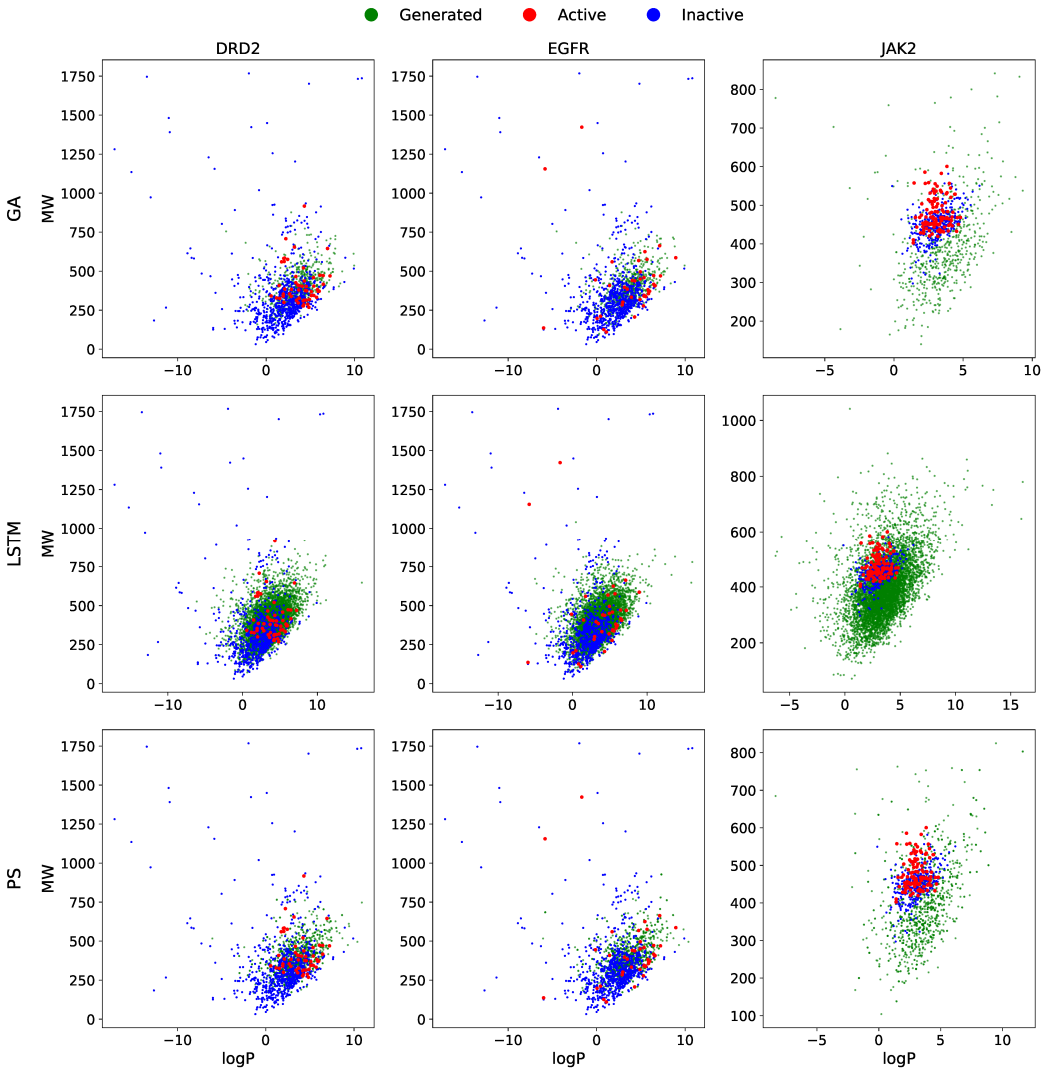


(c) Compounds generated by PS

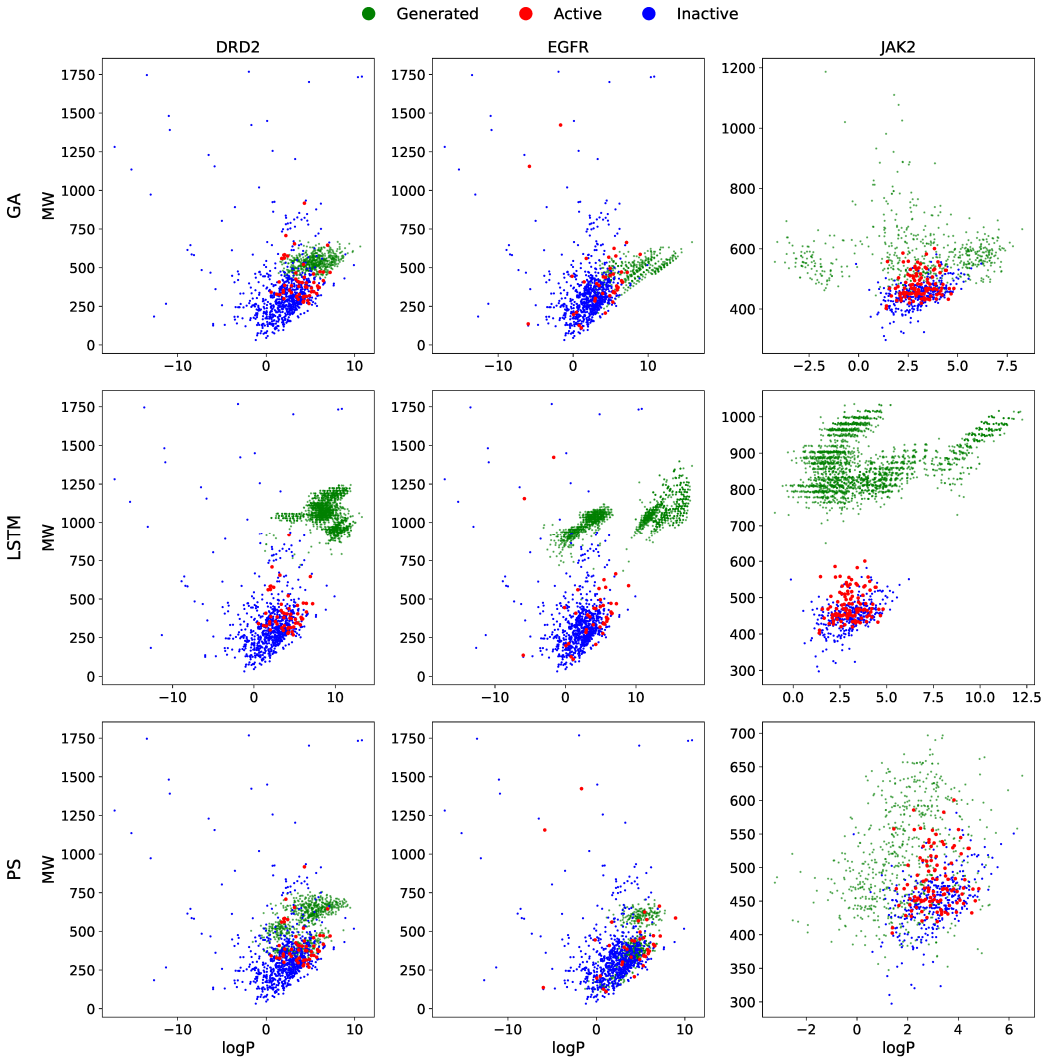


(d) Reference compounds from the training set

Figure S6: Random samples generated for the JAK2 task, with actives from the training set for comparison.

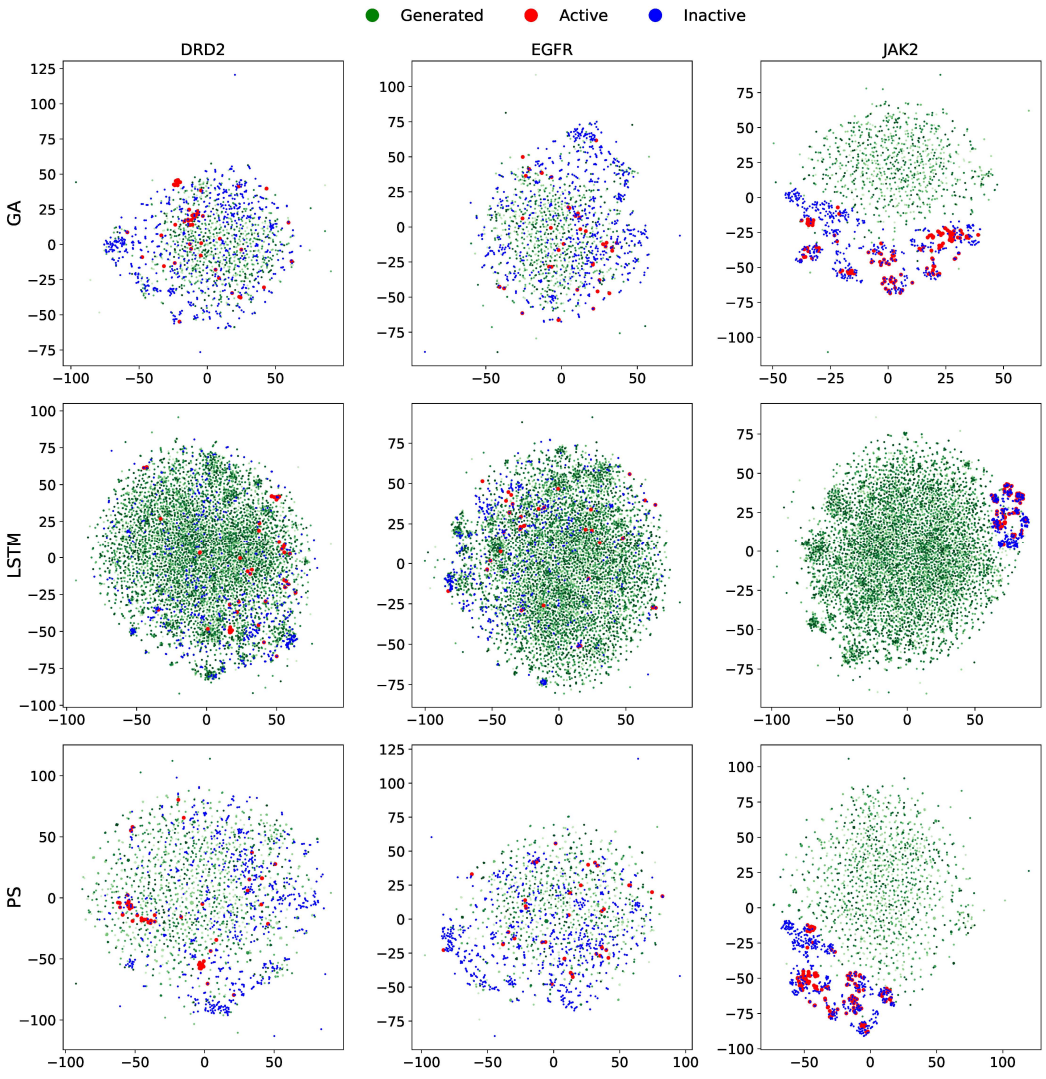



(a) First iteration

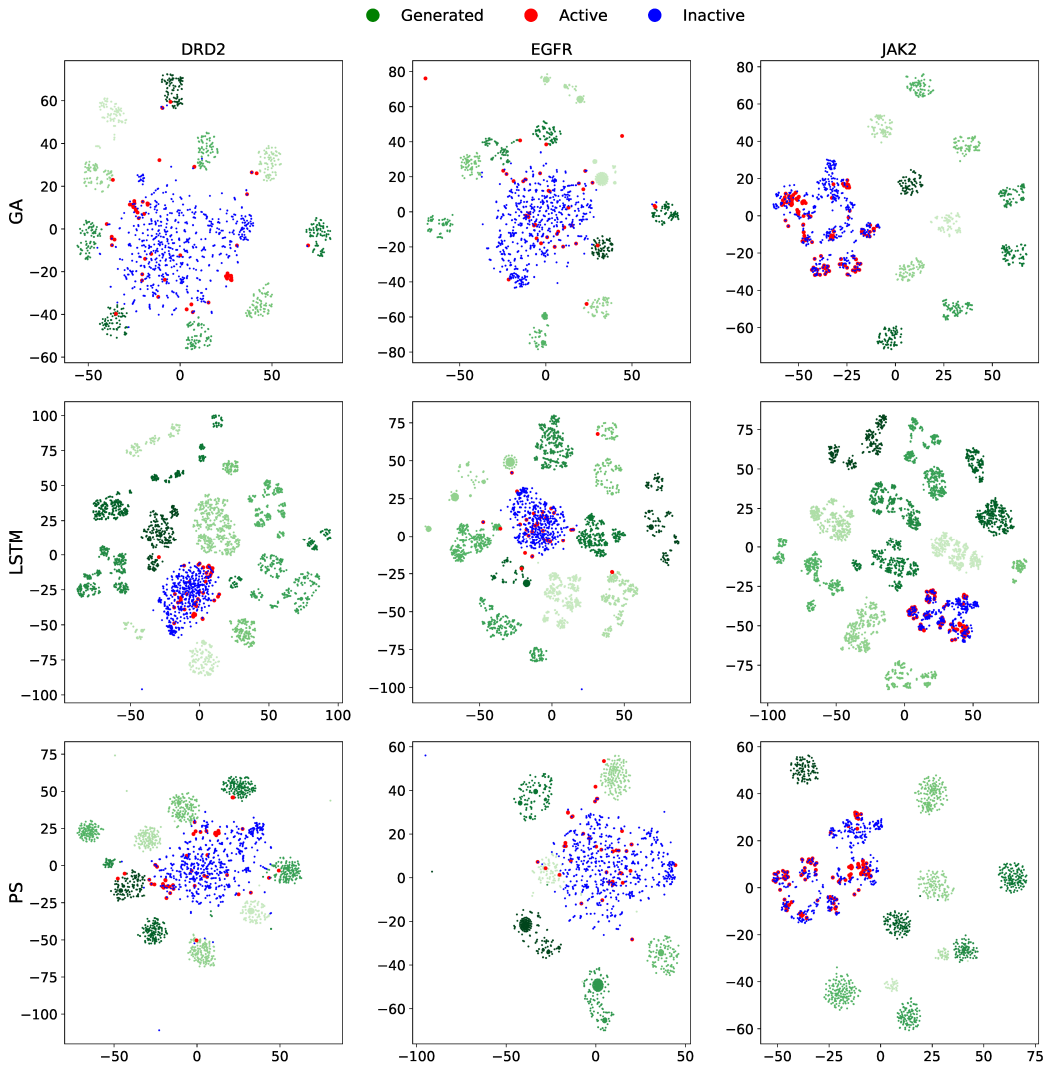



(b) Last iteration

Figure S7: Visualization of logP/MW-values for generated compounds, known actives and known inactives for different optimizers and tasks at the first and last iteration of optimization.




(a) First iteration




(b) Last iteration

Figure S8: t-SNE embedding for generated compounds, known actives and known inactives for different optimizers and tasks at the first and last iteration of optimization. The embedding is based on Tanimoto distance on the same ECFP4 fingerprints as used for training the classifiers. We used the t-SNE implementation from scikit-learn v0.22.1 [4] with standard settings and precomputed metric. The different shades for the generated molecules indicate independent runs.

- logP: water-octanol partition coefficient calculated according to [5] as implemented in [2]
- MW: molecular weight as implemented in [2]
- Length: length of SMILES strings
- Diversity: mean pairwise Tanimoto distance between generated molecules
- NN act: mean Tanimoto distance of generated molecules to nearest training active
- NN ina: mean Tanimoto distance of generated molecules to nearest training inactive

For all average Tanimoto distances we used a subset of size 100 of the generated compounds for computational reasons and binary ECFP4 fingerprints calculated in rdkit [2]. For all curves we first calculated the mean over the generated compounds in a run and then calculated the median and interquartile ranges depicted based on them.

There is a prominent increase in MW and SMILES length for the LSTM optimizer, of which the latter reaches its upper bound of 100 as stated in the optimizer settings. This trend can also be observed for the genetic algorithm, but to a lesser degree.

The diversity among the generated molecules decreases during optimization for LSTM and GA, while for PS it drops in the first iterations and then follows a more constant trend. The average distance to training actives also drops for GA and PS during optimization. For the LSTM an interesting pattern can be observed, where this metric first de- and then increases again.

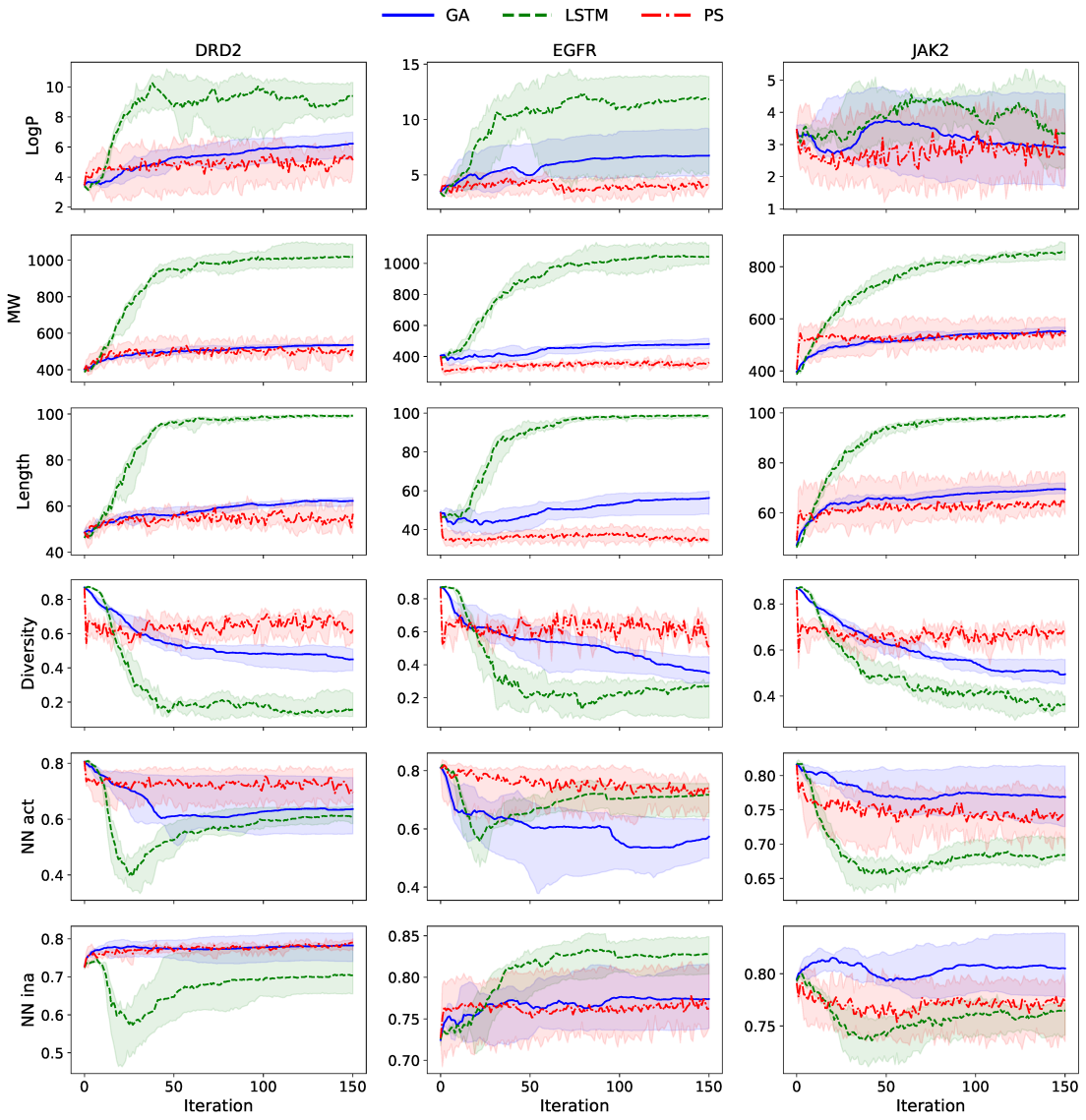



Figure S9: The course of additional metrics during optimization. The bold lines corresponds to medians and the shaded areas correspond to the interquartile range, taken over the means of single runs.

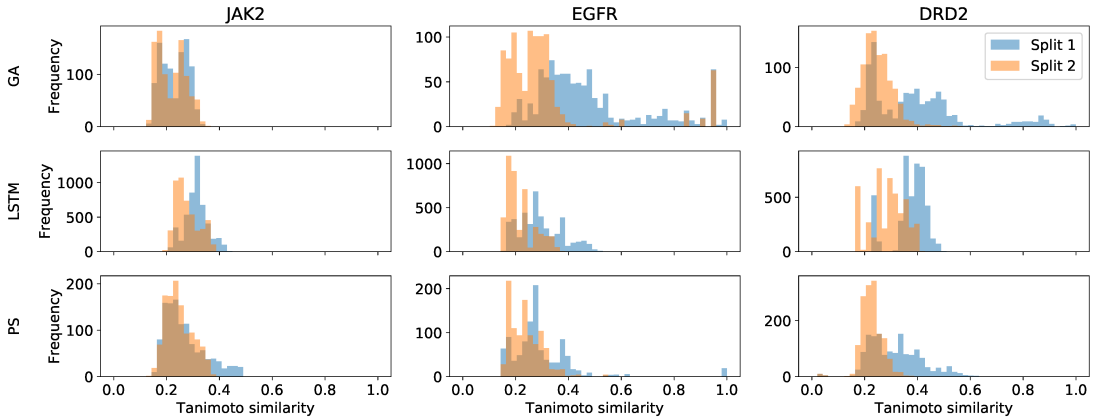



Figure S10: Distribution of nearest neighbour similarities of the optimized molecules to split 1/2. The results of all runs have been merged for each combination of data set and optimizer. In most cases the nearest neighbour similarities are greater for split 1 than for split 2. Tanimoto similarities are based on the same ECFP4 fingerprints that were used for training the classifiers.

# References


[1] Weininger, D. "SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules". In: J Chem Inf Comput Sci 28.1 (1988), pp. 31–36. URL: https://doi.org/10.1021/ci00057a005.

[2] Landrum, G. RDKit: Open-Source Cheminformatics. 2006. URL: http://www.rdkit.org.

[3] Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. "GuacaMol: Benchmarking Models for de Novo Molecular Design". In: J Chem Inf Model 59.3 (2019), pp. 1096–1108. URL: https://doi.org/10.1021/acs.jcim.8b00839.

[4] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. "Scikit-Learn: Machine Learning in Python". In: J Mach Learn Res 12.85 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

[5] Wildman, S. A. and Crippen, G. M. "Prediction of Physicochemical Parameters by Atomic Contributions". In: J Chem Inf Comput Sci 39.5 (1999), pp. 868–873. URL: https://doi.org/10.1021/ci9903071.