# 1. Method

## 1.1. Dataset

Compounds active against the activity classes were extracted from ChEMBL database (version 29) 1 based on the following criteria; less than 1000Da, target confidence score of 9, interaction relationship type ‘D’, an exact potency value given as *Ki* or *Kd* values. In this study, MMP-cliffs 2 were used as the definition of ACs with target independent criteria 3. MMPs with the potency difference of greater than , where and represent mean and standerard deviation of the *pKi* values in the corresponding activity classes, were defined as ACs and those of less than one was defined as non-ACs. The other MMPs with the potency difference of greater than one and less than the criteria were discarded. MMPs were generated with computationally efficient algorithm using Husian-Lee approach 4 implemented in the previous study 5. For MMP generation, substructure exchange was limited in no more than 13 heavy atoms and maximum difference between substituent of a compound was limited in no more than eight heavy atoms. After MMP generation, MMPs having small core with less than 10 heavy atoms were discarded. Activity classes used in the further analysis were selected with two distinct criteria corresponds to the data splitting approach mentioned later. For ‘data leakage possibly included’ data set, activity classes with greater than 50 MMPs and having both AC and non-AC were used, which resulted in 100 activity classes. For ‘data leakage excluded’ data set, activity classes with at least 20 compounds labeled as AC were used, which resulted in 42 activity classes. The details of selected activity classes are provided in **Supplementally Table 1**.

## 1.2. MMP fingerprints

To represent an MMP as a single feature vector, MMP fingerprints 6 which is a concatenated feature vector of individual molecular representation for each substructure were used. Extended connectivity fingerprints of bond diameter 4 (ECFP4) 7 were used as molecular representation for the substructures in present study. ECFP4 for the core and two substituents were calculated for each substructure. During these fingerprints calculation, features within bond diameter of one were eliminated from ECFP feature collection to clarify the contributions of features over bond diameter of two. For each part, identifiers corresponding to the features were sorted in ascending order and assigned to bits in the fingerprints vectors in the same order to prevent feature collision and make features contribute to AC prediction as many as possible. After the ECFP4 calculation for the substructures, XOR operation and AND operation was applied to the two substituent fingerprints to represent unique and common features in the two substituents separately and focus on the chemical transformation. Finally, three fingerprints representing features of core, unique features between substituents, and common features between substituents were concatenated into one vector. MMP fingerprints calculation was conducted with inhouse Java and Python scripts based on *OEChem toolkit* 8.

## 1.3. Condensed graph of reaction representation

To apply graph neural network approach to AC prediction, An MMP was represented in a single graph using the condensed graph of reaction (CGR) approach 9,10. The CGR formalism was originally conceived to combine reactants and products graphs based on a superposition of invariant parts. The resulting CGR forms a completely connected graph in which each node represents an atom and each edge a bond. In a CGR, the shared core of an MMP and the two exchanged substituent fragments form a single pseudo-molecule. The subgraph coming from the low potent compound was connected with the core via a single bond and the high potent compound a hypothetical zero-order bond. An MMP was converted into a pseudo-molecule using an in-house Python script with the *RDKit* API 11.

## 1.4. Experimental setup

To find best approach for AC prediction, four fingerprints-based approaches which were support vector machine (SVM), extreme gradient boosting (XGboost), random forest, and fully connected neural network (FCNN), and a graph-based approach which was message passing neural network (MPNN) were used. For FCNN and MPNN, two distinct models which differ at molecular representation for the input were build. As control calculations, nearest neighbor (1NN) and 5NN were also compared. All models except the two neural networks were implemented with scikit-learn 12 and the neural newtworks *PyTorch* 13. Hyperparameters of models listed in **Supplementally Table 2** were optimized with *Optuna* library 14. The hyperparameter search with *Optuna* for each model was performed for 100 times. The rest of parameters not listed in **Supplementally Table 2** except C in SVM were set to their default.

### 1.4.1. Support vector machine

SVM is a supervised learning method that aims to define the hyperplane separating given training instances with two class labels with maximizing the margin from the hyperplane 15. SVM was originally developed as linear classification method. If linear classification is not possible, SVM can be easily extend to nonlinear classification using kernel functions. In present study, MMPkernel 6 which is a product of two individual Tanimoto kernel 16 calculating core-wise and substituent-wise similarity was used as the kernel function. The parameter ‘class\_weight’ was set to ‘balanced’. The hyperparameter C was selected from a value from to divided into equal 10 steps in leaner log space with grid search.

### 1.4.2. Random forest

RF is a supervised learning method that is an ensemble of multiple decision trees generated from randomly chosen training instances using bootstrapping 17. RF predict for test instances based on the majority of the individual prediction of decision trees. The parameter ‘class\_weight’ was set to ‘balanced’.

### 1.4.3. Extreme gradient boosting

XGboost is a supervised learning method that is also ensemble of decision trees using gradient boosting 18. Gradient boosting iteratively generates decision trees so that each decision trees minimize the residual error from previous models. XGBoost is a computationally efficient and accurate extension of gradient boosting, which is achieved by parallelizing the decision tree construction.

### 1.4.4. Fully-connected neural network

A FCNN consists of a series of connected perceptrons stored in several layers. Each perceptron receives signals from previous layer transform into scaler value with activation function and send it to next layer as a signal. In this study, two distinct FCNNs were implemented based on their input; FCNN with single feature vector represented by MMPfingerprints (simply called as FCNN), FCNN with three separated ECFP4 representing core and two substituents (FCNN\_sep). For FCNN, input MMPfingerprints were mapped into probability indicating how likely the MMP forms AC. The number of nodes in hidden layers was monotonically reduced. In FCNN\_sep, the individual fingerprints were input to several hidden layers and the outputs were concatenated into a vector, which was sent to following hidden layers and transformed into the probability with softmax layer. The number of nodes in hidden layers for both individual substructures and concatenated feature vectors were also monotonically reduced. Rectified Linear Unit (ReLU) 19 was used as activation function except final layer. Binary cross-entropy with balance factor weighted by the number of positive samples was used as loss function and the loss was optimized by the Adam optimizer 20. This learning rate was managed by optim.lr\_scheduler.StepLR scheduler in PyTorch. For the scheduler, the parameter gamma was set to 0.1, and the step size, which is defined by dividing the number of training iterations by the number of steps, is a hyperparameter. The batch size was set to 128 if the number of MMPs in a training set was greater than 128; otherwise, it was equaled to the size of training size. Training steps were performed for 50 epochs during the hyperparameter search and for 100 epochs during best parameter fitting.

### 1.4.5. Message passing neural network

MPNN is a graph convolutional neural network approach which accepts molecular graph as the input and can learn the way to convert input molecular graphs to its optimal feature vectors. During MPNN training step, feature vector on each atom iteratively merged with information from its neighbor atoms and bonds so as to minimize the loss function. The initial features on each atom and bonds are listed in **Supplementally Table 3**. The transformed feature vectors of each atom are merged into single vector and connected to fully-connected neural network with several hidden layer whose output was the probability. In present study, the MPNN architecture referenced previous study 21 originally proposed by Tang et al 22. Same as FCNN and FCNN\_sep, two distinct MPNN were implemented corresponding to their input; MPNN with single CGR (simply called as MPNN), MPNN with three separated subgraphs representing core and two substituents (MPNN\_sep). In MPNN\_sep, feature vectors for each substructure were calculated with individual MPNN and concatenated into single vector which connected to fully-connected neural network. Activation function, loss function, optimizer, scheduler of optimizer, batch size, epochs, and the number of hyperparameter search were set to same setup as FCNN.

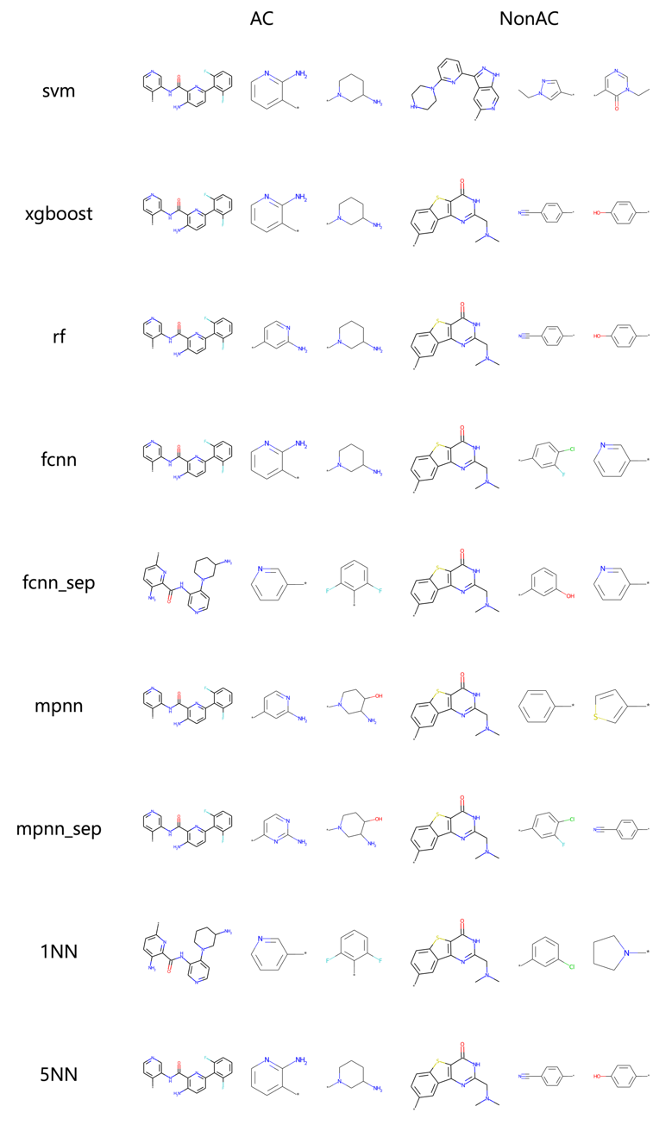
### 1.4.6. Model construction and hyperparameter optimization

The models were built for each activity class. When a data set randomly is separated into training and test set, one compound could be shared among several MMPs. This shared compound appears in both training and test set simultaneously, which causes that a MMP in the training set having compounds shared with a test MMP would be highly focused in the similarity evaluation between training and test samples. This data leakage prevents to evaluate whether the models have learnt chemical features or not. To consider influence from data leakage, two data splitting approach were conducted. In ‘data leakage possibly included’ splitting, data sets were randomly separated into training set (80%) and test set (20%). In ‘data leakage excluded’ splitting, advanced cross-validation (AXV) approach proposed by Horvath et al 10. was referenced. First, for each activity class, compounds used for making MMPs were collected and *n* compounds were randomly chosen as ‘kept-out’ pool. Then, MMPs were recursively selected and assigned to training or test set. If neither compound of the MMP was shared with kept-out pool, the MMP was assigned to training set. If both compounds were in kept-out pool, the MMP was assigned to external test set. If one compound was in kept-out pool, the MMP was no longer used for the further analysis. Both data splits were performed three times for each activity class. The pairs of training set and external test set were common among machine learning models using same seeds. For each data splitting, machine learning models were trained with three-fold internal cross-validation to optimize hyperparameters of selected machine learning methods. The performances of the machine learning models were calculated by taking average of three trials using different data splits.

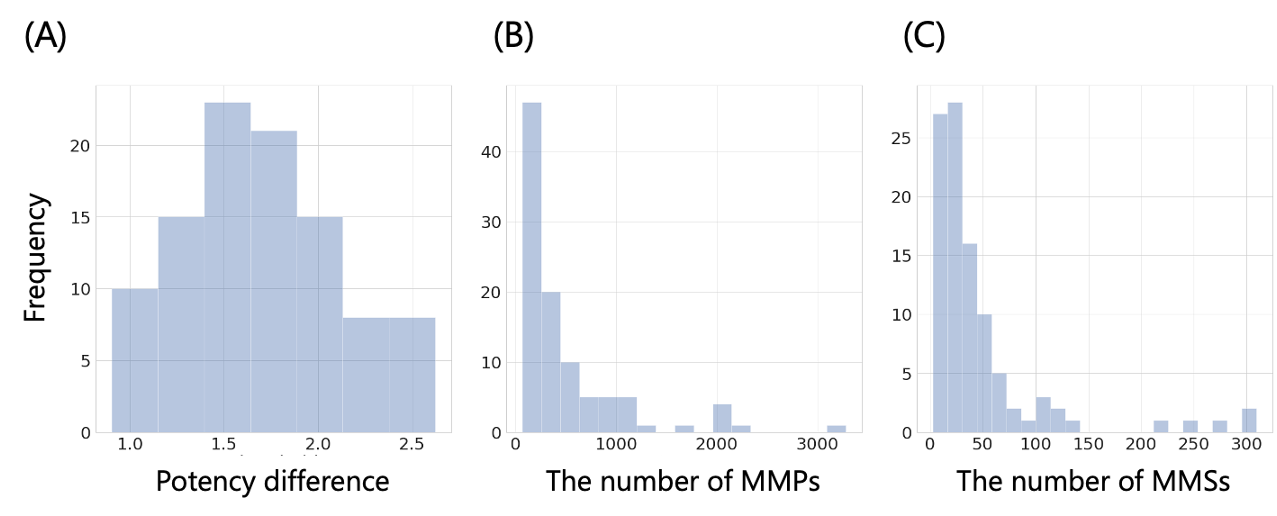
## 1.5. Model measurements

To evaluate performance of the models, balanced accuracy (BA) 23, recall, precision, and Matthew’s correlation coefficient (MCC) 24 were calculated. The four measures are defined below,

where TP, TN, FP, FN, TPR, and TNR stand for true positives, true negatives, false positives, false negatives, true positive rate, and true negative rate, respectively.

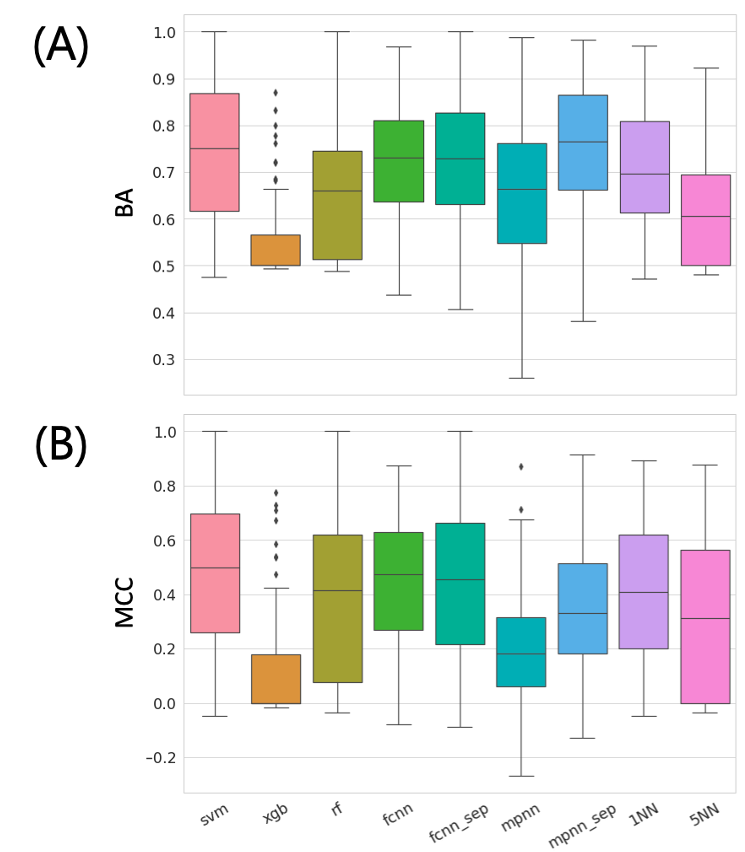
**Figure 1. Exemplary MMPs**

Exemplary MMPs which were accurately predicted as AC or non-AC by each machine learning approach are depicted. These MMPs are composed of compounds active against ChEMBL4523.



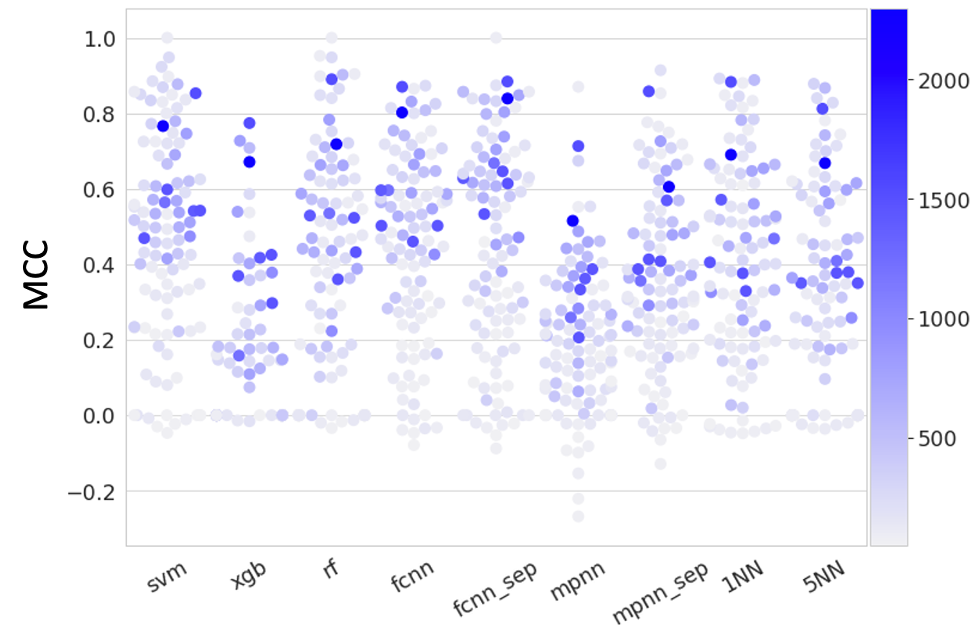
**Figure 2. Histograms of activity classes**

Histograms of (A) the potency difference criteria for target-independent AC definition, (B) the number of MMPs in an activity class and (C) the number of MMSs in an activity class are shown.



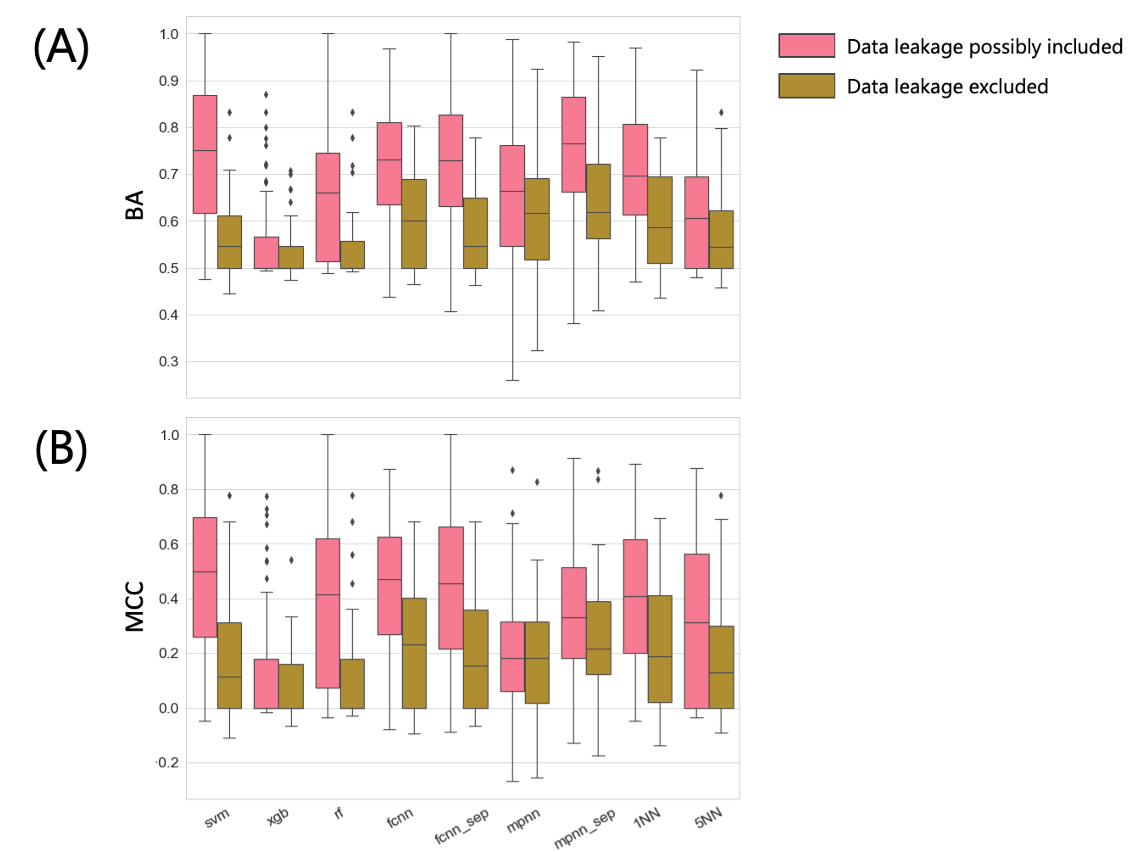
**Figure 3. Comparison of overall performances**

The performance distributions for nine machine learning approach using 100 activity classes are reported in bar plot. The performances were evaluated with (A) balanced accuracy (BA) and (B) Matthew’s correlation coefficient (MCC). Each model was built with data-leakage possibly included (randomly separated) training set.



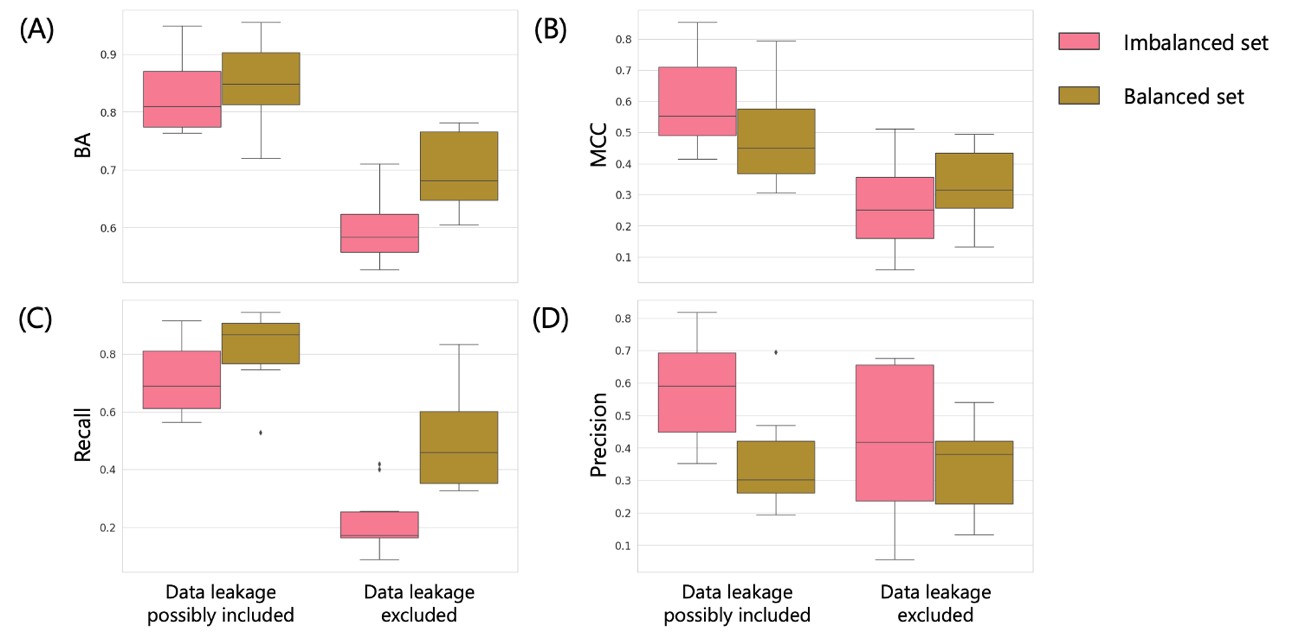
**Figure 4. Effect of training set size**

Swarm plot of MCC values for nine machine learning approach using 100 activity class are colored with blue based on the number of MMPs in the training set. The darker the color is, the more MMPs is in the training set.



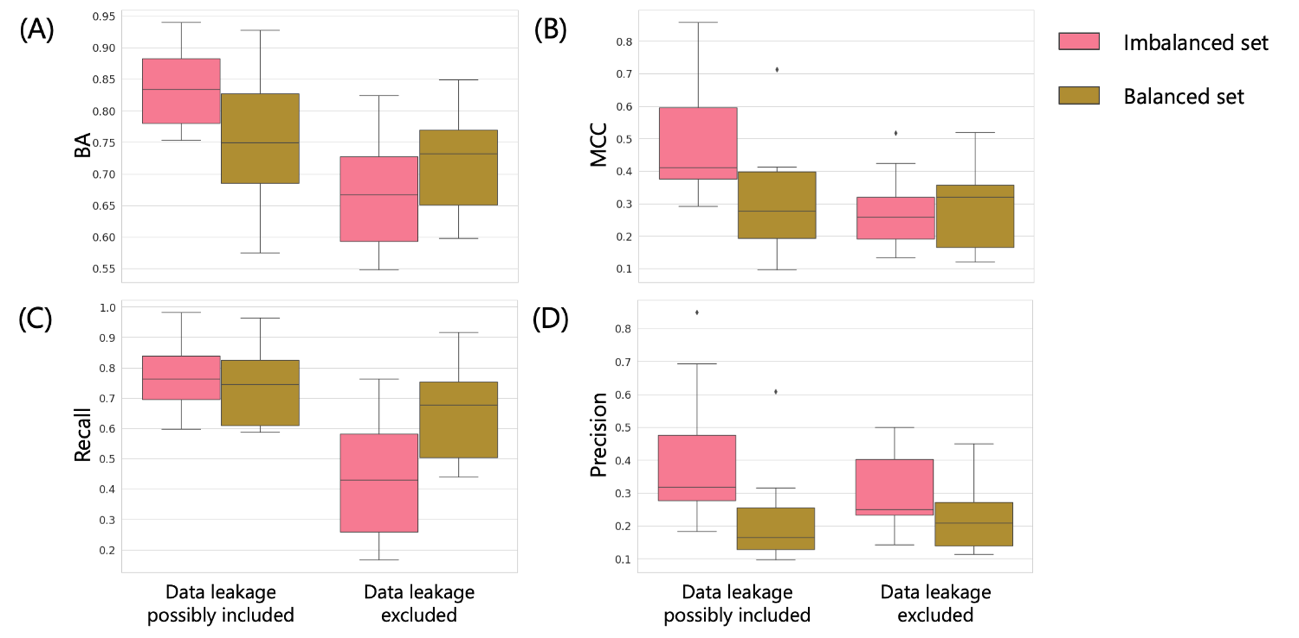
**Figure 5. Comparison of data leakage**

The performance distributions with two distinct data split approach are reported in bar plot. The red bars indicate the performances with data leakage possibly included set and brawn data leakage excluded. The performances were evaluated with (A) balanced accuracy (BA) and (B) Matthew’s correlation coefficient (MCC). The performance distributions for 42 activity classes which were common in the two data set were compared.



**Figure 6. Effect of data set balance for SVM**

The performances with SVM between balanced and imbalanced set were compared in bar plot. The models were built for top 10 activity classes with AC MMPs. The performances were evaluated with (A) BA, (B) MCC, (C) recall, and (D) precision. Generating balanced set was performed for both types of data set; left column: data leakage possibly included set, right: data leakage excluded set. Balanced set was generated by randomly selecting samples from imbalanced set so that the number of AC and non-AC MMPs were same.



**Figure 7. Effect of data set balance for MPNN**

The performances with SVM between balanced and imbalanced set were compared in bar plot. The models were built for top 10 activity classes with AC MMPs. The performances were evaluated with (A) BA, (B) MCC, (C) recall, and (D) precision. Generating balanced set was performed for both types of data set; left column: data leakage possibly included set, right: data leakage excluded set. Balanced set was generated by randomly selecting samples from imbalanced set so that the number of AC and non-AC MMPs were same.

(1) Mendez, D.; Gaulton, A.; Bento, A. P.; Chambers, J.; De Veij, M.; Félix, E.; Magariños, M. P.; Mosquera, J. F.; Mutowo, P.; Nowotka, M.; Gordillo-Marañón, M.; Hunter, F.; Junco, L.; Mugumbate, G.; Rodriguez-Lopez, M.; Atkinson, F.; Bosc, N.; Radoux, C. J.; Segura-Cabrera, A.; Hersey, A.; Leach, A. R. ChEMBL: Towards Direct Deposition of Bioassay Data. *Nucleic Acids Res* 2018, *47* (D1), gky1075-. <https://doi.org/10.1093/nar/gky1075.>

(2) Hu, X.; Hu, Y.; Vogt, M.; Stumpfe, D.; Bajorath, J. MMP-Cliffs: Systematic Identification of Activity Cliffs on the Basis of Matched Molecular Pairs. *J Chem Inf Model* 2012, *52* (5), 1138–1145. <https://doi.org/10.1021/ci3001138.>

(3) Hu, H.; Stumpfe, D.; Bajorath, J. Second-Generation Activity Cliffs Identified on the Basis of Target Set-Dependent Potency Difference Criteria. *Future Med Chem* 2019, *11* (5), 379–394. <https://doi.org/10.4155/fmc-2018-0299.>

(4) Hussain, J.; Rea, C. Computationally Efficient Algorithm to Identify Matched Molecular Pairs (MMPs) in Large Data Sets. *J Chem Inf Model* 2010, *50* (3), 339–348. <https://doi.org/10.1021/ci900450m.>

(5) Wawer, M.; Bajorath, J. Local Structural Changes, Global Data Views: Graphical Substructure−Activity Relationship Trailing. *J Med Chem* 2011, *54* (8), 2944–2951. <https://doi.org/10.1021/jm200026b.>

(6) Heikamp, K.; Hu, X.; Yan, A.; Bajorath, J. Prediction of Activity Cliffs Using Support Vector Machines. *J Chem Inf Model* 2012, *52* (9), 2354–2365. <https://doi.org/10.1021/ci300306a.>

(7) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *J Chem Inf Model* 2010, *50* (5), 742–754. <https://doi.org/10.1021/ci100050t.>

(8) *OEChem TK*; OpenEye Scientific Software Inc, Santa Fe.

(9) Luca, A. de; Horvath, D.; Marcou, G.; Solov’ev, V.; Varnek, A. Mining Chemical Reactions Using Neighborhood Behavior and Condensed Graphs of Reactions Approaches. *J Chem Inf Model* 2012, *52* (9), 2325–2338. <https://doi.org/10.1021/ci300149n.>

(10) Horvath, D.; Marcou, G.; Varnek, A.; Kayastha, S.; León, A. de la V. de; Bajorath, J. Prediction of Activity Cliffs Using Condensed Graphs of Reaction Representations, Descriptor Recombination, Support Vector Machine Classification, and Support Vector Regression. *J Chem Inf Model* 2016, *56* (9), 1631–1640. <https://doi.org/10.1021/acs.jcim.6b00359.>

(11) *RDKit: Cheminformatics and Machine Learning Software.* <http://www.rdkit.org/> (accessed 2022-10-03).

(12) Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Müller, A.; Nothman, J.; Louppe, G.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, É. Scikit-Learn: Machine Learning in Python. *J Mach Learn Res* 2011, *12* (85), 2825–2830.

(13) Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*; Vancouver, Canada, 2019; Vol. 32.

(14) Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A Next-Generation Hyperparameter Optimization Framework; The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; Anchorage, AK, USA, 2019; pp 2623–2631. <https://doi.org/10.1145/3292500.3330701.>

(15) Vapnik, V. N. The Nature of Statistical Learning Theory. 2000. <https://doi.org/10.1007/978-1-4757-3264-1.>

(16) Ralaivola, L.; Swamidass, S. J.; Saigo, H.; Baldi, P. Graph Kernels for Chemical Informatics. *Neural Networks* 2005, *18* (8), 1093–1110. <https://doi.org/10.1016/j.neunet.2005.07.009.>

(17) Breiman, L. Random Forests. *Mach Learn* 2001, *45* (1), 5–32. <https://doi.org/10.1023/a:1010933404324.>

(18) Friedman, J. H. Greedy Function Approximation: A Gradient Boosting Machine. *Ann Statistics* 2001, *29* (5). <https://doi.org/10.1214/aos/1013203451.>

(19) Vinod, N.; E, H. G. Rectified Linear Units Improve Restricted Boltzmann Machines. *Rectified Linear Units Improve Restricted Boltzmann Machines*; 2010; pp 807–814.

(20) Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. *Arxiv* 2014. <https://doi.org/10.48550/arxiv.1412.6980.>

(21) Maeda, I.; Sato, A.; Tamura, S.; Miyao, T. Ligand-Based Approaches to Activity Prediction for the Early Stage of Structure–Activity–Relationship Progression. *J Comput Aid Mol Des* 2022, *36* (3), 237–252. <https://doi.org/10.1007/s10822-022-00449-2.>

(22) Tang, B.; Kramer, S. T.; Fang, M.; Qiu, Y.; Wu, Z.; Xu, D. A Self-Attention Based Message Passing Neural Network for Predicting Molecular Lipophilicity and Aqueous Solubility. *J Cheminformatics* 2020, *12* (1), 15. <https://doi.org/10.1186/s13321-020-0414-z.>

(23) Brodersen, K. H.; Ong, C. S.; Stephan, K. E.; Buhmann, J. M. The Balanced Accuracy and Its Posterior Distribution; 2010 20th International Conference on Pattern Recognition; 2010; pp 3121–3124. <https://doi.org/10.1109/icpr.2010.764.>

(24) Matthews, B. W. Comparison of the Predicted and Observed Secondary Structure of T4 Phage Lysozyme. *Biochimica Et Biophysica Acta Bba - Protein Struct* 1975, *405* (2), 442–451. <https://doi.org/10.1016/0005-2795(75)90109-9.>