Model Training and Predictions:

For every SMILES (simplified molecular-input line-entry system), we generated the morgon circular fingerprint (1) as the feature vector using RDKit (2). Morgon fingerprinting is the process of taking every atom and its neighbors of an SMILES and using the presence or absence of a substructure as a feature. Because there are infinite possible features, we set the number of features to 2048 at the expense of sometimes not being able to determine which of multiple features is present. The way this is accomplished is by assigning a random number from 1 to 2048 to every possible feature and marking the ones present in each molecule. 2048 features is the default choice in RDKit and it is good enough to distinguish almost all the features present in our dataset. In our specific implementation, we used chiral fingerprints where 2 chiralities of the same molecule count as different structures. We made this decision because the chiralities of a molecule may effect its ability to bind to proteins. Kekulizeing a molecule is the act of taking into account the resonenses of various double bonds. When generating the molecule in RDKit, we set the SanitizeFlags.SANITIZE\_KEKULIZE off because we wanted to be able to run the code on some molecules that can not be kekulized.

We used scikit-learn (3) to make predictions of whether molecules bind to SweetTrac1 using the average classifications of 1000 SGD (stochastic gradient decent) classifiers with elastic net penalty. This is a linear model where a positive output means binding and a negative output means not binding. The predict\_proba function was used to calculate the predicted probabilities of weather the molecule binds. This clips the output between -1 and 1 and returns the percent of the way from -1 to 1 as the probability of binding. In order to save on computation time, we deleted any features that were identical across all samples (elastic net penalty already sets the weights of features with 0 variance to 0). This reduced the number of features to 795. We set the class weight of negatives to 0.005 because our training data set is very unbalanced. Without setting the class weight this way, the model has a strong bias toward predicting molecules as not binding. Using leave 1 sample out cross validation, the model has an accuracy of approximately 80% on positives and 85% on negatives. Using class weights of 1 each results in an accuracy of approximately 15% on positives and 99% on negatives. Weighting schemes between 0.005 and 1 tended to have the same problems. We chose to use an SGD classifier because models with a lot of parameters, like deep neural networks, fail to generalize when they are only trained on a small amount of data. Additionally, unlike random forest models, SGD classifiers can handle unbalanced data. The SGD classifier has 1 parameter for each of the 795 feature and a bias term. We have 173 molecules in our training dataset, which is less than the 796 parameters in the model. In order to deal with having more parameters than data, we used the elastic net penalty because it replaces a problem that has infinite potential solutions to a problem that has 1 solution. This works by trying to minimize the magnitudes of the parameters along with the error of the predictions. The elastic net penalty also has the property that it tries to use as little parameters as possible (by setting parameter weights to 0). This was a desirable property because it mimics the intuition to try to look for only the most important parameters and use them.

We also created an algorithm that assigns a number to each atom representing how useful it is in generating a positive prediction from our model. From here on, we will call a submolecule of a molecule any molecule that can be generated by removing a subset of its nonhydrogen atoms (and the bonds between them and any atom). We will also call any submolecule that is classified as a hit as a positive submolecule. When generating these submolecules, some are not going to be physically plausible, so RDKit is unable to kekulize them. The number we assign to each atom is the percent of positive submolecules that the atom could appear in when the submolecule is overlayed on the full molecule (Fig. \_\_\_\_\_\_\_\_\_). For example, lets say (C-C-O) is the molecule and (C-C-O), (C-O), (C O), and (C) are the positive submolecules. Then, the scores of the atoms would be (C: 3/4, C: 4/4, O: 3/4).

Additionally, we applied the model to the Santa Cruz Biotech stock list to identify potential positive molecules with binding affinity to SweetTrac1. We filtered out any molecules that we could not find the chiral SMILES for, the molecules that were substantially heavier than our training data set, any molecules more expensive than \_\_\_\_\_\_\_\_\_$. We ordered the molecules with a prediction value higher than 0.8.

All code is available in the supplemental as a Jupyter Notebook file (4). Fill in the training and testing dataset file names along with the desired file name for the substructure scores image and the desired file name for the test molecule hits list. The code outputs the image of the substructure scores of the training dataset as a .svg file and the hits in your test dataset as a .xlsx file.

(1) Rogers, David, and Mathew Hahn. "Extended-connectivity fingerprints." Journal of chemical information and modeling 50.5 (2010): 742-754.

(2) RDKit: Open-source cheminformatics. https://www.rdkit.org

(3) Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

(4) Kluyver, Thomas et al. "Jupyter Notebooks – A Publishing Format For Reproducible Computational Workflows". IOS Press, 2016, pp. 87-90., https://ebooks.iospress.nl/publication/42900.