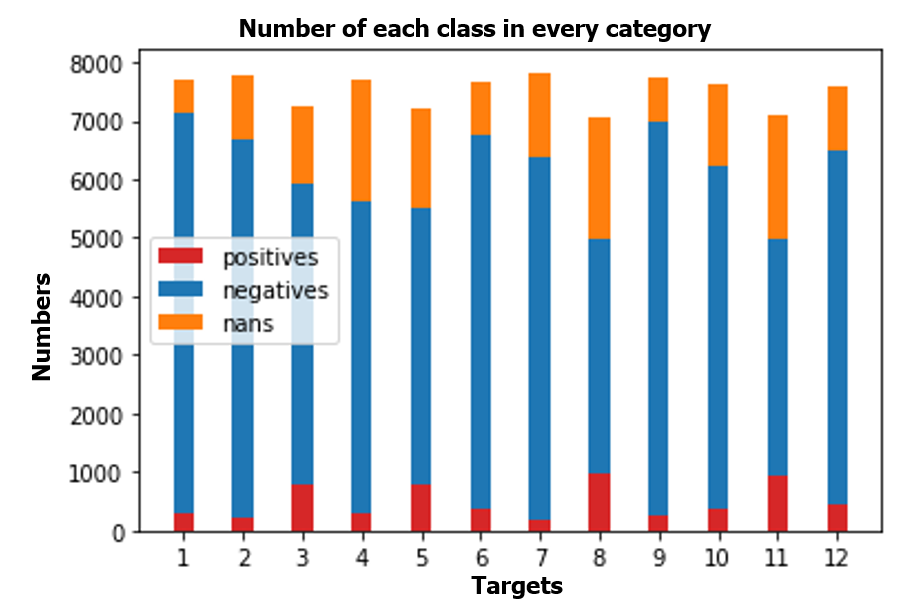
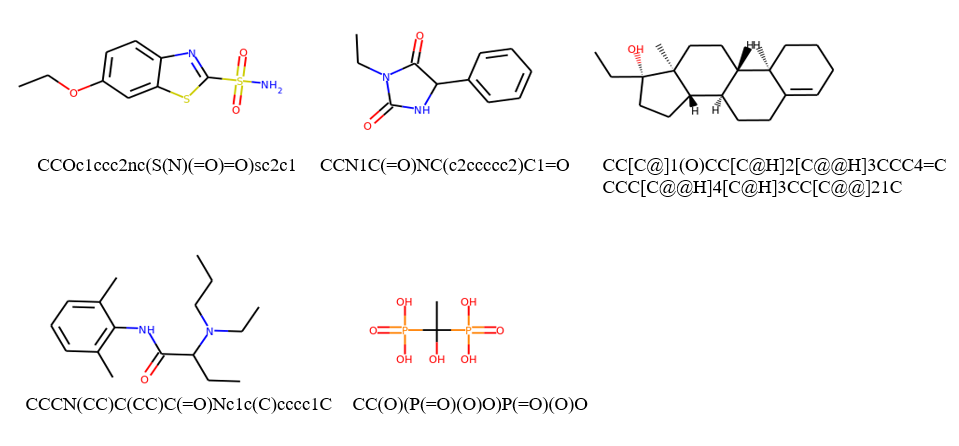
**1.Dataset**

The dataset contains 12 categories (Targets) and 8014 samples, some labels are missing. To see the proportion of the positives, negatives, and nans, I plotted the number of each for every category as **Fig 1**.



**Figure1. Number of positive samples/negative samples/nan samples for each of 12 categories.**

Samples are molecules presented by SMILE codes, the python tool RDKIT [1] gives us the graph of each sample, the first five molecules are printed as **Fig 2**.



**Figure 2. First five molecules(samples) images. The SMILE code is indicted below each graph.**

To find features for each sample, I used Morgan fingerprints generated by RDKIT tool based on Python3, setting the radius=4 and nBits=128, I collected 128 features for each modular.

**2. Method**

The multilabel problems are often solved by two ways. First is to transfer the problem to single label problems. The widely acknowledged algorithms include Binary Relevance (BR), Classifier Chains and Label Powerset (LP). Another option uses adapted algorithms such as MLKNN to perform classification on the whole data instead of separating into subsets. However, the hypothesis of BR method is that the categories are independent from each other [2]. I calculated the correlations between 12 targets using a nonparametric method, based on the values (Target labels), the Spearman correlation is calculated for every two targets. The table is shown as below (**Table 1**).

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Target1** | **Target2** | **Target3** | **Target4** | **Target5** | **Target6** | **Target7** | **Target8** | **Target9** | **Target10** | **Target11** | **Target12** |
| **Targer1** | 1 | 0.570943 | 0.363223 | 0.402888 | 0.383247 | 0.50608 | 0.362172 | 0.012799 | 0.51225 | -0.04087 | 0.279171 | 0.389903 |
| **Targer2** | 0.570943 | 1 | 0.387416 | 0.499178 | 0.490151 | 0.597708 | 0.585047 | 0.190506 | 0.607162 | 0.165739 | 0.398822 | 0.606424 |
| **Targer3** | 0.363223 | 0.387416 | 1 | 0.424276 | 0.379294 | 0.407978 | 0.368369 | 0.155468 | 0.464918 | 0.054814 | 0.361922 | 0.396124 |
| **Targer4** | 0.402888 | 0.499178 | 0.424276 | 1 | 0.517829 | 0.453682 | 0.518101 | 0.271817 | 0.532327 | 0.200056 | 0.442837 | 0.539414 |
| **Targer5** | 0.383247 | 0.490151 | 0.379294 | 0.517829 | 1 | 0.526956 | 0.464989 | 0.219449 | 0.510347 | 0.134184 | 0.396457 | 0.511885 |
| **Targer6** | 0.50608 | 0.597708 | 0.407978 | 0.453682 | 0.526956 | 1 | 0.503755 | 0.146571 | 0.584845 | 0.125635 | 0.386545 | 0.59913 |
| **Targer7** | 0.362172 | 0.585047 | 0.368369 | 0.518101 | 0.464989 | 0.503755 | 1 | 0.26366 | 0.589109 | 0.201905 | 0.463115 | 0.66455 |
| **Targer8** | 0.012799 | 0.190506 | 0.155468 | 0.271817 | 0.219449 | 0.146571 | 0.26366 | 1 | 0.162186 | 0.58254 | 0.216118 | 0.261533 |
| **Targer9** | 0.51225 | 0.607162 | 0.464918 | 0.532327 | 0.510347 | 0.584845 | 0.589109 | 0.162186 | 1 | 0.105138 | 0.421703 | 0.62873 |
| **Targer10** | -0.04087 | 0.165739 | 0.054814 | 0.200056 | 0.134184 | 0.125635 | 0.201905 | 0.58254 | 0.105138 | 1 | 0.118999 | 0.254795 |
| **Targer11** | 0.279171 | 0.398822 | 0.361922 | 0.442837 | 0.396457 | 0.386545 | 0.463115 | 0.216118 | 0.421703 | 0.118999 | 1 | 0.47024 |
| **Targer12** | 0.389903 | 0.606424 | 0.396124 | 0.539414 | 0.511885 | 0.59913 | 0.66455 | 0.261533 | 0.62873 | 0.254795 | 0.47024 | 1 |

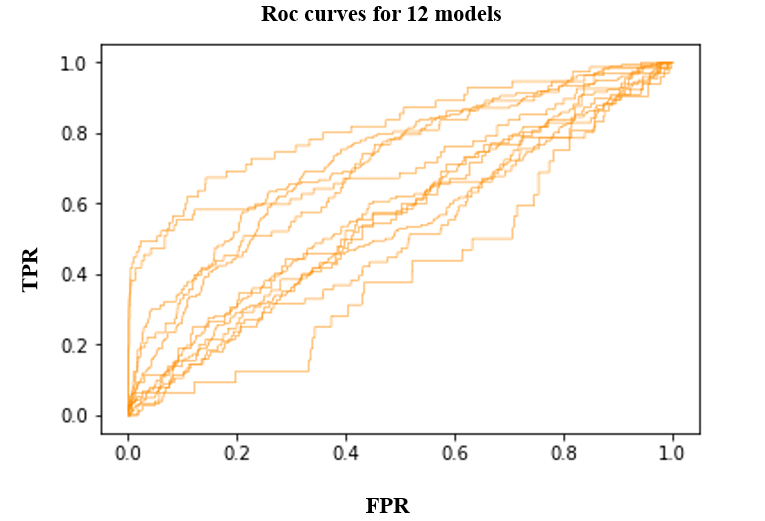
**Table 1.** The correlations of each 2 targets.

Other articles suggest that 40% of similarities between proteins are assumed to share same protein functions [3]. Some correlations in the above table are larger than 0.5, thus, I used Classifier Chains in this study. The classifier chain built 12 models for each label, and each next classifier is trained on not only the generated features, but also the precious classifiers in the chain. Thus, the classifier chain kept the label correlation into consideration and the order of labels can make a great change on model performance. For example, if we would like to know the prediction on target 5 only, we can re-order the labels in dataset and have ‘Target5’ as the last classifier in the chain, the algorithm will make use of other labels as features for model training.

I randomly split 80% of the whole dataset as training as the remaining 20% as testing, built by SkLearn[4] tool, I used SVM algorithm in classifier chain with linear kernel. Note that all missing labels are replaced to the negatives.

**3. Results**

The average accuracy is 64% on testing data and the ROC curves for each classifier is shown as **Fig 3**. The mean value of AUC is 0.615.



**Figure 3. ROCs for 12 models in classifier chain.**

Since we used SVM algorithm with linear kernel, we can interpret the feature significance by the absolute value of coefficients. I output the most significant features for each classifier and the 7th features showed significance among others.

**Reference:**

1. Landrum, G. (2006). RDKit: Open-source cheminformatics. 2006. Google Scholar.
2. Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. Machine learning, 85(3), 333.
3. Luo, Y., Zhao, X., Zhou, J., Yang, J., Zhang, Y., Kuang, W., ... & Zeng, J. (2017). A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information. Nature communications, 8(1), 573.
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.
5. Yu, H. F., Jain, P., Kar, P., & Dhillon, I. (2014, January). Large-scale multi-label learning with missing labels. In International conference on machine learning (pp. 593-601).

**Things I would have liked to try but haven’t had time to:**

1. Some articles proposed method for multilabel classifications with missing labels, they labeled the positives as (1), negatives as (0) and nan as (-1), I would like to try those published papers such as [5].
2. Function Draw.DrawMorganBit in Rdkit.Chem.Draw is supposed to give me the substructure of a modular for every bit, thus I can output the significant bit(7th feature in this experiment) that affects the prediction. But I keep getting the error module ‘rdkit.Chem.Draw’ has no attribute ‘DrawMorganBit’ when the user guide clearly clarified on that(http://www.rdkit.org/docs/GettingStartedInPython.html#morgan-fingerprints-circular-fingerprints).