Report on predicting molecule activity by the Chemprop algorithm

**Introduction**

Regarding the research on screening for novel antibacterial compounds in small molecule libraries we have this project to follow up on a new dataset with further analysis. This report includes prediction performance of applying directed message passing neural network (DMPNN) algorithm proposed by the Chemprop module [] on Library\_10 dataset. As the key differences between these analyses and previous published one [] regarding the computational aspects, we have two new approaches. First, is that we tried to embed new set of features generated by the Transformers that depicted in Figure 1. Second, we applied more comprehensive analyses dealing with completely unseen molecules in test sets for choosing the optimal cut-off on predicted probabilities by our model.

**Framework Overview**

Based on the **Figure 1**. This framework comprised by three main sections. First is the molecules preparation, we use SMILES version of small molecules to train our model. Second, we have three different methods of feature generation includes, 200 RDKit features, DMPNN features, and Transformer features.

Third, each of these features generated separately and for each set we have a linear layer of feed forward neural network that learns the pattern between these features and their labels. Finaly the model output a probability for each molecule that can binarized to active and inactive labels.

A diagram of a diagram

Description automatically generated

**Figure 1. Overview of the model.** This model receives SMILES version of small molecules and generates different types of features. Features include DMPNN, RDKit and Transformer ones. Next, the features use as input to the feed forward linear layers to do binary classification.

**Dataset overview**

The Library\_10 dataset includes around 26000 molecules that extracted from the open-source database ChEMBL. After applying data preprocessing steps include removing Nans and duplicates, we reached to exactly 26901 unique molecules. As the mainstream way of dividing dataset for machine learning tasks we divided the dataset randomly into 80:10:10 portions on train, validation, test data sources. In Figure 2, we showed the distribution of whole datasets based on the number of ‘Active’ and ‘Inactive’ ones.

A blue and orange pie chart

Description automatically generated

**Figure 2. Distribution of molecules in the Library\_10.** It shows that 34.9% of total 26901 molecules are active and 65.1% are Inactive.

**Results**

As we have different sets of features, we introduced different models that comes from different configuration of features. In this section we compared the results of each model to show up the power of each type of features or combination of them plus the feed forward neural network (FFNN).

Model-1: DMPNN features + FFNN

Model-2: RDKit + DMPNN features + FFNN

Model-3: RDKit + DMPNN + Transformer features + FFNN

Model-4: DMPNN + Transformer features + FFNN

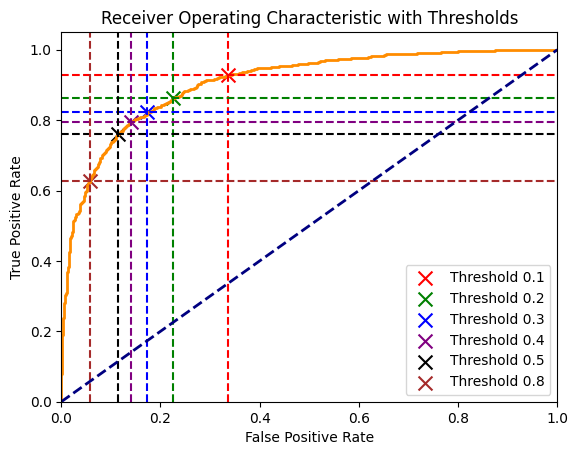
**Table 1. Results of different models with different configurations on Library\_10 dataset.** It reports AUROC and AUPRC scores after hyperparameters optimization.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | AUROC | AUPRC | F1-Score |
| Model-1 | 90.0% | 84.1% | 82.7% |
| Model-2 | 89.5% | 82.8% | 82.0% |
| Model-3 | 89.5% | 82.8% | 82.0% |
| Model-4 | **90.6%** | **84.7%** | **83.0%** |

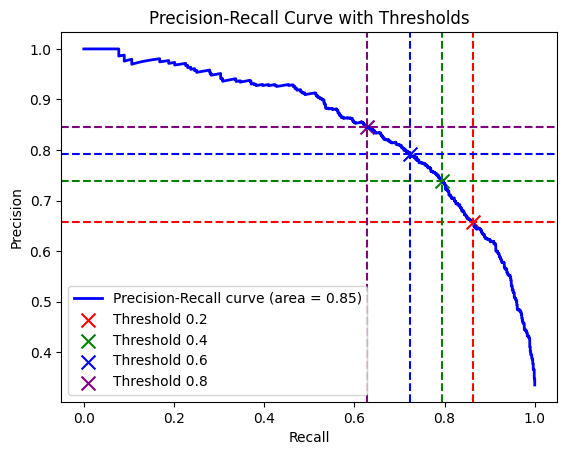
The highlighted scores are the best ones, and the underline shows the second-best scores.

Based on the reported results on **Table 1**. It seems that the Model-4 has the best performance. However, it uses less features compare with the other methods, but it outperforms the other methods by getting 0.6%, 1.1%, and 1.1% improvement to the Model-1, Model-2, and Model-3 respectively.

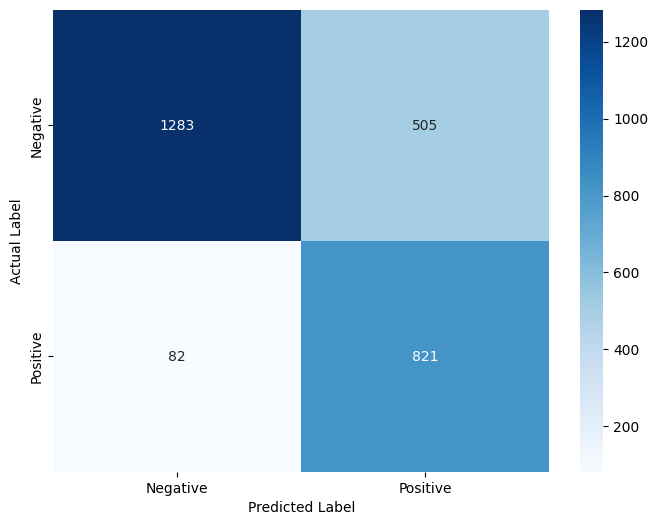
To have better insight about performance of the Model-4 we generated AUROC and AUPRC plots with highlighted thresholds in **Figure 3** and show up the confusion matrix of prediction on test set in **Figure 4**.



**Figure 3. The ROC-curve on test set predictions with the model.** The different thresholds show their affect on finding the optimal one to binarize the predicted values.



**Figure 4. The precision-recall curve.** This plot shows the affect of different thresholds that help to select the optimal one in a way that maximize the recall value.



**Figure 5. The confusion matrix on test set predictions.** This confusion matrix shows that our model could predict 821 out of 903 positive labels truly. Also, predicted 1283 out of 1788 negative labels truly.

As the positive or active labels are important for us, we tried to binarize the predicted probabilities in a way to minimize active labels loss. To demonstrate the result of that we plotted the Figure 5 that represents we missed only 82 out of 903 active labels which is an acceptable performance.

**Choosing the optimal cut-off**

When determining the optimal cutoff for predicted values by machine learning models, particularly to maximize the F1-score, a sophisticated approach involves leveraging the thresholds generated by the ROC curve method. This technique is highly effective in identifying a threshold that balances precision and recall, crucial for scenarios where both false positives and false negatives carry significant consequences.

Here's a concise explanation of the method:

1. **ROC Curve Thresholds:** Initially, you generate a list of potential thresholds using the ROC curve method. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The thresholds extracted from this curve represent different trade-offs between TPR and FPR, providing a comprehensive set of cutoff points to evaluate.
2. **Iterative F1-Score Calculation:** For each threshold derived from the ROC curve, you calculate the corresponding F1-score. This involves classifying the predicted probabilities into binary outcomes based on the current threshold and then computing precision (the proportion of true positive predictions to all positive predictions) and recall (the proportion of true positive predictions to all actual positives). The F1-score is then determined using these values, giving a single measure that balances both precision and recall.
3. **Optimal Threshold Selection:** After computing the F1-scores for all the thresholds provided by the ROC curve, the optimal threshold is identified as the one that maximizes the F1-score. This threshold best balances the model's precision and recall, making it ideal for situations where it's crucial to manage the trade-offs between detecting positive instances and avoiding false alarms.
4. **Model Performance Evaluation:** Finally, the optimal threshold is used to classify the predicted probabilities in the evaluation dataset, allowing for a realistic assessment of the model's performance based on the selected cutoff.

This approach, by integrating thresholds from the ROC curve with the maximization of the F1-score, ensures a data-driven method to fine-tune classification models for optimal performance, especially in domains where balancing the types of classification errors is critical. In our case, after applying this analysis we got the optimal cut-off as 0.118.

**Prediction on PubChem Database**

The PubChem dataset is a publicly available dataset of molecules with around 57 million molecules. In a random selection we chose 10k molecules to do prediction about their activity/inactivity. The challenge in doing binary prediction for completely unseen data without having the true labels is to choose the optimal cut-off. As a solution we tried to select the 10k molecules in a way that their feature`s distribution is in same range with the test set that we had before. To examine our hypothesis, we plot the distribution of the standard deviations of molecules in two datasets based on the Transformer features and MPNN features in **Figure 6** and **Figure 7** respectively.

A graph of a normal distribution

Description automatically generated

**Figure 6. The standard deviation comparison.** The plot displays the distribution of the standard deviations of each molecule for both datasets based on the Transformer features.

A graph of a standard deviation

Description automatically generated

**Figure 7. The standard deviation comparison.** The plot displays the distribution of the standard deviations of each molecule for both datasets based on the MPNN features.

Based on the similarity between test portion features and PubChem features distributions we may apply the same optimal cut-off for predicted probabilities of the PubChem dataset which was 0.118. After applying the 0.118 cut-off we got predicted binary values in a form that is percentage of 1s is 28.48% and percentage of 0s is 71.52% so it gave us around 2800 active molecules that may includes incorrect ones (False Positivses) but we are sure that we minimized missing true positives.

Update 15 Feb 2024

Based on the above plots, while there is some similarity in feature variability between the two datasets (as shown by the overlap). To get a fuller picture of how similar the datasets are in terms of molecular features, we should complement this analysis with direct comparisons of the feature vectors using similarity metrics like cosine similarity, Euclidean distance, etc. This can help us understand not only if the datasets have similar variability but also if they are similar in the actual features that represent molecular characteristics. The report of similarity scores reported in **Table 2**.

**More predictions with the trained model**

Two datasets named NatProd 1&2 which the first one has 45 SMILES and the other has 224200 molecules. Due to the large size of the second dataset it divided to five equal subsets. The feature distribution of the test set data and the NatProd datasets based on STD of each molecule provided in **Figure 8** and the Cosine similarity scores provided in **Table 2**. As the test set and NatProd datasets seems to be similar based on their features, we chose the cut-off equal to 0.118 as the same optimal cut-off used before.

**Table2. Cosine similarity scores between different subsets of NatProd and PubChem with test set used during the training based on Transformer features.**

|  |  |
| --- | --- |
| Dataset | Cosine similarity |
| Pubchem 10k | 0.091 |
| Subset 1 | 0.448 |
| Subset 2 | 0.587 |
| Subset 3 | 0.602 |
| Subset 4 | 0.607 |
| Subset 5 | 0.608 |

A group of graphs showing different sizes of data

Description automatically generated with medium confidence

**Figure 8. The standard deviation plots of five subsets of NatProd dataset with test set.**

**Update Feb 23**

After preparing test results on PubChem (10k molecules) and Natural product datasets we realized that our model is predicting very false positives that make us high costs to investigate their truthiness. To further investigate on the reason of high amount of false positives we used a trained model applied in the research done by Rahman et al 2022 (Chengyou`s model). This model trained with another dataset which we call Seline dataset which has a huge difference with our previously used dataset (Library\_10) and that is the balance between active and inactive molecules.

On top of that, we should mention that the model that trained on Rahman et al 2022 research and the model that we used are both same on main principle that is sing chemprop and DMPNN algorithm and our model used Transformer based features and trained based on random split of the dataset instead of scaffold split that done in Chengyou`s model. So, they are almost similar.

To have a better understanding of the data portions we have below statistics of the datasets and their results.

**Original Selin dataset:**   
actives: 0.866%

in actives: 99.13%

**Natural products dataset`s predictions using the model trained with Seline dataset (Chengyou`s model):**

actives: 2.53%

in actives: 97.47%

-------------------------------------------------------------------------------------------------------------------------------

**Original Library\_10 dataset:**

actives: 35.29%

in actives: 64.71%

**Natural products dataset`s predictions using the model trained with Library\_10 dataset (Hamid`s model):**

actives: 34.77%

in actives: 65.22%

As a result, my assumption about the reason of the problem with Library\_10 is about its balance and the similarity of its balance with what we expect to have. To be more specific, we expect to have so rare active compounds in a random dataset, but we use a dataset that includes lots of actives during training, so it seems that it has caused bias to the model. I am saying it seems because it doesn`t have specific rule and dealing with balancing of the dataset and it is relying on the problem`s details.

Regarding choosing the best dataset we should take into account below considerations:

**Considerations**

1. **Increased Challenge in Learning the Minority Class**: Reducing the representation of the active class will make it even harder for the model to learn to identify such cases accurately, potentially leading to a higher number of false negatives.
2. **Reflecting Real-world Distribution**: In some cases, making the dataset more unbalanced to reflect the real-world distribution of classes can help in preparing the model for how it will be applied in practice, especially if false positives are much more costly than false negatives.
3. **Precision vs. Recall Trade-off**: Be prepared for a trade-off between precision and recall. Increasing the imbalance might reduce false positives (increase precision) but at the cost of missing more true positives (decrease recall).

**Strategies**

1. **Precision-Recall Balance**: Focus on metrics that balance precision and recall, such as the F1 score, to evaluate your model. Given your interest in reducing false positives, you might prioritize precision but also keep an eye on recall to ensure that the model still captures a reasonable proportion of the active cases.
2. **Anomaly Detection Models**: If the active cases become significantly rare, consider using anomaly detection algorithms. These models are designed to identify rare events or observations and might be more effective in this context.
3. **Cost-sensitive Training**: Adjust the cost function to penalize false positives more heavily, encouraging the model to be more conservative in predicting the minority class. This can be a more direct way to control the model's behavior concerning the imbalance.
4. **Threshold Adjustment**: Adjust the decision threshold to be more stringent for classifying an observation as the active class. This can reduce false positives but will also affect the number of true positives.
5. **External Validation and Domain Expertise**: In such extreme cases, involving domain experts in the loop for validation and interpretation of the results becomes even more crucial. Experts might help in understanding the trade-offs and ensuring that the model's predictions align with practical needs and expectations.
6. **Use Ensemble Methods**: Ensemble methods that aggregate predictions from multiple models can sometimes be more robust to imbalance. Techniques like bagging and boosting might help in improving the detection of the minority class, even when its representation is further reduced.