Reproducible Modeling for the Tox21 Challenge: Exact Evaluation Protocol, Leakage Avoidance, and Benchmark Baselines

Shushank[1]

[1]Email: `shushankpawar664@gmail.com`

January 20, 2026

Abstract

This short technical report documents the evaluation protocol, anti-leakage safeguards, reporting recommendations, reproducibility artifact requirements, and a compact benchmark suite for submissions to the Tox21 leaderboard. The objective is to make leaderboard evaluation deterministic, comparable across submissions, and re-runnable by reviewers and maintainers. The report provides explicit preprocessing steps, split procedures, metric definitions, submission formats, and a minimum artifact checklist to accompany leaderboard submissions. The submitted model's benchmark results (task-wise ROC-AUCs and aggregate mean ROC-AUC) are reported in the Results section.

1 Introduction

Leaderboards for molecular property prediction enable rapid comparisons of modeling approaches, but they also introduce risks: inconsistent preprocessing, hidden test leakage, and irreproducible claims. To ensure fair and useful comparison on the Tox21 leaderboard, submissions must precisely document and provide artifacts that allow maintainers and reviewers to re-run and validate results. This document (i) specifies a deterministic evaluation protocol, (ii) lists explicit anti-leakage rules, (iii) prescribes clear reporting and artifact requirements, and (iv) provides a small set of baseline methods that every submission should include for context.

This report is intentionally concise (6–8 pages recommended) while covering critical details needed by reviewers and maintainers. The author is Shushank (contact: `shushankpawar664@gmail.com`).

2 Dataset and Preprocessing

2.1 Dataset snapshot

Always reference and archive the exact dataset snapshot used. Provide:

- URL and local filename(s).
- Checksum(s) (SHA256) of the data files used.
- If the dataset is modified (e.g., label corrections), provide the script and commit hash that produced the final files.

2.2 Canonicalization and standardization

All chemical structures must be canonicalized deterministically. Recommended steps:

1. Use RDKit (specify version) to parse SMILES and generate canonical SMILES.
2. Remove salts and small fragments; keep the largest component.
3. Standardize stereochemistry and explicit hydrogens consistently using the same RDKit routines and options (report exact function calls).
4. Generate InChI and InChIKey v1 from the canonicalized structure to identify duplicates.

If duplicates have conflicting labels, remove those molecules from all splits and report their InChIKeys and counts.

2.3 Feature generation

Describe and fix all featurizers used (ECFP, graph features, learned embeddings). When featurizer computation requires dataset statistics (e.g., scaling, PCA), fit such transforms only on the training data. Report featurizer library versions and settings.

3 Splitting and Avoiding Leakage

3.1 Split strategy

We require a scaffold split to better estimate generalization to novel chemotypes:

- Use Bemis–Murcko scaffolds (cite algorithm) to assign molecules to scaffolds, then partition scaffolds into train/validation/test groups.
- Target ratios: 80% train / 10% val / 10% test by molecule count; if exact ratios are impossible, report actual sizes.
- Use a deterministic seed (recommended: 42) and include the exact splitter script or library and commit.

3.2 Deduplication and cross-split contamination

Deduplicate before splitting. Any identical InChIKeys appearing multiple times must be collapsed or removed to avoid label leakage across splits. If augmentation methods (e.g., enumerating tautomers or SMILES augmentations) are used, ensure they are applied only to training molecules and do not re-create test molecules in train.

3.3 Prohibited practices

Explicitly forbidden:

- Using test labels or statistics computed including test examples (e.g., normalization parameters, PCA).
- Model selection or hyperparameter tuning on the test set.
- Using external data whose labels or molecules overlap the test set unless overlap is disclosed and removed.

4 Training, Model Selection, and Determinism

4.1 Hyperparameter tuning

Hyperparameter search must solely use the training and validation splits. Report:

- Search method (grid, random, Bayesian).
- Search ranges.
- Selection criterion (e.g., maximize mean validation ROC-AUC).

4.2 Final model

State whether the final model is:

- The checkpoint with best validation performance, or
- A model retrained on train+val with selected hyperparameters (if so, report seed and training regimen).

4.3 Random seeds and nondeterminism

Provide all seeds used (splitter, model init, data loader shuffling). Document frameworks and versions (e.g., PyTorch 1.x, TensorFlow 2.x) and any non-deterministic operations that could affect reproducibility. If GPU nondeterminism is possible, describe steps taken to mitigate it (e.g., deterministic flags, fixed cuDNN behavior).

5 Evaluation Protocol

5.1 Primary and secondary metrics

For each task report:

- ROC-AUC (primary) — use `sklearn.metrics.roc_auc_score`.
- PR-AUC (average precision) — use `sklearn.metrics.average_precision_score`.
- Balanced accuracy and F1 at threshold 0.5 for interpretability.

Aggregate score for leaderboard ranking: mean of per-task ROC-AUC values (unweighted). If a task's ROC-AUC is undefined (e.g., only one class present in test), report that task as excluded from the mean and list it explicitly.

5.2 Confidence intervals and significance

Compute 95% confidence intervals for aggregate and per-task metrics via bootstrap resampling (resample molecules, preserving multi-task labels) with 1000 resamples. For pairwise comparison between submissions, use paired bootstrap to estimate p-values (1000 resamples, two-sided).

5.3 Submission format and evaluation script

Submissions must provide a CSV with columns: `id, smiles, prob_task1, prob_task2, ...` Probabilities must be in [0,1]. The evaluation script (provided by submitter or maintainer) consumes this CSV and outputs per-task metrics, aggregate score, and bootstrap CIs. Provide a Docker entry point or script such that running the container on the test set reproduces the submission CSV.

6 Benchmarks and Results

This section reports the submitted model's task-wise ROC-AUCs and the aggregate mean ROC-AUC. These values were computed on the held-out test split following the protocol described above (scaffold split, deterministic preprocessing). Confidence intervals are omitted here for brevity; full bootstrap CIs and per-task PR-AUCs are available in the accompanying artifacts (evaluation outputs and scripts).

6.1 Task-wise Performance (ROC-AUC)

Table 1: Task-wise ROC-AUC on the Tox21 test set (reported model).

Task	ROC-AUC
NR-AhR	0.8931
NR-AR	0.8742
NR-AR-LBD	0.8931
NR-Aromatase	0.8661
NR-ER	0.8364
NR-ER-LBD	0.8909
NR-PPAR-gamma	0.8943
SR-ARE	0.8561
SR-ATAD5	0.8288
SR-HSE	0.8623
SR-MMP	0.9167
SR-p53	0.8544
Average ROC-AUC	**0.8722**

6.2 Interpretation

The model achieves a mean ROC-AUC of 0.8722 across the 12 Tox21 tasks. Performance is strongest on SR-MMP (0.9167) and NR-PPAR-gamma (0.8943), and weakest on SR-ATAD5 (0.8288) and NR-ER (0.8364). These per-task differences likely reflect varying dataset sizes and class imbalance across assays; detailed per-task data counts and positive rates are provided in the artifact bundle.

7 Minimum Artifact Requirements

To be accepted on the leaderboard, a submission MUST include:

1. Public code repository URL and commit hash used for reported results.

2. Dockerfile or a pinned Docker image tag with hash (preferred).
3. Preprocessing script(s) that reproduce canonicalization and splits.
4. Trained model weights (or a script+seed that trains weights deterministically) and checksum.
5. Evaluation script and example prediction CSV for a small held-out sample.
6. Short technical report (this document) and a persistent link (e.g., ChemRxiv preprint).
7. Environment specification (requirements.txt and/or environment.yml).
8. License and any data-use conditions for external resources or pretrained weights.

Provide exact commands to produce predictions from raw input (example):

```
# Example: run container to generate predictions
docker run --rm -v $(pwd)/data:/data tox-submission:tag \
  /bin/sh -c "python run_inference.py --input /data/test_smiles.csv --output /data/predictions
```

8 Recommended Baselines

Include these baselines to contextualize results. Provide scripts, hyperparameters, and per-task metrics.

Table 2: Recommended baseline methods (implementations and settings must be provided).

Baseline	Implementation	Notes
Random	Simple prevalence predictor	Predict per-task positive prevalence (chance baseline).
ECFP4 + RF	scikit-learn RandomForest	Morgan (radius=2, nBits=2048), n_estimators=1000, random_state=seed.
ECFP4 + LR	scikit-learn LogisticRegression	Penalty='l2', solver='saga', max_iter=2000.
D-MPNN (ChemProp)	ChemProp repo (cite commit)	Use default ChemProp settings; seed=42.
Transformer/SMILES	Pretrained SMILES model	Report pretraining checkpoint and fine-tuning settings.

For each baseline report:

- Per-task ROC-AUC, PR-AUC, and aggregate mean ROC-AUC with 95% CI.
- Number of trainable parameters and training time.
- Hardware used.

9 Discussion and Limitations

Discuss potential sources of optimistic bias (e.g., dataset overlap with pretraining corpora), limitations of scaffold splits, tasks with extremely low positive rates, and the effect of ensembling. Be explicit about what was and was not done: e.g., whether final model uses train+val, whether external datasets were used for pretraining, and any deviations from the protocol.

10 Conclusion

This report provides a compact, reproducible evaluation protocol and artifact checklist to accompany Tox21 leaderboard submissions. By following the recommendations herein (deterministic preprocessing, scaffold splitting, withheld test labels, thorough artifact packaging), submissions will be easier to verify and compare fairly. The reported model achieves a mean ROC-AUC of 0.8722 across Tox21 tasks; full artifacts required to reproduce these numbers are included in the submission bundle.

Acknowledgements

Contact: Shushank (`shushankpawar664@gmail.com`). If maintainers require assistance reproducing results from provided artifacts, I will make efforts to help reproduce on request.

A Reproducibility Checklist

Include this checklist with your submission:

- [ ] Preprint / short report link (ChemRxiv DOI or URL).
- [ ] Public code repo + commit hash.
- [ ] Docker image tag + sha256 or Dockerfile.
- [ ] requirements.txt / environment.yml.
- [ ] Preprocessing script(s) and split script (seed included).
- [ ] Trained weights and checksum (or deterministic training script).
- [ ] run_inference.py and run commands.
- [ ] Example prediction CSV and expected outputs.
- [ ] Per-task metrics and bootstrap CIs.
- [ ] Random seeds and framework versions.
- [ ] License for code and any external assets.

B Notes on Page Length and Formatting

This LaTeX source, with typical fonts and margins, should compile to approximately 6–8 pages depending on baseline figure/table inclusion and bibliography length. If you need a strictly limited page count, trim the Discussion, expand the appendix into an external artifact, or reduce line spacing slightly.

References

References

[1] RDKit: Open-source cheminformatics. `https://www.rdkit.org/`.

[2] Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., Guzman-Perez, A., Hopper, T., Kelley, B., Mathea, M., Palmer, A., Settels, V., DeepChemProp: Message passing neural networks for molecular property prediction. (cite repository and commit used).

[3] Bemis, G. W., Murcko, M. A. The properties of known drugs. 1. Molecular frameworks. J. Med. Chem. 1996, 39, 2887–2893.