

Project Work in Machine Learning and Data Mining: Polymer Property Prediction

Filippo Teodorani

Università di Bologna

December 19, 2025

Competition Overview

NeurIPS Open Polymer Prediction 2025

Goal:

- Predict key physicochemical properties of polymers from their molecular structure.

Target properties:

- **T_g** — Glass Transition Temperature
- **FFV** — Fractional Free Volume
- **T_c** — Crystallization Temperature
- **Density**
- **R_g** — Radius of Gyration

Test labels are hidden on Kaggle and evaluated on a private leaderboard.

Dataset Structure

- Input column: **SMILES** strings representing polymer units.
- Each row corresponds to a polymer or monomer structure.
- **Train:** SMILES + target properties.
- **Test:** SMILES only, no labels.

Main challenge: converting SMILES into meaningful numerical features.

Example row from the training dataset

id	SMILES	Tg	FFV	Tc	Density	Rg
87817	*CC(*)c1ccccc1C(=O)OCCCCCC	NaN	0.374645	0.205667	NaN	NaN

SMILES and Molecular Representation

What are SMILES?

- A compact text format describing molecular structures.
- Encodes atoms, bond types, branching, stereochemistry, and ring closures.
- Widely used in cheminformatics and fully supported by RDKit.

Why SMILES matter in ML workflows:

- They are the starting point for all molecular featurization.
- Machine learning models cannot use raw SMILES strings directly.
- Must be transformed into:
 - *Descriptors* (numeric physicochemical properties)
 - *Fingerprints* (bit vectors capturing structural patterns)
- These derived features allow models to learn structure–property relationships.

Missing Data in Training Set

- A big problem of this dataset was the presence of missing data, in particular for the target variable **Tg**

Target	Number of non-NA values
Tg	511
FFV	7030
Tc	737
Density	613
Rg	614

Preprocessing: RDKit Descriptors

RDKit Descriptors:

- Numerical properties extracted from molecular graphs of polymers.
- Computation process in this project:
 - Convert SMILES strings to molecule objects using `Chem.MolFromSmiles`.
 - Handle invalid SMILES with a safe wrapper (`safe_mol_from_smiles`).
 - Use `MoleculeDescriptors.MolecularDescriptorCalculator` to compute 200+ descriptors per molecule.
 - Examples: molecular weight, topological polar surface area, H-bond donors/acceptors, ring counts.
- Output: a structured numeric feature matrix aligned with the dataset.

These descriptors are then used as input for machine learning models.

Preprocessing: Morgan Fingerprints

Morgan Fingerprints (ECFP):

- Binary vectors representing local substructures around each atom.
- Computation process in the project:
 - Convert SMILES strings to RDKit molecule objects.
 - Compute circular fingerprints using `AllChem.GetMorganFingerprintAsBitVect`.
 - Each fingerprint is a 1024-bit binary vector capturing substructure presence.
 - Robust handling of invalid SMILES: missing/invalid molecules produce zero vectors.
- Captures molecular connectivity and structural motifs not explicitly in RDKit descriptors.
- Widely used in chemoinformatics tasks.

These fingerprints complement RDKit descriptors with high-dimensional structural information for machine learning models.

Feature Selection: Filtering Steps

Step 1: Variance Threshold

- Remove constant or near-constant features that provide little information.
- Implemented with `sklearn.feature_selection.VarianceThreshold`.
- Keeps only features that vary across the dataset.

Step 2: Correlation Filtering

- Remove highly correlated features ($|\text{corr}| > 0.93$)
- Prevents redundancy and multicollinearity.
- Calculated using pairwise correlation matrix of features.

Feature Selection: Model-based Importance

Step 3: LightGBM Feature Importance

- Train a lightweight LightGBM regression model on a single target (T_g).
- Extract feature importances from the trained model.
- Keep top k features (300) for downstream modeling.
- Reduces dimensionality → faster training, lower overfitting risk.

Outcome:

- Dataset size reduced from thousands of features to a more compact set.
- Ensures most informative features are preserved for prediction.

Model Choice: Why LightGBM?

Reasons for choosing LightGBM:

- Excellent performance on tabular datasets with mixed features.
- Handles high-dimensional input efficiently (hundreds of RDKit descriptors + 1024-bit fingerprints).
- Robust to multicollinearity between features.
- Fast training and low memory usage compared to other gradient boosting libraries.

Model Implementation: LightGBM

Training setup:

- Separate regression model for each target property (Tg, FFV, Tc, Density, Rg).
- 5-fold cross-validation to estimate out-of-fold performance.
- Input features: selected RDKit descriptors + Morgan fingerprints.

Outcome: Efficient, accurate predictions while retaining interpretability.

Evaluation Metrics: MAE and Weighted MAE

Mean Absolute Error (MAE):

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$

Weighted Mean Absolute Error (WMAE):

$$\text{WMAE} = \frac{\sum_{t \in \text{targets}} w_t \sum_{i \in t} |y_i - \hat{y}_i|}{\sum_{t \in \text{targets}} n_t}$$

- WMAE accounts for different ranges and sample sizes of target properties.
- Used in the NeurIPS Open Polymer Prediction 2025 competition for official scoring.

Results

Cross-validation results (OOF MAE):

- $T_g = 88.070$
- $FFV = 0.021$
- $T_c = 0.077$
- Density = 0.108
- $R_g = 3.904$

Private leaderboard score: 0.11318 on hidden test set.

Conclusions and Outlook

- Combining RDKit descriptors and Morgan fingerprints provides rich molecular information.
- Feature selection significantly reduces noise and improves model stability.
- LightGBM is an effective baseline for polymer property prediction.
- Results show good generalization on the private test set.
- The model could be further improved with advanced ensembling, hyperparameter tuning, or additional descriptors.
- Current approach provides a strong starting point, with possible $WMAE < 0.10$ on the hidden test set.