Predicting Glass Transition Temperature of Chemical Compounds Using Features Generated from SMILES Descriptors with Machine Learning Models

Sunny Kaushik
IMT2021007

***Abstract*—Accurate prediction of the glass transition temperature (Tg) of chemical compounds is crucial for understanding their behavior in diverse applications. This study employs a comprehensive suite of machine learning models, including Linear Regression, Ridge Regression, Random Forest, Gradient Boosting, XGBoost, AdaBoost, CatBoost, and Neural Networks, to predict Tg values using features generated from SMILES (Simplified Molecular Input Line Entry System) descriptors and functional group counts. To enhance prediction accuracy, new features such as the number of different atom types were introduced. Neural networks were implemented using a sequential model with dense layers and dropout for regularization. Our findings reveal that Extreme Gradient Boosting, utilizing the complete feature set, achieves superior performance with the lowest mean absolute error (MAE) and a high R^2 score, underscoring its effectiveness in Tg prediction. This research highlights the importance of feature engineering and the potential of ensemble methods and neural network techniques in advancing predictive models for glass transition temperatures.**

I. Introduction

Glass transition temperature (Tg) is a critical property that determines the behavior of materials as they transition from a rigid glassy state to a more flexible state. Accurate prediction of Tg is essential for a wide range of applications, from material science to atmospheric studies. Despite its importance, the experimental determination of Tg for a vast array of chemical compounds remains limited due to resource-intensive processes.

Recent advancements in machine learning provide promising alternatives for Tg prediction. Studies by Alzghoul et al. (2014) and Tao et al. (2019) have demonstrated the potential of support vector machines and random forest models for Tg prediction, albeit on limited datasets. Building on this foundation, our study leverages a larger and more diverse dataset, incorporating innovative features such as SMILES descriptors and branching information to enhance model performance.

The Simplified Molecular Input Line Entry System (SMILES) is a notation that allows a user to represent a chemical structure in a way that can be used by a computer program. SMILES strings encode molecular structures using short ASCII strings, making them compact and easy to manipulate computationally. This representation includes information about atoms, bonds, connectivity, and chirality, enabling the generation of detailed molecular descriptors.

Descriptors derived from SMILES strings capture various structural, topological, and electronic properties of molecules. In this study, we utilized the RDKit library to transform SMILES strings into a comprehensive set of 208-bit descriptors. These descriptors provide a rich source of information about the molecular properties and are instrumental in improving the predictive power of machine learning models.

We employed several machine learning models, including Linear Regression, Ridge Regression, Random Forest, Gradient Boosting, XGBoost, AdaBoost, CatBoost, and Extra Trees, to predict Tg. Additionally, we explored the impact of newly introduced features, such as the number of sulfur atoms and the degree of branching, on model accuracy. Neural networks were also implemented using a sequential model with dense layers and dropout for regularization. Our methodology involved extensive data preprocessing, feature engineering, and model evaluation using robust metrics like mean absolute error (MAE), root mean squared error (RMSE), and the coefficient of determination (R^2).

This paper presents a comprehensive analysis of the models' performance, highlighting the superior accuracy achieved by the Extreme Gradient Boosting model when utilizing the enriched feature set. Our findings underscore the significance of feature selection, ensemble learning techniques, and neural network approaches in developing reliable predictive models for Tg, paving the way for future research and practical applications in this domain.

II. Dataset

The dataset utilized in this study comprises various chemical compounds represented through SMILES (Simplified Molecular Input Line Entry System) descriptors. It includes the glass transition temperature (T_g) as the target variable, along with a comprehensive set of features generated from the SMILES descriptors. These features encompass atomic counts, molecular properties, and structural descriptors, including the number of different atom types, functional group counts, and molecular weights. Extensive data preprocessing and feature

engineering were conducted to ensure the robustness of the predictive models. The dataset's diverse range of features enables a detailed analysis and accurate prediction of T_g using advanced machine learning algorithms and neural networks.

A. Feature Descriptions

The features in the dataset are detailed as follows:

- **MaxEStateIndex / MinEStateIndex / MaxAbsEStateIndex / MinAbsEStateIndex**:
  EState Indices: These indices represent the Electrotopological State of atoms in a molecule. Higher values can indicate higher electronic interactions. These indices can influence the T_g by affecting the electronic environment within the polymer, potentially impacting intermolecular interactions and rigidity.
- **qed**:
  Quantitative Estimate of Drug-likeness: A metric that combines multiple properties to estimate how "drug-like" a compound is. While primarily used in pharmacology, in the context of polymers, this could relate to molecular properties that influence stability and interactions within the polymer matrix, thus affecting T_g.
- **MolWt / HeavyAtomMolWt / ExactMolWt**:
  Molecular Weight: The mass of the molecule. Exact molecular weight accounts for isotopic distribution. Higher molecular weight typically leads to higher T_g due to increased chain entanglement and restricted molecular motion.
- **NumValenceElectrons / NumRadicalElectrons**:
  Valence Electrons: Number of electrons in the outermost shell.
  Radical Electrons: Number of unpaired electrons.
  Valence electrons can influence the bonding and overall stability of the polymer structure, while radical electrons may indicate potential reactive sites, both of which can affect T_g.
- **Partial Charges (MaxPartialCharge / MinPartialCharge / MaxAbsPartialCharge / MinAbsPartialCharge)**:
  Partial Charges: Reflect the distribution of electron density in a molecule.
  These charges can influence the intermolecular forces such as dipole-dipole interactions and hydrogen bonding, which in turn can affect the T_g.
- **FpDensityMorgan1 / FpDensityMorgan2 / FpDensityMorgan3**:
  Fingerprint Densities: Represent the density of certain molecular substructures (Morgan fingerprints).
  Higher densities of specific substructures can indicate more complex or rigid frameworks, potentially leading to higher T_g.
- **BCUT Metrics (e.g., BCUT2D_MWHI, BCUT2D_MWLOW)**:
  BCUT Descriptors: Measures of molecular properties derived from topological distance matrices.
  These descriptors can capture spatial and electronic properties that influence the packing and interactions of polymer chains, thus affecting T_g.
- **BalabanJ / BertzCT / Chi Indices**:
  Topological Indices: Various descriptors that reflect the topological nature of the molecule (e.g., connectivity, shape).
  These indices can impact T_g by affecting the overall geometry and rigidity of the polymer structure.
- **HallKierAlpha / Ipc / Kappa1 / Kappa2 / Kappa3**:
  Shape Indices: Reflect the molecular shape and branching.
  The shape and branching of polymer chains can significantly affect T_g by influencing the ease of molecular motion and chain packing.
- **PEOE_VSA / SMR_VSA / SlogP_VSA**:
  Volumetric Surface Areas: Descriptors calculated based on the molecular surface areas and electrostatic properties.
  Larger surface areas and specific electrostatic properties can influence the intermolecular forces and packing density, thereby affecting T_g.
- **TPSA**:
  Topological Polar Surface Area: Related to the ability of the molecule to interact with water.
  Higher TPSA values can indicate stronger interactions with polar molecules or solvents, which might affect the flexibility and mobility of polymer chains, influencing T_g.
- **VSA_EState**:
  Volumetric Surface Area based EState: A combination of surface area and EState descriptors.
  These combined properties can provide insights into the overall electronic and spatial characteristics that affect T_g.
- **FractionCSP3**:
  Fraction of SP3 Carbon Atoms: Indicates the fraction of carbon atoms that are sp3 hybridized, which affects molecular geometry and flexibility.
  A higher fraction of sp3 carbons typically leads to more flexible and less rigid polymer chains, which can lower T_g.
- **HeavyAtomCount / NHOHCount / NOCount / NumAliphaticCarbocycles / NumAliphaticHeterocycles / NumAromaticCarbocycles / NumAromaticHeterocycles / NumRotatableBonds / RingCount**:
  Structural Counts: Various counts of atoms, functional groups, rings, and bonds.
  These counts provide a direct measure of the molecular structure's complexity and rigidity, influencing the thermal properties and T_g of the polymer.
- **fr_Descriptors (e.g., fr_Al_COO, fr_ArN, etc.)**:
  Functional Group Counts: Counts of specific functional groups within the molecule.
  The presence and concentration of specific functional groups can significantly impact the intermolecular interactions and thermal stability, thereby affecting T_g.

III. Methodology

A. Machine Learning Methods

In this section, we will provide a detailed explanation of the machine learning models used in this study. Each model will be described in a separate subsection to provide clarity and depth of understanding.

1) Decision Trees: A decision tree is a flowchart-like structure used for decision-making and predictive modeling. It consists of nodes representing decisions or tests on features, branches representing the outcomes of those decisions, and leaf nodes representing final predictions or outcomes. The key components of a decision tree include:

- **Root Node:** The topmost node in the tree that represents the initial feature test.
- **Decision Nodes:** Intermediate nodes that represent tests on features.
- **Leaf Nodes:** Terminal nodes that provide the final prediction or outcome.

The decision tree algorithm works as follows:

1) **Select Best Feature:** At each node, the algorithm selects the best feature to split the data based on a criterion such as Gini impurity or information gain.
2) **Split Data:** The data is split into subsets based on the selected feature.
3) **Repeat:** This process is repeated recursively for each subset until a stopping condition is met (e.g., maximum depth, minimum samples per leaf).

The Gini impurity for a node t is calculated as:

$$G(t) = 1 - \sum_{i=1}^{n} p_i^2$$

where p_i is the proportion of samples belonging to class i.

2) Random Forest: Random Forest is an ensemble learning method that combines multiple decision trees to improve predictive performance and reduce overfitting. Each tree in the forest is trained on a random subset of the data with replacement (bootstrap sampling), and a random subset of features is considered for each split.

Key steps in the Random Forest algorithm:

1) **Bootstrap Sampling:** Randomly sample the data with replacement to create multiple subsets.
2) **Train Trees:** Train a decision tree on each subset using a random subset of features for splitting.
3) **Aggregate Predictions:** Combine the predictions of all trees (e.g., by averaging for regression or majority voting for classification).

The final prediction $\hat{y}$ for regression is the average of individual tree predictions:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} \hat{y}_t$$

where T is the number of trees and $\hat{y}_t$ is the prediction from tree t.

3) Extra Trees: Extra Trees, or Extremely Randomized Trees, is similar to Random Forest but introduces more randomness in the tree-building process. Instead of selecting the best split based on a criterion, Extra Trees selects random split points for each feature.

Key differences from Random Forest:

- **Random Splits:** Split points are selected randomly rather than based on Gini impurity or information gain.
- **No Bootstrapping:** Extra Trees use the entire dataset to train each tree instead of bootstrap samples.

The algorithm increases model diversity and reduces overfitting by introducing additional randomness.

4) Linear Regression: Linear Regression is a simple and widely used statistical method for modeling the relationship between a dependent variable y and one or more independent variables X. The relationship is modeled as a linear equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

where:

- y is the dependent variable.
- $x_1, x_2, \ldots, x_n$ are the independent variables.
- β_0 is the intercept.
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients.
- ϵ is the error term.

The objective is to minimize the sum of squared residuals:

$$\min \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

where $\hat{y}_i$ is the predicted value for observation i.

5) Gradient Boosting: Gradient Boosting is an ensemble technique that builds models sequentially, where each new model corrects the errors of the previous models. It combines weak learners (typically decision trees) to form a strong learner.

Key steps in the Gradient Boosting algorithm:

1) **Initialize Model:** Start with an initial model (e.g., the mean of the target values for regression).
2) **Compute Residuals:** Calculate the residuals (errors) from the current model.
3) **Train New Model:** Train a new decision tree to predict the residuals.
4) **Update Model:** Add the new tree to the existing model with a learning rate η:

$$F_m(x) = F_{m-1}(x) + \eta h_m(x)$$

where $F_m(x)$ is the updated model, $F_{m-1}(x)$ is the previous model, and $h_m(x)$ is the new tree.

6) Extreme Gradient Boosting (XGBoost): Extreme Gradient Boosting (XGBoost) is an optimized version of Gradient Boosting that includes several enhancements to improve speed and performance. Key features of XGBoost include:

- **Regularization:** Adds $L1$ and $L2$ regularization terms to the loss function to prevent overfitting.
- **Sparsity Awareness:** Efficient handling of sparse data.

- **Parallel Processing:** Utilizes parallel computing to speed up training.

The objective function in XGBoost includes a regularization term:

$$\mathcal{L}(\Theta) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(h_k)$$

where $\Omega(h_k)$ is the regularization term for tree k.

XGBoost builds trees sequentially, with each tree improving the residual errors of the previous trees, similar to Gradient Boosting but with additional optimizations.

7) Grid Search CV Mechanism: Grid Search with Cross-Validation (Grid Search CV) is a technique used to find the optimal hyperparameters for a machine learning model. It systematically works through multiple combinations of parameter values, cross-validating as it goes to determine which set of parameters provides the best performance.

Key steps in Grid Search CV:

1) **Define Parameter Grid:** Specify a range of values for each hyperparameter you want to optimize. For example, for a Random Forest model, you might vary the number of trees (`n_estimators`), the maximum depth of the trees (`max_depth`), and the number of features considered for splitting (`max_features`).
2) **Cross-Validation:** Split the training data into k subsets (folds). Train the model on $k-1$ folds and validate it on the remaining fold. Repeat this process k times, each time with a different validation fold.
3) **Evaluation:** For each combination of hyperparameters, compute the average performance metric (e.g., accuracy, MAE) over the k folds.
4) **Select Best Parameters:** Choose the hyperparameter combination that results in the best average performance metric.

Formally, the goal is to minimize the cross-validation error:

$$\min_{\theta \in \Theta} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(y^{(i)}, \hat{y}^{(i)})$$

where Θ is the set of all hyperparameter combinations, $\mathcal{L}$ is the loss function, $y^{(i)}$ are the true values, and $\hat{y}^{(i)}$ are the predicted values for the i-th fold.

Grid Search CV helps in finding the most effective model configuration, ensuring that the model is neither underfitting nor overfitting the data.

8) Bagging and Boosting: Bagging (Bootstrap Aggregating) and Boosting are two fundamental ensemble learning techniques that improve model performance by combining the predictions of multiple base models.

Bagging: Bagging aims to reduce variance and prevent overfitting by training multiple models on different random subsets of the training data (created using bootstrapping). The final prediction is typically obtained by averaging the predictions (for regression) or voting (for classification).

Key steps in Bagging:

1) **Bootstrap Sampling:** Create multiple random samples of the training data with replacement.
2) **Train Models:** Train a separate model on each bootstrap sample.
3) **Aggregate Predictions:** Combine the predictions of all models. For regression, this is usually done by averaging; for classification, by majority voting.

Mathematically, the final prediction $\hat{y}$ for regression is:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} \hat{y}_t$$

where T is the number of models and $\hat{y}_t$ is the prediction from the t-th model.

Boosting: Boosting focuses on reducing bias and improving model performance by sequentially training models, where each new model attempts to correct the errors of the previous models. The models are trained on weighted versions of the training data, emphasizing the samples that were previously mispredicted.

Key steps in Boosting:

1) **Initialize Model:** Start with an initial base model (e.g., a decision tree).
2) **Compute Residuals:** Calculate the residuals (errors) from the current model.
3) **Train New Model:** Train a new model to predict the residuals.
4) **Update Model:** Add the new model to the existing ensemble with a weight. This process is repeated for a predefined number of iterations.

Mathematically, the updated model $F_m(x)$ is:

$$F_m(x) = F_{m-1}(x) + \eta h_m(x)$$

where η is the learning rate, $h_m(x)$ is the new model, and $F_{m-1}(x)$ is the previous model.

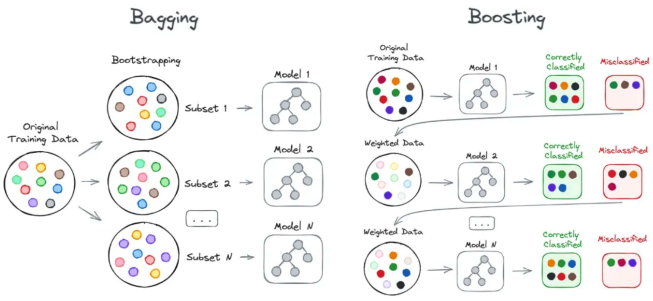

Fig. 1: Bagging and Boosting

Difference between Gradient Boosting and XGBoost:

- **Gradient Boosting:** Combines weak learners, typically decision trees, by optimizing a differentiable loss function in a sequential manner. Each tree attempts to correct the errors of the previous tree.
- **XGBoost:** An enhanced version of Gradient Boosting with additional features such as regularization to prevent overfitting, parallel processing for faster computation, and efficient handling of sparse data.

XGBoost's objective function includes a regularization term to control model complexity:

$$\mathcal{L}(\Theta) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(h_k)$$

where $\Omega(h_k)$ is the regularization term for tree k.

9) Ridge Regression: Ridge Regression, also known as Tikhonov regularization, is a linear regression technique that includes a regularization term to prevent overfitting. This regularization term penalizes large coefficients, thus shrinking them towards zero but not exactly to zero, which helps in dealing with multicollinearity.

The Ridge Regression equation is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

with the objective of minimizing the following cost function:

$$\min \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} \beta_j^2$$

where:

- y is the dependent variable.
- $x_1, x_2, \ldots, x_n$ are the independent variables.
- β_0 is the intercept.
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients.
- ϵ is the error term.
- λ is the regularization parameter controlling the strength of the penalty.

The regularization term $\lambda \sum_{j=1}^{n} \beta_j^2$ helps to prevent overfitting by penalizing large coefficient values, thus improving the model's generalization to new data.

10) AdaBoost: AdaBoost, short for Adaptive Boosting, is an ensemble learning method that combines multiple weak learners, typically decision stumps (trees with a single split), to create a strong learner. AdaBoost works by iteratively training weak learners, with each subsequent learner focusing more on the misclassified samples from the previous iterations.

Key steps in the AdaBoost algorithm:

1) **Initialize Weights:** Assign equal weights to all training samples.
2) **Train Weak Learner:** Train a weak learner (e.g., a decision stump) on the weighted dataset.
3) **Compute Error:** Calculate the error rate of the weak learner.
4) **Update Weights:** Increase the weights of the misclassified samples and decrease the weights of correctly classified samples.
5) **Combine Learners:** The final model is a weighted sum of the weak learners, where weights are determined by the accuracy of each weak learner.

The final prediction $\hat{y}$ for AdaBoost is given by:

$$\hat{y} = \sum_{t=1}^{T} \alpha_t h_t(x)$$

where:

- T is the number of weak learners.
- α_t is the weight of the t-th weak learner.
- $h_t(x)$ is the prediction from the t-th weak learner.

AdaBoost effectively improves the accuracy by focusing on difficult-to-classify samples and combining multiple weak models into a strong model.

11) CatBoost: CatBoost, short for Categorical Boosting, is a gradient boosting algorithm specifically designed to handle categorical features. It provides several innovations to improve the handling of categorical data and the efficiency of training.

Key features of CatBoost include:

- **Categorical Feature Encoding:** CatBoost uses a combination of target-based statistics and one-hot encoding to handle categorical features effectively.
- **Ordered Boosting:** To prevent target leakage, CatBoost uses a special ordered boosting technique that processes data in a sequential manner.
- **Efficient Training:** CatBoost is optimized for efficient training and can handle large datasets with high-dimensional categorical features.

The objective function in CatBoost includes both the loss function and regularization terms to control overfitting:

$$\mathcal{L}(\Theta) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(h_k)$$

where:

- $l(y_i, \hat{y}_i)$ is the loss function measuring the error between true and predicted values.
- $\Omega(h_k)$ is the regularization term for the k-th tree.

CatBoost builds trees sequentially, with each tree improving the residual errors of the previous trees, similar to other gradient boosting methods but with optimizations for categorical data and efficient computation.

B. Neural Networks

Neural networks, particularly deep learning models, have shown significant promise in capturing complex patterns in data due to their ability to model non-linear relationships. In this study, we employed a neural network architecture to predict the glass transition temperature (T_g) of chemical compounds using features generated from SMILES descriptors.

Methodology:

The methodology for implementing neural networks involved several key steps, including data preprocessing, model architecture design, training, and evaluation.

1) Data Preprocessing: The dataset was loaded and preprocessed to ensure its suitability for training the neural network. The preprocessing steps included:

1) **Loading the Dataset:** The dataset containing SMILES descriptors and T_g values was loaded using `pandas`.
2) **Separating Features and Target:** The features (`X`) and target variable (`y`) were separated, with `X` containing all features except the target (T_g) and SMILES strings.
3) **Handling Missing Values:** Infinite values were replaced with NaN, and missing values were imputed using the mean strategy with `SimpleImputer`.

4) **Data Splitting:** The dataset was split into training and testing sets using `train_test_split`, with 80% for training and 20% for testing.
5) **Feature Scaling:** The features were standardized using `StandardScaler` to ensure that they are on a similar scale, which is crucial for neural network training.

2) Neural Network Architecture: We designed a neural network model using the `TensorFlow` and `Keras` libraries. The architecture consisted of:

- **Input Layer:** The input layer matched the number of features in the dataset.
- **Hidden Layers:** Three hidden layers were used:
  - Two layers with 128 neurons each and ReLU (Rectified Linear Unit) activation function, followed by Dropout layers with a dropout rate of 0.2 to prevent overfitting.
  - One layer with 64 neurons and ReLU activation function.
- **Output Layer:** A single neuron output layer with a linear activation function to predict the continuous target variable (T_g).

The model was compiled with the Adam optimizer, the mean squared error (MSE) loss function, and the root mean squared error (RMSE) as a metric.

3) Training and Early Stopping: The neural network was trained on the training set with the following configurations:

- **Epochs:** The model was trained for up to 100 epochs.
- **Batch Size:** A batch size of 32 was used.
- **Validation Split:** 20% of the training data was used as a validation set.
- **Early Stopping:** Early stopping was employed to monitor the validation loss with a patience of 10 epochs. If the validation loss did not improve for 10 consecutive epochs, training was halted, and the best model weights were restored.

4) Evaluation: The trained neural network was evaluated on the test set using the mean squared error (MSE) and the coefficient of determination (R^2) metrics. The model's performance was compared with other machine learning models, including Ridge Regression, AdaBoost, and CatBoost.

Impact and Importance:

- **Handling Complex Non-linear Relationships:** Neural networks are capable of capturing complex non-linear relationships in the data, which might be missed by traditional linear models. This ability is crucial for accurately predicting T_g, which can be influenced by intricate molecular interactions and structural properties.
- **Feature Interactions:** Neural networks can automatically learn and represent interactions between features, which is beneficial when dealing with high-dimensional data generated from SMILES descriptors.
- **Prevention of Overfitting:** Techniques like dropout and early stopping help in preventing overfitting, ensuring that the model generalizes well to unseen data.
- **Comparison with Traditional Models:** By comparing neural networks with traditional machine learning models, we can assess their relative strengths and determine the best approach for T_g prediction.

The use of neural networks in this study highlights their potential in improving the accuracy of predictive models for the glass transition temperature of chemical compounds, complementing traditional machine learning techniques.

C. SMILE descriptors

In this section, we will discuss the significance of SMILES (Simplified Molecular Input Line Entry System) descriptors and their importance in computing RDKit descriptors. In the previous section, we examined various machine-learning methods. This method can be utilized to predict the glass transition temperature by using descriptors generated from the SMILES representation.

SMILES: SMILES, which stands for Simplified Molecular Input Line Entry System, is a notation that allows a user to represent a chemical structure in a way that can be easily used by computer programs. It uses a line of text to describe the structure of molecules in a manner that is both human-readable and machine-processable

Key rules for the computation of SMILES:

1) **Rule One: Atoms and Bonds**
   SMILES supports all elements in the periodic table. An atom is represented using its respective atomic symbol. Upper case letters refer to non-aromatic atoms; lower case letters refer to aromatic atoms. If the atomic symbol has more than one letter the second letter must be lowercase.
   Bonds are denoted as shown below:
   - - Single bond
   - = Double bond
   - # Triple bond
   - * Aromatic bond
   - . Disconnected structures

   Single bonds are the default and therefore need not be entered. For example, 'CC' would mean that there is a non-aromatic carbon attached to another non-aromatic carbon by a single bond, and the computer would identify the structure as the chemical ethane. It is also assumed that the bond between two lowercase atom symbols is aromatic. A blank terminates the SMILES string.
2) **Rule Two: Simple Chains** By combining atomic symbols and bond symbols simple chain structures can be represented. The structures that are entered using SMILES are hydrogen-suppressed, that is to say, that the molecules are represented without hydrogens. The SMILES software understands the number of possible connections that an atom can have. If enough bonds are not identified by the user through SMILES notation, the system will automatically assume that the other connections are satisfied by hydrogen bonds. **Some Examples:**

CC	CH_3CH_3	Ethane
$C=C$	CH_2CH_2	Ethene
CBr	CH_3Br	Bromomethane
$C\#N$	$C=N$	Hydrocyanic acid
$Na.Cl$	$NaCl$	Sodium chloride

The user can explicitly identify the hydrogen bonds, but if one hydrogen bond is identified in the string, the SMILES interpreter will assume that the user has identified all hydrogens for that molecule.

HC(H)=C(H)(H) Ethene

Because SMILES allows entry of all elements in the periodic table and also utilizes hydrogen suppression, the user should be aware of chemicals with two letters that could be misinterpreted by the computer. For example, 'Sc' could be interpreted as a sulfur atom connected to an aromatic carbon by a single bond, or it could be the symbol for scandium. The SMILES interpreter gives priority to the interpretation of a single bond connecting a sulfur atom and an aromatic carbon. To identify scandium the user should enter [Sc].

3) **Rule Three: Branches**

A branch from a chain is specified by placing the SMILES symbol(s) for the branch between parenthesis. The string in parentheses is placed directly after the symbol for the atom to which it is connected. If it is connected by a double or triple bond, the bond symbol immediately follows the left parenthesis.

Some Examples:

$CC(O)C$	2-Propanol
$CC(=O)C$	2-Propanone
$CC(CC)C$	2-Methylbutane
$CC(C)CC(=O)$	2-Methylbutanal
$c1c(N(=O)=O)cccc1$	Nitrobenzene
$CC(C)(C)CC$	2,2-Dimethylbutane

4) **Rule Four: Rings**

SMILES allows a user to identify ring structures by using numbers to identify the opening and closing ring atom. For example, in C1CCCCC1, the first carbon has a number '1' which connects by a single bond with the last carbon which also has a number '1'. The resulting structure is cyclohexane. Chemicals that have multiple rings may be identified by using different numbers for each ring. If a double, single, or aromatic bond is used for the ring closure, the bond symbol is placed before the ring closure number.

Some Examples:

C=1CCCCC1	Cyclohexene
c1ccccc1	Benzene
C1OC1CC	Ethyloxirane
1cc2ccccc2cc1	Naphthalene

5) **Rule Five: Charged Atoms**

Charges on an atom can be used to override the knowledge regarding valence that is built into SMILES software. The format for identifying a charged atom consists of the atom followed by brackets which enclose the charge on the atom. The number of charges may be explicitly stated (-1) or not (-).

Some Examples:

CCC(=O)O-1	Ionized form of propanoic acid
c1ccccn+11CC(=O)O	1-Carboxylmethyl pyridinium

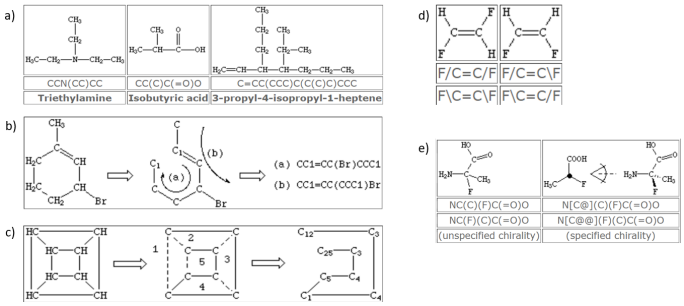

Fig. 2: Examples of SMILES

Figure 2 shows some more examples of SMILE representation a) Representation is constructed using the fundamental rules of branching, chaining, and atom naming. b) Using the ring rule, the structure is broken in the middle and named accordingly. c) The structure is broken multiple times and named. If an atom has multiple branch breaks, the naming convention will include annotations indicating the number of branch breaks. d) Cis and trans molecules are represented using the symbols '/' and '\', respectively. e) Chiral center molecules are represented using the '@' symbol. Refer to [5] and [6] references for the better understanding.

RDKIT: RDKit is a widely-used open-source software toolkit designed for cheminformatics, which encompasses the storage, retrieval, analysis, and manipulation of chemical information. Developed to facilitate drug discovery and chemical research, RDKit provides a robust set of tools that can handle a variety of tasks in computational chemistry and bioinformatics.

In this study, we utilized RDKit, an open-source cheminformatics toolkit, to process molecular structures, generate molecular descriptors, and apply these descriptors in machine learning (ML) models to predict the glass transition temperature (Tg) of molecules.For implementation refer to [7] of the github link

Methodology:

- **Step 1: Processing SMILES Strings -** SMILES (Simplified Molecular Input Line Entry System) strings are a compact way to represent chemical structures using short ASCII strings. RDKit can convert SMILES strings into molecular objects that can be further analyzed and manipulated
- **Step 2: Generating Molecular Descriptors -** Molecular descriptors are numerical values that describe various aspects of a molecule's structure and properties. RDKit provides tools to calculate a wide array of descriptors(approximately 200 descriptors), which can be used as features in machine learning models.
- **Step 3: Preparing the Dataset -** We prepared a dataset containing SMILES strings of polymers and their corre-

sponding Tg values. The dataset was split into training and testing sets to build and evaluate the ML models.

- **Step 4: Building the Machine Learning Model -** We have tried different machine learning algorithms to model the relationship between molecular descriptors and Tg. As outlined in the methodology section, we have employed all the machine learning algorithms listed in the machine learning methods section.
- **Step 5: Model Evaluation and Interpretation -** The performance of the model was evaluated using the Mean Absolute Error (MAE),Root Mean Squared Error (RMSE), and Coefficient of Determination (R2 Score) metrics which are discussed in detail in the future section. The importance of each descriptor in predicting Tg was analyzed to interpret the model.

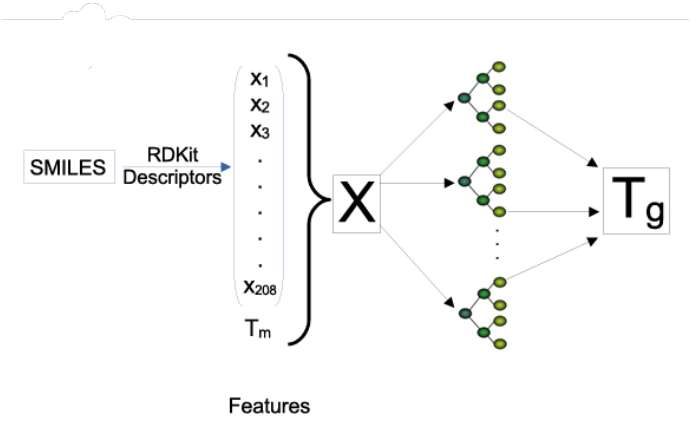

Fig. 3: Schematic representation of the workflow of SMILES mode. It uses molecular descriptors from a SMILES string.

IV. Data Augmentation

Data augmentation is a crucial step in enhancing the quality and quantity of data available for training machine learning models. In this study, the initial dataset comprised only SMILES (Simplified Molecular Input Line Entry System) descriptors and the corresponding glass transition temperature (T_g) values for various chemical compounds. While the SMILES descriptors provide a compact representation of the molecular structure, they alone do not fully capture the physical and chemical properties essential for accurate T_g prediction.

To address this limitation, we employed the RDKit library to extract additional chemical properties from the SMILES descriptors. These properties included various electrostatic and topological features, such as maximum absolute charge, maximum partial charge, and other related descriptors. However, these properties did not encompass the physical characteristics of the compounds, particularly those related to molecular mass and composition, which are known to significantly influence thermal properties like the glass transition temperature.

Recognizing the importance of molecular mass and composition, we augmented the dataset by including detailed compositional features. Specifically, we calculated the number of different types of atoms and functional groups present in each molecule. These features were derived using SMARTS (SMiles ARbitrary Target Specification) patterns to identify specific substructures within the molecules. The newly added features included:

- **Number of Methyl Groups (`num_CH3`):** Representing the count of `CH3` groups in the molecule.
- **Number of Methylene Groups (`num_CH2`):** Representing the count of `CH2` groups.
- **Number of Methine Groups (`num_CH`):** Representing the count of `CH` groups.
- **Number of Carbon Atoms (`num_C`):** Total number of carbon atoms, excluding those in `CH3`, `CH2`, and `CH` groups.
- **Number of Hydroxyl Groups (`num_OH`):** Count of `OH` groups, indicating alcohol or phenol presence.
- **Number of Ether Linkages (`num_COC`):** Count of `COC` groups, representing ether linkages.
- **Number of Carbonyl Groups (`num_OC`):** Count of `C=O` groups, indicating the presence of carbonyl groups.
- **Number of Nitrogen Atoms (`num_N`):** Total number of nitrogen atoms in the molecule.
- **Number of Halogen Atoms (`num_Hal`):** Count of halogen atoms, including fluorine, chlorine, bromine, and iodine.

By incorporating these features, it provides a more comprehensive representation of the molecular structure, capturing both chemical and physical properties. The inclusion of these detailed compositional features significantly improved the accuracy of the predictive models. The augmented dataset allowed the models to better understand the relationship between molecular structure and the glass transition temperature.

The impact of data augmentation on model performance was substantial. After adding these new features, we observed a marked improvement in prediction accuracy, as reflected in lower mean squared error (MSE) and higher R^2 scores. This enhancement underscores the importance of considering both chemical and physical properties in predictive modeling of thermal properties.

The data augmentation process successfully enriched the dataset with critical compositional information, leading to more accurate and reliable predictions of glass transition temperature. This novel approach highlights the value of integrating detailed molecular composition features into machine learning models, paving the way for future research and applications in materials science and chemistry.

V. Evaluation Metrics

To evaluate the performance of the machine learning models used in predicting the glass transition temperature (Tg), we utilized three key metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the Coefficient of Determination (R^2 score). Each of these metrics provides unique insights into the accuracy and reliability of the models.

A. Mean Absolute Error (MAE)

Mean Absolute Error (MAE) measures the average magnitude of errors in a set of predictions, without considering their direction. It is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

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

where:

- n is the number of observations.
- y_i is the actual value.
- $\hat{y}_i$ is the predicted value.

MAE is a linear score, which means that all the individual differences are weighted equally in the average. It is easy to understand and interpret, making it a popular metric for regression problems. Lower MAE values indicate better model performance, as they signify smaller average errors.

B. Coefficient of Determination (R^2 Score)

The Coefficient of Determination, commonly known as the R^2 score, is a statistical measure that explains the proportion of the variance in the dependent variable that is predictable from the independent variables. It provides an indication of goodness-of-fit and typically ranges from 0 to 1.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where:

- y_i is the actual value.
- $\hat{y}_i$ is the predicted value.
- $\bar{y}$ is the mean of the actual values.

The R^2 score can be interpreted as the percentage of the response variable variation that is explained by the model. An R^2 score of 1 indicates that the model perfectly explains the variability of the response data, whereas an R^2 score of 0 indicates that the model does not explain any of the variability in the response data. Higher R^2 values indicate better model performance.

VI. Comparing Results Before and After Performing Data Augmentation

The impact of data augmentation on the accuracy of the predictive models was evaluated by comparing the performance of various machine learning models before and after the inclusion of additional compositional features. The table below summarizes the results:

Model	Metric	Before Augmentation	After Augmentation
Lasso Regression	MSE	8579.30	8579.30
	R^2 Score	0.4039	0.4039
Linear Regression	MSE	4554.45	4406.33
	R^2 Score	0.6836	0.6939
Random Forest	MSE	2336.13	2204.42
	R^2 Score	0.8377	0.8469
Gradient Boosting	MSE	3082.55	2778.67
	R^2 Score	0.7858	0.8070
XGBoost	MSE	2262.11	2236.85
	R^2 Score	0.8428	0.8446

TABLE I: Comparison of model performance before and after data augmentation

The results demonstrate that the inclusion of additional features, such as the number of different kinds of atoms and functional groups, led to an improvement in the accuracy of the predictive models. This can be observed from the decrease in Mean Squared Error (MSE) and the increase in the coefficient of determination (R^2 Score) for most models.

Best Results After Augmentation:

- **Best Parameters:** {colsample_bytree: 0.8, learning_rate: 0.1, max_depth: 7, n_estimators: 150, subsample: 1.0}
- **Best MSE:** 2018.41
- **Best R^2 Score:** 0.8598

Best Results Before Augmentation:

- **Best Parameters:** {colsample_bytree: 0.8, learning_rate: 0.1, max_depth: 7, n_estimators: 150, subsample: 0.8}
- **Best MSE:** 2042.12
- **Best R^2 Score:** 0.8581

We can clearly see that after including these features, the accuracies improved. Therefore, this approach can be considered a novel and effective strategy for predicting the glass transition temperature of chemical compounds.

References

[1] Gianluca Armeli, Jan-Hendrik Peters, and Thomas Koop, "Machine-Learning-Based Prediction of the Glass Transition Temperature of Organic Compounds Using Experimental Data," *ACS Omega*, vol. 8, no. 13, pp. 12298–12309, 2023.

[2] Ralph G. Beaman, "Relation between (apparent) second-order transition temperature and melting point," *Journal of Polymer Science*, vol. 9, no. 5, pp. 470–472, 1952.

[3] Raymond F. Boyer, "Relationship of first-to second-order transition temperatures for crystalline high polymers," *Journal of Applied Physics*, vol. 25, no. 7, pp. 825–829, 1954.

[4] Andrzej Nowok, Hubert Hellwig, Kajetan Koperwas, Wioleta Cieślik, Mateusz Dulski, Piotr Kuś, Marian Paluch, and Sebastian Pawlus, "Structure–Glass transition temperature relationship for non-polymeric molecules: The concept of internal plasticizing effect," *Journal of Molecular Liquids*, 2024, p. 124222.

[5] Anderson, E., G.D. Veith, and D. Weininger. 1987. SMILES: A line notation and computerized interpreter for chemical structures. Report No. EPA/600/M-87/021. U.S. Environmental Protection Agency, Environmental Research Laboratory-Duluth, Duluth, MN 55804

[6] Hunter, R.S., F.D. Culver, and A. Fitzgerald. 1987. SMILES User Manual. A Simplified Molecular Input Line Entry System. Includes extended SMILES for defining fragments. Review Draft, Internal Report, Montana State University, Institute for Biological and Chemical Process Control (IPA), Bozeman, MT.

[7] https://github.com/gashawmg/moleculardescriptors/blob/main/Molecular

[8] Weininger, D. 1988. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. Journal of Chemical Information and Computer Science 28: 31-36.

[9] Zhang, K., Li, X., Jin, Y., Jiang, Y. 2022. Machine learning glass caging order parameters with an artificial nested neural network. Soft Matter 18(33): 6270-6277.

[10] Brunton, S.L., Proctor, J.L., Kutz, J.N. 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences 113(15): 3932-3937.

[11] Clegg, P.S. 2021. Characterising soft matter using machine learning. Soft Matter 17(15): 3991-4005.