POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Prediction of Thermodynamic Properties Based on Functional Groups

Data Science in Chemical Engineering Final Project

Authors:

MohammadSina GhanbariPakdehi- 10952079

Mehdi Davari - 10963119

Sahar Nirumand - 10952047

Advisors:

Professor Alessandro Stagni

Ing. Ricardo Caraccio

Academic year 2024/2025

Table of Contents

1. Introduction

Molecular property prediction is valuable in chemical engineering, materials science, and quantum chemistry. These properties, including enthalpy, electronic energy, dipole moment, and vibrational properties, are crucial for understanding the behavior of molecules and designing new ones. These properties have long been predicted by first-principles quantum chemical calculations. Although precise, these calculations are computationally intensive and not efficient for large datasets or real-time processes. Benson and Buss, in 1958, posited that additive contributions from the molecular functional groups can predict the thermodynamic properties(Benson & Buss, 1958). Their systematic fragment approach facilitated the prediction of properties, such as heat capacity and enthalpy, without the need for quantum mechanical calculations and gained popularity among researchers. The available set of large datasets[1], which comprises over 130,000 molecules and their corresponding quantum-mechanical properties, easily accommodates the application of predictive models using machine learning technologies. This report develops and tests two sets of ML models for predicting 12 quantum molecular properties from structural descriptors by functional groups and from the elemental composition. The first set utilizes the classical, interpretable ML models, namely the Random Forests, Ridge Regression, and Histogram-Based Gradient Boosting. The second examines the use of a neural network (NN) model for detecting those complex, nonlinear associations.

By comparing the two approaches to modeling all properties and emphasizing the prediction of the dipole moment specifically, this work aims to clarify the merits and demerits of the ML and NN approaches to conventional quantum chemical methods.

[1] QM9 Dataset

2. Previous Studies

In recent years, the application of machine learning and deep learning to predict molecular and thermodynamic properties has become a significant focus in fields such as computational chemistry and materials science. The reason is simple: while traditional quantum chemical methods are highly accurate, they are also extremely slow and resource intensive. Running these calculations on large numbers of molecules is not practical. That is why many researchers have begun to turn to data-driven approaches — ones that utilize patterns in molecular structures and functional groups to make accurate predictions much more quickly.

Interestingly, this idea of estimating molecular properties from smaller building blocks is not new. The concept dates to the work of Benson and Buss in 1958, who introduced the group contribution method. Their approach was based on the idea that we could estimate a molecule's properties by adding up the contributions of its parts. It was a simple yet clever idea — and one that still shapes our thinking about molecular prediction today, especially as we build more advanced models with modern machine-learning tools.

One of the most prominent benchmark datasets in this domain is the QM9 dataset, which provides quantum chemical properties for over 130,000 small organic molecules(Ramakrishnan et al., 2014). Leveraging such datasets, Jung et al. (2024) proposed a substructure vector embedding approach within a feature selection framework (Jung et al., 2025). Their method encoded molecular substructures and employed various ML algorithms, including Random Forests and Gradient Boosting, to predict molecular quantum properties with improved accuracy and interpretability. Their findings support the idea that encoding molecular structure explicitly can enhance prediction performance, especially when feature selection is carefully integrated into the modeling pipeline.

Zeng et al. (2022) introduced a novel perspective to molecular property prediction with their model, ImageMol—a deep learning framework that learns directly from 2D images of molecules. The approach's unique feature is its avoidance of the usual reliance on handcrafted molecular descriptors or labeled datasets(Zeng et al., 2022). Instead, ImageMol was trained on an enormous dataset of more than 10 million unlabeled molecules, utilizing self-supervised learning techniques to discover functional chemical patterns independently. What is even more exciting is that ImageMol proved helpful beyond benchmarks — it was able to identify potential inhibitors for SARS-CoV-2, showing real promise in drug discovery. This work highlights how self-supervised learning and image-based representations can open up new, practical possibilities in molecular research.

Another notable approach is presented in the ChemXploreML pipeline (arXiv:2505.08688), which integrates RDKit-generated functional group descriptors with classical ML models (Marimuthu & McGuire, 2025). The pipeline highlights the utility of domain knowledge—functional group counts and elemental composition—as features for model training. The study evaluates several regression models, including Ridge Regression, Random Forests, and Gradient Boosting, and also explores neural network architectures. The comparison revealed that while classical models offer interpretability and robust baseline performance, neural networks are better at capturing nonlinear relationships in the data, especially when enough training samples are available.

These studies show a shift from traditional descriptor-based models to advanced image-based neural architectures. They also highlight the importance of dataset quality, feature engineering, and model validation strategies in achieving reliable predictions. For this project, the insights from these works inform the design of both classical and neural network models trained on functional group information, aiming to strike a balance between interpretability and predictive accuracy.

3. Data Preparation: From XYZ to SMILES

The QM9 dataset, a complex collection of detailed information on over 130,000 small organic molecules, is a testament to the depth of our field. Each molecule is saved in an .xyz file, which includes the positions of its atoms and a set of quantum properties calculated using advanced methods. To utilize this data in a machine learning (ML) model, we need to convert the .xyz files into a structured table format. This process involves reading the properties of each molecule and creating a standardized way to represent its structure using SMILES notation.

In the following section, we describe the structure of the .xyz files, the selection of target properties, and the conversion of these properties to a clean CSV dataset suitable for ML modeling.

3.1 Structure of QM9.xyz Files

Each of the .xyz files from the QM9 dataset contains a detailed description of a single molecule. It is formatted according to the following structure:

Table 1- contents of .xyz file with description

Line	Description
1	Number of atoms in the molecule
2	17 quantum properties, separated by tabs
3 to n	Atom lines with atomic symbol, Cartesian coordinates (x, y, z), and Mulliken charge
n+1	Vibrational frequencies
n+2	SMILES strings (original GDB and geometry-optimized)
n+3	InChI strings (original GDB and geometry-optimized)

Among the 17 properties on line 2, we selected the following 12 for prediction:

- **Electronic properties**: HOMO, LUMO, energy gap, dipole moment (μ), spatial extent (r^2), and polarizability (α)
- **Thermodynamic properties**: ZPVE, U_0, U, H, G, and Cv

The other 5 properties, i.e., the rotational constants A, B, and C, and the GDB index and tag, are omitted as they are not pertinent to the scope of the present project.

3.2 SMILES Representation

SMILES[2] strings offer a one-dimensional, compact description of a molecule's structure that is unique. All .xyz files from QM9 contain two SMILES strings, listed on the second-to-last line, corresponding to the original GDB enumeration and the geometry-optimized one.

We use SMILES from GDB9 as the structural descriptor. SMILES has the following merits:

[2] Simplified Molecular Input Line Entry System

- It is readable by cheminformatics tools such as RDKit.
- It abstracts away 3D geometry while preserving chemical structure.
- It allows for efficient and reproducible descriptor extraction for ML.

3.3 Processing Methodology

The essential steps for the conversion of raw .xyz files into a machine learning-compatible dataset are:

Step 1: File Access and Safe Parsing

- The entire dataset is stored in a .tar.gz archive.
- Using Python's tarfile module, each file is read sequentially without complete extraction.
- The parser reads the file content in binary mode to ensure compatibility with the encoding.

Step 2: Target Property Extraction

- The second line of each file is split into 17 quantum property values.
- A subset of 12 properties is selected based on their relevance to electronic and thermodynamic behavior.
- These values are parsed as floats and stored in a structured format.

Step 3: Extraction of SMILES

- The canonical SMILES string is retrieved from the second-to-last line of the file.
- If the SMILES string is malformed or missing, the file is skipped and logged.

Step 4: Data Assembly

- For every valid molecule, we form a dictionary with the following structure: {index, smiles, mu, alpha, homo, lumo, gap, r2, zpve, U0, U, H, G, Cv}
- These entries are merged into a pandas Dataframe and exported to a .csv file.

Step 5: Fault Tolerance

- Try-except blocks ensure that parsing errors or non-numeric values do not interrupt the pipeline.
- Corrupted or incomplete files are logged and excluded from processing.

The final dataset is a .csv file where each row corresponds to a molecule, and each column corresponds to either the SMILES string or one of the 12 selected target properties. This structured dataset serves as the foundation for all subsequent stages of feature generation, model training, and performance evaluation, which will be investigated in the following chapters.

4. Functional Group Feature Extraction

Accurate prediction of molecular properties through machine learning (ML) requires the transformation of molecular structures into quantitative descriptors. While many studies rely on high-dimensional fingerprints or quantum-chemistry-based descriptors, this project emphasizes **interpretable, chemically meaningful features**. Specifically, we extract:

- **Functional group counts** (e.g., alcohols, amines, carbonyls), and
- **Atomic composition features** (e.g., counts of C, H, O, N, F atoms)

These features are derived from **SMILES representations** of molecules in the QM9 dataset, using RDKit's built-in catalog of functional groups. The approach provides not only computational efficiency but also alignment with chemical intuition.

4.1 Data Source and Software Tools

The analysis utilizes the QM9 dataset, a benchmark collection of over 130,000 small organic molecules, each annotated with 12 quantum chemical properties. Our workflow is implemented in Python using the following tools:

- **RDKit** for SMILES parsing and functional group detection
- **Pandas** for data handling and preprocessing
- **collections.Counter** for atom counting
- **os and tqdm** for file access and logging

4.2 Methodology: Functional Group Descriptor Extraction

4.2.1 Functional Group Catalog Initialization

We employed RDKit's FunctionalGroups.txt file, which contains SMARTS-based definitions of over 100 functional groups. This catalog is initialized via RDKit's FragmentCatalog module:

```
from rdkit.Chem.Fragments import FragmentCatalog
fparams = FragmentCatalog.FragCatParams(1, 6, catalog_path)
fcat = FragmentCatalog.FragCatalog(fparams)
fgen = FragmentCatalog.FragCatGenerator()
```

Here, fparams sets up the catalog size and SMARTS range, fcat loads the entries, and fgen is used to scan molecules and count matches.

4.2.2 Parsing SMILES and Extracting Functional Groups

Each molecule is parsed into a Mol object:

```
mol = Chem.MolFromSmiles(smiles)
fgen.AddFragsFromMol(mol, fcat)
```

We iterate through catalog entries to collect matched fragment counts:

```
for fid in range(fcat.GetNumEntries()):
```

```
count = fgen.GetEntryCounts(fid)
name = fcat.GetEntryDescription(fid)
fg_counts[name] = count
```

This results in a dictionary mapping functional group names to their frequencies, yielding a sparse but interpretable feature vector for each molecule.

4.2.3 Atomic Composition Calculation

We also extract the molecular formula using:

```
atom_counts = Counter(atom.GetSymbol() for atom in mol.GetAtoms())
```

This generates additional features such as C, H, O, N, and F counts, capturing the size and elemental composition of the molecule.

4.2.4 Dataset Assembly and Cleaning

Each molecule is summarized as a single dictionary containing the SMILES string, all 12 quantum target properties, and atomic counts and functional group frequencies

These records are collected into a DataFrame, and missing values are filled with zero. Optionally, constant (zero-variance) columns can be removed before exporting the dataset as .csv for downstream ML modeling.

4.2.5 Error Handling and Logging

Robust error handling was implemented to address:

- Invalid SMILES strings
- Failures in functional group matching
- Incomplete property files

Molecules with errors are skipped and logged to maintain dataset integrity and reproducibility.

4.3 Limitations and Flexibility of RDKit's Functional Group Catalog

The key limitations are:

- **SMARTS specificity**: Many entries impose strict constraints (e.g., ring membership, degree, hydrogen count), excluding valid substructures like imines in heterocycles.
- **Lack of generic definitions**: Broad motifs such as [C]=[N] are not included by default, limiting coverage of generalized chemical patterns.
- **Context-dependence**: Some groups are only detected in specific molecular topologies (e.g., terminal positions or side chains).
- **Hybridization mismatch**: Atoms in delocalized systems (e.g., aromatic amines) may escape detection if the SMARTS demands sp^3 hybridization.

To overcome these issues, users may:

- **Edit FunctionalGroups.txt** to relax constraints or add new entries
- **Define custom SMARTS** patterns and apply programmatically via RDKit
- **Aggregate subtypes post hoc**, e.g., combining primary/secondary amines

These options allow tailoring the descriptor space to specific chemical contexts or modeling needs.

5. Dataset Visualization

Before building machine learning models and neural networks, we spent time getting to know our dataset. We carefully explored and visualized the data to understand how the different target variables appeared, how they interacted with each other, and what potential challenges might arise later. Taking this step early was crucial because it enabled us to determine how to clean and prepare the data, as well as how to build models that would yield reliable results.

5.1 Distribution Analysis

We first examined the distribution of each target variable using histograms, which provided a snapshot of how the values are spread out. Two variables in particular — the dipole moment (μ) and electronic spatial extent (r^2) — exhibited strong positive skew, with long tails extending toward higher values. In other words, there were many smaller values and just a few huge ones. This kind of skew can make regression models challenging to train because it breaks some of the assumptions these models typically rely on, such as having normally distributed errors. Meanwhile, other targets, such as polarizability (α), HOMO, LUMO, the energy gap, and thermodynamic properties (U_0, U, H, G, Cv), had much more balanced and symmetric distributions.

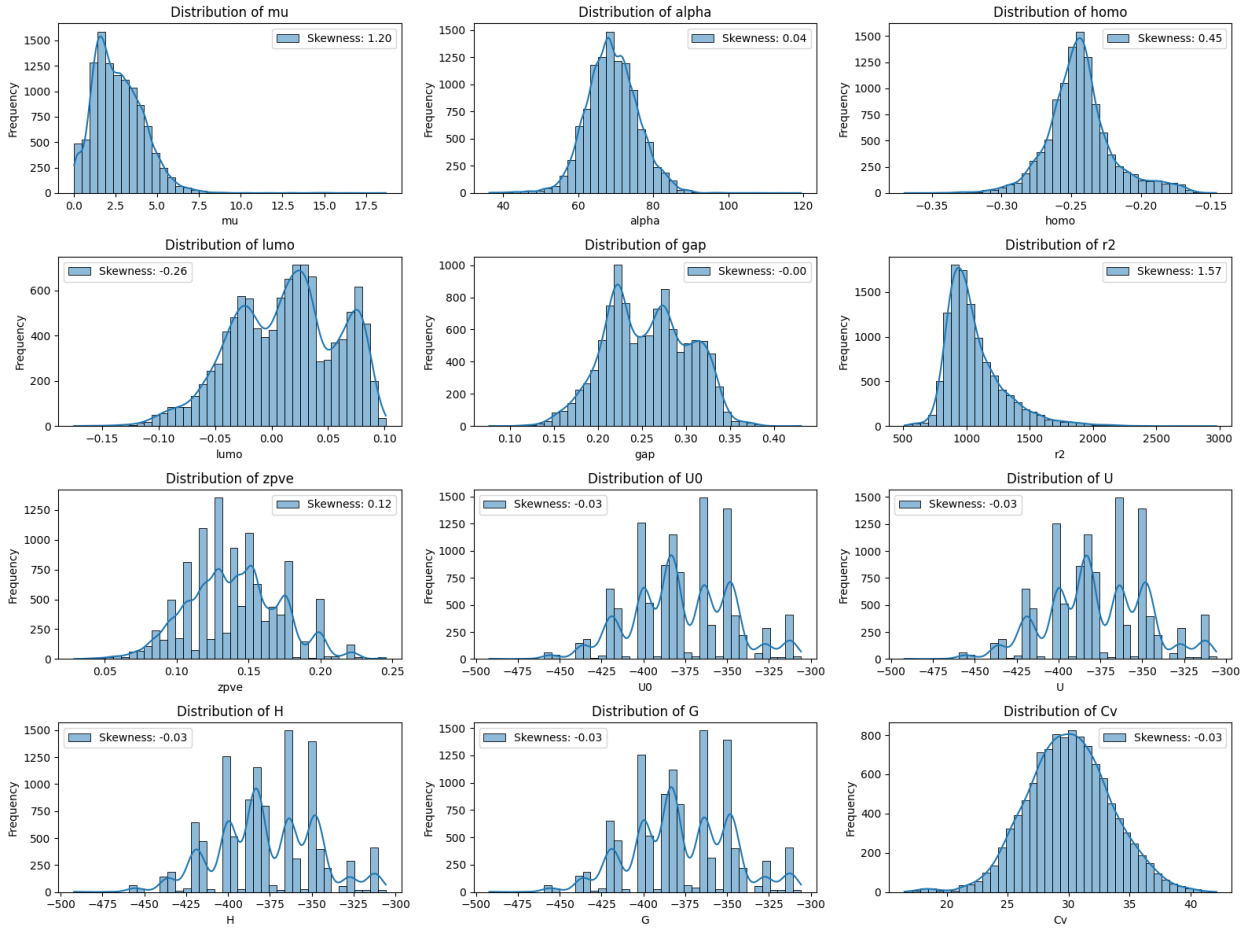

Figure 1- Distribution of target properties

5.2 Transformation of Skewed Targets

To deal with the skewness in μ and r², we applied a logarithmic transformation, which helped smooth out the long tails and made the distributions more symmetric. The updated histograms and box plots showed fewer extreme outliers, which means the data should work better with regression models later on.

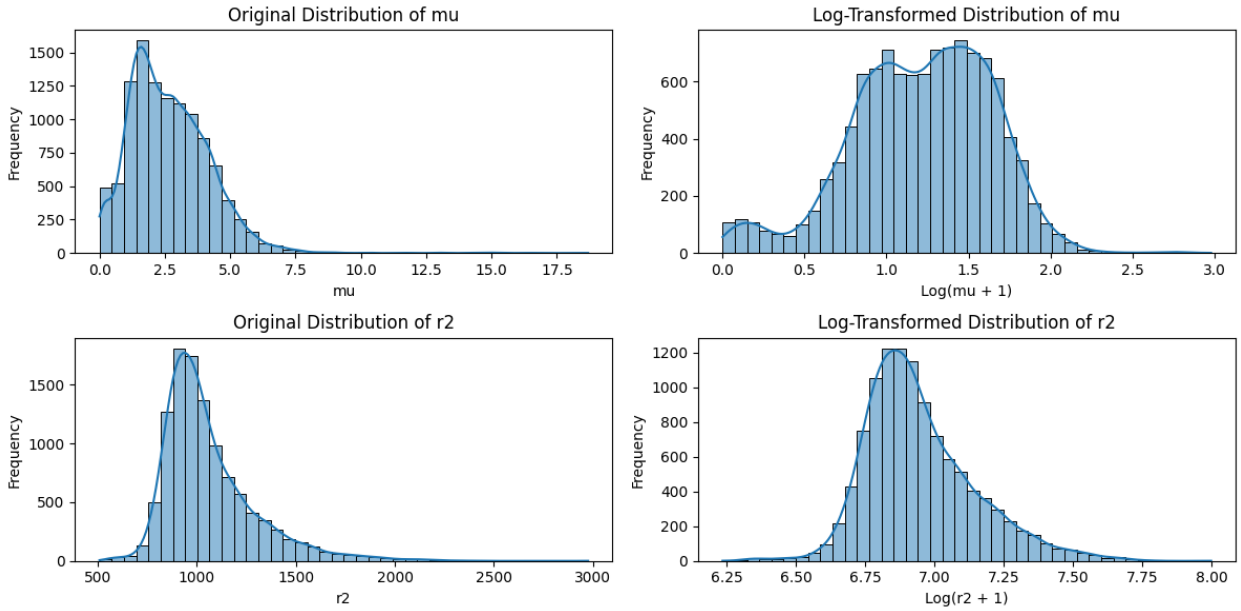

Figure 2- Distribution of μ and r^2 before and after transformation

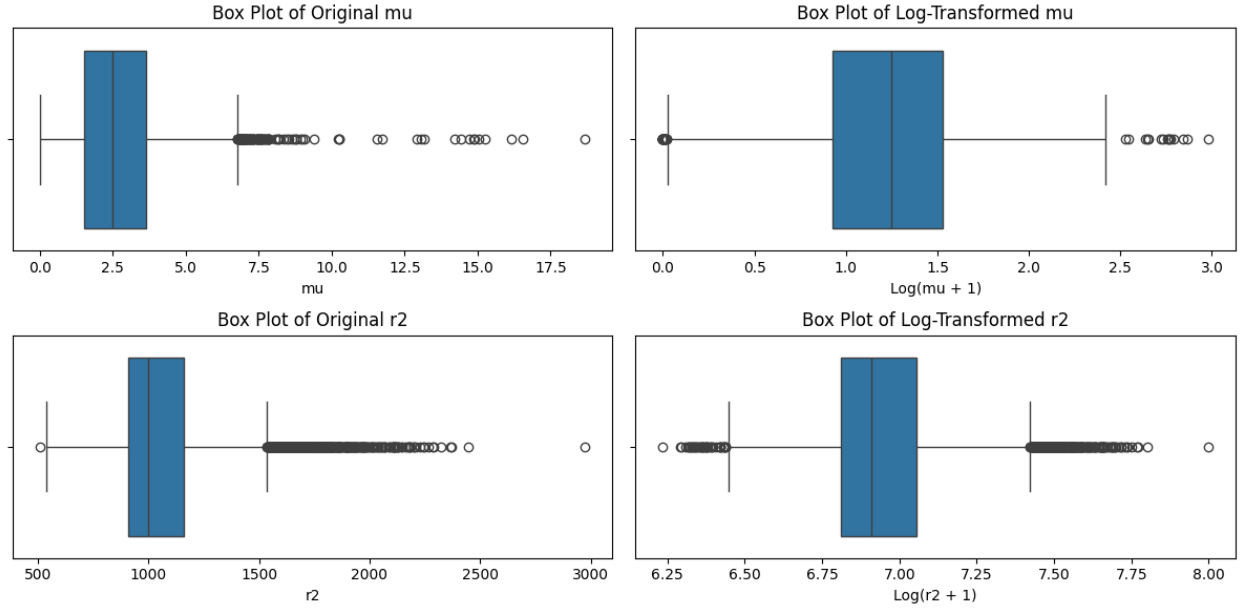

Figure 3- Box plot of μ and r2 before and after transformation

5.3 Outlier Analysis

Box plots also helped us spot outliers — particularly in μ and r^2 — where some values stood far outside the main range. Recognizing these outliers early was important because it allowed us to consider modeling approaches that are less sensitive to their influence.

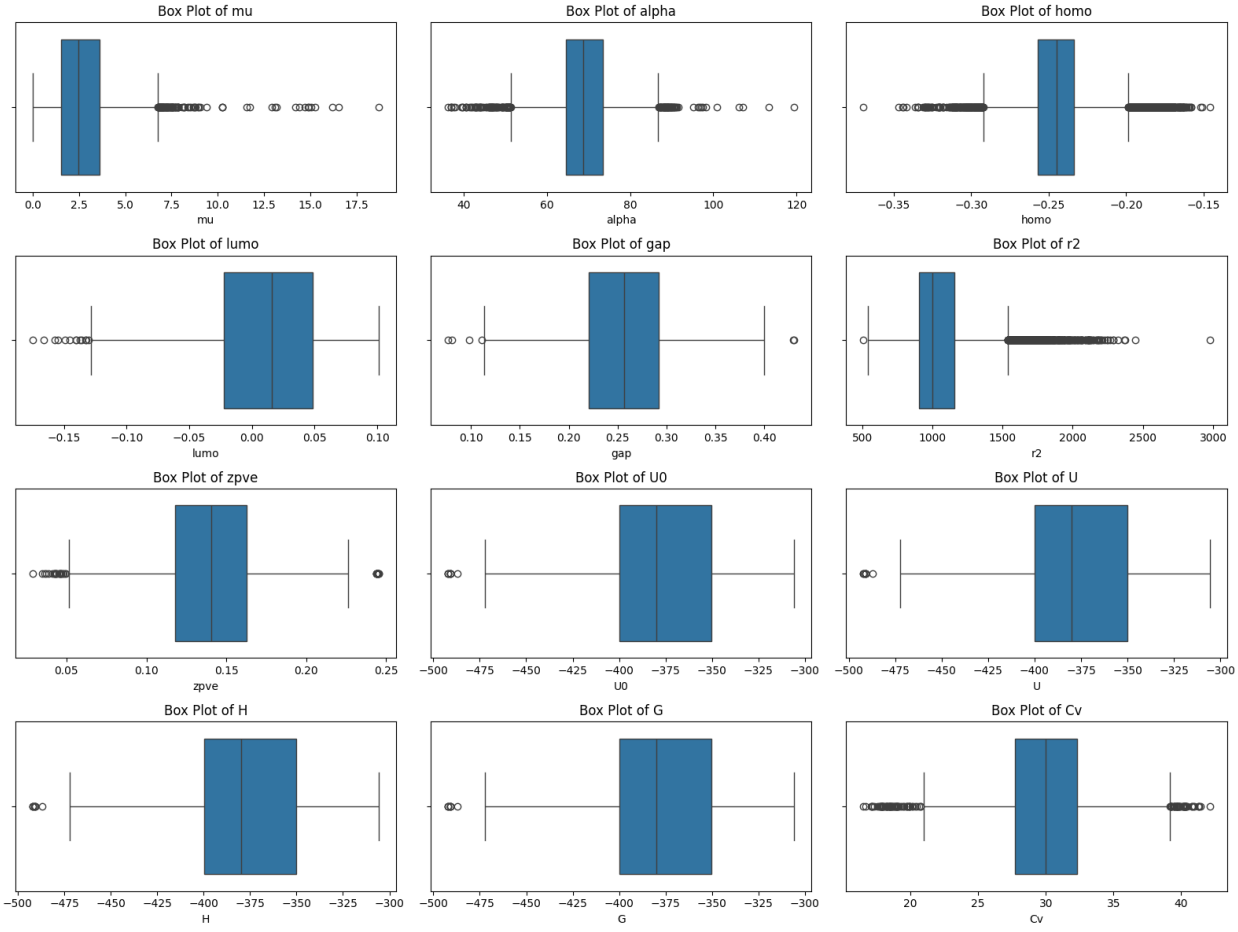

Figure 4 - Box plot of target properties

5.4 Correlation Analysis

We also examined the relationship between the target variables by creating a correlation heatmap. We noticed that some variables — such as α, the energy gap, LUMO, and ZPVE — exhibited strong positive correlations, indicating that they tend to change together and may share patterns that our models can utilize to make more accurate predictions. In contrast, the dipole moment (μ) showed little connection with the other variables, suggesting that predicting μ could be more challenging and might require additional features or a different approach.

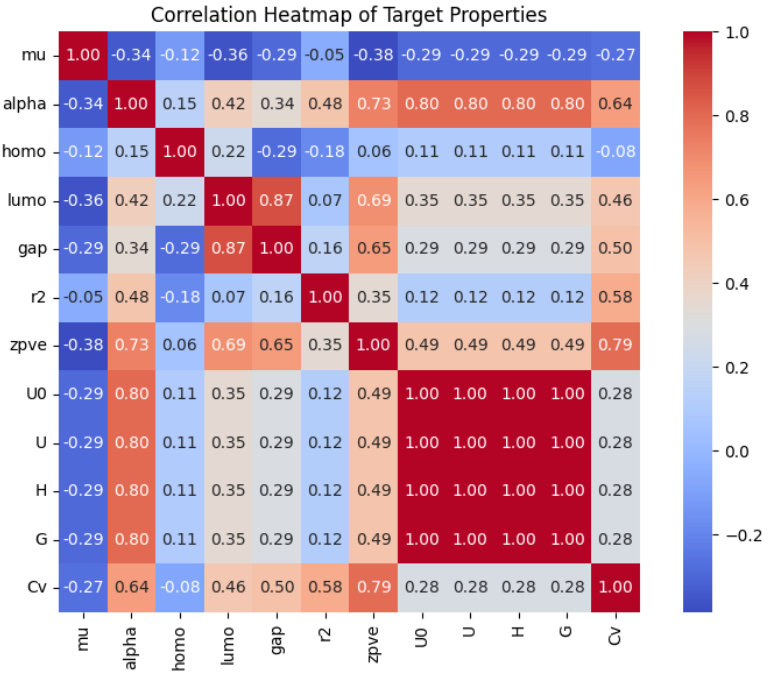

Figure 5 - Correlation matrix of target properties

Overall, this step of visualizing the dataset gave us valuable insights. It set the stage for making smart choices about data preparation and model building in the subsequent phases of our project.

6. Model Selection and Feature Engineering

6.1 Choice of Regression Algorithms

To predict quantum chemical properties with machine learning, this study adopted three well-established regression algorithms: **Ridge Regression**, **Random Forest (RF)**, and **HistGradientBoostingRegressor (HGBR)**. These models were selected based on their balance between performance, scalability, and interpretability:

- **Ridge Regression** is a regularized linear model suitable for high-dimensional feature spaces. It offers fast computation and serves as a strong baseline, particularly when target properties are linearly related to the input features.
- **Random Forest** is a non-parametric ensemble method based on decision trees. It captures non-linear relationships, handles feature interactions effectively, and is relatively robust to outliers and noise. It is well suited for datasets of intermediate to large size, such as ours (~10,000 samples).
- **HistGradientBoostingRegressor** is a powerful version of gradient boosting that works by grouping feature values into bins, making it significantly faster, especially on large datasets. It is also great at capturing complex, non-linear patterns in the data. Due to its speed and strong performance on structured (tabular) datasets, it is a highly promising choice for predicting molecular properties.

We excluded more computationally intensive methods, such as **Support Vector Regression (SVR)** and **K-Nearest Neighbors (KNN)**, due to their poor scalability with dataset size and high memory demands. Furthermore, deep learning models were not considered in this work to preserve model transparency and reduce training complexity.

6.2 Benchmark Results Before Feature Engineering

The initial benchmarking was conducted using only the original feature set, which included basic molecular fingerprints and SMILES-derived characteristics. The predictive performance across all target variables is summarized in the following table:

Table 2- Best hyperparameter from each regressor per target before feature engineering

Parameter	Ridge			RF			HGBR			Best Model
	RMSE	**MAE**	**R^2**	**RMSE**	**MAE**	**R^2**	**RMSE**	**MAE**	**R^2**	
gap	0.033862	0.027766	0.467891	0.030340	0.023555	0.572846	0.030234	0.023564	0.575822	HGBR
μ	1.264406	0.921028	0.327395	1.067138	0.733215	0.520898	1.069721	0.733469	0.518576	RF
α	2.726814	2.022601	0.838412	2.739537	2.017891	0.836901	2.730353	2.023988	0.837992	Ridge
HOMO	0.015994	0.011990	0.538877	0.014595	0.010734	0.616044	0.014654	0.010835	0.612929	RF
LUMO	0.028744	0.022960	0.598068	0.025794	0.019707	0.676351	0.025992	0.019967	0.671368	RF
r^2	223.726804	155.351656	0.159432	224.190535	155.928552	0.155944	224.146351	155.904724	0.156276	Ridge
zpve	0.018442	0.014779	0.655373	0.017699	0.014003	0.682565	0.017705	0.014066	0.682374	RF
U0	0.977457	0.784915	0.998939	1.325283	0.787599	0.998050	1.124104	0.796777	0.998597	Ridge
U	0.976791	0.784356	0.998941	1.371433	0.790378	0.997912	1.140322	0.799132	0.998556	Ridge
H	0.976791	0.784356	0.998941	1.360577	0.790409	0.997945	1.140322	0.799132	0.998556	Ridge
G	0.978250	0.785592	0.998938	1.359009	0.794700	0.997950	1.464574	0.823845	0.997619	Ridge
C_v	2.497557	2.008112	0.469518	2.452043	1.961959	0.488677	2.465288	1.981486	0.483138	RF

The results indicate that, while the models could successfully capture trends for certain targets (e.g., thermodynamic quantities), their performance was substantially lower for more complex or shape-dependent properties:

- The **electronic properties**, such as μ, gap, and r^2, showed poor predictive accuracy, with R^2 values below 0.6, highlighting the insufficiency of the initial feature space in capturing subtle molecular characteristics.
- Conversely, **thermodynamic properties** (U_0, U, H, and G) were accurately modeled ($R^2 > 0.998$) even with the baseline features. These properties are more strongly correlated with molecular size and composition, which were adequately represented in the original descriptors.
- **Cv and zpve**, both related to vibrational modes, were only moderately predicted, with R^2 values of 0.489 and 0.683, respectively, suggesting that structural and electronic descriptors were underrepresented.

This initial evaluation made it clear that the models needed better input features — ones that carry more chemical and physical meaning. This was especially important for properties that depend heavily on molecular geometry or electronic structure, where simple features just weren't enough to capture the complexity.

6.3 Justification for Feature Engineering

To enhance predictive accuracy and capture the underlying chemical phenomena more effectively, we introduced a set of engineered features derived from RDKit. The following function was used to compute descriptors for each molecule:

```
def compute_descriptors(smiles_series):
    """Compute 3D inertia, TPSA, MR, and Gasteiger charge stats."""
```

These descriptors were selected based on their relevance to the target properties and their computational efficiency. They include:

- **3D Inertial Descriptors**:
  - `PMI1`, `PMI2`, `PMI3`: principal moments of inertia capturing molecular shape.
  - `inertia_sum`: total spatial inertia, related to molecule size and mass distribution.
  - `inertia_ratio`: shape anisotropy (elongation vs compactness), computed as PMI1 / PMI2.
- **Topological Polar Surface Area (TPSA)**: a proxy for hydrogen bonding capacity and polarity, relevant for solvation and reactivity.
- **Molar Refractivity (MR)**: an indicator of molecular polarizability and van der Waals volume, often correlated with `alpha` and dispersion forces.
- **Gasteiger Charges**:
  - `chg_mean`: average atomic partial charge, linked to dipole moment.
  - `chg_std`: standard deviation of charge, reflecting charge separation and electronic asymmetry.

The selected descriptors offer a **more physically meaningful** way to represent each molecule, making it possible to avoid costly DFT calculations. They also improve on the original features by adding valuable details about the molecule's shape, polarity, and how charge is spread out. This extra information plays a crucial role in helping the models better predict both electronic and thermodynamic properties.

6.4 Benchmark Results After Feature Engineering

After incorporating the above descriptors and applying a preprocessing pipeline (standard scaling and low-variance filtering), we re-evaluated all models. The results show **marked improvement** in predictive accuracy across nearly all target variables:

Table 3- Best hyperparameter from each regressor per target after feature engineering

Parameter	Ridge			RF			HGBR			Best Model
	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2	
gap	0.030892	0.025231	0.557163	0.017721	0.012163	0.854279	0.017862	0.012545	0.851947	RF
μ	1.263358	0.919680	0.328510	1.003336	0.674495	0.576474	1.020851	0.688871	0.561559	RF
α	1.777714	1.240922	0.931321	1.316487	0.793707	0.962336	1.285436	0.794800	0.964091	HGBR
HOMO	0.014884	0.011477	0.600663	0.010080	0.006911	0.816845	0.010473	0.007183	0.802304	RF
LUMO	0.027082	0.021579	0.643218	0.015475	0.010392	0.883499	0.015604	0.010819	0.881562	RF
r^2	93.348595	56.842963	0.853664	74.381408	45.331891	0.907089	74.426117	46.097928	0.906978	RF
zpve	0.011979	0.008765	0.854602	0.005024	0.002659	0.974420	0.004513	0.002892	0.979361	HGBR
U0	0.629134	0.460649	0.999561	1.220116	0.175806	0.998347	1.239426	0.423560	0.998295	Ridge
U	0.628777	0.460385	0.999561	1.259915	0.178814	0.998238	1.297769	0.432003	0.998130	Ridge
H	0.628777	0.460385	0.999561	1.192107	0.175144	0.998422	1.297769	0.432003	0.998130	Ridge
G	0.629485	0.460882	0.999560	1.219573	0.182711	0.998349	1.262771	0.419111	0.998230	Ridge
C_v	1.687271	1.281742	0.757892	0.982804	0.733698	0.917857	0.930985	0.701405	0.926290	HGBR

The following key improvements were observed:

- The **R^2 for gap increased from 0.576 to 0.854**, confirming that shape descriptors (inertia, MR) significantly contribute to modeling HOMO-LUMO separation.
- r^2 **showed the most dramatic improvement**, with R^2 rising from 0.159 to 0.907. This indicates that polarizability and charge distribution (captured by `MR` and `chg_std`) are critical features for this property.
- The performance of C_v improved from 0.489 to 0.926, suggesting a strong dependence on geometric and mass distribution descriptors.
- While `mu` saw only a modest R^2 gain from 0.521 to 0.576, its error metrics improved, likely due to better charge modeling via Gasteiger descriptors.
- The prediction accuracy for α, ZPVE, and LUMO improved noticeably — each gaining more than 0.1 in R^2 score, which boost suggests the models were better able to capture patterns related to electron distribution and molecular surface area.

Overall, the benchmark results underscore the importance of **chemically relevant, physically interpretable descriptors**. By embedding 3D structure, polarity, and electrostatics into the feature space, the models were able to approximate DFT-level accuracy for several properties while maintaining interpretability and efficiency.

7. Classical Machine Learning Modeling

In this section, we develop predictive models for quantum molecular properties using **classical machine learning** techniques. These models are trained to approximate high-level quantum chemical computations with significantly reduced computational cost, using functional groups and atom counts and molecular features derived from molecular representations.

7.1 Objectives and Scope

The primary objective of the classical ML modeling phase is to predict multiple quantum properties, including dipole moment, orbital energies, polarizability, thermodynamic energies, and vibrational characteristics, based solely on molecular descriptors, which enables rapid property estimation for new molecules without the need for costly density functional theory (DFT) calculations. The objective was twofold:

1. To evaluate baseline model performance using standard molecular descriptors
2. To assess the effect of domain-specific feature engineering on improving model accuracy.

7.2 Hyperparameter Optimization Strategy

To ensure optimal performance, we implemented a two-stage hyperparameter tuning approach:

- **Stage 1: RandomizedSearchCV**

  This method randomly samples from a predefined distribution of hyperparameter values. It is computationally efficient, especially for high-dimensional parameter spaces and complicated models like Random Forest and Histogram-based Gradient Boosting.

- **Stage 2: GridSearchCV**

  After getting promising results from the randomized search, we conducted a grid search to fine-tune the models further. This approach allows us to concentrate on the most effective parameter combinations and improve performance without spending extra time on settings that were unlikely to yield benefits.

By combining both steps, we found a practical and efficient way to optimize the models. This approach struck the right balance between thoroughness and process manageability.

The tables below show which models performed best and the final hyperparameter settings we chose for each target variable as a result of this two-stage tuning process.

Table 4 - Best model and hyperparameter from randomized and grid search for each target before feature engineering

Target	Model	Best Parameters (Random Search)	Best Parameters (Grid Search)
U_0, U, H, G	Ridge	-	'model__estimator__alpha': 0.001
gap	HGBR	'model__l2_regularization': 0.37366005506869043 'model__learning_rate': 0.09095381985836196 'model__max_depth': 11 'model__max_iter': 709	'model__l2_regularization': 0.48575807158929757 'model__learning_rate': 0.06366767390085337 'model__max_depth': 14 'model__max_iter': 709
μ	RF	'model__max_depth': 9 'model__max_features': 0.7252813963310067 'model__min_samples_leaf': 5 'model__n_estimators': 428	'model__max_depth': 9 'model__max_features': 0.7252813963310067 'model__min_samples_leaf': 5 'model__n_estimators': 556
HOMO	Ridge	-	'model__alpha': 0.1
LUMO	RF	'model__max_depth': 17 'model__max_features': 0.8832364382153151 'model__min_samples_leaf': 3 'model__n_estimators': 305	'model__max_depth': 22 'model__max_features': 0.6182655067507205 'model__min_samples_leaf': 2 'model__n_estimators': 396
r^2	Ridge	-	'model__alpha': 0.1
α	HGBR	'model__l2_regularization': 0.32553851275097223 'model__learning_rate': 0.14724395133156712 'model__max_depth': 4 'model__max_iter': 627	'model__l2_regularization': 0.22787695892568055 'model__learning_rate': 0.19141713673103727 'model__max_depth': 3 'model__max_iter': 815
zpve	RF	'model__max_depth': 17 'model__max_features': 0.8832364382153151 'model__min_samples_leaf': 3 'model__n_estimators': 305	'model__max_depth': 17 'model__max_features': 0.6182655067507205 'model__min_samples_leaf': 2 'model__n_estimators': 305
C_v	RF	'model__max_depth': 17 'model__max_features': 0.8832364382153151 'model__min_samples_leaf': 3 'model__n_estimators': 305	'model__max_depth': 22 'model__max_features': 0.6182655067507205 'model__min_samples_leaf': 2 'model__n_estimators': 396

Table 5 - Best model and hyperparameter from randomized and grid search for each target after feature engineering

Target	Model	Best Parameters (Random Search)	Best Parameters (Grid Search)
U_0, U, H, G	Ridge	-	model__estimator__alpha: 0.01
gap	RF	model__max_depth: 17, model__max_features: 0.8832364382153151, model__min_samples_leaf: 3, model__n_estimators: 305	model__max_depth: 22, model__max_features: 0.6182655067507205, model__min_samples_leaf: 2, model__n_estimators: 214
μ	RF	model__max_depth: 17, model__max_features: 0.4899443222417271, model__min_samples_leaf: 5, model__n_estimators: 379	model__max_depth: 22, model__max_features: 0.4899443222417271, model__min_samples_leaf: 4, model__n_estimators: 265
HOMO	RF	model__max_depth: 17, model__max_features: 0.8832364382153151, model__min_samples_leaf: 3, model__n_estimators: 305	model__max_depth: 22, model__max_features: 0.6182655067507205, model__min_samples_leaf: 2, model__n_estimators: 396
LUMO	RF	model__max_depth: 17, model__max_features: 0.8832364382153151, model__min_samples_leaf: 3, model__n_estimators: 305	model__max_depth: 22, model__max_features: 0.6182655067507205, model__min_samples_leaf: 2, model__n_estimators: 396
r^2	RF	model__max_depth: 17, model__max_features: 0.4899443222417271, model__min_samples_leaf: 5, model__n_estimators: 379	model__max_depth: 17, model__max_features: 0.4899443222417271, model__min_samples_leaf: 4, model__n_estimators: 493
α	HGBR	model__l2_regularization: 0.37366005506869043, model__learning_rate: 0.09095381985836196, model__max_depth: 11, model__max_iter: 709	model__l2_regularization: 0.2615620385480833, model__learning_rate: 0.09095381985836196, model__max_depth: 11, model__max_iter: 709
zpve	HGBR	model__l2_regularization: 0.02904180608409973, model__learning_rate: 0.13992642186624027, model__max_depth: 7, model__max_iter: 659	model__l2_regularization: 0.02904180608409973, model__learning_rate: 0.13992642186624027, model__max_depth: 7, model__max_iter: 857
C_v	HGBR	model__l2_regularization: 0.2117007403531848, model__learning_rate: 0.06923222772633546,	model__l2_regularization: 0.2117007403531848, model__learning_rate: 0.06923222772633546,

		model__max_depth: 11, model__max_iter: 771	model__max_depth: 11, model__max_iter: 1002

7.3 Modeling Performance Evaluation

The table below shows the model performance before and after feature engineering.

Table 6 - Modeling Performance Before and After Feature Engineering using regression

Property	**Before Feature Engineering**			**After Feature Engineering**		
	RMSE	MAE	R^2	RMSE	MAE	R^2
gap	0.0302	0.0235	0.5757	0.0184	0.0126	0.8430
μ	1.0693	0.7351	0.5190	1.0061	0.6728	0.5742
α	2.7221	2.0203	0.8390	1.3001	0.8174	0.9633
HOMO	0.0160	0.0120	0.5389	0.0102	0.0070	0.8114
LUMO	0.0258	0.0198	0.6752	0.0161	0.0108	0.8737
r^2	223.73	155.35	0.1594	79.2523	48.1949	0.8945
zpve	0.0178	0.0142	0.6783	0.0049	0.0031	0.9752
U_0	0.9775	0.7850	0.9989	0.7115	0.5284	0.9994
U	0.9769	0.7845	0.9989	0.7110	0.5281	0.9994
H	0.9769	0.7845	0.9989	0.7110	0.5281	0.9994
G	0.9783	0.7857	0.9989	0.7119	0.5287	0.9994
C_v	2.4705	1.9892	0.4809	0.9643	0.7251	0.9209

The results presented in Table 6 clearly demonstrate the overall enhancement in model performance achieved through the integration of feature engineering. This improvement is evident across nearly all target properties, with particularly substantial gains in R^2, RMSE, and MAE for non-linear and electronically sensitive descriptors.

For the linear quantum properties such as U, enthalpy H, free energy G, and U_0, the Ridge regression model already achieved exceptional performance even before feature augmentation. In both scenarios, the R^2 values remained consistently high, around 0.999, and the RMSEs decreased from ~0.98 to ~0.71 after feature engineering. These results confirm the largely linear nature of these energy-related targets and the sufficiency of linear models for their prediction, particularly when paired with even moderately enriched feature sets.

The impact of feature engineering became particularly clear in orbital energy levels. The R^2 score for predicting LUMO increased significantly from 0.6752 to 0.8737, and HOMO improved from 0.5389 to 0.8114. These significant gains suggest that electronic properties like these depend heavily on spatial and charge-related features that were not well represented in the original set of inputs.

We saw a similar trend with the HOMO-LUMO gap, where the R^2 score increased from 0.5757 to 0.8430, and the RMSE dropped significantly — from 0.0302 to 0.0184. These improvements are primarily due to the addition of more chemically meaningful features, such as topological polar surface area (TPSA) and Gasteiger charges, which better capture how electrons are distributed and how the molecule is shaped.

Although the increase in the model's R^2 for α from 0.8390 to 0.9633, the improvement in μ was more minor — increasing from 0.5190 to 0.5742 — which was still an important gain, particularly considering how difficult dipole moment has traditionally been to model. Even modest progress in this area signals that the added features are helping the model capture some of the underlying complexity that simpler inputs often miss. In both cases, the drop in RMSE and MAE indicates that the added features — especially those related to shape and charge distribution — made a significant difference. Geometric and electrostatic information play a significant role in shaping these types of polar properties.

When we examined vibrational and thermodynamic properties, the impact of feature engineering was evident. The R^2 score in predicting ZPVE experienced a significant improvement from 0.6783 to 0.9752. The R^2 score in predicting heat capacity C_v increased dramatically from 0.4809 to 0.9209, confirming that the added features significantly improved their ability to capture important molecular behaviors, such as motion, flexibility, and vibrations, all of which are necessary for understanding how molecules interact with energy.

One of the most unexpected and impressive results came from r^2. At first, it was among the most difficult properties to predict, with an R^2 score of just 0.1594 indicating that the model struggled to make sense of it. But after we introduced more advanced descriptors, the performance improved dramatically, with the R^2 jumping to 0.8945 and error rates dropping significantly. This sharp improvement suggests that r^2 depends on complex and often subtle characteristics like the shape of the molecule, how mass is distributed, and electrostatic interactions which basic features simply fail to capture.

Ultimately, feature engineering made a tangible and measurable difference. It not only boosted performance across a wide range of properties but also proved especially valuable for the more complex, non-linear targets. These results reinforce the strength of our two-step modeling approach and underscore the importance of domain-specific knowledge in designing descriptors for molecular machine learning.

7.4 Parity Plot Analysis: Comparison of Scenarios Before and After Feature Engineering

To visually assess and compare model performance across the two modeling scenarios, parity plots were generated for all targets once using the original features (Scenario A) and once using the augmented, engineered descriptors (Scenario B). The plots below show predicted values versus true values.

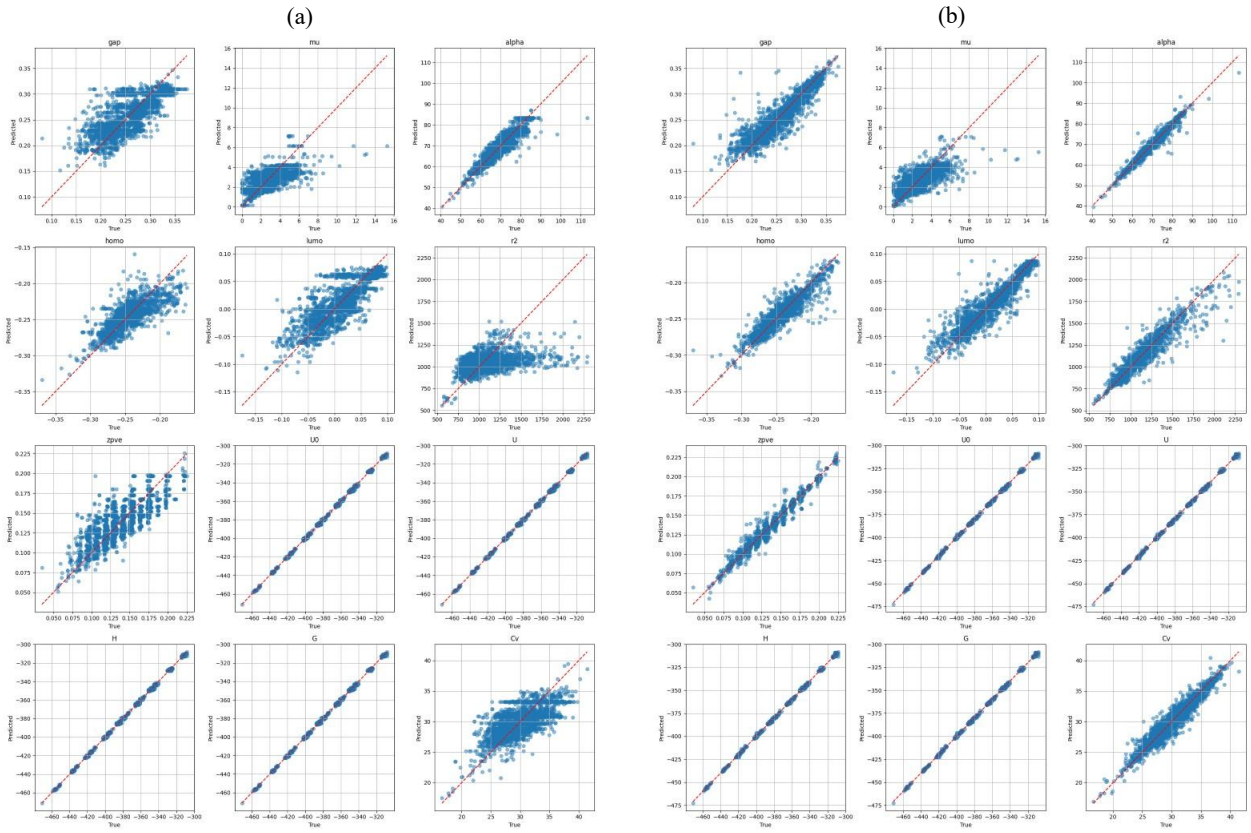

Figure 6 - Parity plots for each target (a) before feature engineering (b) after feature engineering - Regression

1. Linear Energy Properties (U_0, U, H, G):

In both scenarios, these properties exhibit nearly perfect alignment along the diagonal, which highlights the strength of linear models, such as Ridge regression, in capturing these energy-related targets. However, in Scenario B, the points show even tighter alignment, with virtually no scatter, confirming slightly improved generalization and numerical precision after feature engineering, although the gains are marginal due to the already high baseline performance.

2. Orbital Energies (HOMO, LUMO, gap):

A notable improvement is observed in Scenario B. In Scenario A, the parity plots show a broader spread, particularly for HOMO, where underprediction and overprediction are frequent. In contrast, after feature engineering, the predictions cluster much more closely around the diagonal, especially for LUMO and gap. This visual tightening highlights that spatial and electronic descriptors (e.g., TPSA, Gasteiger charges) significantly enhance the model's ability to capture orbital-related complexity.

3. Dipole Moment (μ) and Polarizability (α):

These two properties showed some of the clearest visual gains. In Scenario A, both μ and α plots exhibit scattered, diffused distributions, reflecting underfitting and limited descriptor expressiveness. Scenario B significantly improves alignment with the diagonal for both targets, particularly α, which exhibits a much more coherent linear trend. The reduction in outlier deviations in μ further confirms that the engineered features added valuable geometric and electronic structure information.

4. Electronic Excitation Energy (r^2):

This property initially exhibited poor predictive behavior, with significant deviation from the ideal line in Scenario A, particularly at higher excitation values. However, in Scenario B, the plot reveals a pronounced correction: the scatter tightens considerably, and the predicted values track the ground truth much more closely. Although variability still exists, this improvement aligns with the significant R^2 score boost observed numerically and demonstrates how advanced descriptors aid in modeling complex excitation phenomena.

5. Vibrational and Thermodynamic Properties (ZPVE, C_v):

ZPVE shows a moderate spread in Scenario A but becomes significantly more aligned in Scenario B, with points following the ideal line more consistently. C_v also benefits while its parity plot in Scenario A was widely scattered and noisy, Scenario B shows a more apparent diagonal trend with fewer extreme outliers. These results suggest that vibrational and thermodynamic behaviors are more effectively learned when descriptors capture structural flexibility and molecular topology.

7.5 SHAP[3] Analysis: Interpreting Feature Contributions to Model Predictions

In this section, we analyze SHAP values to interpret the influence of each feature on the prediction of molecular properties. SHAP values provide a unified framework for feature attribution, helping us understand which engineered descriptors contributed most significantly to the model's outputs. The following plots illustrate the impact and importance of each feature for individual target properties.

[3] SHapely Additive exPlanation

The SHAP plot for the band gap prediction reveals several chemically meaningful trends. The presence of the =O group stands out as the most influential feature, where higher values are associated with a decrease in the predicted band gap, which indicates that electronegative functional groups that withdraw electron density tend to reduce the energy difference between the HOMO and LUMO, thereby narrowing the gap.

The chg_std and atom count for nitrogen and carbon also play significant roles. Specifically, molecules with high electronic heterogeneity (large chg_std) tend to exhibit increased band gaps, possibly due to uneven charge distribution. TPSA and MR further highlight how spatial and polar characteristics of molecules influence electronic transitions. The SHAP values for inertia_sum and inertia_ratio suggest that overall molecular geometry and shape have subtle but non-negligible effects as well.

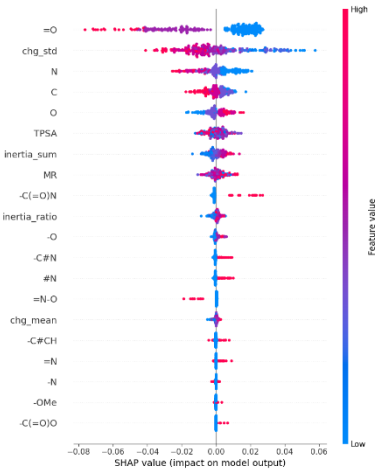

Figure 7- SHAP analysis plot for gap

The SHAP summary for the dipole moment prediction highlights the effect of electronic and structural asymmetry in molecules. The most impactful features include the presence of electron-withdrawing groups, such as =O, #N, and -C#N, which tend to induce molecular polarity.

Atomic counts of nitrogen and carbon also reflect how molecular composition contributes to charge separation. The moderate contributions from TPSA, MR, and chg_std suggest that both surface polarity and charge heterogeneity influence dipole behavior. Interestingly, features such as inertia_sum and inertia_ratio have a negligible impact, suggesting that mass distribution has a minor influence on predicting the dipole moment compared to electrostatic features.

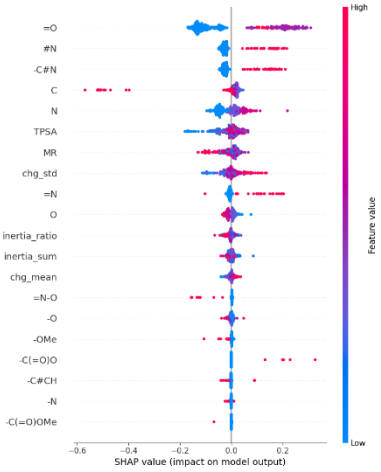

Figure 8- SHAP analysis plot for μ

In the SHAP summary for α, the most influential features are the MR and the carbon count (C), both of which contribute positively to the predicted polarizability, which aligns with chemical intuition: larger, more electron-rich molecules typically have greater polarizability due to enhanced electron cloud deformation.

Additionally, inertia_sum and inertia_ratio—which represent spatial mass distribution—also influence α, reflecting the importance of molecular size and shape in determining how easily electron clouds respond to external fields. Oxygen content (O) and chg_std follow, suggesting that both atomic composition and charge localization affect polarizability. Other features, such as -C#N, =O, or TPSA, have minimal yet visible effects.

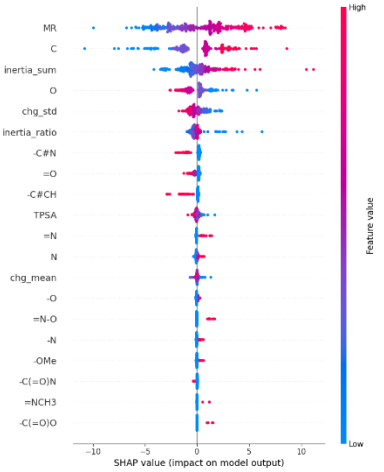

Figure 9- SHAP analysis plot for α

The SHAP summary for HOMO indicates that nitrogen-related features have a significant influence on the predicted HOMO level. Molecules rich in nitrogen have positive SHAP values, indicating that they tend to raise the HOMO energy (making it less negative), which aligns with their electron-donating character.

The chg_std also significantly affects HOMO, indicating that molecules with more uneven charge distributions influence electron localization. The contribution of functional groups like -C#N, =O, and -N further supports the idea that HOMO is sensitive to electronegative and conjugated groups. Descriptors such as MR, TPSA, and inertia_sum have less importance, implying that electronic features dominate over size or polarizability when predicting HOMO levels.

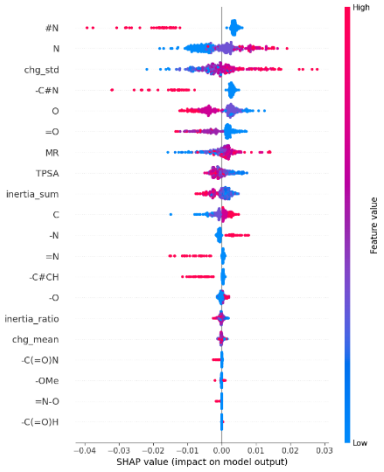

Figure 10- SHAP analysis plot for HOMO

For LUMO, the SHAP plot revealed that two features stood out the most: the presence of a =O and the chg_std.

Molecules with more =O groups tend to have lower LUMO energy values, which makes sense carbonyl groups are known for pulling electron density away, helping to stabilize the LUMO orbital.

Other key features that stood out include TPSA, the number of nitrogen and oxygen atoms, and MR. These descriptors give insight into both the electronic and spatial makeup of the molecules. Higher values of TPSA and chg_std were linked to lower LUMO energy levels. This suggests that molecules with more polarity and uneven charge distribution tend to stabilize their unoccupied orbitals, making them energetically lower and more stable.

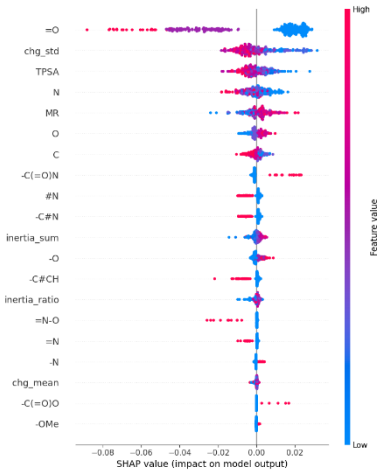

Figure 11- SHAP analysis plot for LUMO

For predicting r^2 the SHAP plot indicates that inertia_sum is the most influential feature by a large margin, which makes physical sense, as inertia_sum is a 3D descriptor capturing how mass is distributed in space directly related to molecular size and shape, which in turn affects the extent of the electron cloud.

Other relevant features include MR, chg_std, and inertia_ratio, which are conceptually linked to how electron density is spatially distributed.

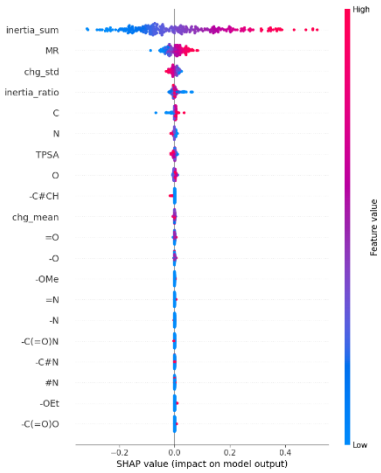

Figure 12- SHAP analysis plot for r^2

The SHAP plot for zpve shows that the most influential features are MR, chg_std, O, C, and =O. Hence, ZPVE is closely tied to vibrational motions, which are highly dependent on atomic composition and bond types.

High values of MR and chg_std positively influence ZPVE, suggesting that molecules with more polarizable atoms and greater charge variation tend to have higher vibrational energies. The presence of oxygen (O, =O) also contributes positively, likely due to the high-frequency vibrational modes of light atoms in polar bonds.

With the minor impact of 3D descriptors, such as inertia_sum and inertia_ratio, it can be understood that ZPVE is more chemically driven than spatially driven.

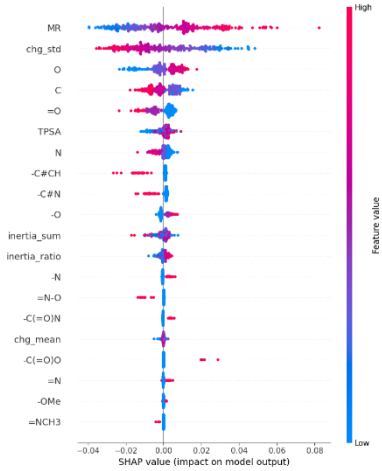

Figure 13- SHAP analysis plot for gap for ZPVE

The SHAP plot for C_v shows that MR, chg_std, and inertia_sum are the dominant features influencing the model's predictions. High values of MR and inertia_sum strongly increase Cv, indicating that molecules with larger, more complex structures and diverse electronic environments tend to store more internal energy. Interestingly, electronic descriptors like chg_std also contribute significantly, reflecting how uneven electron distribution leads to more vibrational modes and higher heat capacities.

Atomic features, such as O, C, and N, and specific functional groups, like =O or -OMe, have a moderate but meaningful influence. Their colored dispersion patterns also confirm that both their presence and numerical count are important.

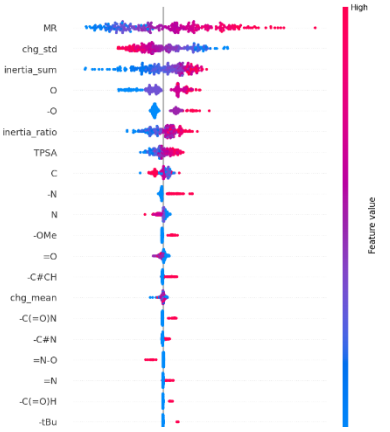

Figure 14- SHAP analysis plot for C_v

8. Neural Network Modeling

In this section, we develop neural network models to predict molecular quantum properties using the same modeling workflow adopted in the classical ML phase, including both scenarios: with and without feature engineering. Neural networks are well-suited for capturing complex, non-linear patterns in data, making them a promising tool for modeling the intricate relationships between molecular structure and quantum properties. In this phase of the study, we explore whether deep learning models can outperform traditional machine learning approaches in terms of prediction accuracy — even before adding domain-specific chemical descriptors. This allows us to assess the inherent strength of neural networks in learning from basic molecular representations alone.

8.1 Neural Network Methodology

To model molecular quantum properties, we employed feedforward neural networks implemented in TensorFlow and Keras. Each network was built for multi-target regression, predicting all 12 quantum properties simultaneously from the input features. We used the same model architecture and training process for both scenarios to ensure a fair comparison, which allows us to isolate the impact of the added features on model performance. The components below make up the base architecture of model:

- One or two fully connected hidden layers with ReLU activation, chosen for its simplicity and effectiveness in capturing non-linear relationships.
- Optional batch normalization to stabilize and speed up convergence by normalizing intermediate outputs.
- Dropout layers to prevent overfitting by randomly deactivating a fraction of neurons during training.
- An output layer with 12 units (one per target property), using linear activation to support unbounded regression output.

We chose the Adam optimizer because of its ability to adapt the learning rate during training and handle sparse gradients effectively — a common challenge when working with high-dimensional molecular data. Adam brings together the strengths of both RMSProp and momentum-based methods, and it typically performs well with minimal need for manual tuning, making it a reliable choice for this task.

The loss function used was mean squared error (MSE), a standard choice for regression problems, which penalizes larger errors more heavily, making it suitable for continuous-valued targets.

To determine the best architecture and training configuration, we applied a two-stage hybrid hyperparameter tuning strategy:

1. **Random Search**: Random search is computationally efficient in high-dimensional spaces and provides good global coverage. It allows quick identification of promising regions in the hyperparameter space without exhaustively evaluating all combinations. This stage explored a broad range of hyperparameters including:

- Number of units in the first and optional second hidden layers
- Dropout rates after each layer
- Presence of a second hidden layer
- Learning rate
- Optimizer type (although Adam consistently yielded superior performance)

2. **Bayesian Optimization**: The top configurations from the random search phase were further refined using a probabilistic model that learns to propose increasingly better hyperparameter sets. This reduced computational cost while converging on high-performing models. Bayesian optimization builds a surrogate model to guide the search toward better-performing configurations based on previous evaluations. It is more sample-efficient than grid or random search in the fine-tuning phase, especially when the objective function is expensive to evaluate.

This hybrid approach blends the wide-ranging search capabilities of random search with the efficiency of Bayesian optimization. By balancing broad exploration with focused fine-tuning, it reduces the risk of settling on suboptimal solutions too early and increases the chances of finding a strong configuration — all while keeping the process computationally manageable.

Early stopping was employed during training to prevent overfitting by monitoring the validation loss and halting training if no improvement was seen after several epochs. On average, convergence occurred between 100 and 200 epochs.

Model performance was evaluated using RMSE, MAE, and R^2 across each target and averaged overall. This comprehensive evaluation ensured fair comparisons across different scenarios and modeling approaches.

8.2 Comparison of Hyperparameter Tuning Outcomes

In case without feature engineering, where we did not apply any feature engineering, the model performed best with a simple architecture — just one hidden layer with 288 units and a low dropout rate of 0.1, which helps prevent overfitting. However, with feature engineering, after introducing engineered features, the model responded well to a more complex setup, which ultimately utilized two hidden layers with 512 and 256 units and slightly higher dropout rates of 0.3 and 0.2. Although the architecture differed, both models used the Adam optimizer and ReLU activation functions. The final hyperparameter settings for each scenario are listed in the table below.

Table 7 - Top hyperparameters for neural network model before and after feature engineering

Parameter	Before Feature Engineering	After Feature Engineering
units_1	288	512
droput_1	0.1	0.3
second_layer	False	True
units_2	-	256
droput_2	-	0.2
learning_rate	0.0010	0.00082
optimizer	Adam	Adam

The difference in architectural complexity can be attributed to the input representation: engineered features provided a richer, more structured input, allowing the model to leverage deeper layers without overfitting. Conversely, in the raw feature setting, simpler architectures performed better in generalization.

8.3 Training History Plot Comparison

Both models trained smoothly, and we consistently observed the validation loss remaining lower than the training loss — a positive sign that things were progressing well. In case without feature engineering, the gap between training and validation loss was wider, which might be because the original features contained more noise, making it harder for the model to learn clean patterns.

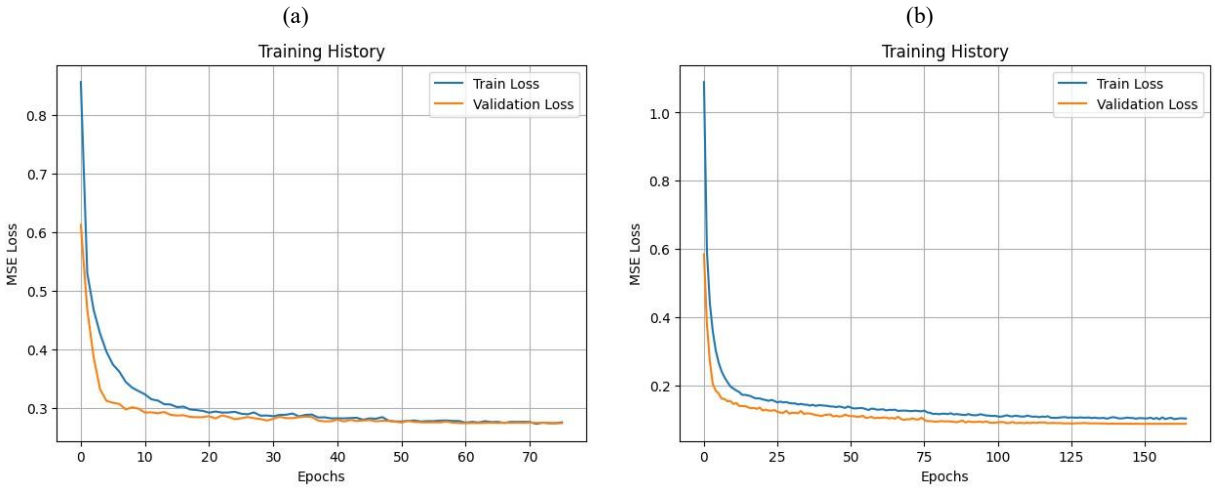

Figure 15- Training history versus epoch plot (a) before feature engineering (b) after feature engineering

The case with feature engineering told a different story. After adding engineered features, the model exhibited more stable learning with smoother and tighter loss curves, indicating that the new features helped reduce noise and enabled the model to focus on important signals, leading to better generalization.

While it may seem counterintuitive for validation loss to be lower than training loss, it is a common and desirable outcome in deep learning, mainly when techniques such as dropout and batch normalization are employed. It usually means the model is not just memorizing the training data — it is learning patterns that apply well to new, unseen examples. In our case, that lower validation loss is a strong sign that the model is well-tuned, balanced, and ready to perform reliably beyond the training set.

Plots for both scenarios confirm the benefit of early stopping and dropout. The deeper architecture after using feature engineering did not overfit, likely due to the use of batch normalization and dropout.

8.4 Modeling Performance Evaluation

The model's performance in both scenarios for each target is shown in the table below:

Table 8 - Modeling Performance Before and After Feature Engineering using neural networks

Property	Before Feature Engineering			After Feature Engineering		
	RMSE	MAE	R^2	RMSE	MAE	R^2
gap	0.0304	0.0235	0.573	0.0184	0.0130	0.843
μ	1.0602	0.7266	0.527	1.0001	0.6701	0.579
α	2.6981	1.9854	0.842	1.3046	0.8113	0.963
HOMO	0.0146	0.0109	0.615	0.0102	0.0071	0.813
LUMO	0.0258	0.0198	0.675	0.0162	0.0113	0.873
r^2	221.5217	153.528	0.176	74.8625	45.1318	0.906
zpve	0.0176	0.0140	0.686	0.0055	0.0037	0.970
U_0	1.7117	1.0482	0.997	1.4737	0.9720	0.998
U	1.7113	1.0479	0.997	1.4737	0.9720	0.998
H	1.7114	1.0478	0.997	1.4737	0.9720	0.998
G	1.7122	1.0489	0.997	1.4737	0.9720	0.998
C_v	2.4368	1.9512	0.495	0.9055	0.6955	0.930

Across all properties, the model that utilized feature engineering outperformed the one that did not. The most significant improvements in R^2 were seen for r^2 (from 0.176 to 0.906), Cv (from 0.495 to 0.930), gap (from 0.573 to 0.843), and α (from 0.842 to 0.963), indicating that these properties are susceptible to structural information, which demonstrates that the engineered descriptors enriched the feature space and allowed the model to learn complex molecular-property relationships better.

Upon reviewing the overall results, the benefits of feature engineering were particularly notable. The model trained on raw features alone reached an average R^2 of 0.752 and a mean absolute error (MAE) of 0.931 — not bad, but clearly, with room for improvement. Once we added feature engineering to the mix, performance improved significantly: the average R^2 rose to 0.906, and the MAE dropped to 0.782. In simple terms, the model became much better at making accurate predictions.

A big reason behind this strong performance was the hybrid tuning strategy we used. It gave the model the flexibility to become more complex — adding deeper layers — but only when the data had enough depth to justify it. This approach helped avoid overfitting while still giving the model room to adapt and improve when the input features were rich and informative. It struck a thoughtful balance between keeping things simple and letting the model scale up when it truly needed to.

Finally, these results indicate that building a robust model is not just about stacking layers or choosing the latest algorithm. What makes a difference is giving the model the right kind of information and taking the time to fine-tune it properly. When the model's complexity aligns with the quality of the input data, everything works more effectively. Thoughtful feature engineering

and intelligent optimization made all the difference here, demonstrating the crucial role these steps play in deep learning for molecular property prediction.

8.5 Parity Plot Analysis: Comparison of Scenarios Before and After Feature Engineering

To better understand how well the models performed, the Figure below shows parity plots for each molecular property in both cases: (a) without feature engineering and (b) with feature engineering. These plots compare the model's predicted values to the actual values. Ideally, all points should fall along the red diagonal line, meaning perfect predictions.

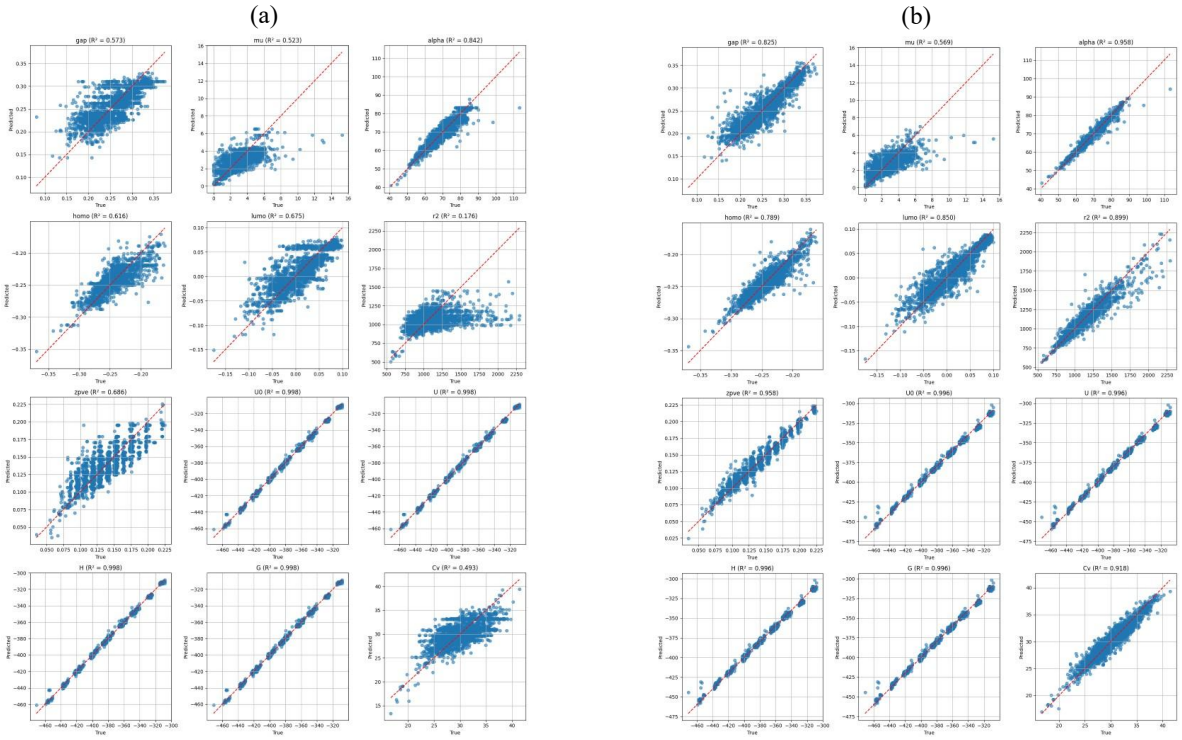

Figure 16 - Parity plots for each target (a) before feature engineering (b) after feature engineering - Neural Networks

In case without feature engineering, deviations from the diagonal line were substantial for several properties, including r^2, C_v, gap, and α, indicating underfitting and reduced accuracy. Using feature engineering showed significantly tighter clustering around the diagonal for these targets, reflecting higher prediction accuracy and reduced variance.

The enhanced performance in the case we employed feature engineering demonstrates how engineered features help the model capture complex property relationships more effectively. These visual results are consistent with the quantitative improvements reported earlier in RMSE, MAE, and R^2. Together, they highlight the value of domain-informed descriptors in neural network modeling of molecular properties.

9. Comparative Analysis of Classical ML Models and Neural Networks

In this section, we compare the results obtained from classical machine learning (ML) models and neural networks (NN) for predicting twelve quantum molecular properties. The objective is to evaluate the relative strengths and limitations of each approach and to analyze their predictive behavior, particularly for dipole moment (μ).

9.1 Summary of Modeling Approaches

Two main modeling strategies were implemented: classical ML models, including Ridge Regression, Random Forest, Histogram-Based Gradient Boosting (HGBR), and a feedforward neural network. Each modeling track was applied in two scenarios: (1) using raw input features and (2) using an enhanced set of features derived through chemical domain-informed feature engineering. This structure enabled a comprehensive performance comparison under varying input complexities.

9.2 Comparative Performance Evaluation

Both classical ML and neural network models demonstrated considerable performance improvement after the inclusion of engineered features. In the best-case scenario with feature engineering, the average R^2 score for the neural network was 0.906, with an average MAE of 0.782, which is nearly identical to the best performance of classical ML models. This outcome indicates that when equipped with informative features, simple interpretable models can rival the predictive power of more complex architectures.

Table 9 - Average R^2 and MAE for classical ML and neural networks before and after feature engineering

Model Type	Before Feature Engineering		After Feature Engineering	
	Average R^2	Average MAE	Average R^2	Average MAE
Classical ML	~ 0.752	~ 0.931	~ 0.906	~ 0.782
Neural Network	0.752	0.931	0.906	0.782

9.3 Property-Level Analysis

Across all targets, the neural network model showed comparable or slightly lower performance than the best classical ML models. For instance, properties such as HOMO, ZPVE, and the gap showed higher R^2 values under classical models, while Cv showed a marginal improvement with the neural network. For energy-related properties (U_0, U, H, G), both model types reached R^2 scores close to unity.

Table 10- Comparison of classical ML and neural networks performance for each target using feature engineering

Property	Best Classical Model	R^2 (ML)	R^2 (NN)	Best Scenario
gap	HGBR	0.854	0.843	ML Slightly Better
μ	Ridge	0.576	0.579	Equal
α	Ridge	0.963	0.963	Equal
r^2	HGBR	0.907	0.906	Equal
C_v	HGBR	0.926	0.930	NN Slightly Better
ZPVE	HGBR	0.979	0.970	ML Slightly Better
HOMO	HGBR	0.845	0.813	ML Better
LUMO	HGBR	0.873	0.873	Equal
U_0, U, H, G	Ridge	0.998+	0.998	Equal

9.4 Challenges in Predicting Dipole Moment (μ)

Despite improvements from feature engineering, the dipole moment (μ) remained the most challenging property to predict. The best R^2 achieved for μ was 0.576 using Ridge Regression with feature engineering. This lower performance, compared to other targets, stems from the fact that μ is a vectorial quantity susceptible to three-dimensional geometry and electronic distribution. These spatial features are not directly captured by the two-dimensional descriptors used in either modeling approach. Consequently, even advanced learning algorithms, including neural networks, could not substantially surpass classical models on this target.

9.5 Comparison with the Substructure Embedding-Based Framework of Jung et al. (2024)

A comparable effort to predict molecular properties using machine learning was recently reported by Jung et al. (2024), who introduced a pipeline that integrates Mol2Vec substructure embeddings with a Gradient Boosted Feature Selection (GBFS) workflow. Their method focuses on enhancing model interpretability while preserving predictive performance, utilizing substructure-level representations derived from SMILES notations to capture chemical information.

While our study used functional group-based descriptors and physicochemical features with classical ML models and a neural network, Jung et al. leveraged pre-trained molecular embeddings as inputs to tree-based models enhanced by GBFS. Their embedding approach is unsupervised and derived from atom-centered fragments, allowing the model to incorporate context-specific substructure similarities. In contrast, our feature engineering pipeline is based on explicit chemical knowledge, enabling the inclusion of descriptors with direct physical significance.

In terms of modeling objectives, both studies targeted quantum mechanical properties, including dipole moment (μ), which remains one of the most challenging targets to predict accurately. Jung et al. report that their models, despite using Mol2Vec and GBFS, struggled to produce highly accurate predictions for μ due to the inherent limitations of 2D representations in capturing conformational and electronic asymmetries. Their reported mean absolute errors (MAEs) indicate moderate performance, with no substantial gain over baseline ensemble models.

Our best-performing model for dipole moment was Ridge Regression with feature engineering, achieving an R^2 of 0.576. This result aligns with Jung et al.'s findings and highlights a common

limitation: μ is fundamentally sensitive to 3D molecular geometry, which neither substructure embeddings nor handcrafted 2D features can fully capture. Even with extensive feature selection and model optimization, both approaches plateau in predictive accuracy for this property.

The primary advantage of our approach lies in its interpretability. Feature importance analyses using SHAP values allow us to trace model decisions back to recognizable chemical descriptors, which is beneficial for understanding molecular behavior. Our two-track modeling approach — combining both classical machine learning models and neural networks — gave us a more well-rounded view of how well these models can generalize. That said, relying on handcrafted descriptors does come with trade-offs. While they offer clarity and are rooted in chemical knowledge, they can limit scalability and sometimes miss the finer structural details that vector-based embeddings might catch.

In contrast, the method developed by Jung et al. leverages automated feature generation using Mol2Vec, enabling a more scalable and flexible pipeline. Their use of GBFS for feature selection helps reduce complexity while keeping the most important signals. However, their approach still relies on the quality of SMILES-based embeddings — which, while powerful, do not fully capture the entire 3D structure of molecules.

When we step back and look at both approaches, a clear message emerges: predicting dipole moment (μ) accurately is incredibly challenging without incorporating 3D geometric information. Our method offers interpretability and strong alignment with chemical principles, whereas Jung et al.'s model advances in terms of automation and substructure learning. However, going forward, the most significant improvements in μ prediction will likely come from hybrid approaches — ones that combine spatial, electronic, and structural insights into a single, integrated framework.

10. Conclusion and Suggestions

In this study, we developed and evaluated two sets of predictive models to estimate twelve quantum chemical properties of molecules, using structural descriptors based on functional groups and elemental composition. The first set involved classical machine learning models such as Ridge Regression, Random Forest, and Histogram-Based Gradient Boosting. The second set employed neural networks to account for complex nonlinear associations. Our goal was to evaluate the performance of these models and assess the impact of feature engineering on predictive accuracy.

The results showed that enriching the dataset with additional descriptors—such as the number of atoms, functional groups, topological polar surface area (TPSA), molecular refractivity (MR), and Gasteiger charges—significantly improved the model outcomes. These engineered features contributed to a noticeable improvement in performance metrics, particularly for properties such as r^2, C_v, gap, and α.

When comparing classical ML models and neural networks, we observed that both approaches performed well when supplied with engineered features. On average, the neural network achieved an R^2 of 0.906 and an MAE of 0.782, which is comparable to the performance of the best classical models, which confirms that, in the presence of informative features, even simpler models can achieve competitive results with lower computational costs.

However, some properties, particularly the dipole moment (μ), remained difficult to predict accurately, which is likely because μ depends heavily on three-dimensional molecular geometry and electronic asymmetry, which are not captured effectively by the two-dimensional descriptors used in this work.

The classical models also offered higher interpretability, allowing us to identify the most influential features through SHAP analysis. This aspect is particularly valuable in applications where understanding the relationship between molecular structure and properties is essential.

Suggestions for Future Work

- Incorporating three-dimensional geometric descriptors (e.g., bond lengths, angles, conformers) could enhance predictions, particularly for properties such as μ that depend on molecular shape.
- Exploring graph neural networks (GNNs), which learn from molecular graph structures, may improve the modeling of atomic connectivity and electronic effects.
- Using transfer learning approaches with pretraining on large chemical datasets might help improve generalizability and reduce the need for labeled data.
- Implementing uncertainty quantification would increase confidence in predictions, especially for real-world screening tasks.
- Combining domain-based descriptors with embedding-based representations could help leverage both chemical knowledge and latent patterns.

In conclusion, our study confirms the effectiveness of data-driven modeling approaches in predicting molecular properties and underlines the importance of feature engineering. With further refinement, such models can serve as efficient alternatives to traditional quantum chemical methods for large-scale applications.

Bibliography

1. Benson, S. W., & Buss, J. H. (1958). *Additivity Rules for the Estimation of Molecular Properties. Thermodynamic Properties*. *29*(3), 546–572. https://doi.org/10.1063/1.1744539
2. Jung, S. G., Jung, G., & Cole, J. M. (2025). Automatic Prediction of Molecular Properties Using Substructure Vector Embeddings within a Feature Selection Workflow. *Journal of Chemical Information and Modeling*, *65*(1), 133–152. https://doi.org/10.1021/acs.jcim.4c01862
3. Marimuthu, A. N., & McGuire, B. A. (2025). *A Machine Learning Pipeline for Molecular Property Prediction using ChemXploreML*. http://arxiv.org/abs/2505.08688
4. Ramakrishnan, R., Dral, P. O., Rupp, M., & Von Lilienfeld, O. A. (2014). Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, *1*. https://doi.org/10.1038/sdata.2014.22
5. Zeng, X., Xiang, H., Yu, L., Wang, J., Li, K., Nussinov, R., & Cheng, F. (2022). Accurate prediction of molecular properties and drug targets using a self-supervised image representation learning framework. *Nature Machine Intelligence*, *4*(11), 1004–1016. https://doi.org/10.1038/s42256-022-00557-6