Predicting Enthalpy of Vaporization (del Hvap) from Molecular Descriptors using Machine Learning

CHE657 – COURSE PROJECT

Mentor – Prof. Salman Khan

Group 3: –

- Ayush Omer (230264)
- Harshvardhan Gaur (230467)
- Khushi Jain (230560)
- Om Jee Singh (230718)
- Prince Yadav (230792)

Objective

To develop and evaluate machine learning models that can accurately predict enthalpy of vaporization using only a molecule's structure.

Why is it needed?

- Experimental measurement is accurate but slow and resource-intensive.
- We need a fast, reliable, and data-driven way to estimate this property.

Dataset and Methodology

Dataset:

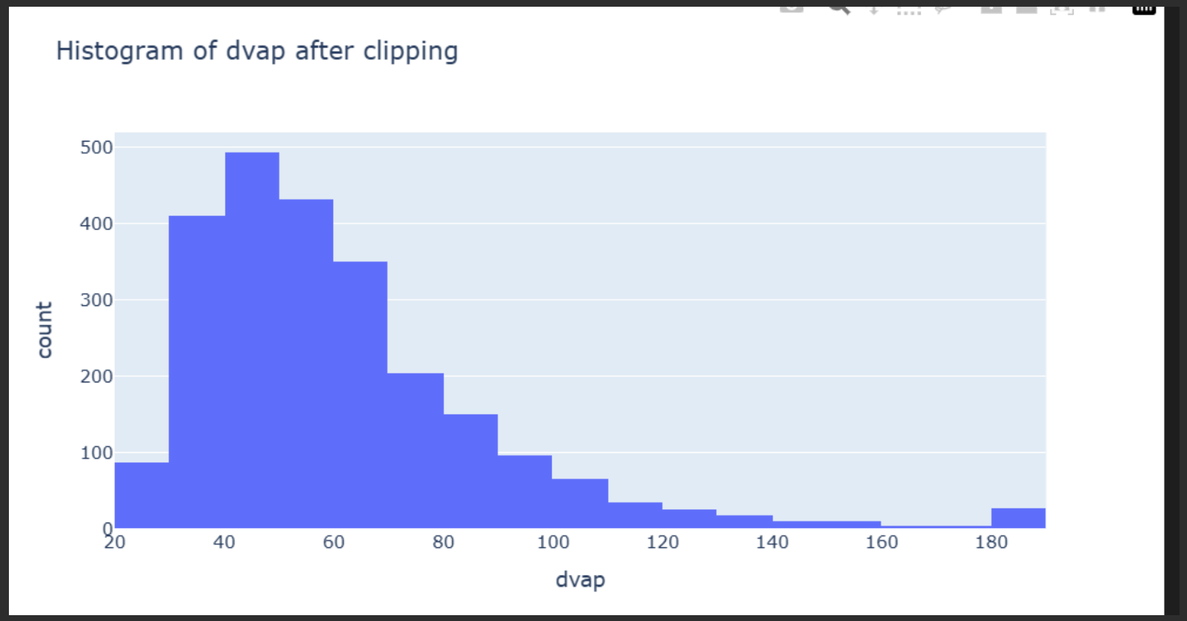
- Volatile Organic Compounds (VOCs) with known enthalpies of vaporization.
- Structure of organic compounds as a string of characters - SMILES

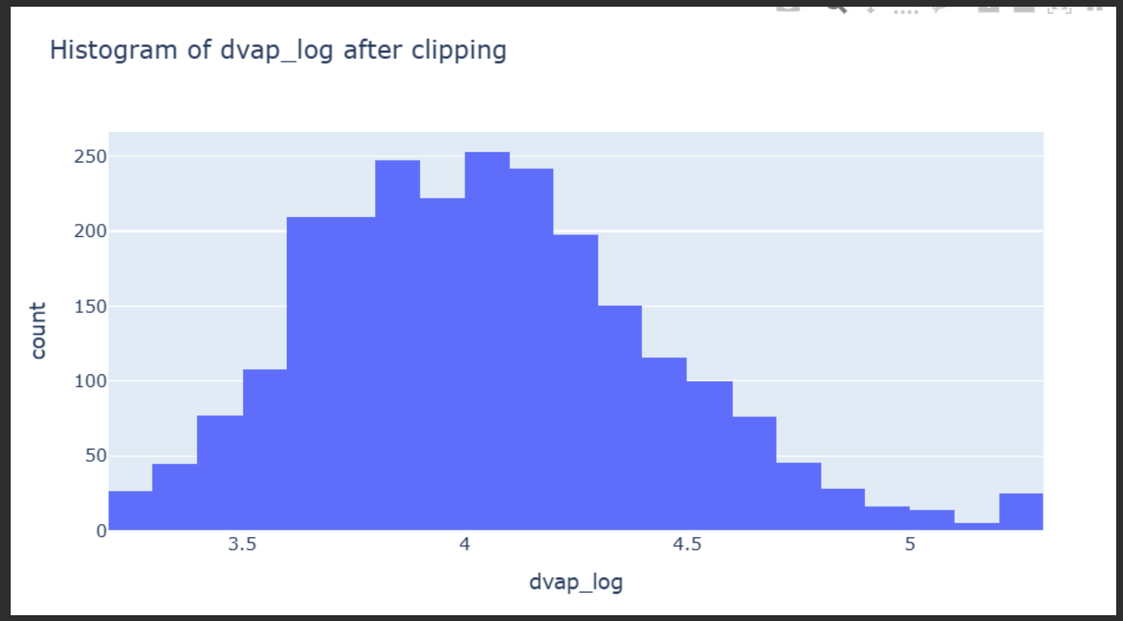
Feature Generation:

- Used the **RDKit** cheminformatics library to convert molecular SMILES strings into numerical features.
- Generated 217 molecular descriptors (e.g., molecular weight, topological, electronic properties) resulting in a final modeling dataset with dimensions of 2,410 compound * (217 features+original data).

DATA PREPROCESSING

- Clipped outliers from the target variable by removing extreme quantiles to remove any NaN values or noise from the data.
- We used a **log-transformed** target variable, as the log transform significantly reduced the skewness of the distribution.





DATA PREPROCESSING

```
final_data["dvap"] = final_data["dvap"].clip(lower=final_data["dvap"].quantile(0.01),
                                             upper=final_data["dvap"].quantile(0.99))

final_data["dvap_log"] = np.log1p(final_data["dvap"])
#plot histogram using plotly
```

```
skewness_log = final_data["dvap_log"].skew()
skewnes_normal = final_data["dvap"].skew()
print(f"Skewness of dvap: {skewnes_normal:.4f}")
print(f"Skewness of dvap_log: {skewness_log:.4f}")
```

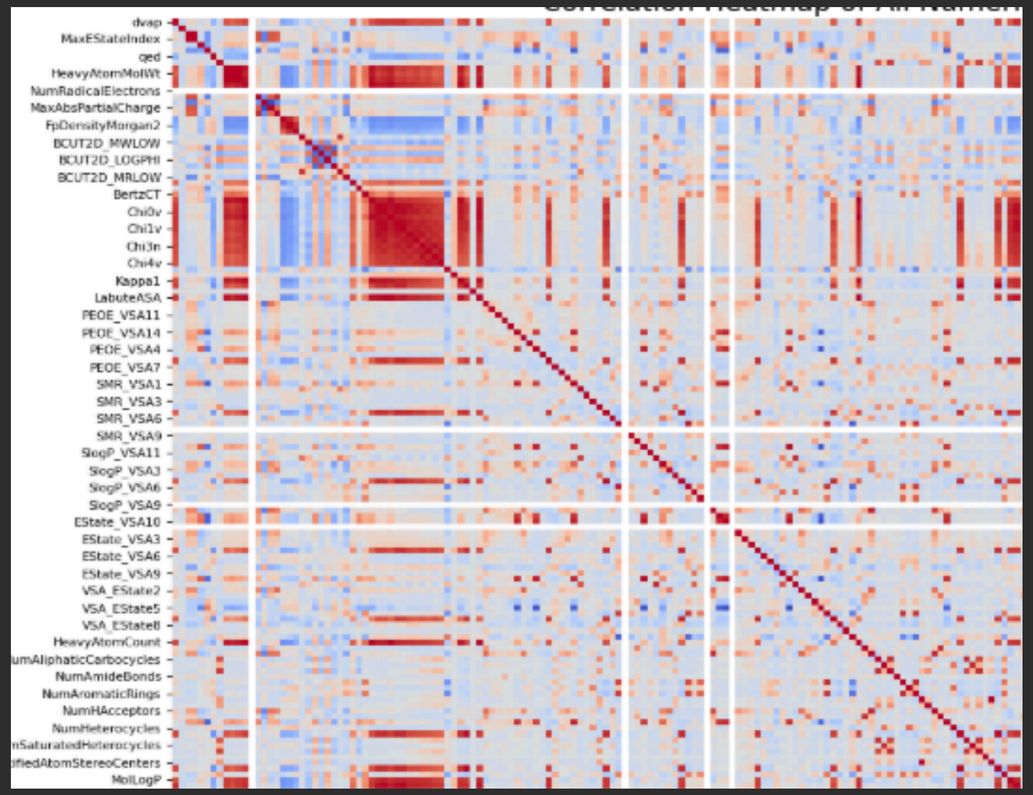
✓ 0.0s

```
Skewness of dvap: 1.7788
Skewness of dvap_log: 0.4871
```

DATA PREPROCESSING

- **Features analysis:**

  On analysing the correlation among the features, we found many features are heavily correlated, which in turn could make our models worse especially in the case of linear regressors.



We used Principal Component Analysis to remove highly correlated features, and only kept those which explain the maximum variance of the dataset (that is, the highest information)

DATA PREPROCESSING

- **Princeple Component Analysis:**

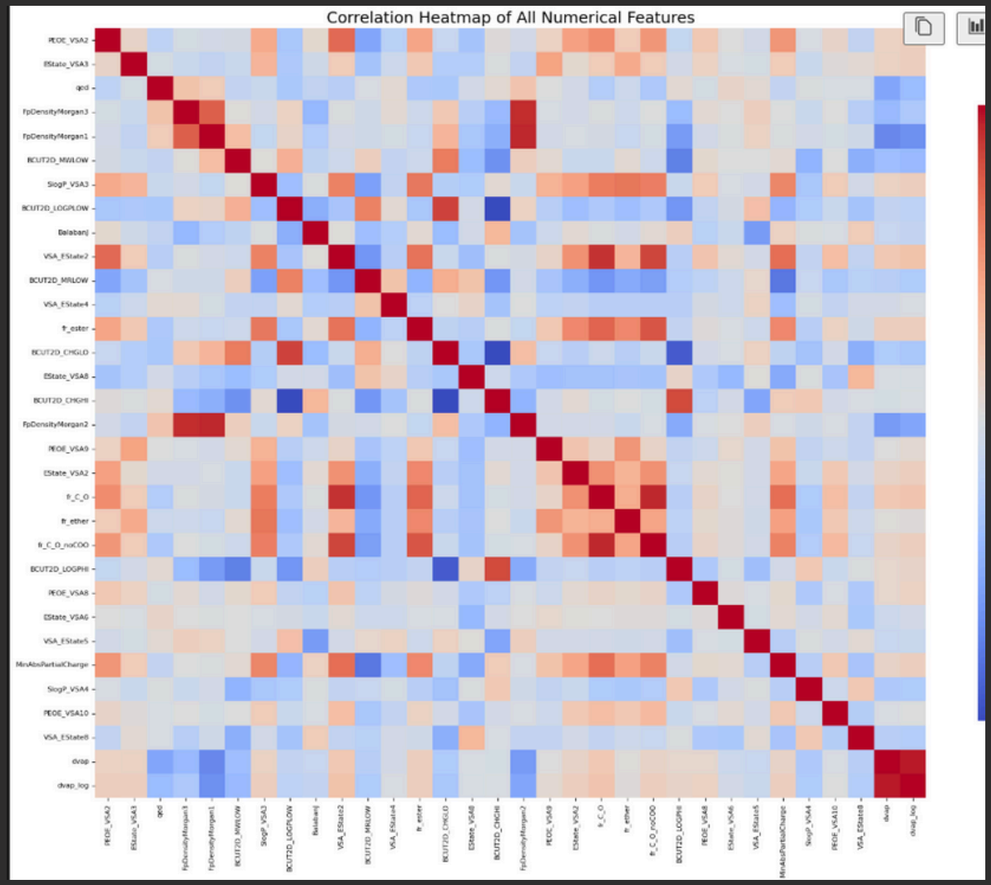
  - Since many features in the dataset were highly correlated, we used Principal Component Analysis (PCA) to evaluate the contribution of each original feature to the overall variance.

  - Instead of using the PCA components directly, we analysed the PCA loadings and explained variance to identify the top 20 most informative original features.

  - This helped to reduce redundancy and multicollinearity, retaining only features that captured the most significant variance in the data while simplifying the model input space.

- **Princeple Component Analysis:**

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# --- Load your dataset ---
# Example: df = pd.read_csv('your_data.csv')
# Assume last column is the target variable
X = data.drop(columns=['dvap'])      # 216 feature columns
y = data['dvap']        # target variable (integer over a range)

# --- Standardize features (important for PCA) ---
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# --- Fit PCA ---
pca = PCA()
pca.fit(X_scaled)

# --- Compute feature importance based on loadings ---
# Each feature's importance = sum of squared loadings across components (weighted by explained variance)
loadings = np.abs(pca.components_) * pca.explained_variance_ratio_[:, np.newaxis]
feature_importance = loadings.sum(axis=0)

# --- Select top 20 original features ---
top20_indices = np.argsort(feature_importance)[::-1][:20]
top20_features = X.columns[top20_indices].tolist()

print("Top 20 most important original features (by PCA):")
print(top20_features)
```



Which Model Works Best?

Baseline Model:

- Linear Regression (plus Ridge & Lasso)

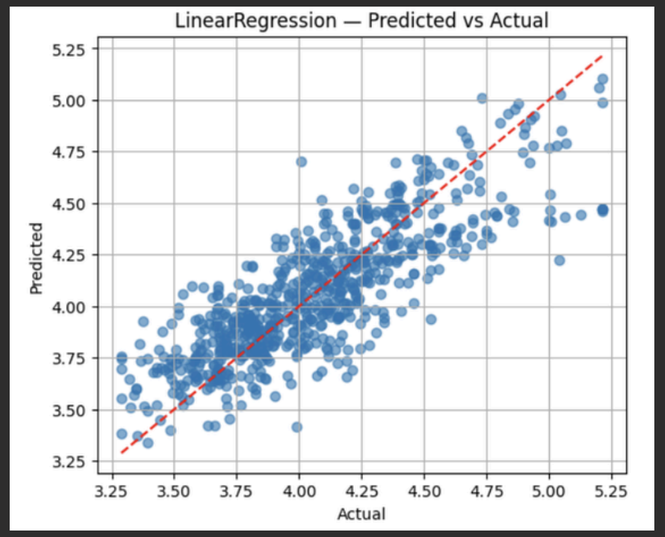
Traditional ML:

- Support Vector Regression (SVR)
- Random Forest

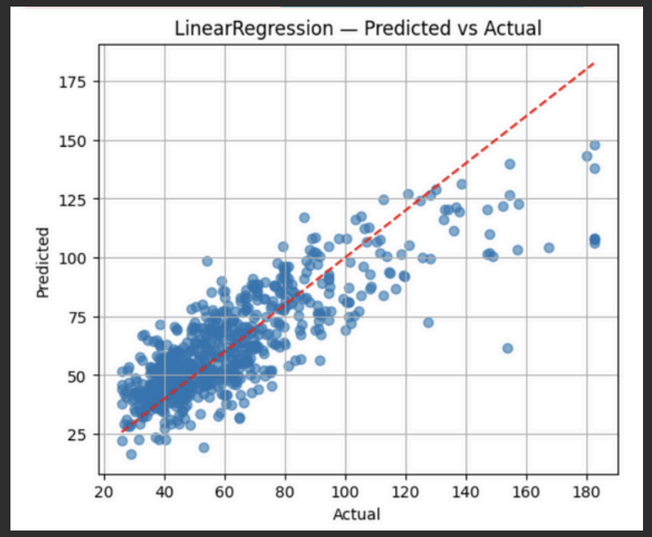
Advanced Ensemble Models (Gradient Boosting):

- Standard Gradient Boosting
- XGBoost
- LightGBM
- CatBoost

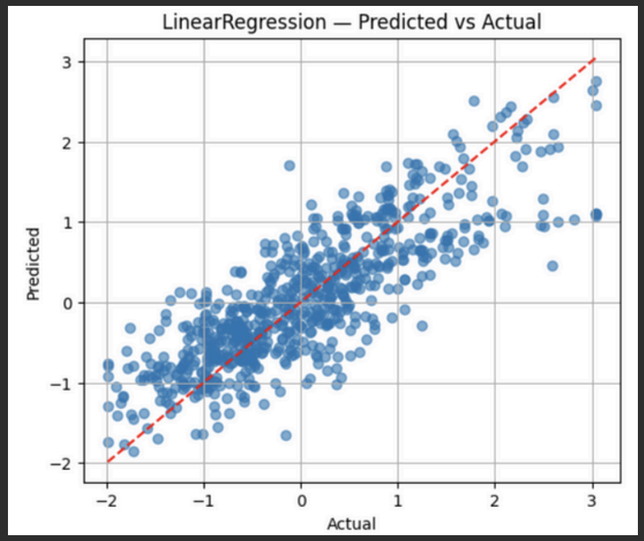
Linear Regression



Log Dvap

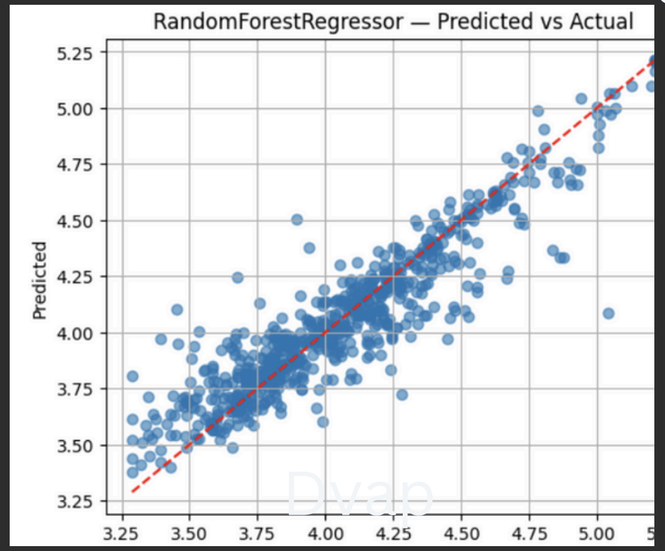


Raw data

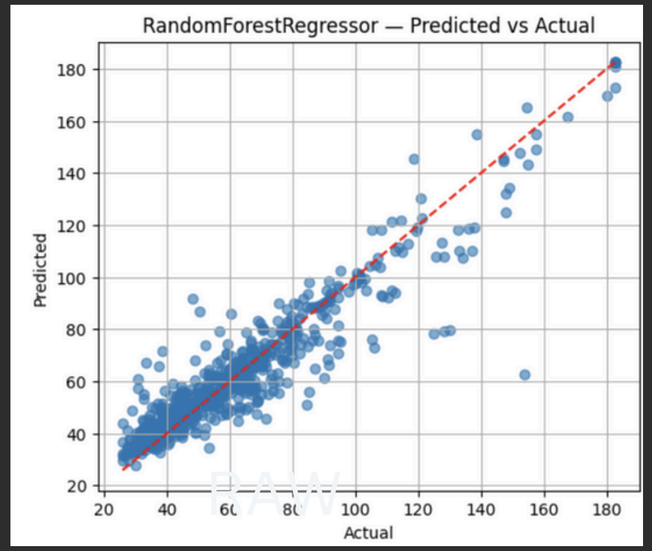


Dvap_Log_Scaled

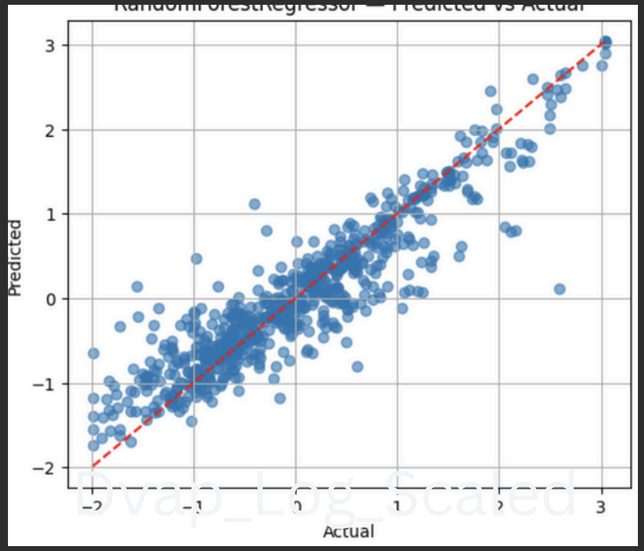
Random Forest Regression



Dvap

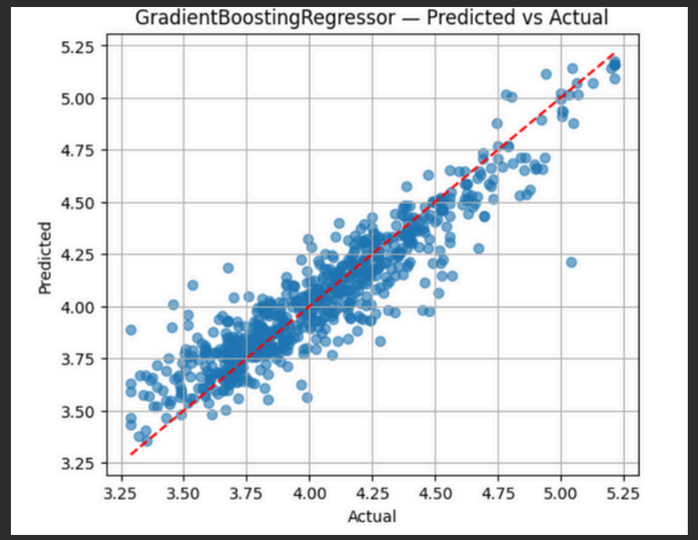


RAW

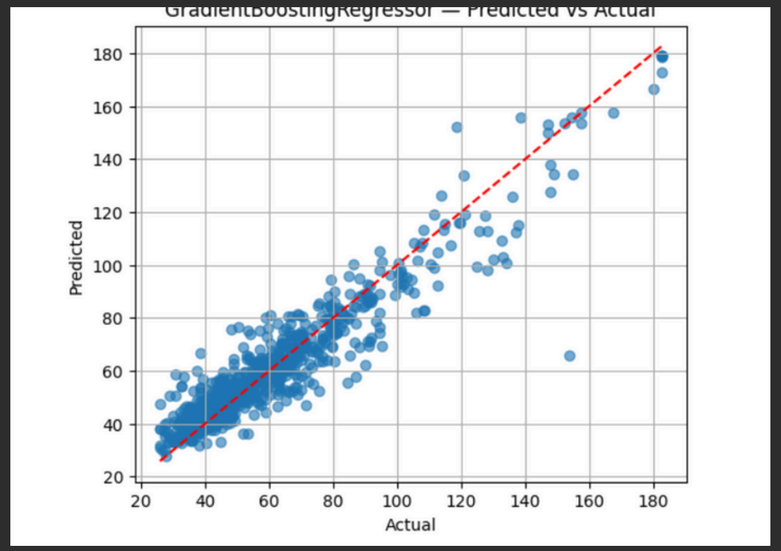


Dvap_Log_Scaled

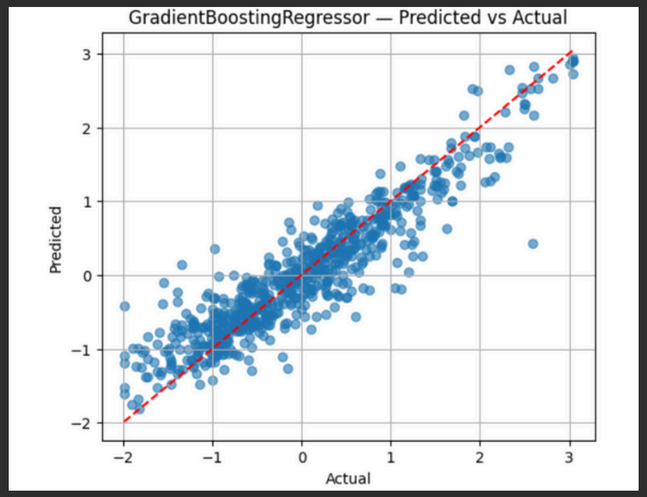
Gradient Boosting Regression



Dvap

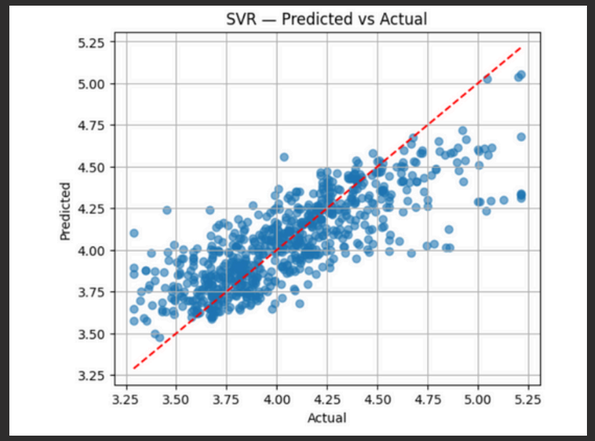


RAW

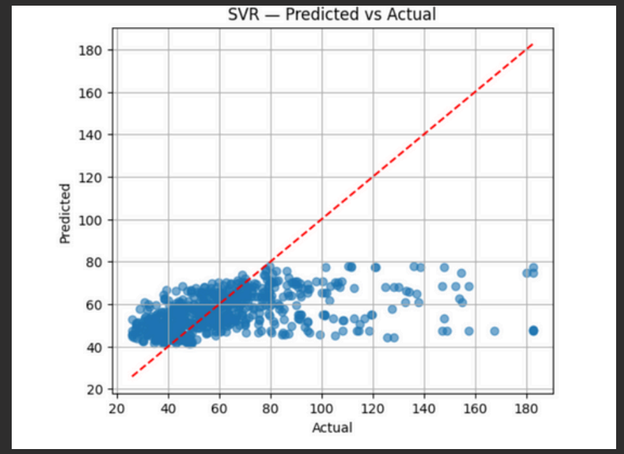


Dvap_Log_Scaled

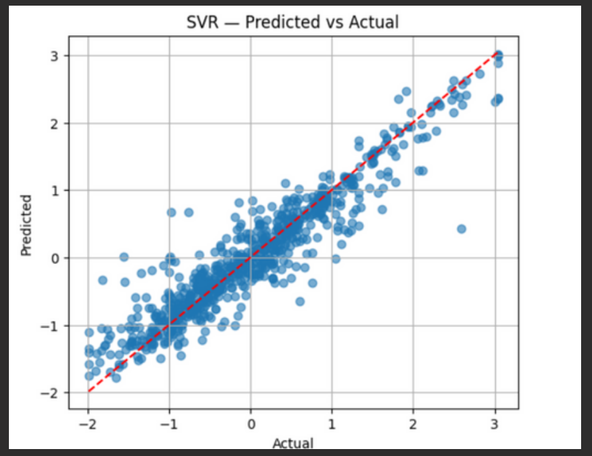
SVR Regression



Dvap

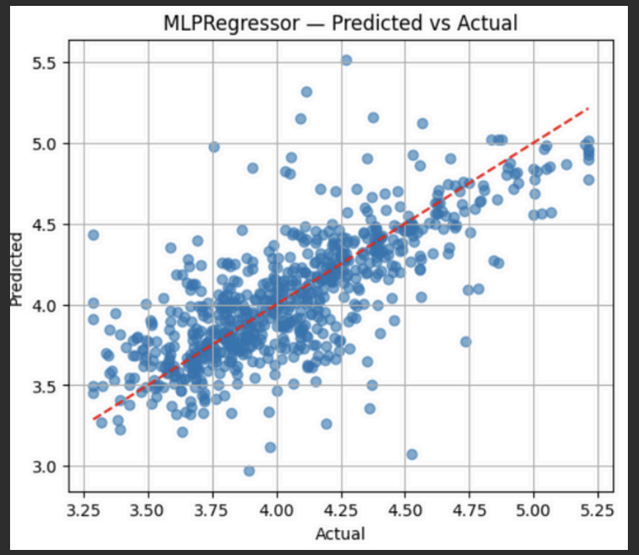


RAW

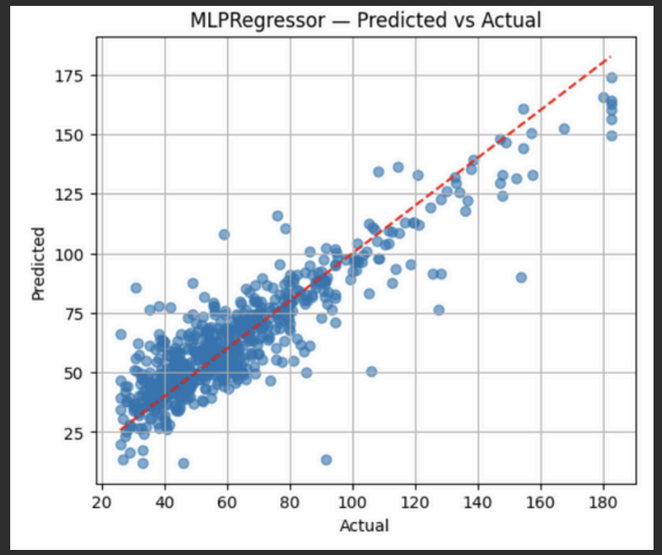


Dvap_Log_Scaled

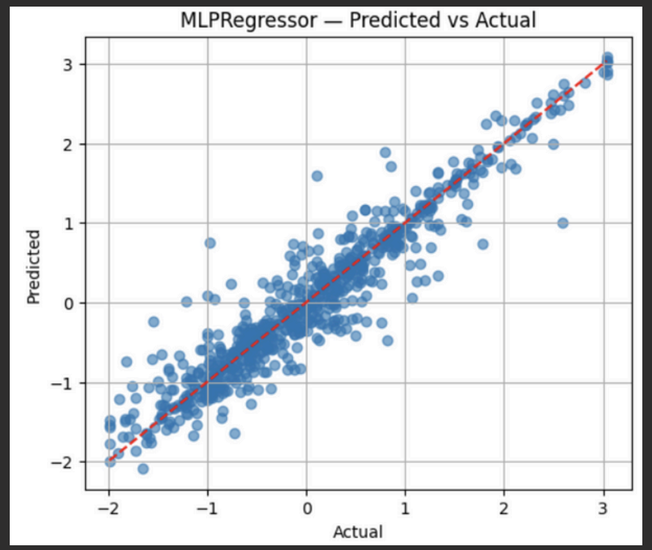
MLP Regression



Dvap

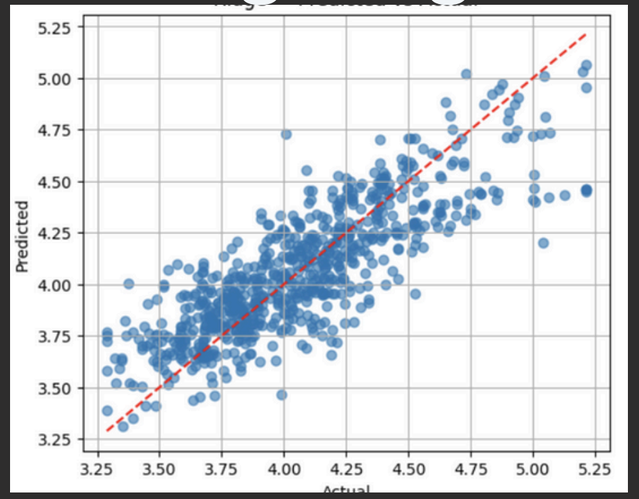


RAW

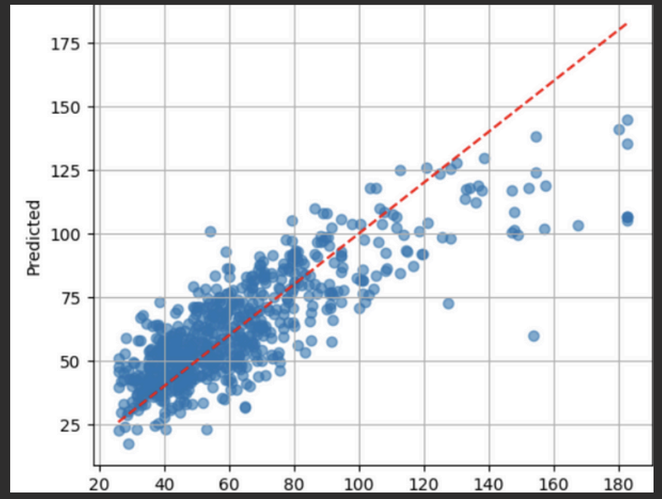


Dvap_Log_Scaled

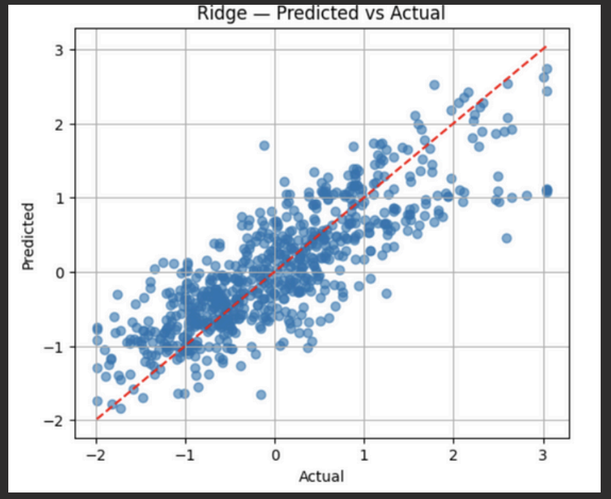
Ridge Regression



Dvap

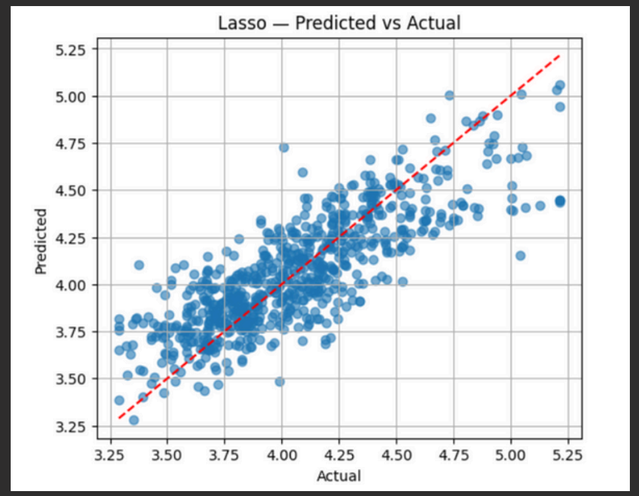


RAW

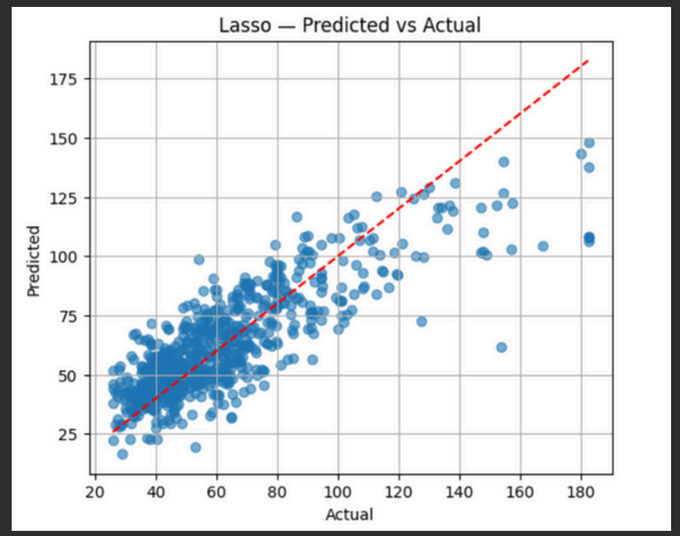


Dvap_Log_Scaled

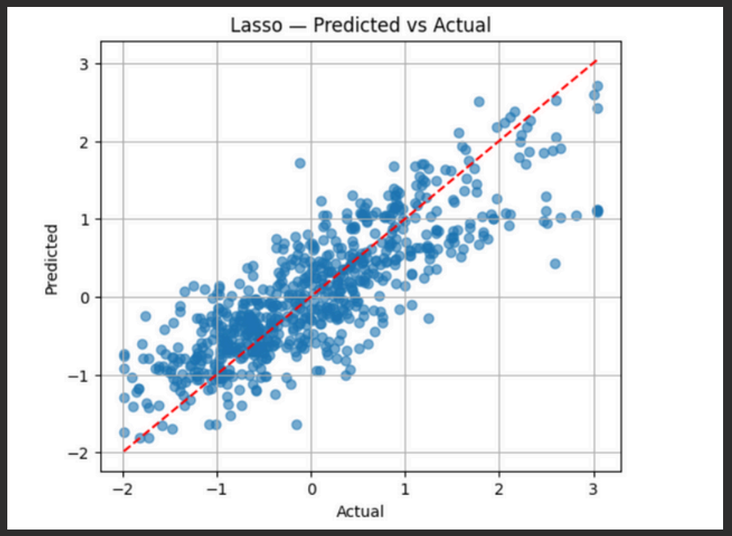
Lasso Regression



Dvap

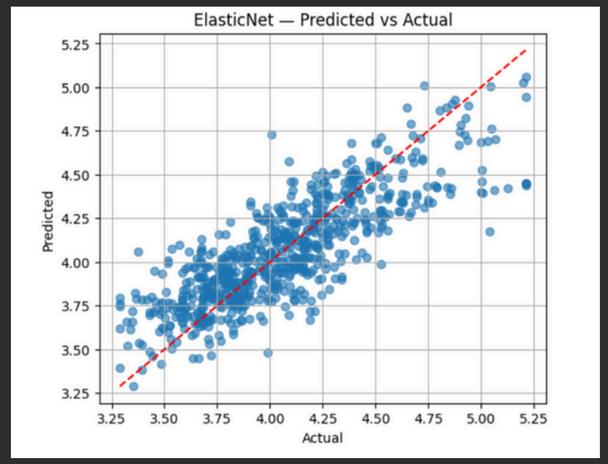


RAW

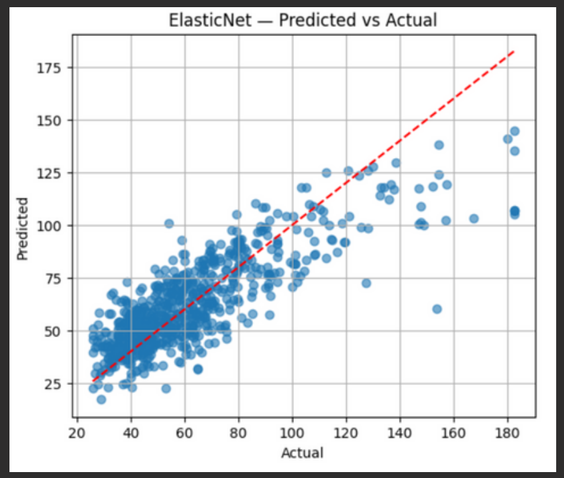


Dvap_Log_Scaled

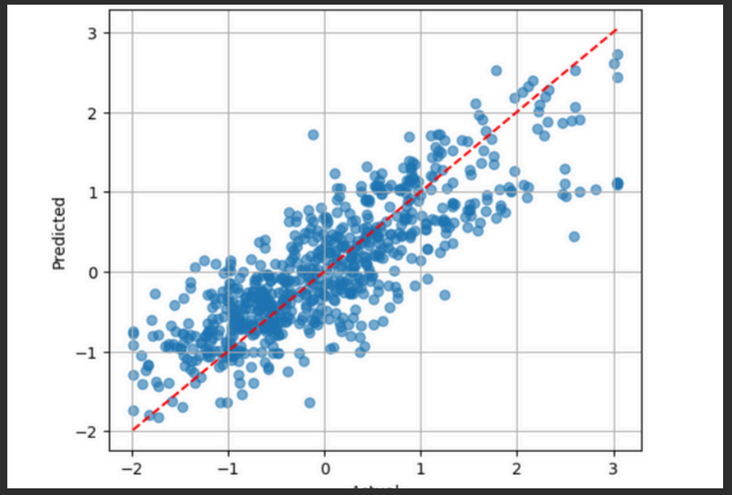
Elastic Net Regression



Dvap

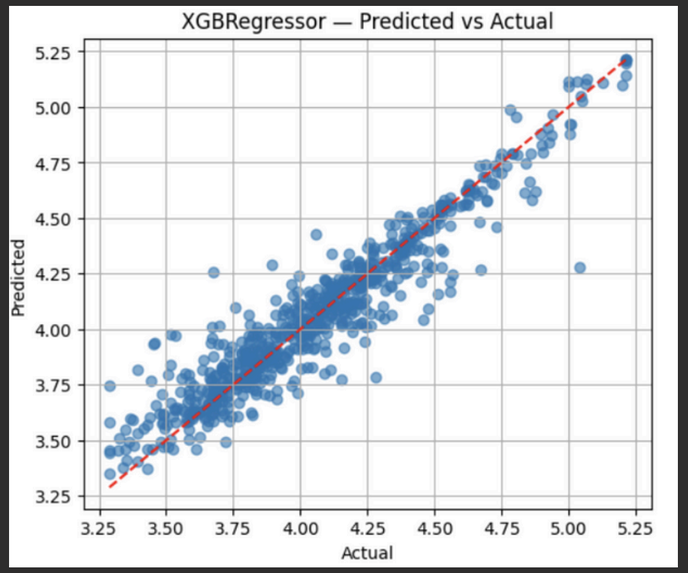


RAW

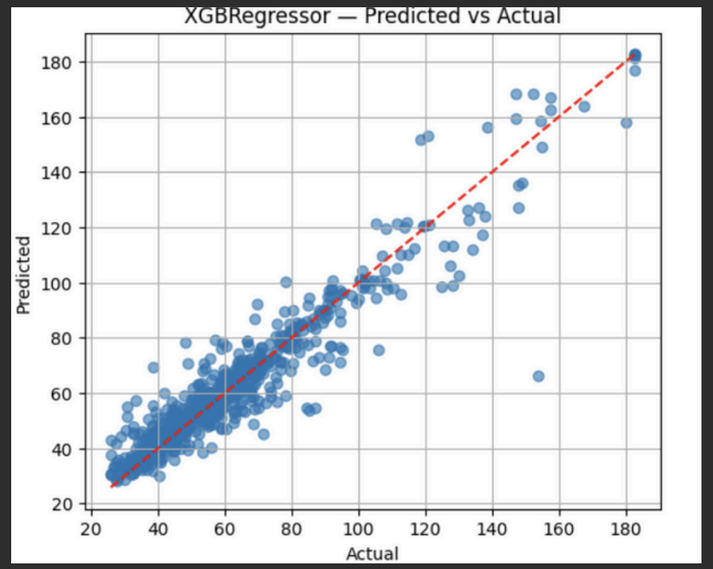


Dvap_Log_Scaled

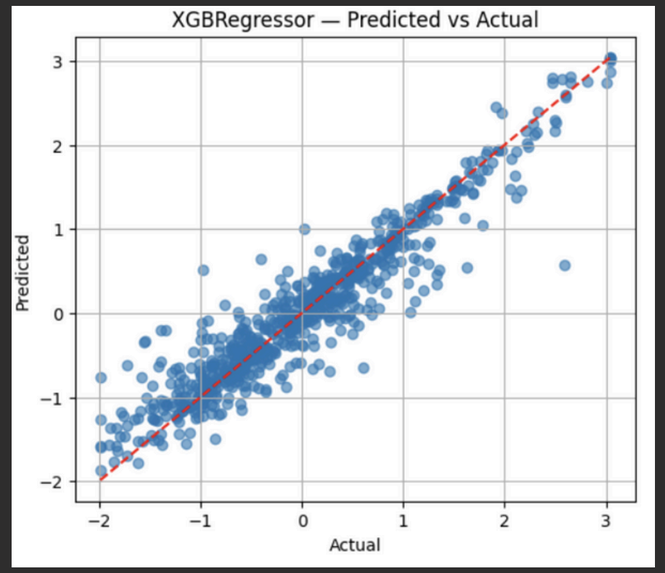
XGB BOOST Regression



Dvap

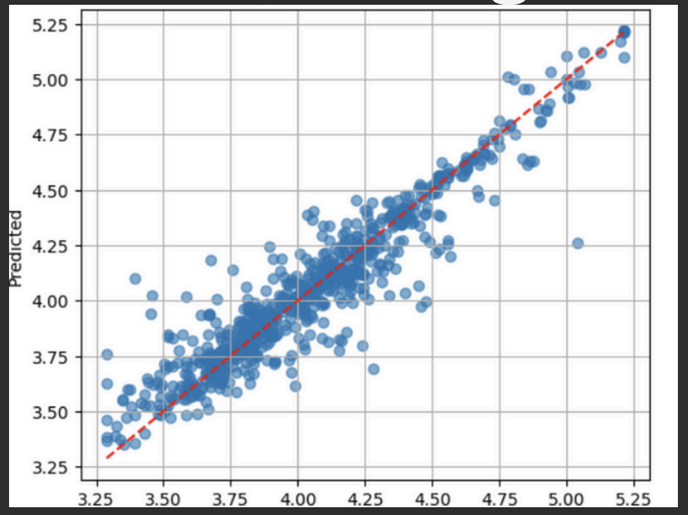


RAW

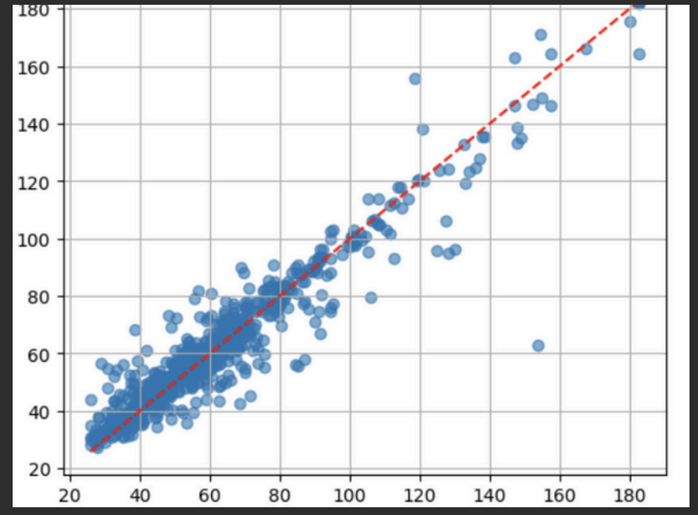


Dvap_Log_Scaled

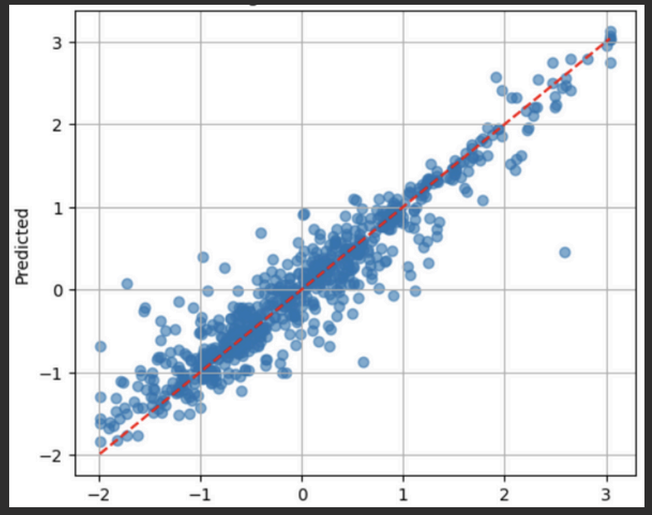
LGBM Regression



Dvap

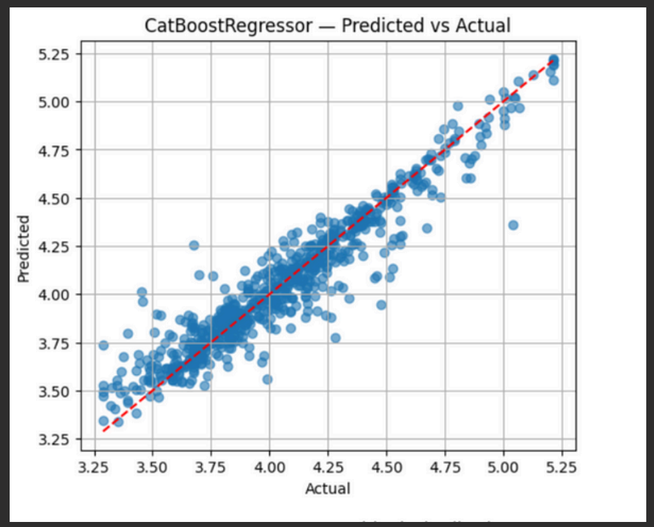


RAW

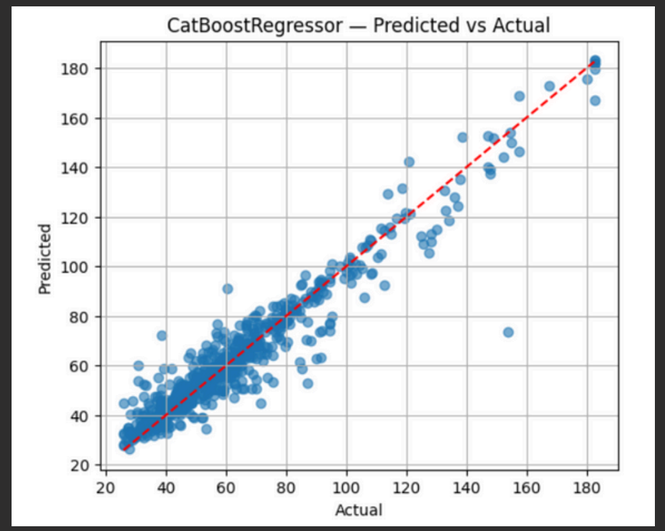


Dvap_Log_Scaled

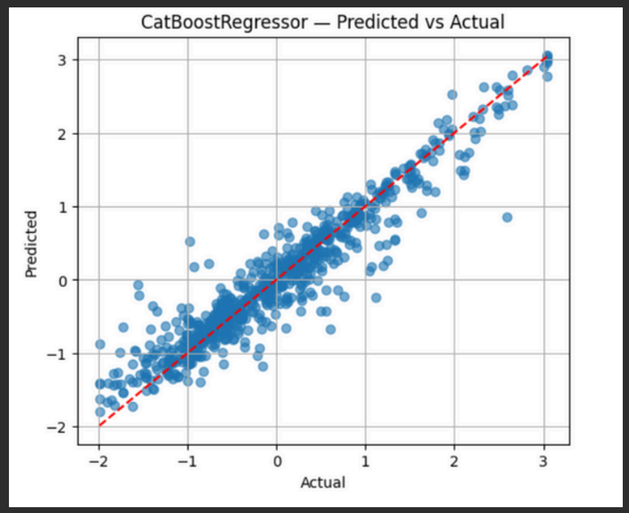
CAT BOOST Regression



Dvap

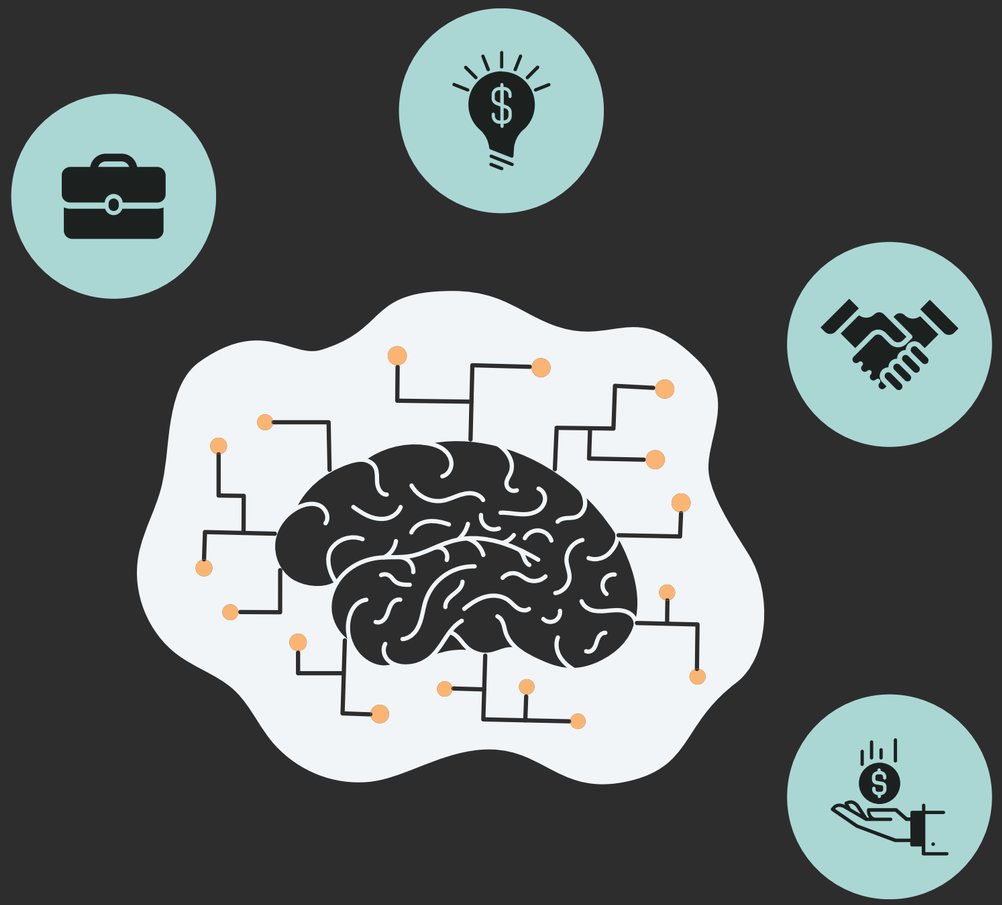


RAW



Dvap_Log_Scaled

Performance Metrics



Coefficient of Determination (R^2)

The "coefficient of determination." It measures the percentage of the variance in dvap that our model can explain. Closer to 1.0 is better.

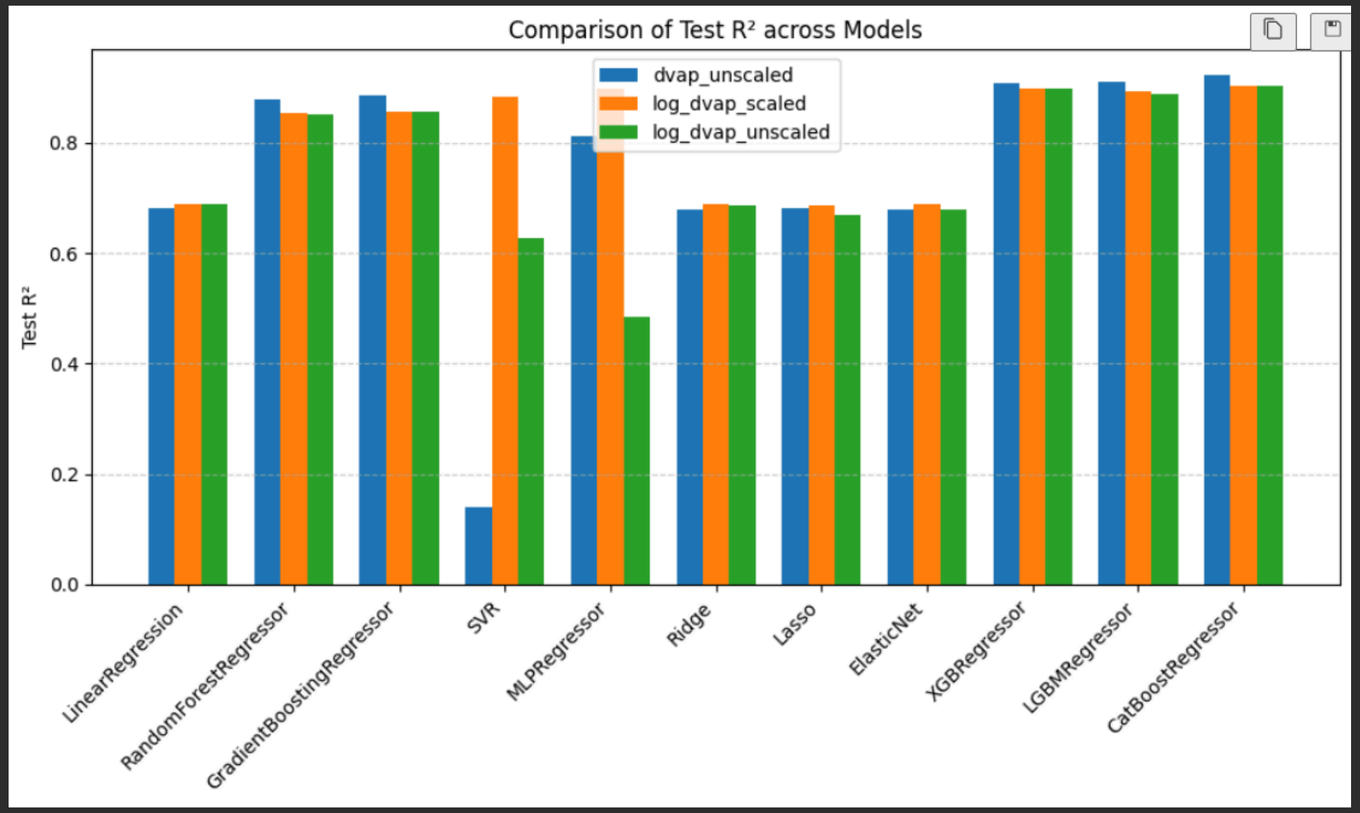
Root mean squared error (RMSE)

Measures the standard deviation of the prediction errors. Lower is better.

Mean absolute error (MAE)

Measures the average size of the prediction errors. Lower is better.

Outcomes and Result



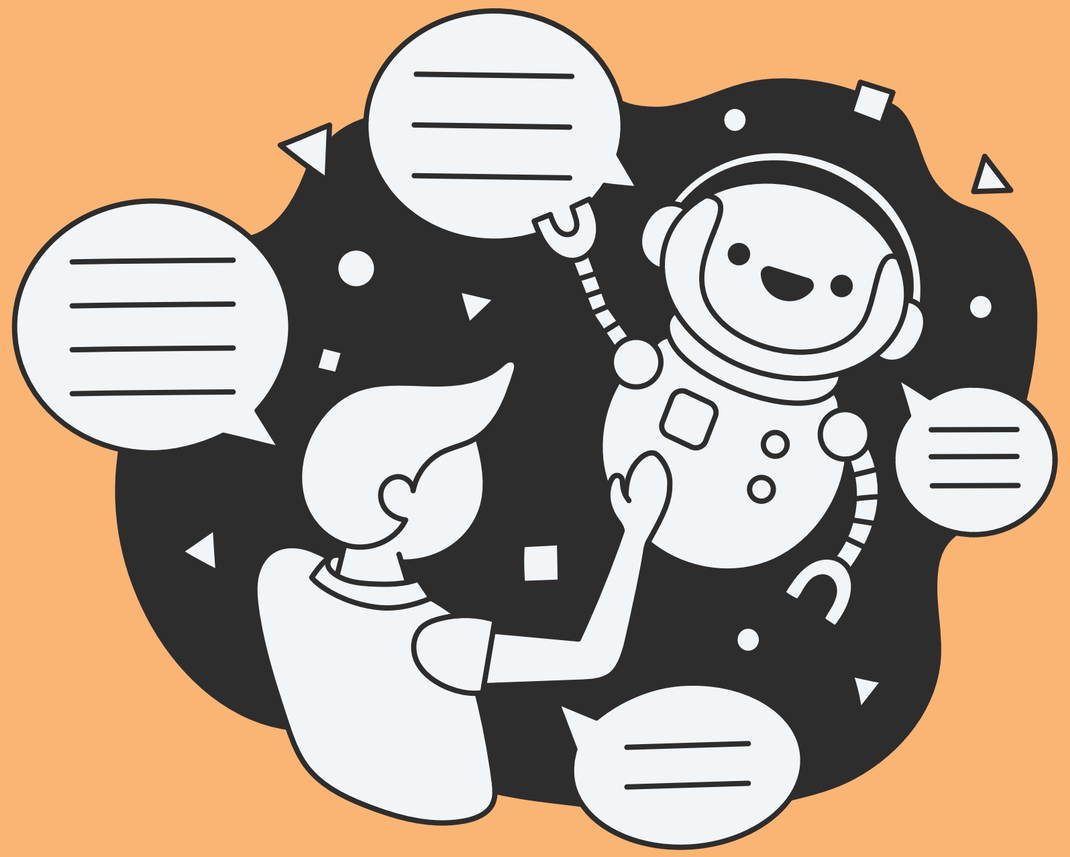
To evaluate model performance on the DVAP prediction task, multiple regression algorithms spanning linear models, tree-based ensemble methods, kernel-based learners, and neural architectures were benchmarked using Test R^2 scores

Key Findings:

- Ensemble-based methods consistently outperformed simpler linear models.
- Traditional linear models showed moderate performance with Test R^2 values around ~0.67–0.70.

Conclusion and Key Innovation

- **Primary Finding:** We successfully demonstrated that machine learning, specifically CatBoost, can predict del Hvap with high accuracy (R^2 = 0.92).
- **Methodology:** A rigorous approach, including analyzing the raw data distribution and systematic feature selection, was essential to prevent bias and achieve this high performance.
- **Key Insight:** Advanced ensemble models are required to capture the complex, non-linear relationships in molecular data; simpler models (Linear Regression, SVR) are insufficient.



Thank You