ML Term Project 2024: Group Number: 198

Gayani Anandagoda, Chen Yihao, Samara Senari

December 22, 2024

1 Introduction:

The term project focuses on the GeckoQ dataset, which comprises atomic structures of 31,637 atmospherically relevant molecules formed by oxidation of $\alpha-$pinene, toluene, and decane. This dataset is designed to facilitate data-driven research in atmospheric science by offering molecular data relevant to aerosol particle growth and the formation of new particles. One of the key molecular properties associated with aerosol particle growth is saturation vapor pressure (pSat), which measures a molecule's ability to condense into the liquid phase.

2 Data:

In this project, the dataset is split into two subsets: a training dataset with 26,637 observations and a testing dataset with 5,000 observations. Additionally, the logarithmic scale of pSat (log_pSat_Pa) is used instead of the raw pSat values. As log_pSat_Pa is a continuous variable, the project focuses on building a regression-based machine learning model to predict log_pSat_Pa for molecules based on their atomic structure. There are 25 features that define the atomic structure of the molecule. One key variable, "parent species," is categorical with seven levels in the training dataset and six levels in the testing dataset (table 1). Additionally, there are 210 missing values for "parent species" in the training dataset and 33 missing values in the testing dataset. Importantly, the missing values for "parent species" are not random; they occur due to difficulties in determining the parent species type. Therefore, instead of removing them, we created a new category labeled "Unknown" to account for these cases.

Level	train	test
Unknown	210	33
apin	6165	1195
apin_decane	46	5
apin_decane_toluene	9	2
apin_toluene	37	5
decane	2218	381
decane_toluene	2	0
toluene	17950	3379

Table 1: Parent Species Levels

2.1 Correlation between the features

We compute the Pearson correlation coefficients for all pairs of numeric variables. The correlation heatmap (figure 1) reveals a strong positive correlation between MW and NumOfAtoms, MW and NumOfN, MW and NumOfO, as well as between NumOfAtoms and NumOfC. The calculated correlation coefficients are summarized in table 2.

Feature1	Feature2	Correlation
NumOfAtoms	MW	0.707
NumOfO	MW	0.88
NumOfN	MW	0.773
NumOfC	NumOfAtoms	0.838

Table 2: varibles with correlation greater than 0.7

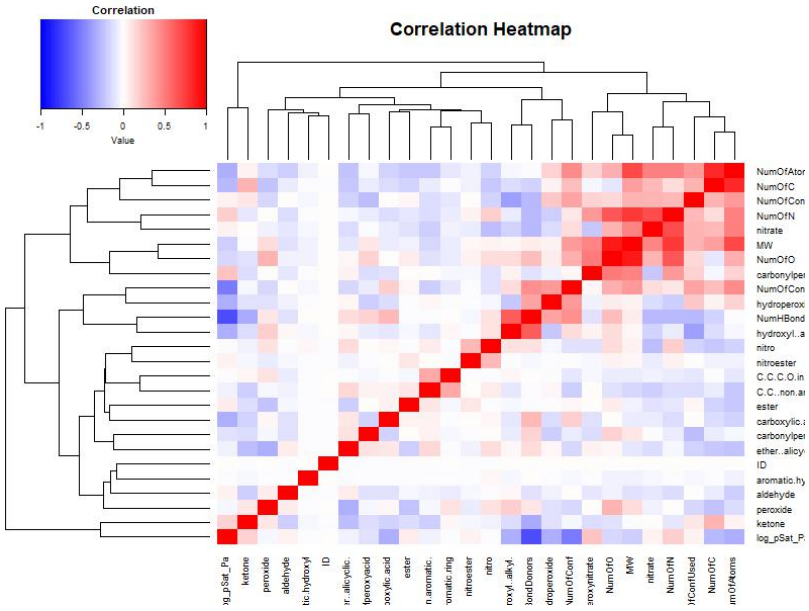

Figure 1: heatmap

3 Model evaluation Based on RMSE of cross validation

3.1 RMSE

RMSE is the most accessible and convenient model evaluation method and it measures the difference between the observed values and the predicted values from the regression model. It provides an indication of how well the model is performing in predicting the continuous target variable, log_pSat_Pa. RMSE gives more weight to large errors due to the squaring operation, making it particularly useful when large prediction errors are more problematic or meaningful.

3.2 K fold cross validation

K-fold cross-validation is used to assess the model's performance on unseen data. We expect the average performance across the folds to provide a robust evaluation of the model. Also, K-fold cross-validation maximizes the use of the available training data for both training and evaluation, leading to more reliable estimates of model performance. It is particularly useful when we do not have a separate validation dataset, as in this project, because it provides a way to simulate multiple train-test splits and assess the model's generalizability. In this project we used 5 fold cross validation.

4 Method:

This section includes a brief description about the each model we used and the method followed.

Dummy and OLS as a base model

A dummy model is a supervised learning model that consistently produces the same constant output, regardless of the values of the covariates. This model is not expected to perform well. As the initial step, we built a dummy model and calculated both the training error and cross-validation error. Next, we developed an OLS (Ordinary Least Squares) model using all the features and evaluated its performance using 5-fold cross-validation with the boot library in R. Since Molecular Weight (MW) is highly correlated with several variables (table x), we trained an OLS model excluding MW to assess whether this adjustment improves the performance of the linear model. The cross-validation errors from both the dummy model and the OLS model served as baselines for assessing the performance of models developed in later steps.

Principal Component Analysis (PCA)

Next, we tried to apply PCA as a dimension reduction and variance reduction method, and the goal was to transform the original correlated predictors into uncorrelated principal components to address multi-collinearity and simplify the model.

Lasso as a regularization

Before moving to non-linear models, Lasso regression was applied as a regularization technique to improve model performance. Lasso is particularly effective in scenarios where the dataset contains a large number of predictors, including those that may not contribute significantly to the predictive power of the model. By using Lasso before exploring non-linear models, we aimed to create a well-regularized linear model,and establish a baseline for comparison with more complex methods. This approach ensures that the transition to non-linear models is justified and that simpler solutions are considered first. We utilized the glmnet package in R to fit a regularized linear model. To determine the optimal value of the regularization parameter lambda , we performed 5-fold cross-validation.

Regression tree

A regression tree is a decision tree-based regression model used for predicting continuous variables. It recursively splits the data into multiple regions, each associated with a predicted value. Specifically, it minimizes the mean squared error (MSE) at each split and outputs the average target value in each leaf node. Unlike linear models, regression trees can capture nonlinear relationships, but single trees may overfit and often benefit from ensemble methods to improve performance. We constructed a regression tree model using the rpart method from the caret package in R, employing 5-fold cross-validation for model evaluation for different `cp` (complexity parameter) values. The `cp` parameter controls the size of the tree by penalizing the number of splits. A smaller `cp` allows for more splits, resulting in a larger, more complex tree, while a larger `cp` leads to smaller, simpler trees by pruning unnecessary splits.

Random Forests

Random Forest is an ensemble learning method based on decision trees. It improves accuracy and stability by building multiple regression trees and averaging their predictions. Using Bagging (Bootstrap Aggregating), it creates individual trees by sampling with replacement from the training data, reducing the risk of overfitting. Additionally, Random Forest selects a random subset of features for splitting at each node, enhancing diversity among trees. Compared to a single tree, Random Forest is more robust to noise and outliers, making it well-suited for high-dimensional data. In this study, we utilized the randomForest package in R to construct two regression models, each with different values for the `mtry` parameter, which determines the number of features considered at each split in the decision trees. Specifically, we evaluated models with `mtry` = 5 and `mtry` = 8. Given the substantial computational costs associated with tuning additional hyperparameters, such as the number of trees (`ntree`) and the minimum node size (`nodesize`), we chose not to optimize these parameters.

Gradient Boosting

Gradient Boosting is an iterative ensemble learning technique that builds a predictive model by combining multiple weak learners, typically decision trees. In our project, we employed Gradient Boosting to enhance prediction performance using the R library xgboost, a widely used library for gradient-boosted decision trees. To optimize model performance, we implemented a mesh tuning approach. In mesh tuning, we try different combinations of parameters through a predefined parameter mesh and evaluate the effect of each set of parameters.

The core idea of this approach is an exhaustive search, finding the most effective parameter by traversing all possible combinations of parameters. By comparing the performance of the RMSE, we were able to find the parameter with the best performance. Perparameters we tuned during the process are,

1. `max_depth`: The maximum depth of each decision tree.
2. `eta(learning rate)`: This parameter controls the contribution of each tree to the final model.
3. `subsample`: The fraction of samples used for training each tree. Using more subsets helps prevent overfitting by introducing randomness.
4. `colsample_bytree`: This parameter is randomly selected for training each tree. This reduces feature dependence and further prevents overfitting.
5. `nrounds`: The number of boosting rounds or trees in the model. Understandably, more nrounds predict a better model.

Support Vector Machine (SVM)

A non-linear kernel, specifically the radial kernel, is used in this analysis. The radial kernel takes the form,

$K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$, where γ is a tuning parameter.

As the value of γ increases, the model becomes more non-linear, allowing it to capture more complex patterns in the data. We utilized the R library caret to model Support Vector Machines (SVMs) with (method = "svmRadial") and for parameter tuning and evaluation. The choice of tuning parameters is critical, as it determines the extent to which the model underfits or overfits the data.

The cost argument specifies the cost of violating the margin. A small `c` value allows for wider margins, resulting in more support vectors on or violating the margin. In contrast, a large `c` value leads to narrower margins, reducing the number of support vectors on or violating the margin. Low cost values correspond to high bias and low variance, while high cost values correspond to low bias and high variance.

To select the optimal combination of `c` and γ , we conducted a grid search over various values of these parameters. For each model, we calculated the Root Mean Squared Error (RMSE) using 5-fold cross-validation. This approach ensures a robust evaluation and helps identify the parameter pair that minimizes prediction error.

5 Results and Discussion:

This section discusses the results we obtained for each model.

Model Evaluation

Method	**RMSE**	**Parameters**
OLS	1.689753	-
OLS sfter doing PCA	1.689753	-
OLS without the varible "MW"	1.702887	-
Lasso	1.944237	`lambda = 0.002651092`
Regression tree	2.344318	`cp = 0.0424216`
Random Forests	1.619372	`mtry = 5`

gradient Boosting	1.399229	`eta = 0.1;`
		`max_depth = 6;`
		`subsample = 1;`
		`colsample_bytree = 0.7;`
		`gamma = 5;`
		`min_child_weight = 1;`
		`nrounds = 200`

SVM	1.557945	`gamma = 0.03125;` `C = 1`

Table 3: summary of the modal performances. Provides 5 fold cross validation errors

Table 3 summarizes the performance of various models applied to the dataset, evaluated using Root Mean Square Error (RMSE) from 5-fold cross-validation. Lower RMSE values indicate better performance. Gradient Boosting achieved the best result (RMSE: 1.399), followed by Support Vector Machines (RMSE: 1.558) and Random Forests (RMSE: 1.619). Ordinary Least Squares (OLS) methods performed similarly (RMSE: 1.690), with no improvement from Principal Component Analysis (PCA). Lasso regression and Regression Trees had higher errors, reflecting weaker performance. The table also lists the key hyperparameters used for each model, highlighting the tuning efforts to optimize their performance.

PCA

PCA did not contribute to the score as much as we expected. This outcome is not unusual, as PCA focuses solely on explaining variance in the predictors rather than enhancing the relationship between predictors and the target variable. The principal components that capture the most variance may not be the most predictive for the target, and reducing dimensions could inadvertently discard relevant information. Additionally, PCA is an unsupervised

method and does not consider the target variable when creating components, potentially leading to suboptimal alignment with the predictive task. In cases where the original data structure already aligns well with the target or where multicollinearity is minimal, PCA may not offer significant benefits. This highlights that PCA is primarily a data transformation technique, not inherently a predictive modeling tool, and alternative methods like feature selection or regularization may be more effective for improving model performance.

Gradient Boosting

In the figure 2, when `eta=0.01`, the red line shows significantly higher and more fluctuating RMSE, indicating underfitting. Comparing `eta=0.1` (green) and `eta=0.2` (blue), the green line performs best at `max_depth=6`, achieving the lowest RMSE. In the figure 3, at `eta=0.01`, RMSE is high across all depths, especially for the red line `(max_depth=3)`. When `eta=0.1`, the green line `(max_depth=6)` performs the best with the lowest RMSE. At `eta=0.2`, RMSE slightly increases, particularly at greater depths. Therefore, `eta=0.1` and `max_depth=6` are the optimal choices. Parameters of the best gradient boosting model is tabullated in table 3.

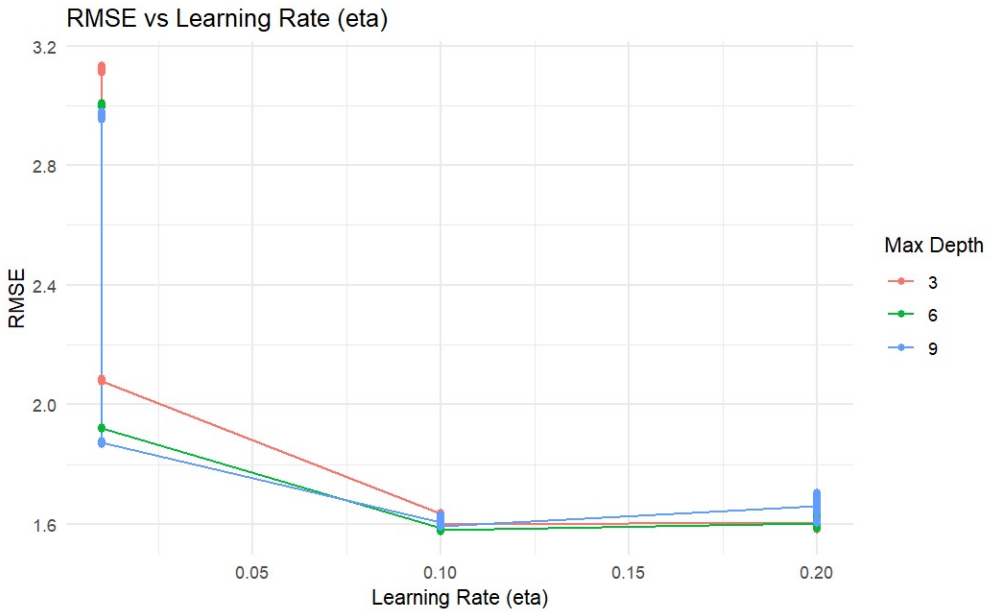

Figure 2: RMSE vs Learning Rate (eta)

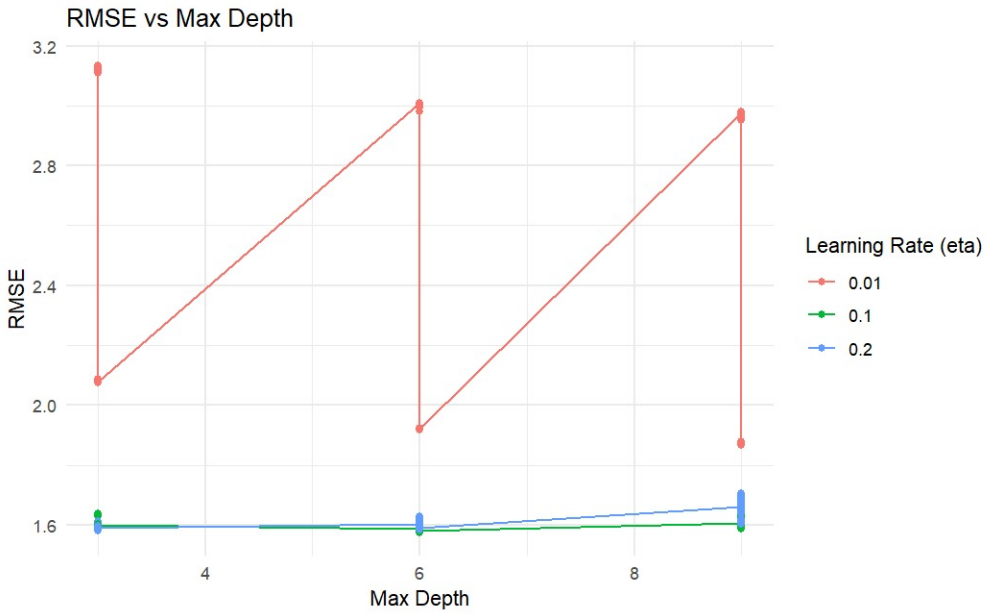

Figure 3: RMSE vs Max Depth

SVM

Figure 4 illustrates how the RMSE changes with different combinations of `c` and γ. From the plot, it is evident that the model with `c` $= 10$, $\gamma = 0.01$ achieves the lowest RMSE of 1.550164. The second lowest RMSE, 1.557945, occurs with `c` $= 1$, $\gamma = 0.03125$. Considering the bias-variance tradeoff, the model with the second lowest RMSE is preferable over the other models.

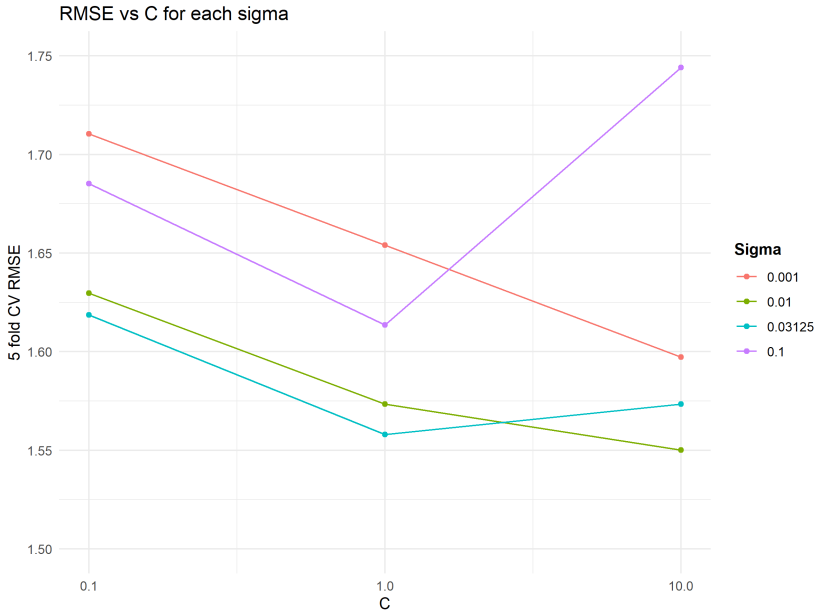

Figure 4: SVM parameter tunning

6 Conclusion:

The Support Vector Machine (SVM) and Gradient Boosting models demonstrated strong performance in addressing the prediction task, with each model effectively capturing the underlying patterns in the data. On the public leaderboard, the SVM model achieved a Kaggle score of 0.7554, while the Gradient Boosting model attained a score of 0.7524. However, there remains room for improvement in fine-tuning the hyperparameters further to optimize the models' accuracy and generalization capabilities. While grid search was used to explore the parameter space, the process required significant computational resources. For instance, a single grid search run for the Gradient Boosting model could take an entire night, making it challenging to test all possible parameter combinations within the current time constraints.

Future work could focus on exploring polynomial models, which we have not yet attempted. As we learned from presentations in the final lecture, polynomial models may be better suited for capturing nonlinear relationships in the data. Additionally, prioritizing feature engineering could further enhance the model's performance by creating more meaningful features or transforming existing ones to provide better inputs. These strategies, combined with more refined parameter tuning, could further improve the accuracy and robustness of the models. Despite the current limitations, the models we developed are both robust and effective, laying a solid foundation for future work.

7 Grading:

Deliverables: Grade 3

The project presented significant challenges, starting with unfamiliarity with the dataset and its variables. Initial struggles to understand the variable names were addressed through background research to grasp the data's context. Various methods and practices from class and exercises were applied, but inconsistencies in data manipulation required several revisions to the approach. Kaggle scores were used as benchmarks, helping the team identify the most effective methods for analysis.

While the project demonstrates a solid understanding of the topic and effective application of research methods, there were areas where improvements could have been made to achieve better results. Specifically, more thorough parameter tuning and exploration of alternative approaches (feature engineering and feature selection) might have enhanced the accuracy and quality of the outcomes. However, the team's focus on achieving a competi-

tive Kaggle score, combined with time constraints, computational challenges, and associated costs, limited the extent of this refinement. These factors impacted the final deliverables, leading the team to conclude that the work merits a grade of 3, reflecting both its strengths and the areas for improvement.

Group work: Grade 5

The group work began with a Zoom meeting to discuss the project's background, objectives, and initial ideas, laying the foundation for effective collaboration. Each member shared insights, identified challenges, and proposed strategies, agreeing to prioritize efficient communication. A WhatsApp group was established as the primary platform for real-time updates, resource sharing, and addressing concerns, fostering continuous alignment and momentum. During a subsequent meeting, tasks were divided based on individual expertise, preferences, and workload balance, ensuring fair contributions.

Frequent updates in the WhatsApp group enabled members to share progress, seek assistance, and address challenges collaboratively. The final step involved integrating individual contributions into a shared document, and collaboratively refining the content for consistency and clarity. The group's open communication, strategic use of digital tools, and equitable task division fostered a positive and productive team environment, resulting in satisfactory outcomes.