IML 2023 Term Project

Topias Pesonen[1]

[1]**015251420, Group 21**

PROJECT PURPOSE AND DATA

The purpose of the project is to predict the saturation vapor pressure using a regression model on the GeckoQ dataset(1). The tools for this project are the following: Python 3.10. with the packages pandas, numpy, matplotlib, seaborn, sklearn, scipy, xgboost.

We decided to go with the standard version of the project, without the topological fingerprints.

The dataset contains 26 columns, including the Id and the target variable. This leaves 24 columns as independent variables to predict the dependent variable, pSat_pa. Other variables are in numeric form, except parentspecies, which is categorical. There is some correlation within the variables, as seen in Figure **??**. For example, it is to be expected that a nitrate might have a positive correlation with the number of Nitrogen atoms.

kThe first metric for the success of the model was the Kaggle competition. Our team did ”ok” in it, even though the results dropped in the private leaderboard. In this report we address some of the potential causes for the issues as well.

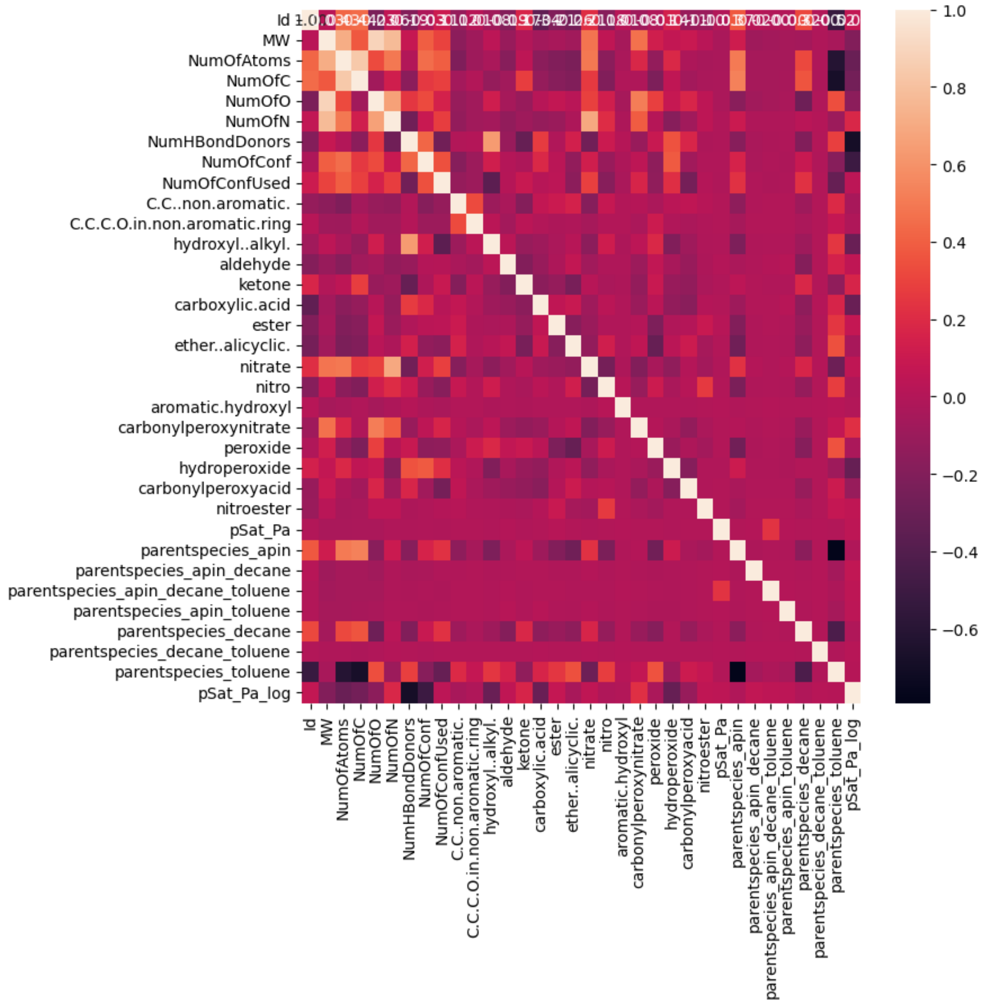

Figure 1. Corralation matrix of the variables

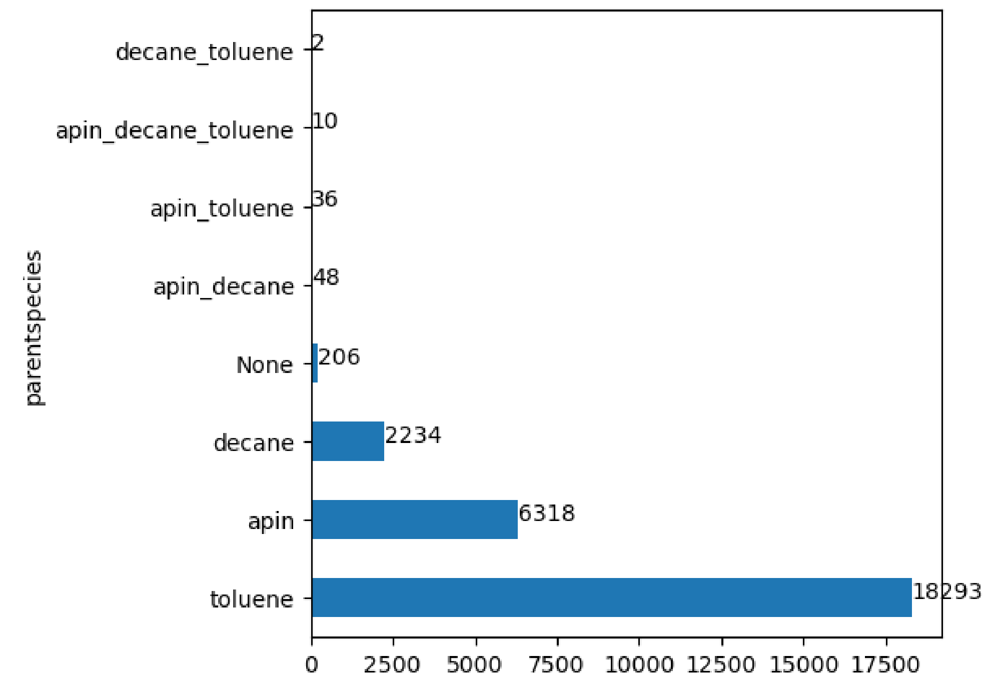

Figure 2. Parentspecies value counts

DATA PREPROCESSING

To start off, we checked if the data contains any missing values. Only the column ’parentspecies’ was missing some data, which we then filled with ’None’. The value counts after the imputation can be seen in Figure 2

After filling the missing spots, we one-hot encoded the variable to make it usable for the models.

Next up was the transformation of the target variable. Figure 3 shows the unprocessed variable. The variance is quite high due to some really high values, but we did as instructed and after the log10-transformation the data looks much more manageable (Figure 4).

After some test runs, we decided to check for outliers and if we could make the model a bit better. We decided to use Z-score to find some outlier rows, and by limiting the Z-score withing [-3,3], we have the final dataset with 109 outlier rows removed we used for the competition submission. For the final results post-competition, however, we decided to not remove the outlies, due to lack of domain knowledge. Even if the R2-score would get marginally better, we consider keeping the outliers better since we don’t understand how they could affect the model in real life production scenario.

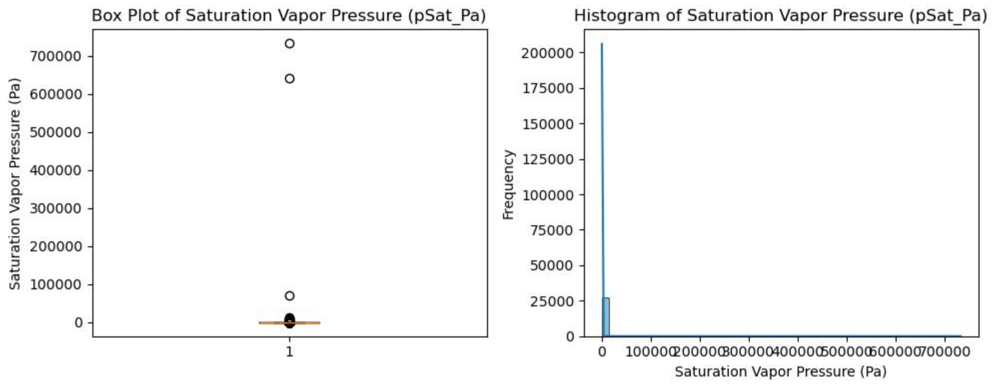

Figure 3. Target variable pSat_Pa in the unprocessed dataset.

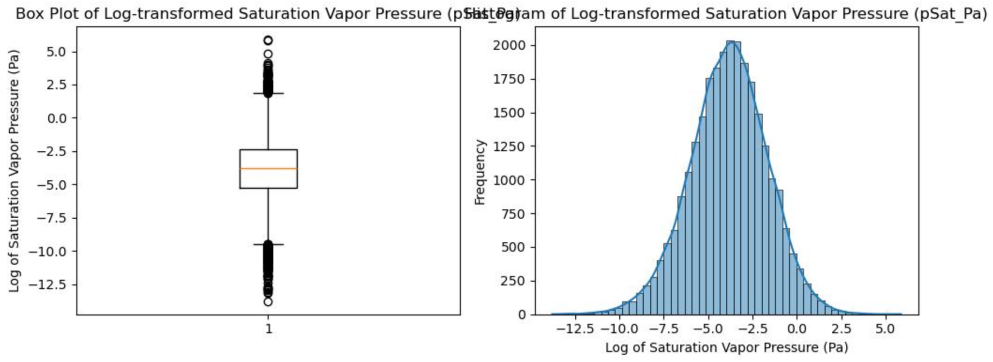

Figure 4. Target variable pSat_Pa after log10-transformation.

INITIAL MODELING AND RESULTS

Even though this is not a trivial regression problem, we started off with a simple linear regression. After splitting the data in to train and validation sets, we ran a LinearRegression from the sklearn-package. This gave us the following results:

Metric	Value
R2 Score	0.7094957076642562
Mean Squared Error	1.3396608554531388

Table 1. First linear regression results.

Now, even though this result doesn't look terrible, we suspect that we can do much better with some more sophisticated modeling. We chose three models to approach the probelm: Random Forest, Gradient Boosting and XGBoost, a more regularized version of Gradient Boosting.

XGBoost

XGBoost or "Extreme Gradient Boosting" is gradient boosted tree algorithm. The algorithm's objective function consists of two parts: training loss and regularization term, that helps us to avoid overfitting (2):

$\text{obj}(\theta) = L(\theta) + \Omega(\theta)$

In the case of our project, we used mean squared error as the loss function L:

$$L(\theta) = \sum_i (y_i - \hat{y}_i)^2$$

XGBoost builds an ensemble of decision trees in a sequential manner. Each tree in the sequence tries to correct the errors or residuals made by the previous tree. The final prediction is a weighted sum of the predictions from all trees.

The regularization term $\Omega(\theta)$ in XGBoost's objective function penalizes the complexity of the model (i.e., the number and depth of trees), helping to avoid overfitting.

We employ the python package xgboost to implement this model.

Parameters

To start with the modeling, we found from sklearn a package GridSearchCV, which can be used to find the parameters that maximize given target, in this case R2-score. We applied this method to all three main models, and found the optimal parameters for them. Parameters we tested for the models are shown in Table 2.

Table 2. Hyperparameter Grids and Best Parameters

Algorithm	Hyperparameter Grid		Best Parameters
Random Forest	'n_estimators':	[100, 200, 300]	'max_depth': 20,
	'max_depth':	[None, 10, 20, 30]	'min_samples_split': 10,
	'min_samples_split':	[2, 5, 10]	'n_estimators': 300
Gradient Boosting	'n_estimators':	[100, 200, 300]	'learning_rate': 0.1,
	'learning_rate':	[0.1, 0.01, 0.001]	'max_depth': 4,
	'max_depth':	[3, 4, 5]	'n_estimators': 300
XGBoost	'n_estimators':	[100, 200, 300]	'learning_rate': 0.1,
	'max_depth':	[3, 4, 5]	'max_depth': 4,
	'learning_rate':	[0.1, 0.01, 0.001]	'n_estimators': 300

For the function GridSearchCV we chose cv=3, due to the heaviness of computation for multiple rounds of cross-validation.

Additionally, we applied simple Principal Components Analysis to the data to see if we can get some better results with better feature selection. This was done with sklearn packages methods PCA and StandardScaler. We selected to retain 95% of the variance and select components as such, which left us with 21 features.

This crude implementation of PCA did not improve the results at all.

Table 3. R2 Scores of the models

Model	R2 Score
Linear Regression	0.709496
Random Forest	0.719011
Gradient Boosting	0.740256
Random Forest with PCA	0.703084
Gradient Boosting with PCA	0.715384
XGBoost	0.748609
XGBoost with PCA	0.725775

Results for the competition

As we can see from the Table 3, XGBoost is the model with the best R2 Score. This is what we went with for the competition. In the public leaderboard, this gave us the 37th place with the score of 0.6770. In the private leaderboard this score dropped slightly to 0.6762, giving us the 52nd place.

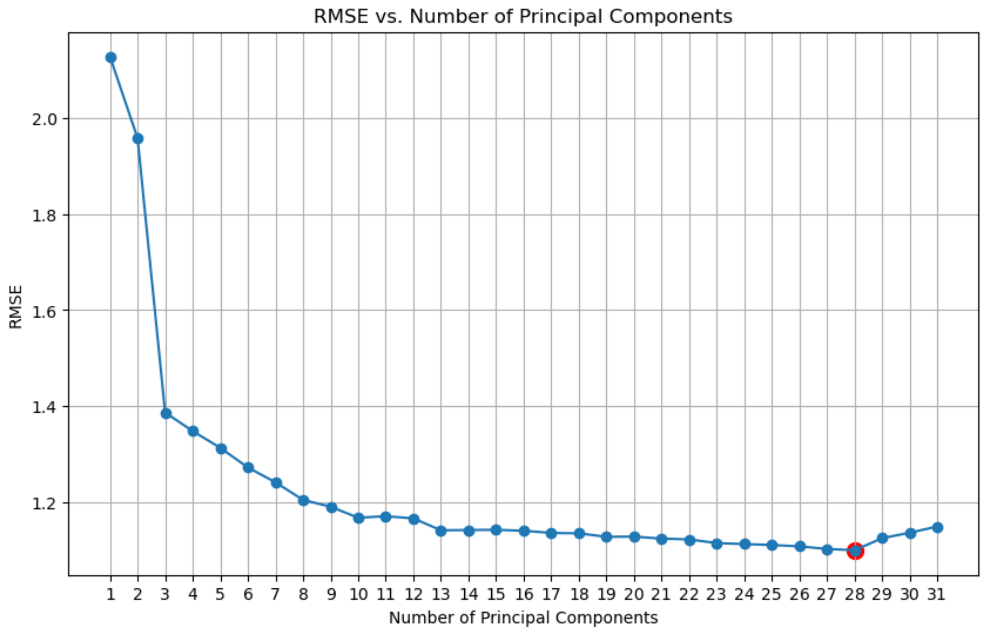

Figure 5. RMSE vs. Number of Principal Components

MODIFICATIONS AFTER THE COMPETITION RESULTS

First things that came to our mind after seeing the results on Kaggle: our model was overfitting. This is apparent due to the quite significant drop in the R2 score with the actual test data. We might be able to address this with improved PCA.

We performed another PCA, this time plotting the Root-mean-square-deviation (RMSE) and R2 scores against the number of components. The plots can be seen in the images 5 and 6.

Based on this, the optimal number of components is 28. These components yield us an R2-score of **0.737**.

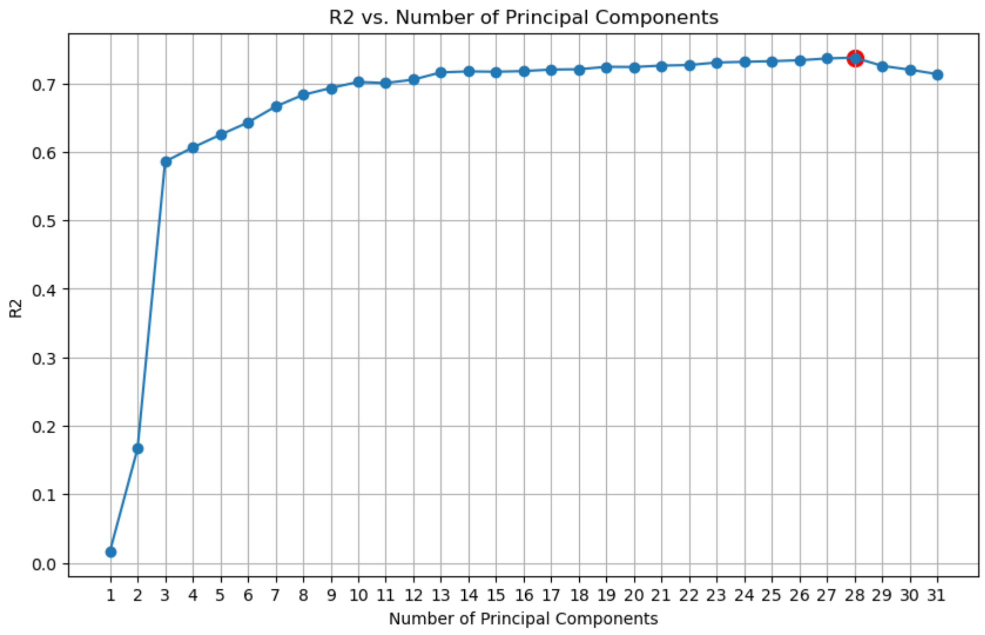

Figure 6. R2 vs. Number of Principal Components

FINAL RESULTS AND DISCUSSION

At this stage, it's obvious our team could have a done a lot better with some more sophisticated data processing and modeling. Our domain knowledge on the subject matter is lacking, to say generously. Therefore the project was done in the dark, with understanding only on the format of the data available, and methods to make the predictions.

XGBoost turned out to be the winner, with the following resulting metrics:

Table 4. Final XGB Performance Metrics

Metric	Value
R2 Score	0.737
Mean Squared Error	1.211
Root Mean Squared Error	1.100

We're sure there's still lots of room for improvement in our approach. For example, creating a custom ensemble model could be a next step to get even better results. All in all, we're quite happy with the R2 score we got from out final model.

SELF-GRADING

Proposed grade: 3

The topic could have been researched better, and the data exploration might have warranted a bit more scrutiny. However, the basic idea was understood and the team was able to start modeling with some acceptable results.

Data pre-processing was done to an acceptable extent, but some more in depth approaches could have been implemented. The same goes for modeling: the models were suitable for the task, and the results were not bad.

The project as a whole is readable and the process clear, but the authors could have gone more in depth with the models and the reasoning behind them. With all this in mind, we think the project deserves a grade in the middle of the pack - not great, not terrible.

REFERENCES

[1] V. Besel, M. Todorović, T. Kurtén *et al.*, “Atomic structures, conformers and thermodynamic properties of 32k atmospheric molecules,” *Sci Data*, vol. 10, p. 450, 2023.

[2] XGBoost Developers. (2023) XGBoost Documentation: Model. [Online]. Available: https://xgboost.readthedocs.io/en/stable/tutorials/model.html