Master's Programme in Data Science

# Final Term Project Report IML24

Joonas Ahovalli, Juuso Saavalainen
Group 74

December 21, 2024

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

# Contents

# 1. Exploratory Data Analysis

## 1.1 Manual exploration

We started the project by diving into the data in our first live session together. We produced a heat map to inspect the correlations among all features (figure 1.1), scatter plots for each feature against the target, and box plots of each feature. Additionally, we used RandomForest to extract an initial view of the importance of the features (figure 1.2). Manual data exploration was done in both R and Python. Libraries such as seaborn+matplotlib and VS code extensions such as data wrangler[†] were utilized in EDA and during the whole project. After an initial overview, we moved to automatic data profiling that provided a broader and structured interactive overview of the dataset.

## 1.2 Data profiling

We utilized *ydata-profiling* [‡] to automate and verify the findings. This tool automates the typical EDA by producing basic visualizations and analytics on a given dataset. The profiling report contains an overview of the entire dataset combined with automatic alerts that show imbalanced features, high correlations, missing values, and other important aspects to notice. Each feature is presented with key statistics and plots, according to the data type. Figures 1.3 and 1.4 show an example of numerical and categorical feature descriptions, respectively. In addition to feature-specific information, a correlation plot is created for the entire data set with a heatmap and tabular format. The interactions of features can be visualized through an interactive hex plot, missing values are visualized, and a sample of the data is provided. Even though this tool has several limitations, for example, a feature with six distinct values might be classified as numerical, whereas with five it is classified as categorical, we found it useful in the given context.

[†]https://code.visualstudio.com/docs/datascience/data-wrangler

[‡]https://docs.profiling.ydata.ai/latest/

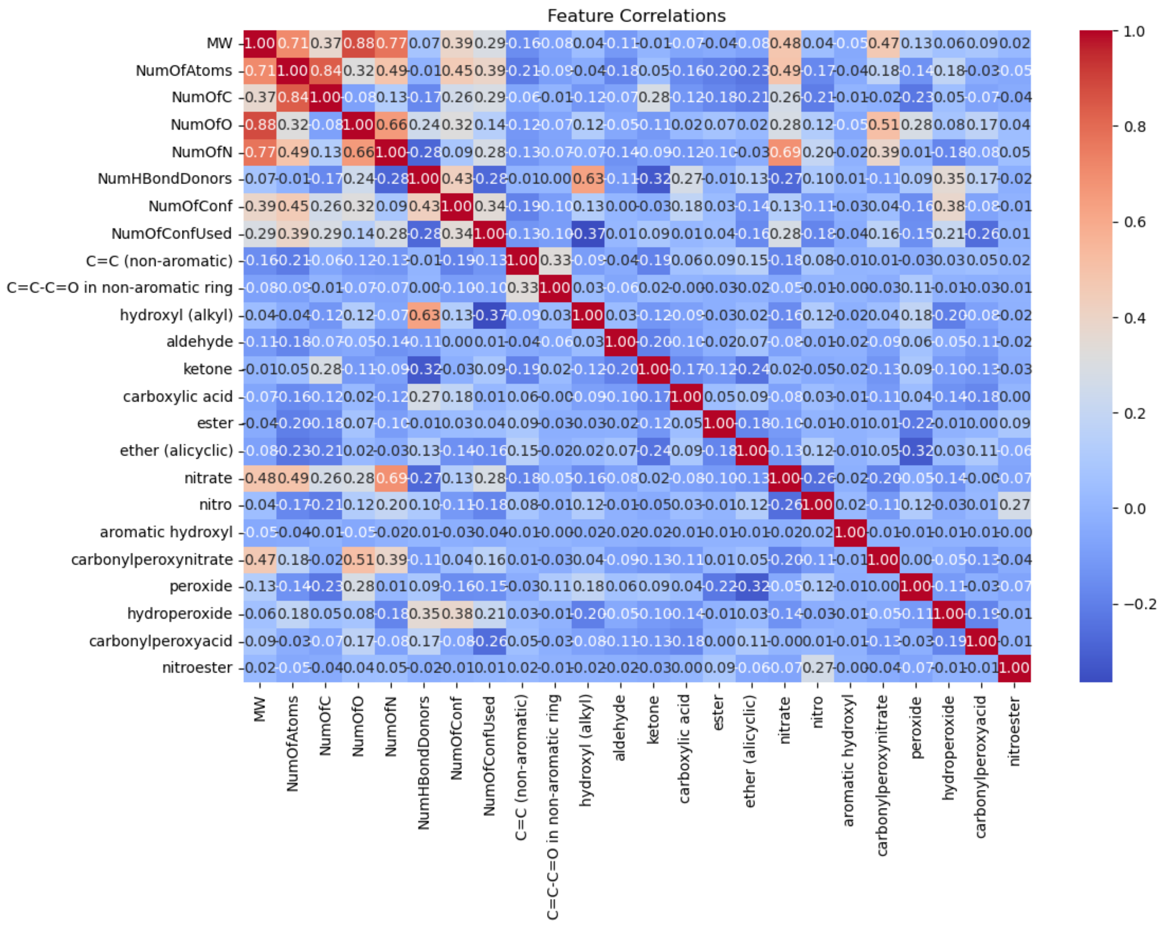


**Figure 1.1:** Heat map depicting feature correlations among the base numerical features.

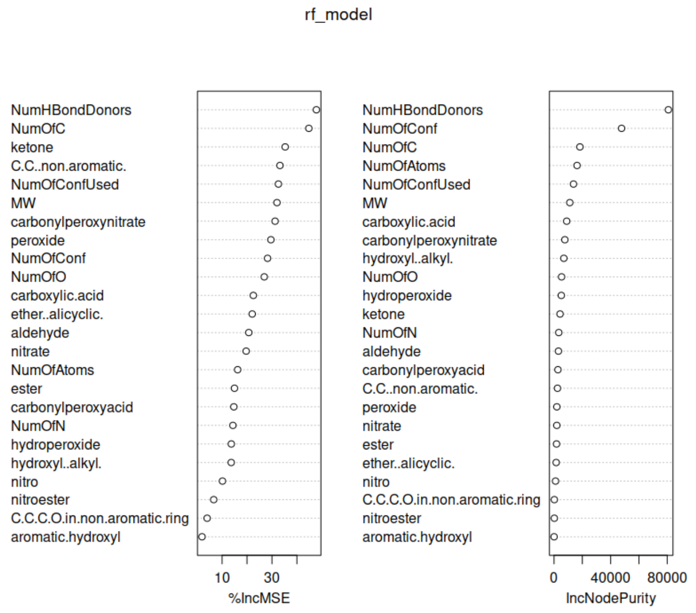
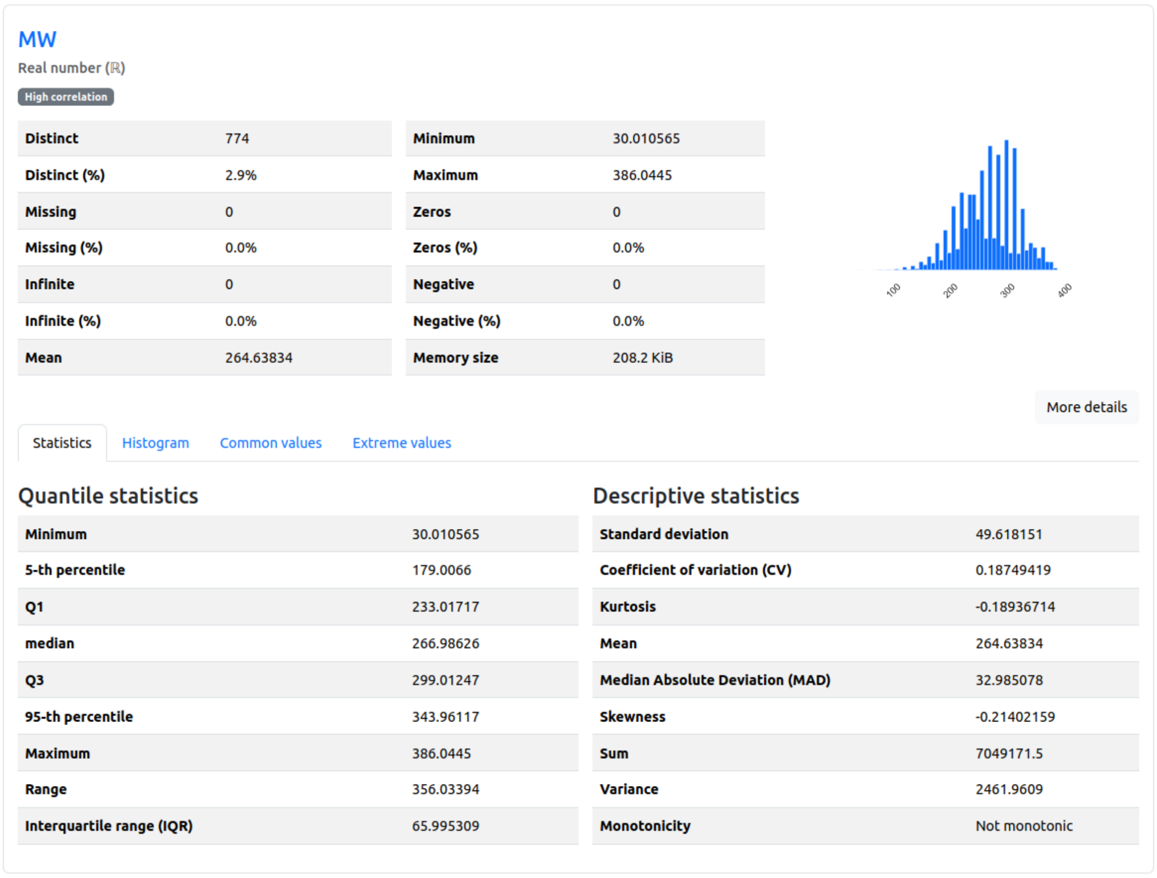


**Figure 1.2:** Initial feature importance with Random Forest. Left: Increase measured with Mean Squared Error (MSE). Right: Increase measured with node purity.



MW

Real number (ℝ)

High correlation

| | | | |
|---|---|---|---|
| **Distinct** | 774 | **Minimum** | 30.010565 |
| **Distinct (%)** | 2.9% | **Maximum** | 386.0445 |
| **Missing** | 0 | **Zeros** | 0 |
| **Missing (%)** | 0.0% | **Zeros (%)** | 0.0% |
| **Infinite** | 0 | **Negative** | 0 |
| **Infinite (%)** | 0.0% | **Negative (%)** | 0.0% |
| **Mean** | 264.63834 | **Memory size** | 208.2 KiB |

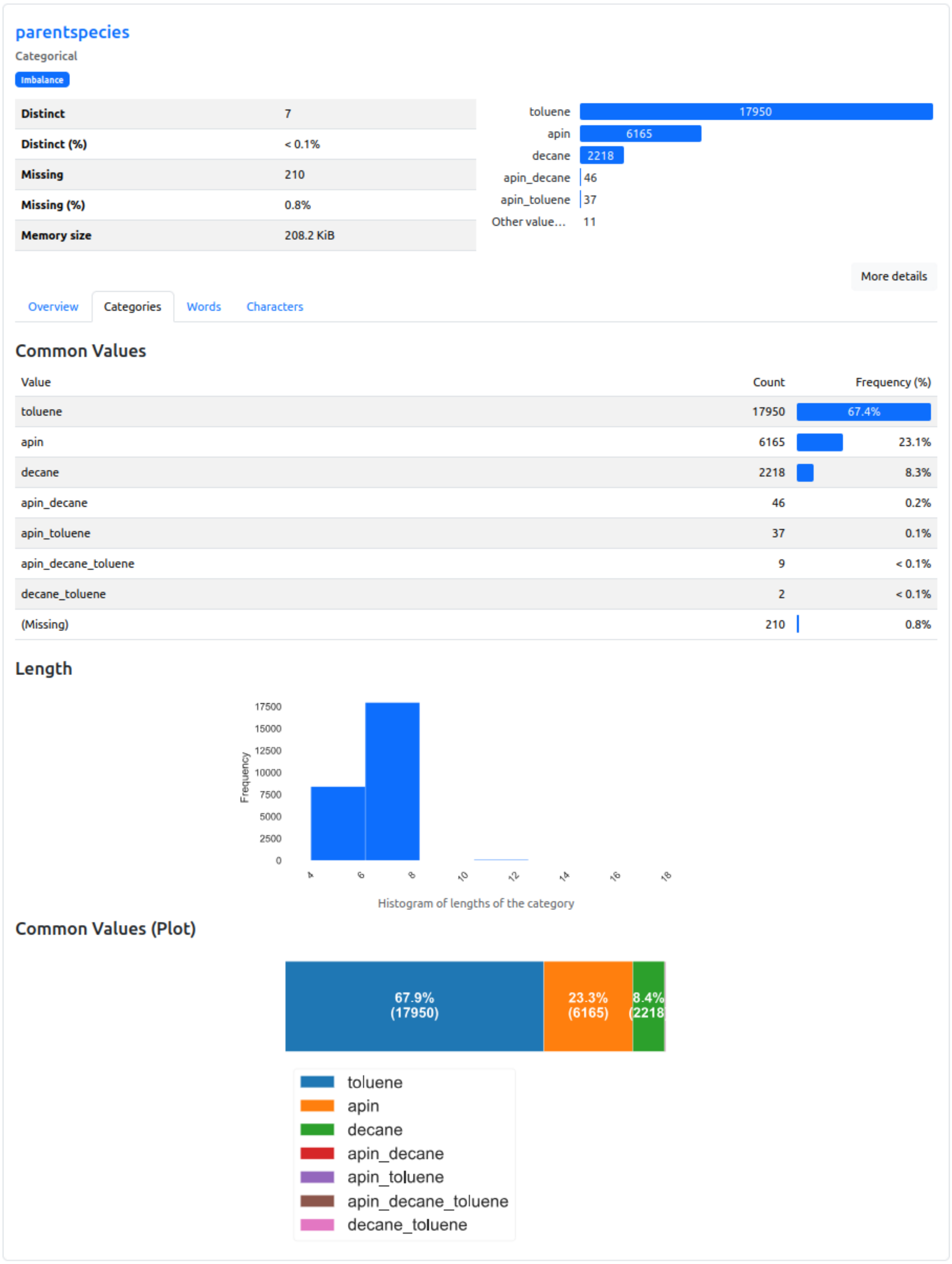
Quantile statistics

| | |
|---|---|
| **Minimum** | 30.010565 |
| **5-th percentile** | 179.0066 |
| **Q1** | 233.01717 |
| **median** | 266.98626 |
| **Q3** | 299.01247 |
| **95-th percentile** | 343.96117 |
| **Maximum** | 386.0445 |
| **Range** | 356.03394 |
| **Interquartile range (IQR)** | 65.995309 |

Descriptive statistics

| | |
|---|---|
| **Standard deviation** | 49.618151 |
| **Coefficient of variation (CV)** | 0.18749419 |
| **Kurtosis** | -0.18936714 |
| **Mean** | 264.63834 |
| **Median Absolute Deviation (MAD)** | 32.985078 |
| **Skewness** | -0.21402159 |
| **Sum** | 7049171.5 |
| **Variance** | 2461.9609 |
| **Monotonicity** | Not monotonic |

**Figure 1.3:** Overview of the statistics of the numerical feature "MW", produced by ydata-profiling. For a more detailed view, the full automated data profiling is saved in HTML format and can be downloaded by double-clicking on the attached HTML file: .



parentspecies

Categorical

Imbalance

| | |
|---|---|
| **Distinct** | 7 |
| **Distinct (%)** | < 0.1% |
| **Missing** | 210 |
| **Missing (%)** | 0.8% |
| **Memory size** | 208.2 KiB |

| | |
|---|---|
| toluene | 17950 |
| apin | 6165 |
| decane | 2218 |
| apin_decane | 46 |
| apin_toluene | 37 |
| Other value... | 11 |

Common Values

| Value | Count | Frequency (%) |
|---|---|---|
| toluene | 17950 | 67.4% |
| apin | 6165 | 23.1% |
| decane | 2218 | 8.3% |
| apin_decane | 46 | 0.2% |
| apin_toluene | 37 | 0.1% |
| apin_decane_toluene | 9 | < 0.1% |
| decane_toluene | 2 | < 0.1% |
| (Missing) | 210 | 0.8% |

Length

Common Values (Plot)

**Figure 1.4:** Overview of the categorical feature "parentspecies", produced by ydata-profiling.

# 2. Preprocessing

## 2.1 Categorical features

The given dataset contains missing values for *"parentspecies"* column only. This column was also the only non numerical feature in the dataset. We used one-hot or dummy encoding to handle this feature. Test data was aligned to match the dummies in train data. We also trained models without this variable as it was not among the important ones when we analyzed the feature importance later.

## 2.2 Feature engineering

Feature engineering or making new features benefits a lot from subject knowledge. As we have little to no knowledge about the context, this part was challenging for us. We used feature importance to find potential features to use to create new ones. Simple ratios like *'MW' / 'NumOfAtoms'* were beneficial throughout the experiments (figure 2.1). Most likely several features could be made similarly with proper knowledge, which would boost the performance significantly. Our best submissions by private and public scores in Kaggle used 2-3 ratios of the most important features, which also seemed to be the sweet spot, as adding more ratios past that gave us negligible model accuracy gains. We noticed improvements with scaling even with LGBM, so we used a standard scaler to scale numerical features into zero mean and unit variance.

## 2.3 Feature tranformations

During the EDA we noticed several variables with skewed distribution. Most notably the *'NumOfConfUsed'* and *'NumOfConf'*. Different methods like squaring and logarithms were tested for features showing skewness. In our final solution, we used log transformation for 'NumOfConf' and binned the 'NumOfConfUsed' into four values. While systematic evaluation of the impact was not conducted, we saw positive results from transformations that were most significant in simpler models like linear regres-

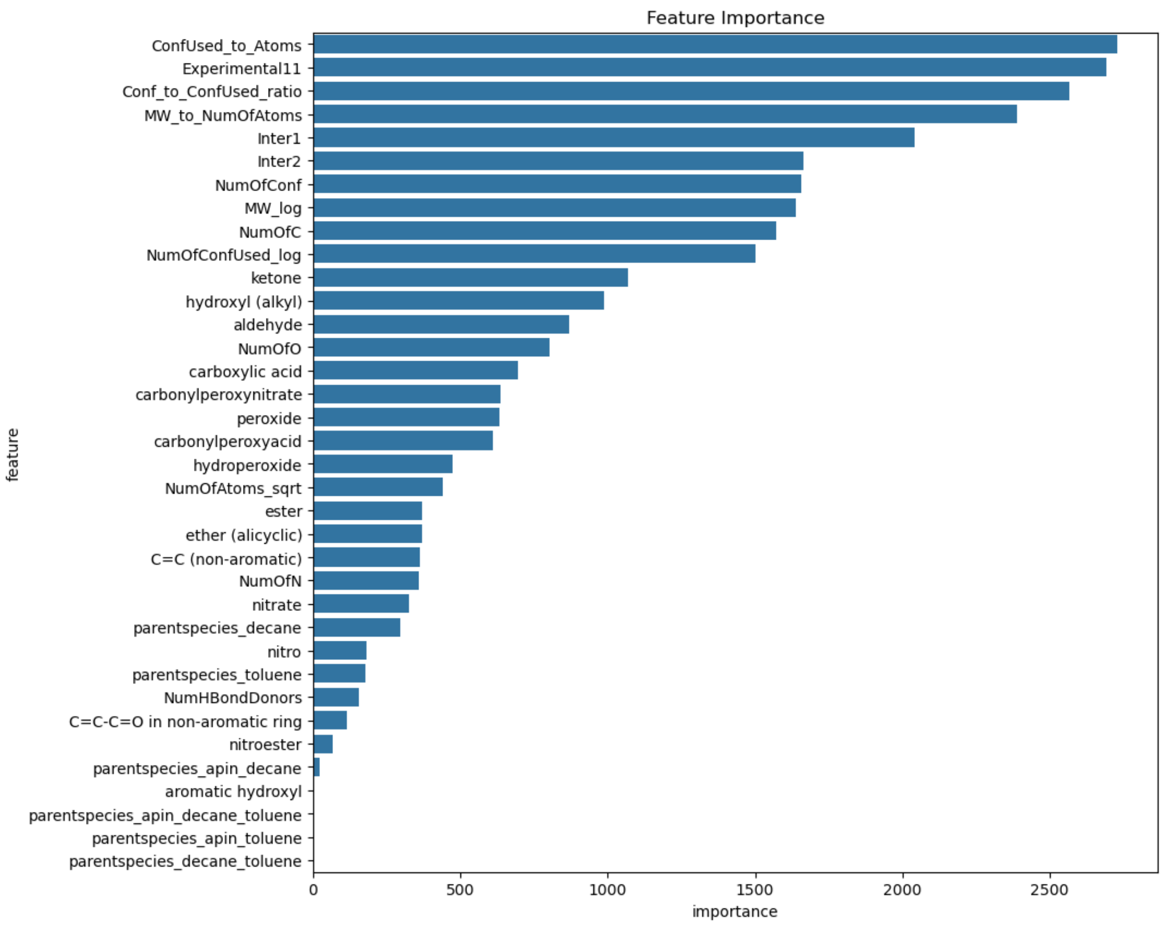


**Figure 2.1:** One of the feature importance plots used during our project, captured from the LGBM training process. As can be seen, custom features based on the ratios of existing important features such as *NumOfConfUsed / NumOfAtoms* and the previously mentioned *MW / NumOfAtoms* performed well by the metric of feature importance.

sion. We also tested PCA with and without additional features but improvement was not gained.

# 3. Modeling

Multiple different approaches were considered in modeling. At first, we evaluated the performance of basic linear regression, RF (Random Forest), XGBoost (eXtreme Gradient Boosting) [2], and LGBM (Light Gradient Boosting Machine) [3]. All of these were tested with similar preprocessing and no additional features. From the initial result, we saw that the out-of-the-box performance was best with the LGBM approach, and it was chosen as the main method for this project. We later evaluated the impact and gains of additional features, different preprocessing steps, and things like PCA to different approaches. LGBM performed best throughout all the experiments. Linear regression out-of-the-box score was increased from 0.7163 -> 0.7390 with transformations and feature engineering. While the gains with LGBM were smaller the starting point was already way better at 0.7427 without hyperparameter tuning. Our best public score 0.7565 was gained by adding transformations, two new features, and hyperparameter tuning. We expected that gradient-boosted decision trees (GBDT) would perform nicely as the nonlinearity was assumed.

## 3.1 Tuning and validation of LGBM

In general, GBDTs are widely used as they offer efficiency, accuracy, and interpretability [3]. LGBM was looking promising from the beginning. It offers speed over many other GBDT methods and is interpretable at least when tree sizes are not huge. One of our models is visualized in figure 3.2, showing how each prediction can be explained with simple steps. LGBM can also be used with little to no preprocessing as it can natively handle categorical features and does not need scaling in general with the training data. It is easy to overfit GBDT, LGBM offers many additional hyperparameters to control the fit and potentially avoid overfitting. We used L1 and L2 regulation terms to avoid overfitting. The tree depth and boosting rounds were also carefully set to reasonable ranges. Ensemble methods and GBDTs are powerful and also suitable for this problem, our decision leaned from XGBoost to LGBM due to the speed and initial result comparisons. However, we expect similar results to be achieved with XGBoost. Overall we wanted to use a method that would be able to handle nonlinearity, would be fast to train, and would also reasonably easy to implement and evaluate.

While the most gains on this problem were found from the preprocessing part, optimal hyperparameters can help avoid overfitting and boost performance. Note that *can* does not mean it will. As our subject knowledge was nonexistent, we tried different approaches to hyperparameter optimization. In the beginning, GridSearch was used, but the computing time started to ramp up quickly, so we moved on to RandomSearch to reduce it. Results from RandomSearch were partially used to define starting ranges for later tuning, reducing the potential search space. After a while, we moved to Optuna [1], which allows more efficient and advanced algorithms to be used in hyperparameter tuning. Optuna was then used in most of our experiments, thanks to the speed and the tools that come with it. Optuna Dashboard*, was used to monitor and log the hyperparameter tuning as an extension to VScode. We used either AutoSampler or a more classic TPEsampler in all experiments. All training and tuning were done with 10-fold Cross-Validation. Mean absolute error (MAE) was used as the main metric during optimization, but $R^2$ and root mean squared error (RMSE) were also evaluated after optimization. We also evaluated the fit by examining the prediction residuals to identify potential outliers and other issues with the fit (figure 3.1). We did not assess the outliers as seen in the residuals. The final submission used the following hyperparameters:

- n estimators: 612
- learning rate: 0.03253420017567533
- num leaves: 123
- max depth: 7
- min child samples: 46
- subsample: 0.7456333280664998
- colsample bytree: 0.9137393037318097
- reg alpha: 0.7026860907781649
- reg lambda: 0.7187304208680759

*https://optuna-dashboard.readthedocs.io/en/latest/getting-started.html

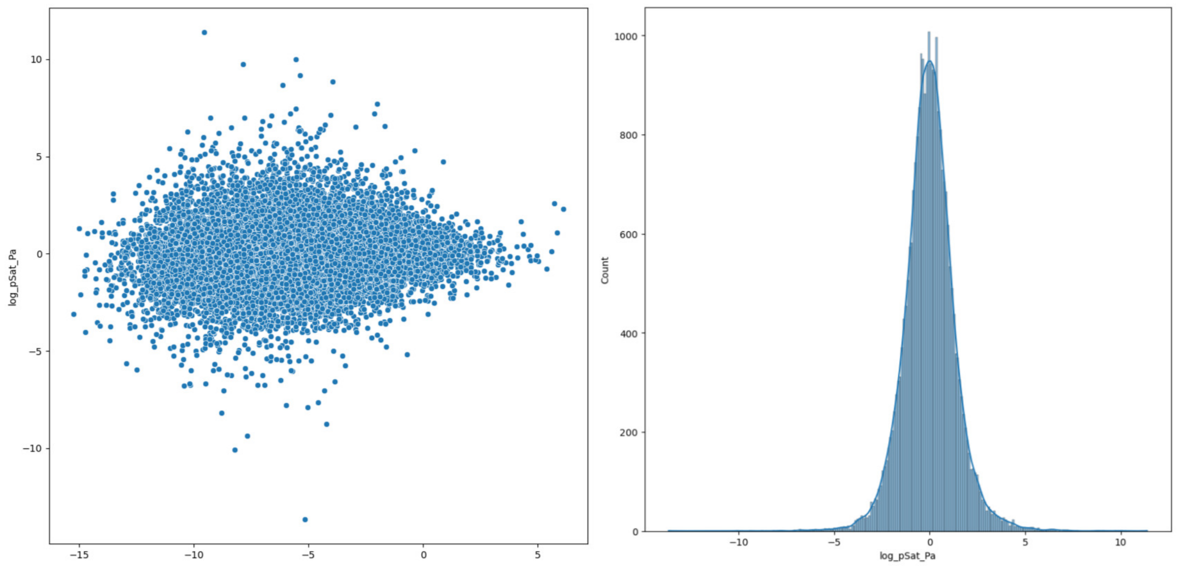


**Figure 3.1:** Evaluation of prediction residuals. Left: Scatterplot of residuals and predicted values. Right: Distribution of the residuals as a histogram.

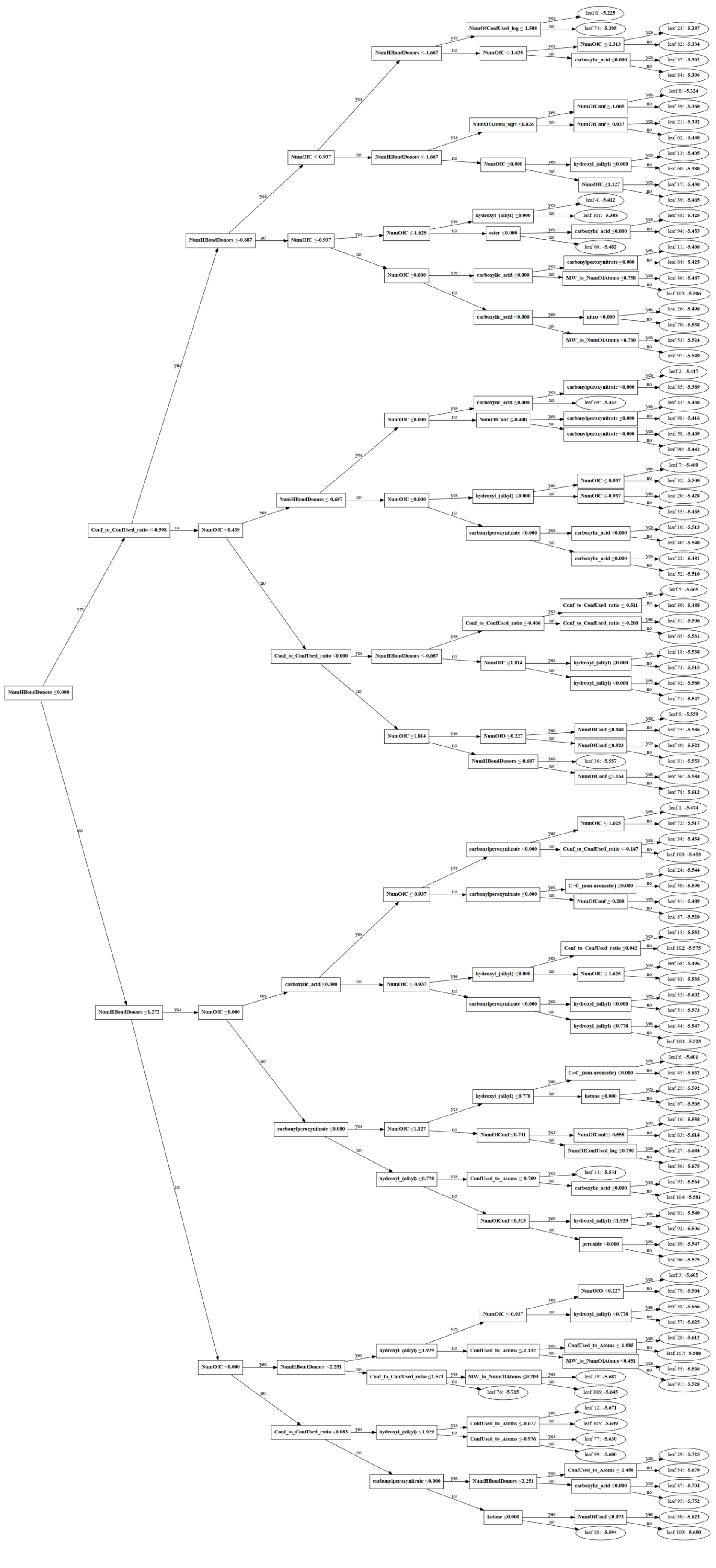


**Figure 3.2:** Visualization of LGBM tree structure.

# 4. Summary

Our final solution was built upon a comprehensive EDA that provided a solid understanding of the data that we have. This information guided the feature engineering, tuning, and modeling. We used proper preprocessing steps and evaluated our solutions and results frequently. Our selected machine learning model LGBM, was carefully tuned and evaluated throughout. We utilized 10-fold Cross-Validation and regularization in validation and tuning ensuring robustness for the model. Our decisions were selected based on our knowledge and insights gained from experiments.

Our submission to Kaggle was chosen based on the public score. It received a public score of **0.7565 (rank 17)** and a private score of **0.7382 (rank 34)**. While the scores are relatively good, they show one of the biggest flaws in our project as the scores indicate overfitting the public leaderboard. The tuning could have been done with generalization in mind, not solely looking at public scores. Even with this it still outperformed the models without additional features and tuning, but was not the best among our submissions.

## 4.1 Discussion and reflection

As a group, we think that the project was successful in many ways. We did not have issues with time and managed our goals so we could do all that we wanted in time. Our project was not perfect and flaws can be pointed out. Most importantly the evaluation of performance in the end could be improved. We both think that we learned a lot during this project and overall are happy with the presented outcomes. Even with logging, keeping the project streamlined and structured all the time was challenging due to the explorative approach. This could be solved by using automated pipelines, we decided to focus on the actual problem as the time was limited.

# 5. Self-Grading

**We propose a grade of 5 for our project and group as a whole.** Our limited in-person group working hours were well utilized, and we managed to adapt meetings accordingly to our needs. The initial planning and estimates on tasks were useful in managing the project. We were able to contribute equally while still working independently and together on parts where it was most beneficial. Overall the collaboration was easy, and we did not face issues with it. Appropriate machine learning methods were applied to the task at hand, and additionally, comprehensive analysis of the results was obtained using multiple visualization methods. The process started off like one would expect without having any knowledge of the problem domain, but we feel like towards the end the methods we decided to stick with were well justified and appropriate for the problem. Despite the challenges of finding time for in-group sessions, we feel like the group work positively enhanced the learning experience. After the presentations, we could see that more effort could have been put into exploring simpler, basic approaches like lasso and linear regression. The time used towards the hyperparameter tuning could have been used for this. However, we used the best approach we could, and doubt that we would've figured out the polynomial regression with regularization even with more exploration. While not as elegant and simple, our approach still shows appropriate steps to handle nonlinearity, skewness, and other issues found in the data. We utilized our prior knowledge on similar projects by including monitoring and tuning with Optuna and automatic data profiling in EDA.

# References

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[2] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785â794, New York, NY, USA, 2016. Association for Computing Machinery.

[3] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3149â3157, Red Hook, NY, USA, 2017. Curran Associates Inc.