Master's Programme in Data Science

Term project IML2023

January 15, 2024

University of Helsinki
Faculty of Science

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

Contents

1. Exploratory Data Analysis

Exploratory data analysis can be considered numerical and graphical "detective work". It involves examining data to find out its implications. Moreover, it is about learning what could be done (Tukey et al., 1977).

Before delving into our detective work, it is important to note that we employed Python, a popular programming language for data analysis, owing to its powerful and efficient libraries, including *pandas*, *numpy*, *matplotlib*, *seaborn*, and *scikit-learn* (McKinney, 2017). *pandas* and *numpy* were utilized for data handling and manipulation, while *matplotlib* and *seaborn* served as visualization libraries. Additionally, *scikit-learn*, a machine learning library, will be discussed in greater detail later in this report.

Our data analysis started with an exploratory process where we familiarized ourselves with the task and the data set. Each of us read the task description and additional relevant materials. Also, we agreed to look at the data set before our initial meeting. The reasoning behind this was to understand the characteristics of the data and our goals. For instance, we summarized and described the features of the data set.

Even though the exploratory process is the first step and the cornerstone in data analysis (Tukey et al., 1977), it is also an iterative process. Even though we conducted our exploratory data analysis well initially, working with the data set and testing different machine learning approaches provided essential insights to understand things better, enabling us to polish or change our approaches.

One of the most crucial tasks in data analysis is creating informative visualizations. During the exploratory process, visualization can help, for example, in identifying necessary data transformations (McKinney, 2017). As depicted in Figure 1.1, the data in the *parentspecies* column is categorical rather than numerical, with missing values. Although these aspects were introduced in the task description, a simple data visualization proves beneficial. However, the most significant value derived from a figure is when it conveys information that was not anticipated (Tukey et al., 1977). For example, illustrating the column-to-column correlation as a correlation heatmap (Figure 1.2), provided insights into the relationships between the variables. Additionally, Figure 1.3 offered a more comprehensive depiction of the correlation between the *pSat_Pa_log* and other features.

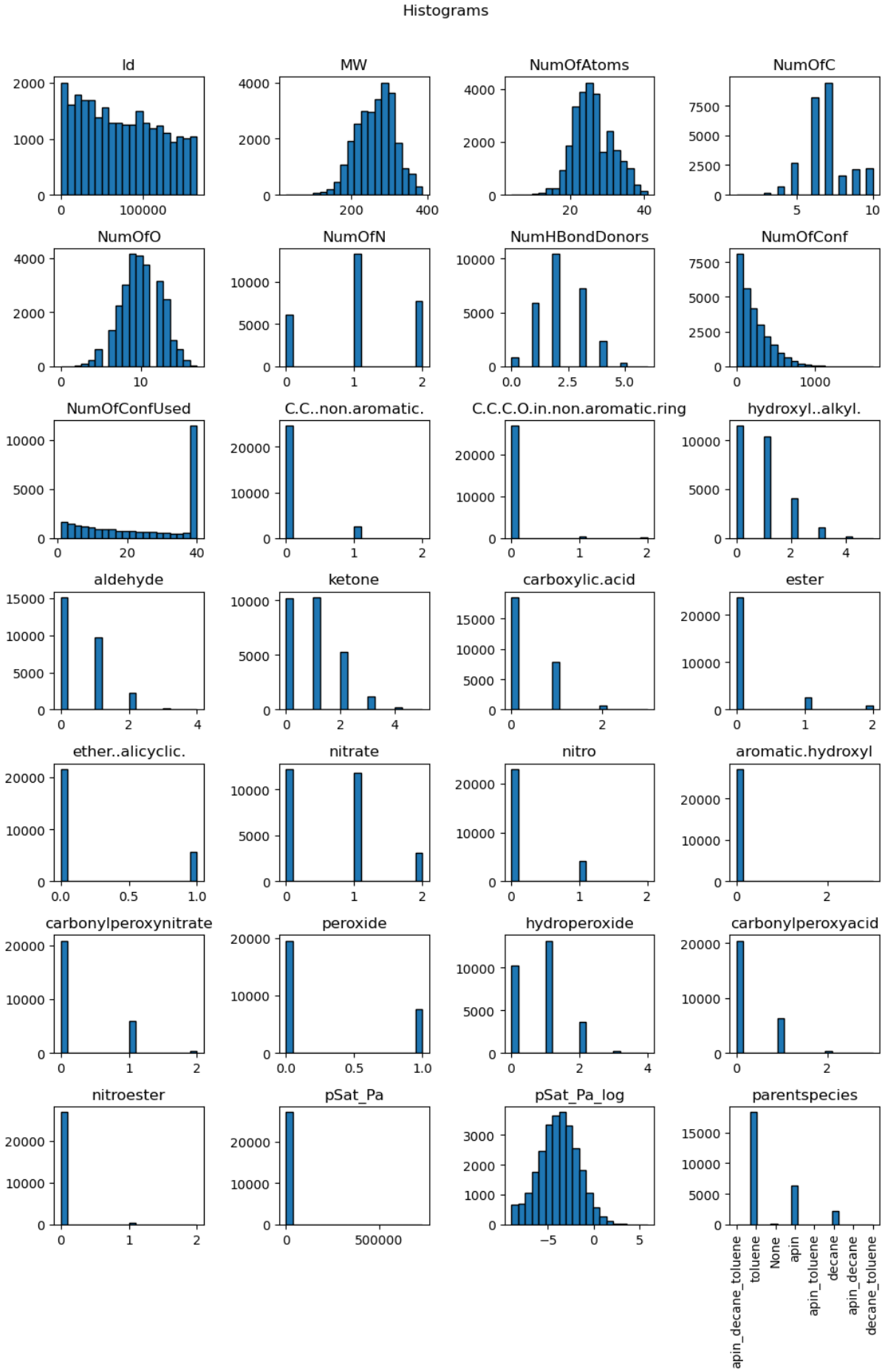

Figure 1.1: Visualizing distributions of feature values with histograms.

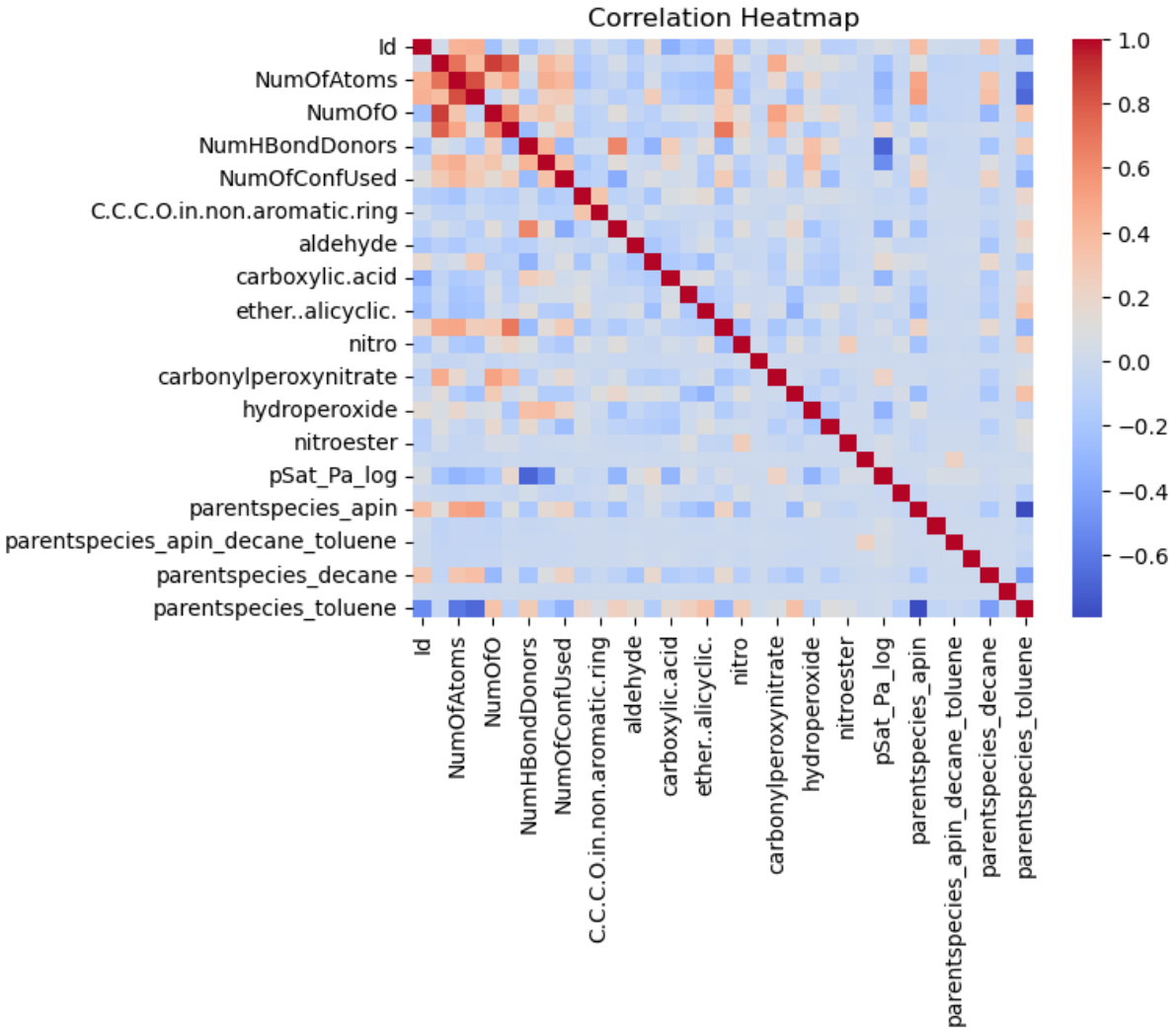

Figure 1.2: Correlation heatmap showing a correlation between the features. As the color gets darker (red or blue), correlation strengthens (positive or negative).

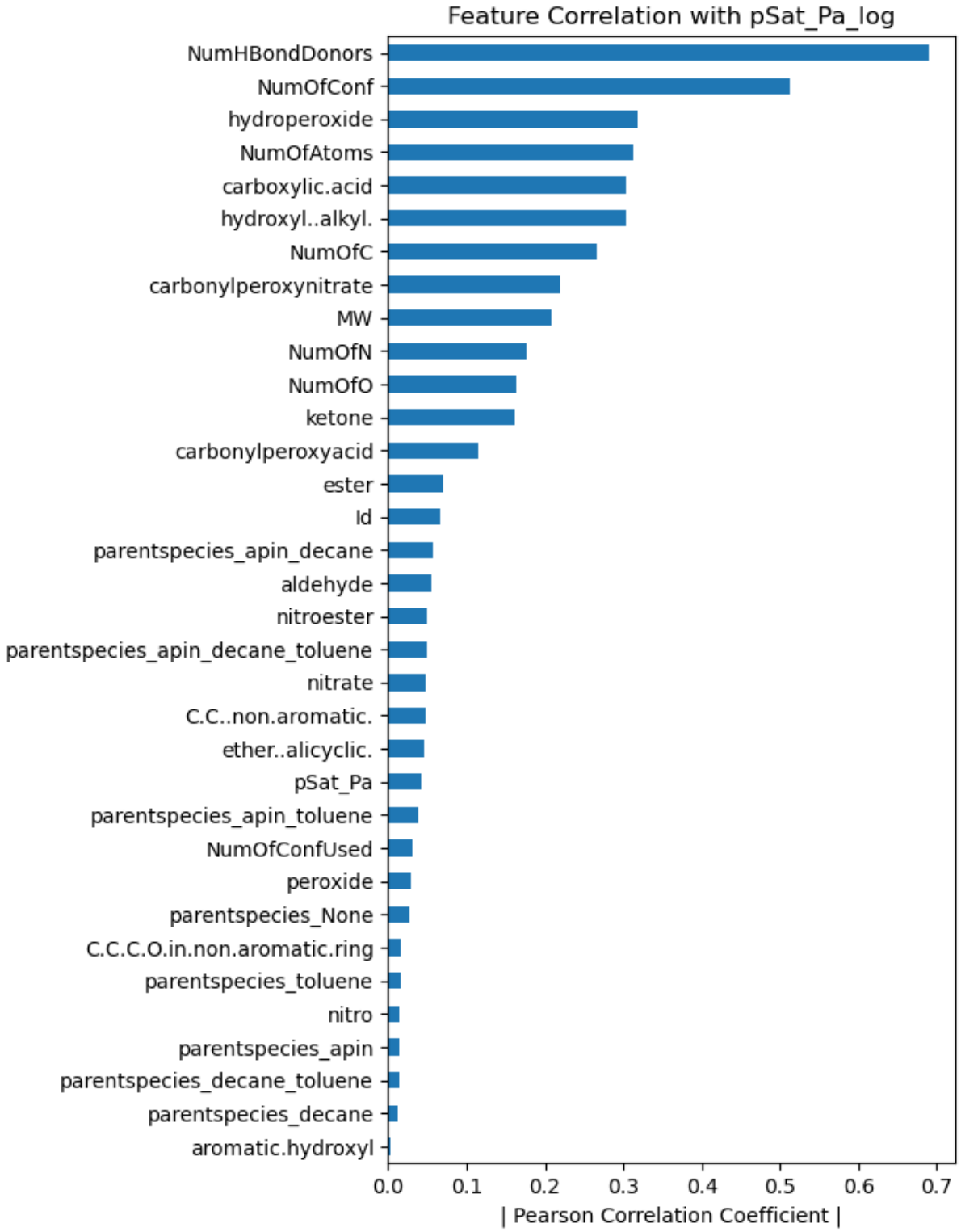

Figure 1.3: Feature correlation showing the absolute values of Pearson correlation coefficients between *pSat_Pa_log* (logarithm with base 10 of *pSat_Pa*) and other features.

2. Preprocessing

Preprocessing influences the model's accuracy as data-driven models use historical data to predict future outcomes. Due to this, if data contains many outliers or "noisy" data, the model might perform poorly (James et al., 2023).

For preprocessing, we first looked at missing values. There appeared to be only missing values in the *parentspecies* column. As mentioned in the EDA section, parent categories are categorical columns. We decided to do one hot encoding for the species to ensure we can use this column in our machine-learning model. One hot encoding is a method used to transform categorical data into a numerical format that machine learning models can effectively utilize (James et al., 2023). It creates new binary columns for each category, ensuring no implied order. However, this technique can increase the dimensionality of the data set, potentially leading to issues like the "curse of dimensionality" in some cases (James et al., 2023). Since we only had seven unique parent species, we decided to move forward with one hot encoding.

After this step, we standardized the data set, where each feature was scaled to have a mean of zero and a standard deviation of one. This process, known as feature scaling or Z-score normalization, ensures that all features contribute equally to the model's performance and are not skewed by their original scale (James et al., 2023). Most of the machine learning approaches we tested benefit or require standardization.

Since it is known that outliers or "noisy" data can lead to poor performance in machine learning algorithms, outlier detection was employed as a step in our process. The strategy was to apply z-score validation to our dataset using a specific threshold. This method identifies outliers by measuring how many standard deviations an element is from the mean (Yoseph et al., 2019). Data points that exceed our set threshold are considered outliers and are treated accordingly. This approach helps clean our data set, ensuring that our machine-learning models are trained on more accurate and representative data. This step is crucial for algorithms like the random forest regressor, which can be sensitive to extreme values in the data (James et al., 2023). After utilizing this method, we found that no improvements were made when we trained our machine-learning algorithms. So the idea was scrapped.

Our preprocessing efforts showed that many techniques had minimal impact. This was mainly because our limited domain knowledge made it difficult to improve the dataset through methods like feature engineering or feature removal. Using correlation analysis for feature elimination didn't capture the complex relationships between features and the target variable. Removing features based on correlation and importance indicators from the random forest model resulted in poor performance rather than improving the model. This experience emphasized the value of domain knowledge in data preprocessing and

feature selection.

3. Machine Learning Approaches

3.1 Exploring Regression Models

Following our detailed exploratory data analysis (EDA) and preprocessing, we are now set to dive into the machine-learning aspect of our project. Choosing the proper machine-learning technique largely depends on the characteristics of our data (James et al., 2023). We'll focus on regression models since we aim to predict a continuous target variable, *pSat_Pa*. Each group member selected a machine algorithm they wanted to learn and apply. The machine learning algorithms we used were from Scikit-learn (Pedregosa et al., 2011), which provides simple and effective open source algorithms with Python. We discussed the results each week and tried to help or give suggestions. All the code was made through Google Colab, where group work was easy.

We began by implementing baseline models like Dummy Regressors. This step was crucial for setting a basic performance benchmark before progressing to more advanced regression algorithms (James et al., 2023). As anticipated, these initial models yielded low performance, which is understandable given their simplicity. We aimed to start with more interpretable models such as Linear Regression, Ridge, Or Lasso Regression. Moreover, if the simpler models were not flexible enough, we would move into more flexible models such as Random Forest and Gradient Boosting Machines. K-Fold Cross-validation was employed with each approach to avoid overfitting, which will be discussed more in later sections.

3.2 Linear Regression

3.2.1 Ordinary Least Squares

In Ordinary Least Squares (OLS) for linear regression, a linear relationship between input and output variables is assumed, and errors are expected to be randomly distributed with a uniform variance (James et al., 2023). Enhancing model efficiency involves removing outliers or non-predictive elements, such as the identifier column (*Id*), and managing influential variables like categorical ones (*parentspecies*). These steps minimize their skewing effect on regression results, which otherwise lead to poorer performance. By aligning the dataset more closely with OLS assumptions through these adjustments, a better model fit and more reliable predictions were achieved.

3.2.2 Ridge

Then, Ridge regression was used to eliminate features that affected the prediction since multicollinearity existed in the data. Ridge regression shrinks coefficients toward zero but never reaches zero (James et al., 2023). Selecting those coefficients that were higher than 0.001 resulted in 24 features into a selection. This was then applied to the OLS model, resulting in a worse prediction than only OLS. However, the R2 score was very close to that of OLS. Ridge regression got a score of 0.65114, and OLS got a score of 0.65163. In Figure 3.1, we can see the importance of each numerical feature-based ridge regression.

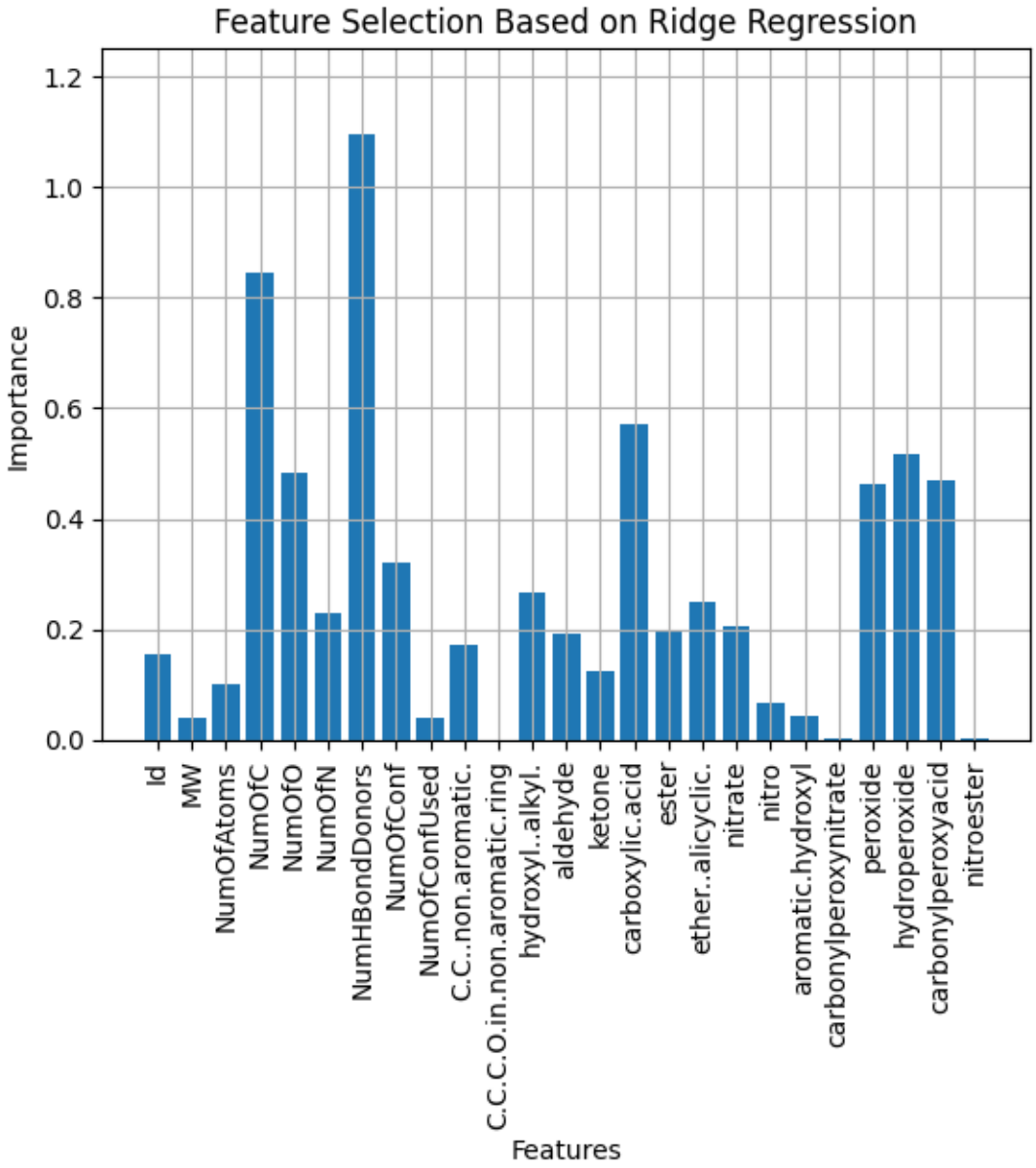

Figure 3.1: Feature selection based on Ridge regression.

3.2.3 Lasso

Lasso was applied to analyze feature selection, setting coefficients of certain features to zero under a large penalty, thus excluding them from the model (Figure 3.2). This doesn't imply the removed features are uninformative; instead, it's part of Lasso's regularization to prevent overfitting (James et al., 2023). The process selected 6 features, yielding a prediction score of 0.62, lower than the 0.65 achieved with OLS. This decrease may be due to information loss from excluded features and the trade-off between model simplicity and predictive accuracy inherent in Lasso's regularization approach.

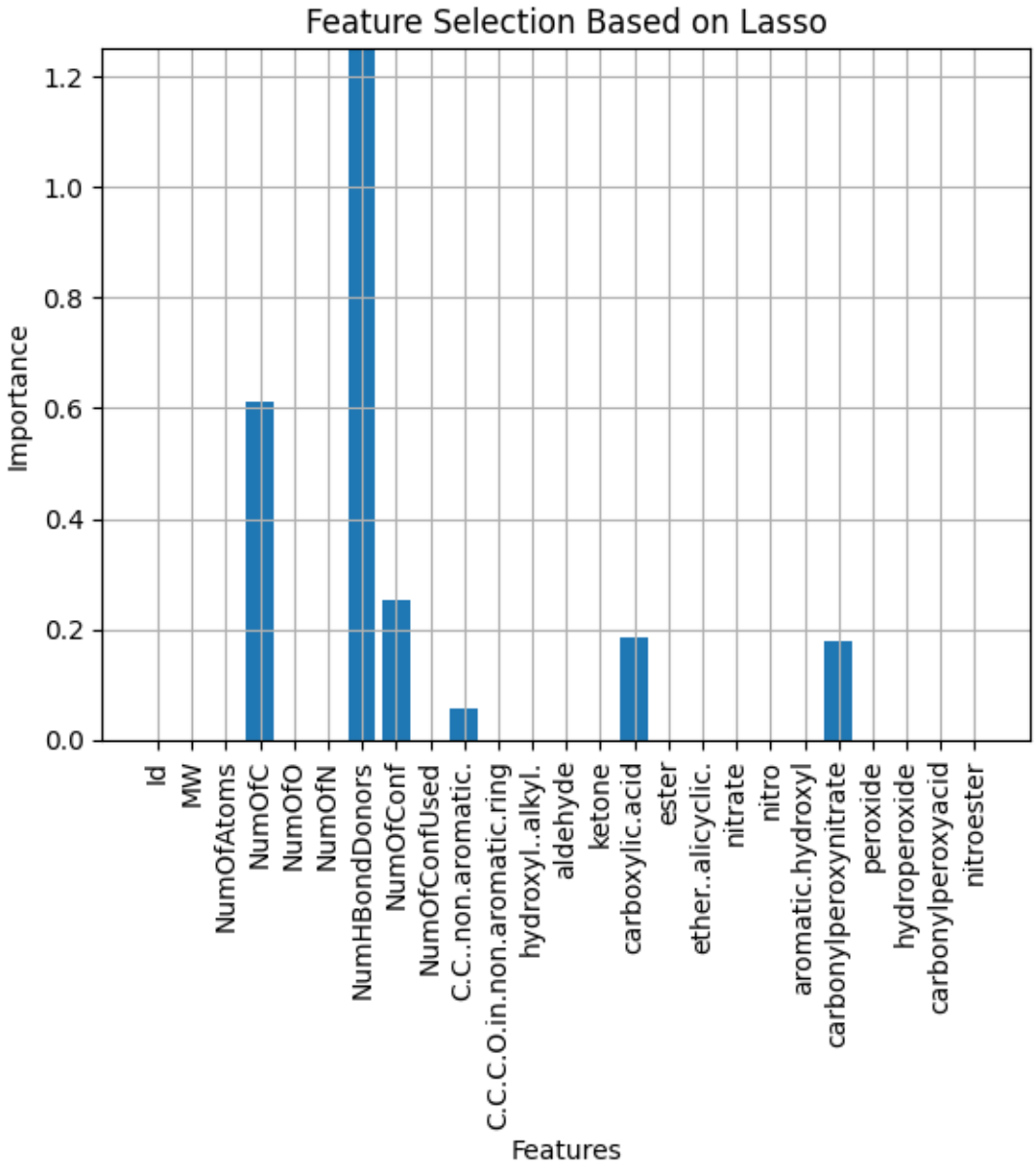

Figure 3.2: Feature selection based on Lasso regression.

3.3 Support Vector Regression

One of the models we tried out was Support Vector Regression (SVR), a variant of Support Vector Machines (SVM) adapted for regression. SVR is sensitive to the scale of the input features (James et al., 2023), so standardization was used with SVR. We tested different features and parameters, such as kernels, margin sizes, and kernel coefficients. The best R2 score with SVR in the Kaggle competition was 0.6682 (with a corresponding public R2 score of 0.677). The model used a radial basis function (RBF) kernel with all features except the column *Id*. However, we did not find a model that could outperform the approach below.

3.4 Random Forest Regression

We then shifted our focus to a random forest regressor. This algorithm strongly fits our data, evidenced by the high R2 score on the test data set. We chose this model for its straightforwardness, making it easier for us as students to understand, and because it aligned with our preference for simpler models, in contrast to more complex ones like Neural Networks. The random forest regressor was selected due to its excellent performance and relative ease of interpretation. Our next objective was to enhance our algorithm through hyperparameter tuning. This process involved identifying the optimal combination of hyperparameters for training our model (James et al., 2023). We employed GridSearch, which systematically tests every possible parameter combination within a grid (James et al., 2023). This approach resulted in a noticeable improvement in the performance metrics of our model. Random forest after hyper-tuning resulted in an average of 0.74337 R2 score with 10 cross-folding. When evaluated in the Kaggle, it resulted in a public R2 score of 0.6905 and a private R2 score of 0.709. Interestingly, the random forest regressor had the best R2 scores, considering all the features.

3.5 PCA

To refine our models further, we employed Principal Component Analysis (PCA) to reduce the dimensionality of the data while retaining as much information as possible (James et al., 2023). However, when combined with Linear Regression or Support Vector Regression, these approaches led to diminished predictive results due to the information loss inherent in reducing dimensions. Also, before implementing PCA, we normalized our data using a Scaler. This step is crucial because PCA aims to maximize the variance of each component (James et al., 2023). Without scaling, given the high values in the

original data, one component might dominate in explaining most of the data, which is not desirable (James et al., 2023). Hence, using PCA without normalization typically results in poorer performance than when normalization is applied, regardless of the model used.

3.6 Validation

To compare and select a model for predictions, we used 5-fold and 10-fold cross-validation. We wanted to be sure that our model did not overfit, so cross-validation was employed. The entire dataset will be trained and validated in cross-validation, ensuring our model does not overfit or become biased. By retaining evaluation scores, we could confirm that the model has relatively identical scores in each iteration. Thus, it is stable. Using 5-fold cross-validation was an approach that we considered after assessing that data is large enough to be split into 5, where 20% of data was used for validation and the rest for training. However, using 10-fold cross-validation leads to less bias than using 5-fold validation since 10-fold validation will use 90% of the data for training and the rest for testing.

To assess the accuracy of our models in each cross-validation iteration, Root Mean Squared Error was used to quantify the average distance between the predicted and actual values in the validation set (James et al., 2023). As another approach to assess the accuracy of the predictions, a mean squared error, which is the average squared error between predicted and actual values, was employed (James et al., 2023). Subsequently, to estimate the model fit, we used the Residual Squared-score, which tells us how much of the variation in the dependent variable is explained by the independent variables (James et al., 2023).

4. Summary

Our final model was developed through a comprehensive process that began with Exploratory Data Analysis (EDA) to understand and prepare our data. Next, we applied one-hot encoding to transform categorical variables into a format suitable for modeling. The core of our model is a Random Forest Regressor, which we carefully fine-tuned. This tuning involved adjusting multiple parameters using GridSearch, a method that helps find the most optimal parameters for our model. We employed a 10-fold Cross-Validation strategy during the hyperparameter search to ensure the model's robustness and avoid overfitting. This approach ensures that our model is accurate, reliable, and efficient when running inference.

The reason why random forest might have worked better than other simpler models is due to handling non-linearity. As previously concluded, the dataset is complex between molecular properties and vapor pressure and is likely non-linear. Random forests are generally adept at handling non-linear relationships without much data transformation.

Also, the random forest regressor did not require standardization. Our best-performing model did not use standardization. We also tried random forests with scaled features, but it did not yield better results. Our standardization technique was improperly implemented, thus negatively affecting the tested approaches.

4.1 Discussion and self-reflection

We met before writing this section, where the team reflected on the entire project. The learning curve was relatively high since we started from understanding how each machine learning algorithm worked to employing these algorithms and then to actual data. However, we got all the algorithms to work for the dataset after trial and tribulations.

The team felt that preprocessing was the hardest part. We did not know how to effectively preprocess such data when we did not know what the features meant. Upon self-reflection and the last mandatory lecture, we understood that we could have done more domain research to understand the data better.

To improve our results, we could have combined our machine learning models or "stacked" them to gain more accurate results, as many did in the mandatory lecture. This led to an open-ended question of whether such an approach would be practical, especially for only a few percent extra accuracy, while losing the interpretability of the model.

In summary, the team learned a lot from when we started the project to after submitting our final results. Even though we faced many issues and a high learning curve, with the help of teamwork, we managed the project well in the end.

5. Grading section

We propose a grade 5 for us. This proposal is based on an organized and professional approach to the given task. The topic was outside of our previous knowledge but with proper EDA, preprocessing, and different machine-learning approaches, we got fairly good results quickly. If we continued with the project, we would have a clear plan of how to proceed, a great team with good communication methods, and the work done so far supporting us.

The Kaggle competition did not treat us properly. We ended up to position 90. with our random submission *submission.csv* having a private R2 score of -1.4567 and a public R2 score of -1.3677. However, our best R2 score was with submission *RF_HYPER.csv* having a public score of 0.6905 and a private score of 0.709. Therefore, we should have positioned as 19. if we look at the leaderboard on the Kaggle competition. We contacted Gauri Pradhan and Kai Puolamäki about this; fortunately, this should not affect our grading.

5. Bibliography

James, G., D. Witten, T. Hastie, R. Tibshirani, and J. Taylor (2023). "Introduction". In: *An Introduction to Statistical Learning: with Applications in Python*. Cham: Springer International Publishing, pp. 1–13. ISBN: 978-3-031-38747-0. DOI: 10.1007/978-3-031-38747-0_1. URL: https://doi.org/10.1007/978-3-031-38747-0_1.

McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Incorporated. URL: https://ebookcentral-proquest-com.libproxy.helsinki.fi/lib/helsinki-ebooks/detail.action?docID=5061179.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Tukey, J. W. et al. (1977). *Exploratory data analysis*. Vol. 2. Reading, MA.

Yoseph, F., M. Heikkilä, and D. Howard (2019). "Outliers Identification Model in Point-of-Sales Data Using Enhanced Normal Distribution Method". In: *2019 International Conference on Machine Learning and Data Engineering (iCMLDE)*, pp. 72–78. DOI: 10.1109/iCMLDE49015.2019.00024.