# IML2024 Term project Report

Patrik Keinonen, Mikko Kallio, Tuuli Toivanen-Gripentrog

2024-12-05

## Introduction

Pate hoitaa

Kirjoita millaiseen malliin päädyttiin, mitkä parametrit ja featuret jne.

**Final Kaggle Score**

# Feature engineering

### Feature correlations

We analyzed the different features and how they correlate between the target column log_pSat_Pa.
We analyzed the correlation values and listed the features which have the strongest correlation with the
target as well the strongest correlation pairs among the features:

```
## Strongest correlation between the target:
## NumHBondDonors          0.689196
## NumOfConf               0.513653
## hydroperoxide           0.314053
## hydroxyl (alkyl)        0.310452
## NumOfAtoms              0.307337
## carboxylic acid         0.304259
## NumOfC                  0.262769
## carbonylperoxynitrate   0.223739
## MW                      0.199574
## NumOfN                  0.183152
## Name: log_pSat_Pa, dtype: float64

## ---------------------

## Strongest correlation pairs:
## NumOfO                MW                0.880358
## NumOfC                NumOfAtoms        0.838402
## NumOfN                MW                0.772575
## NumOfAtoms            MW                0.707009
## nitrate               NumOfN            0.687224
## NumOfN                NumOfO            0.656750
## hydroxyl (alkyl)      NumHBondDonors    0.632023
## carbonylperoxynitrate NumOfO            0.510409
## nitrate               NumOfAtoms        0.492108
## NumOfN                NumOfAtoms        0.491902
## dtype: float64
```
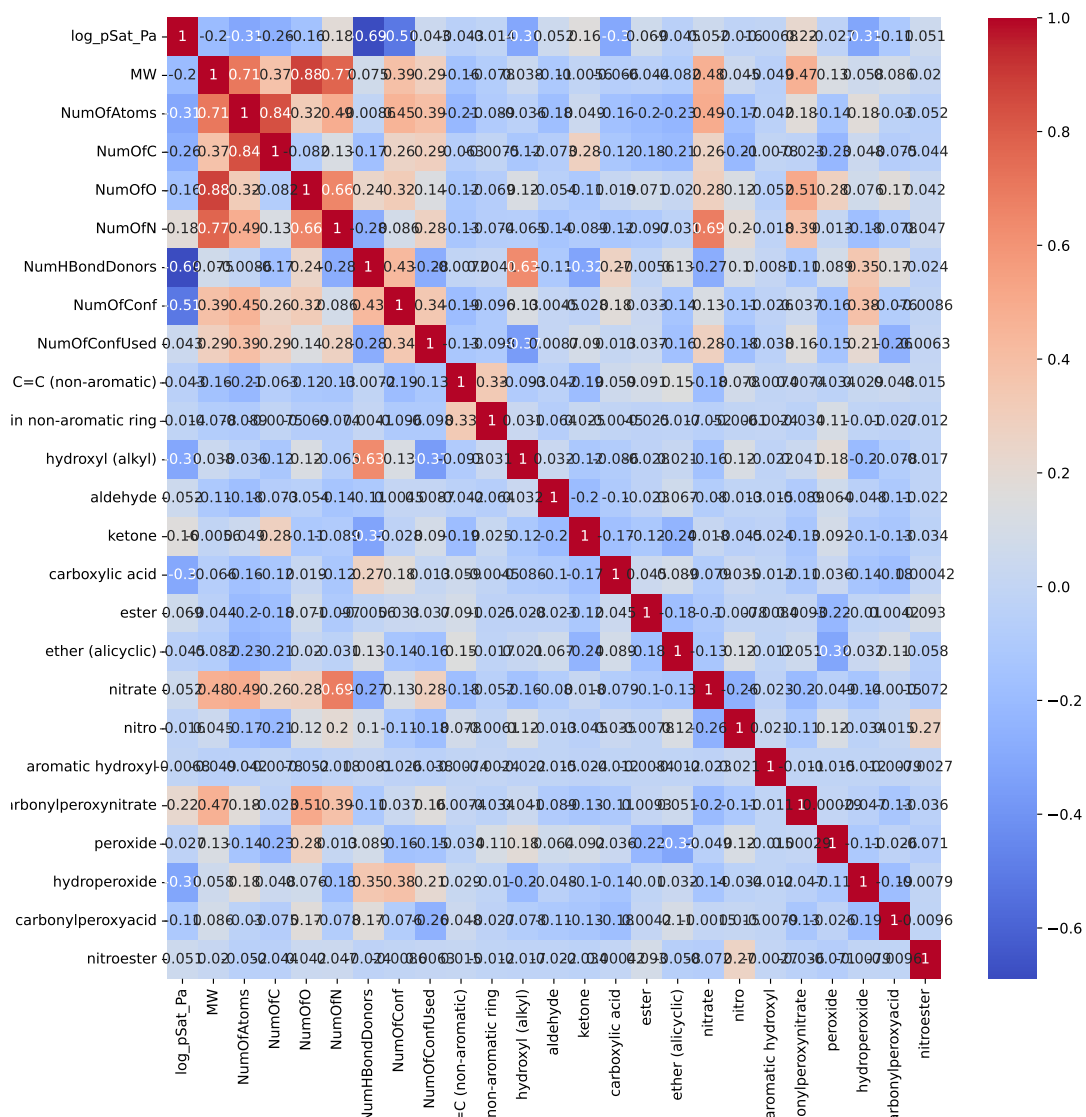
We did not get good results by only selecting a subset of features based on the correlations above. We tried for example to select only the 5 most correlated variables and exclude others as well as drop some of features which were highly correlated with some other feature. This did not improve the models performance. Hence, we moved on to some more advanced feature engineering techniques.

## Data Cleaning:

Replace infinite values and missing data with zero. This ensures the dataset is well-suited for downstream modeling by preventing issues with invalid numerical values.

## Creation of Derived Features:

We quickly realized lacking domain knowledge in chemistry, the competence required analysing molecular data. The principal idea is to compose *meaningful features*, something we were inspired by the discussion

at *Bayesian Data Analysis* course in the context of GLM (generalized linear models). After studying the university guidelines provided in the course page, we interpreted the rules such that we are able to use an external AI consultant. Hence, providing a listing of the features with descriptions to ChatGPT, we asked what formulas in chemistry uses the given covariates as parameters. We received a listing of formulas which we then implemented in the feature engineering phase.

- AtomFraction: Proportion of carbon, oxygen, and nitrogen atoms in the molecule relative to the total atom count.

- Polarity: Ratio of hydrogen bond donors to molecular weight.

- HBondDensity: Ratio of hydrogen bond donors to the total number of atoms.

- GroupDensity_CarboxylicAcid: Concentration of carboxylic acid groups per unit molecular weight

- Unsaturation: Sum of non-aromatic double bonds and C=C-C=O bonds in non-aromatic rings.
  `df['Unsaturation'] = df['C=C (non-aromatic)'] + df['C=C-C=O in non-aromatic ring']`

- ConfigurationalComplexity: Ratio of the number of stable conformers to molecular weight.
  `df['ConfigurationalComplexity'] = df['NumOfConf'] / df['MW']`

- HydrogenBondPotential: Total potential for hydrogen bonding based on hydrogen bond donors and nitrogen/oxygen atoms.
  `df['HydrogenBondPotential'] = df['NumHBondDonors'] + (df['NumOfO'] + df['NumOfN'])`

- PolarGroupCount: Count of polar functional groups.
  `polar_groups = ['hydroxyl (alkyl)', 'aldehyde', 'ketone', 'carboxylic acid', 'ester', 'nitro']`

- AromaticGroupFraction: Proportion of aromatic hydroxyl groups relative to all polar groups.
  `df['AromaticGroupFraction'] = df['aromatic hydroxyl'] / (df[polar_groups + ['aromatic hydroxyl']].sum(axis=1) + 1e-9)`

- MolecularSize: Combined measure of molecular weight and atom count.

- Hydrophobicity: Ratio of carbon atoms to the total atom count.

- DegreeOfUnsaturation: Measure of molecular unsaturation based on carbon and atom counts.

- ShapeCompactness: Ratio of atom count to the number of conformers used in calculations.

- SymmetryIndex: Ratio of symmetric to asymmetric functional groups.

- VolatilityIndex: Proxy for molecular volatility based on carbon-to-polar group ratio and molecular weight.

- PolarityIndex: A combined measure of molecular polarity.

- OxygenToCarbonRatio: Relative proportion of oxygen to carbon atoms.

## Interaction Features:

To capture non-linear relationships, several interaction terms were introduced. We experimented with Sklearn `PolynomialFeatures(degree=2, intercation_only=True)`. This resulted in a convoluted dataset. Principal component analysis was also employed to reduce the complexity of the model. However, we defaulted into a more crude, manual process by defining the interactions by hand using semi-brute-force strategy. Hence, we tried different interactions such as:

- PolarityIndex * FlexibilityRatio
- ShapeCompactness * FlexibilityRatio

in order to model potential synergistic effects between molecular properties.

### Feature transformations

As part of the preprocessing pipeline, we systematically explored various mathematical transformations of the input features to improve the model's performance. Transforming features can help normalize distributions, reduce the influence of outliers, and capture nonlinear patterns.

We developed a function to iterate through features and apply transformations based on a predefined set of transformation functions. Using a brute-force approach, we systematically evaluated each transformation for all features by assessing their impact on model performance.

**Transformation Functions Explored:**

- Logarithmic (log): Compresses large values and expands small ones, effective for reducing skewness.
- Square root (sqrt): Reduces the magnitude of large values while preserving smaller ones.
- Square (square): Amplifies differences in values, highlighting larger magnitudes.
- Cubic (cube): A stronger version of the square transformation to capture higher-order relationships.
- Exponential (exp): Expands small differences in values, useful for features with compressed scales.
- Reciprocal (1/x): Inverts the scale, useful for features with large values that need downweighting.
- Box-Cox: A parameterized transformation that aims to stabilize variance and make data approximately normal (applied only for strictly positive features). [1]
- Yeo-Johnson: Similar to Box-Cox but works with both positive and negative values. [2]

## Trying out different models

We evaluated several models from different families listed below.

- Dummy regressor
- Linear Regression
- Random Forest
- Gradient Boosting Regressor
- Epsilon-Support Vector Regressor SVR [3]

We observed that most promising results were produced by SVR and GBR models, where SVR out-performed that GBR with a small margin given a few different parameterizations. Given the limited amount of time allocated for this project, we decided to commit to SVR without any further analysis.

## Evaluate model with different parameters and features

Engineering of Machine Learning course covers a broad set of tools to practise professional, production level ML system management. Tuning the parameters of a model is a tedious task, which fortunately can be automatized. We implemented a model tuning infrastructure on top of Optuna [4]. Optuna was left running for hours traversing in the parameter space towards (some) optimal set of parameters. Using an iterative process of alternately exploring feature engineering and running hyperparameter tuning in Optuna, we found an improved combination of features and hyperparameters.

No experiments can be done without an adequate measure of model performance. We considered using either separate train-validation split or use cross-validation. After exploring both avenues, because of the benefits discussed in the IML textbook [5], we decided to move forward with cross-validation with five folds.

## References

[1]     "Box Cox Transformation: Definition, Examples," *Statistics How To.* Accessed: Dec. 05, 2024. [Online]. Available: https://www.statisticshowto.com/probability-and-statistics/normal-distributions/box-cox-transformation/

[2]     kjytay, "The Box-Cox and Yeo-Johnson transformations for continuous variables," *Statistical Odds & Ends*. Feb. 2021. Accessed: Dec. 05, 2024. [Online]. Available: https://statisticaloddsandends.wordpress.com/2021/02/19/the-box-cox-and-yeo-johnson-transformations-for-continuous-variables/

[3]     "SVR," *scikit-learn*. Accessed: Dec. 05, 2024. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.svm.SVR.html

[4]     "Optuna - A hyperparameter optimization framework," *Optuna*. Accessed: Dec. 05, 2024. [Online]. Available: https://optuna.org/

[5]     G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor, *An Introduction to Statistical Learning: With Applications in Python*. in Springer Texts in Statistics. Cham: Springer International Publishing, 2023. doi: 10.1007/978-3-031-38747-0.