Advanced Data Analysis Project

Spectral Dataset

Week 3

LUT University

Alejandro Lopez Vargas
Sina Saeedi
Pau Ventura Rodríguez

Visit our GitHub to check the script.

Modelling Goal

The goal of the project is to compare the performance of various regression techniques, specifically Multiple Linear Regression (MLR), Principal Component Regression (PCR), Partial Least Squares (PLS), and k-PLS (Kernel Partial Least Squares) for predicting some plant traits using Spectral dataset that includes wavelength variables.

Let's discuss each of these techniques and modeling approaches:

1. Multiple Linear Regression (MLR): MLR is a traditional regression technique that aims to establish a linear relationship between the predictor variables (in this case, wavelength variables) and the target variable.
2. Principal Component Regression (PCR): PCR combines Principal Component Analysis (PCA) with MLR. First, we perform PCA on the wavelength variables to reduce their dimensionality while retaining most of the information. Then, apply MLR to the reduced set of Principal Components (PCs) instead of the original wavelengths. The number of PCs we decide to retain can impact the model's performance.
3. Partial Least Squares (PLS): PLS is a regression technique that aims to find latent variables (also known as factors) that explain the maximum variance in both the predictor and target variables. By selecting an appropriate number of latent variables, not only we build a predictive model but also achieve dimensionality reduction.
4. Kernel Partial Least Squares (k-PLS): k-PLS is an extension of PLS that employs kernel methods for non-linear regression. Similar to PLS, it aims to find latent variables that capture the relationships between predictor variables (wavelengths) and the target variable. It can be particularly useful when the relationship between variables is non-linear.

Model Calibration, Validation, and Testing Strategy

The strategy that we are going to use for this task, contains the following parts:

1- Data Preparation:

Data Cleaning: This involves handling missing values, outliers, and any data inconsistencies. For that purpose, we can use data-cleaning libraries in Python (e.g., Pandas)

For handling Missing values we have used python missingno library and have found that there are not null values on the input wavelength, which is great news as we can ensure consistency on our dataset. On the traits (response variables) however, we do have a large amount of missing values (some even reaching higher than 90%). This matter will be fixed creating different datasets, one for each trait, as explained in the following sections.

For detecting outliers, we have used multivariate methods such as PCA T-square vs SPEx charts to detect those samples that are outside of the expected range of values. Although this has led us to find irregular samples, we cannot directly drop them as we would need an expert's point of view in the spectral data matter to tell us if those were just a result of mistakes on measurements or just extreme values.

Data Normalization: Ensure that wavelength variables are on a consistent scale. StandardScaler could be used in this part to subtract mean and scale using standard deviation to give each feature the same importance.

2- Data Splitting:

We will divide our dataset into three parts: Calibration, Validation, and Test. The calibration data is used for training and calibrating the models.

The validation data is reserved for assessing model performance during the calibration phase. We will set aside a separate test dataset, which is not

used during model calibration or validation, for the final evaluation of model performance.

Usually, 80% of the data is used in the calibration phase, 10% in validation, and 10% of the data is reserved for testing. Since we created one dataset for each trait, we will have 20 train, validation, and test datasets in total.

The main reason after creating a dataset for each trait is that we have multiple null values on the traits. It is usually a hard task (even a bad practice if it is done without caution) to impute values on the target variables, as this could result in adding a large bias into our model. Furthermore, the inputs for our dataset are supposed to be the wavelengths, thus we cannot use information from some traits to predict others as we do not have that information as input. The best practice is then to drop all null values for a given trait, which will result in having 20 datasets, one for each trait, with a different number of samples.

3- Model Calibration:

For this part, we will train our 4 models (MLR, PCA, PLS, and K-PLS) based on the calibration data. We can use the Scikit-Learn library in Python for the training or just simply write the formulas and do the calculations for each part. Selecting the appropriate number of principal components and latent variables for PCA and PLS is also a part of model calibration. For further examination, we can recalibrate our models by using only the most important wavelength variables. We will then have to train each model 20 times, one for each trait. Note that some models such as PLS can predict multiple target variables, which would be our case but as we have that many null values on targets, we won't be using that feature of the model.

4- Model Validation:

Model validation is crucial to ensure the generalization of our models. We will use our Validation partition of the data to tune the parameters of our models.

We will evaluate the performance of each model (MLR, PCR, PLS, k-PLS) on the validation dataset for all 20 traits by using appropriate regression evaluation metrics such as Mean Squared Error (MSE), and R-squared (R^2) to compare model performance and we will end up choosing the hyperparameters that optimize those scores.

5- Model Testing:

For model testing, we will employ separate data that was not used in the calibration and validation phase. The test data serves as a real-world evaluation of our models. During the model testing phase, we will assess the performance of our calibrated models using various regression metrics such as:

(Note that in the following equations, “n” is the number of predictions, Y_i is the real value for the i-th sample, \hat{Y_i} is the predicted value for the i-th sample and the mean of the real values for the predictions is denoted with a “y” overlined.)

- Mean Squared Error (MSE): measures the average squared difference between the predicted values and the actual values giving us the overall accuracy of the model.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- R-squared (R^2): represents the proportion of variance in the target variable that is explained by the model and provides an overall measure of model performance and fitness.

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

- Mean Absolute Error (MAE): the average absolute difference between predicted and actual values. It is less sensitive to outliers.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\, y_i - \hat{y}_i \,|$$

- Plotting residuals: this will help us check how our model predicts the target variables, and if it predicts better on a certain range of values.
  Residual_i = y_i - \hat{y_i}

6- Comparison and Interpretation:

Compare the results across the four modeling techniques for each of the 20 traits. Analyze which technique performs better in terms of prediction accuracy and robustness, using the previous metrics among others.
Also, we need to consider factors like model complexity, interpretability, and computational efficiency when interpreting the results.

The Mathematical Methods

1- Partial Least Squares (PLS):

PLS aims to maximize the covariance between the predictor variables (X) and the response variable (Y). For that purpose, we try to find the X score vector (T) and the Y score vector (U) after the PCA decomposition and maximize the covariance between T and U. After the decomposition, we can calculate the predictor weight (W) and response weight (U) for calculating T and U.

$$X = TP^T + E \;,\; Y = UQ^T + F$$

$$T = X.W \;,\; U = Y.V$$

$$Maximizing\ Cov(T,\ U)$$

2- Principal Component Regression (PCR):

PCR combines Principal Component Analysis (PCA) with linear regression. First, PCA is applied to the predictor variables (X) to obtain principal components (PCs), and then linear regression is performed on these PCs.

$$Principal\ Components\ (PCs):\ X' = XU,\ where\ U\ is\ the\ matrix\ of\ loadings.$$

$$Linear\ Regression:\ Y = b_0 + b_1PC_1 + b_2PC_2 + \ldots + b_n * PC_n$$

Where b_i represent the i-th component's weight to the response, being b_0 the bias.

3- Kernel Partial Least Squares (K-PLS):

K-PLS extends PLS to handle nonlinear relationships by mapping the data into a higher-dimensional feature space using a kernel function (e.g., radial basis function kernel).

$$T = \Phi(X)W \quad , \quad U = YV$$

Φ(X) represents the kernel-transformed predictor matrix and we aim to maximize Cov(T, U) in the kernel feature space.

In conclusion, K-PLS consists on using PLS on an input matrix that was previously transformed using a kernel. Different kernels include the Radial Basis, Laplacian and Matern.

4- Multiple Linear Regression (MLR):

MLR models the linear relationship between the response variable (Y) and multiple predictor variables (X1, X2, ..., Xn).

$$Y = b_0 + b_1X_1 + b_2X_2 + \ldots + b_n * X_n$$

Where Y is the response variable, X1, X2, ..., Xn are predictor variables, b0 is the intercept, and b1, b2, ..., bn are the coefficients.

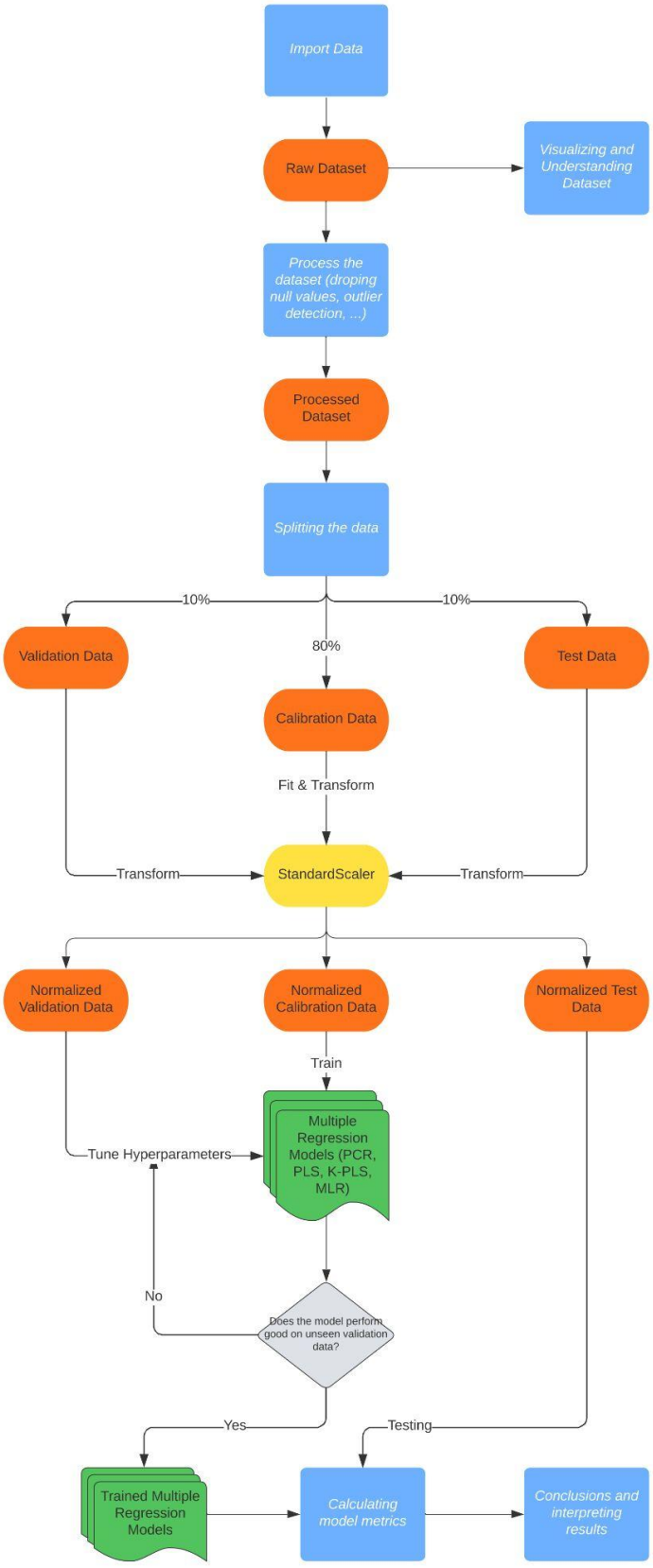
These formulas provide a high-level understanding of the mathematical foundations of each modeling technique. In practice, the specific

calculations may involve more steps and considerations, such as data preprocessing, regularization, and error minimization techniques.

Modeling Roles

For this project we will perform all steps (except modeling) together to make sure that all the concepts are learned by everyone. This might be a little bit more time consuming, but definitely worth it in order to get the learning outcomes. For the modeling part, Alejandro will build the MLR model, Sina will work on the PLS and Pau will work on the PCR. The K-PLS will be done by the three of us as it looks like the toughest model (or at least the one that we have worked less on).

Operations flowchart



Week 6

All results from this report can be found on our github repository: https://github.com/pau-ventrod/ADA_Spectral

Note that we could not include all graphs in this report (one graph for each trait ends up with 20 different graphs), thus all other unshown charts are available on the code.

Multi Linear Regression

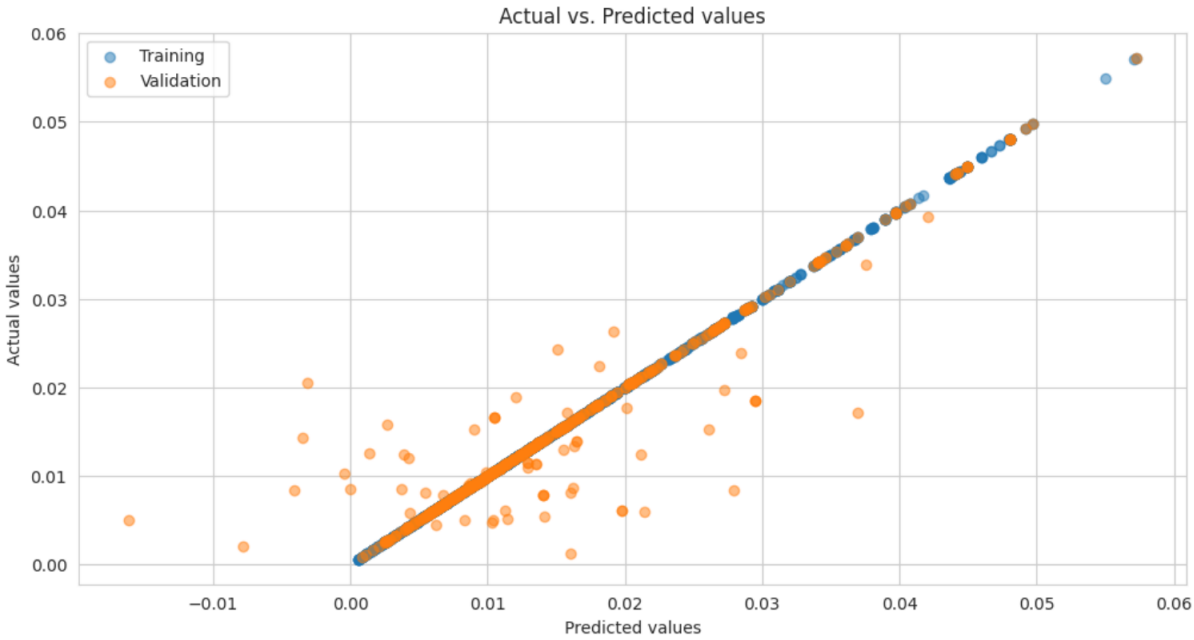
For Multi Linear Regression there were no hyperparameters to tune, so it was straightforward to train the model.

Before performing MLR on our dataset we prepare, normalize and split the dataset. After this, for each target column, missing values are removed and the MLR model is trained (using linear regression) with the train data. Following this, predictions are made using the model and the metrics on R^2 and MSE are computed. Finally, we display the performance of the metrics for each target.

Regarding the output given for each target, we can highlight the significant overfitting present. This means that the model fits too well to the training data and not for the validation data. A clear example is seen in the target Anthocyanin content (μg/cm²), where we can observe a train MSE of 0 and a validation MSE of 37.5590. In addition to a TrainR^2 of 1 and a Validation R^2 of -305.9229. This is a perfect fit for the training set and not for the validation, indicating overfitting.

Another observation we can comment on is that there are negative metrics in R^2, indicating that the model performs worse than one that simply predicts the average of the variable for all observations. An example of this is C content (mg/cm²) with a validation R^2 of -9245695.1716, which is very poor performance on the validation data.

It’s important to note that not all targets exhibit overfitting, and there are some with extremely high MSE values in validation compared to train data, which may suggest scaling issues and outliers.

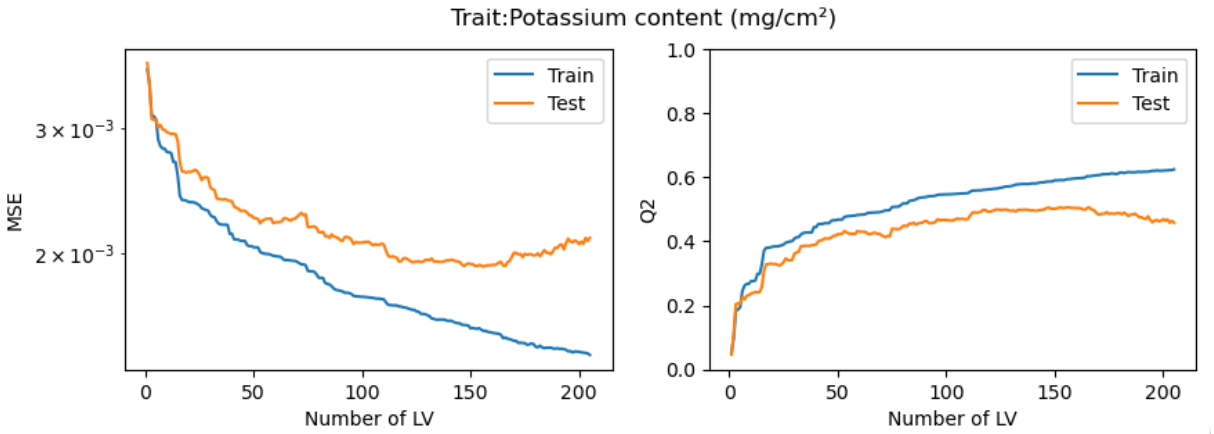


Thanks to this chart, we can see a scatter plot of the actual values contrasted with the predicted values. Ideally, if all the predictions were perfectly calculated, they would be concentrated on the diagonal, because that's where the actual value matches the predicted one. It can be seen that many points, especially the training ones, are close to this line, indicating that the model's fit has been quite good, which means overfitting. However, there are certain points that deviate significantly from this line, especially those from the validation set. Moreover, it can be observed that there are certain areas of the chart where there's a higher concentration compared to other areas, suggesting that there are regions of data that the model predicts with greater accuracy than other zones.

Principal Components Regression

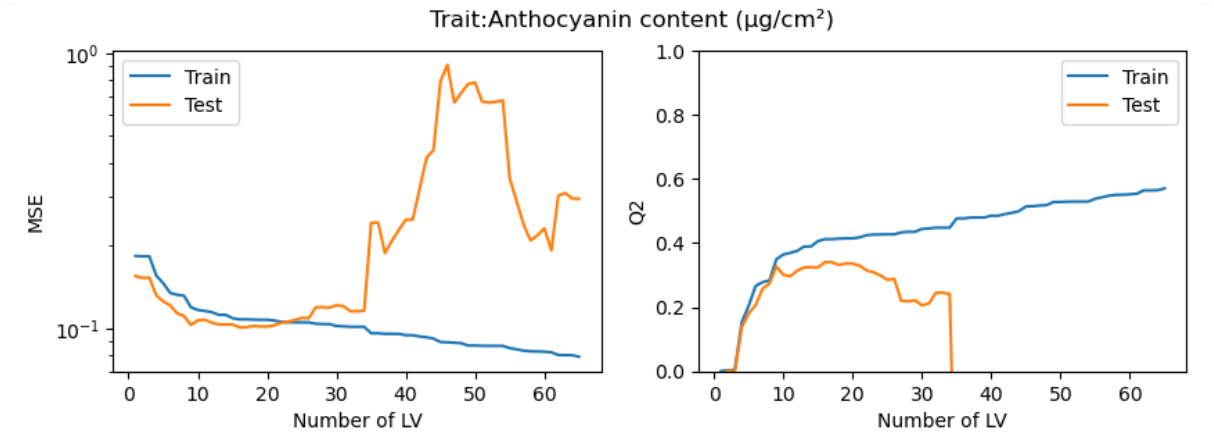
PCR combines Principal Component Analysis (PCA) with linear regression. As we have a different dataset X,y for each trait, ending up with a total of 20 datasets, we have trained a model for each one of them.

We used the validation dataset to choose the best number of components to use of the PCA. For doing so, we compared the Q2 and MSE for a big set of components for each trait until we found the optimal score.



We chose the number of components when we found that increasing it would only result in increasing the MSE on the test dataset while having a plateau on the Q2 score.

Some other traits had higher variances on the scores due to the small amount of data:



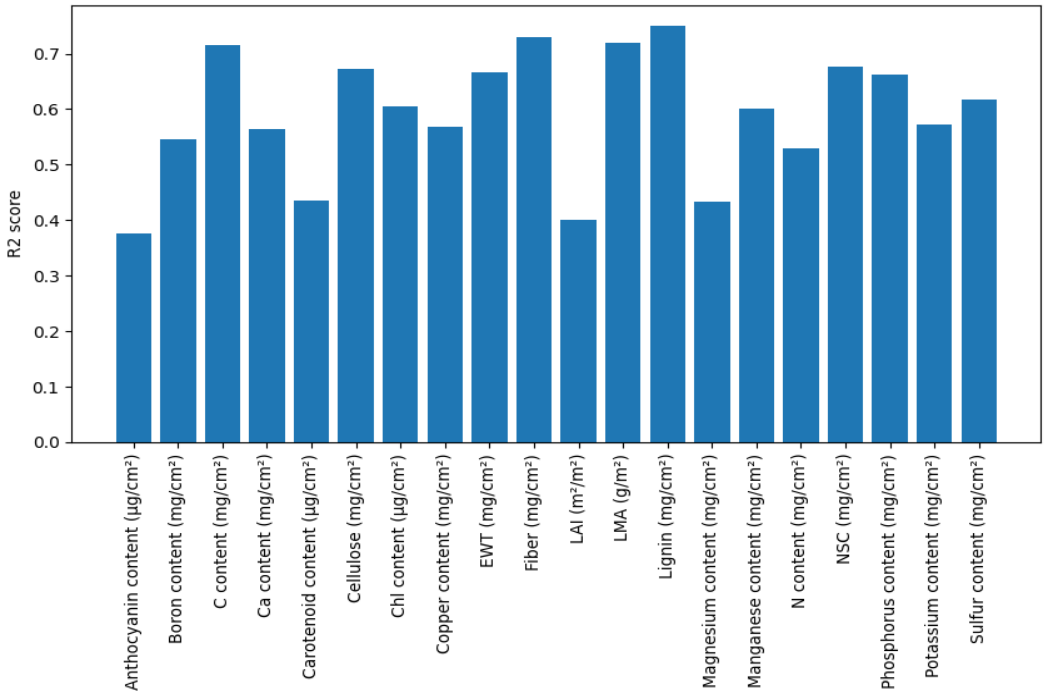
The number of components maxes out at the minimum between the number of samples and the number of columns from a dataset. This makes it very time-expensive to find the optimum number of components, as it would require to make 20 searches on each one of the 1700 components, only to find out that with 100 components it was enough. To avoid this, we have implemented early stopping of 50. This concept makes the search algorithm stop when the MSE has not improved in over 50 components, and keeps the best number of components.

Note that, although with 2 components we managed to reach almost 90% of the variance with PCA, only using 2 components results in a noticeable underperforming model.

Partial least squares (PLS)

Partial Least Squares (PLS) regression is a multivariate statistical technique used for modeling the relationship between a set of predictor variables (X) and a response variable (Y). In our case, we have 20 datasets (X1, X2, ..., X20) and 20 corresponding response variables (Y1, Y2, ..., Y20). For each response variable, we individually train a PLS model using training data and subsequently assess its performance with test data. This results in the creation of 20 distinct PLS models. To determine the optimal number of components for each model, we generate plots displaying R-squared values against different component numbers. By analyzing these plots, we can then determine the optimal number of components for each specific PLS model.

After choosing the optimal number of components for each model, then we train and test our PLS models. Based on performance metrics, the highest R-squared value was observed for Lignin, which was approximately 0.75, while the lowest R-squared value was found for Anthocyanin content, at 0.38. Among the traits, LMA, Lignin, Fiber, and C content are the four with R-squared values exceeding 0.7.



Kernel Partial least squares (K-PLS)

Kernel Partial Least Squares (K-PLS) is an extension of Partial Least Squares (PLS) that incorporates a kernel function, such as the Gaussian kernel. Kernels transform the original data into a higher-dimensional space, making it easier to capture complex relationships between data points. In our case we have used the Gaussian kernel. Here's are some points that should be considered in case of K-PLS with a Gaussian kernel:

1. Kernel Transformation: K-PLS starts by transforming the input data (X) using a kernel function. The Gaussian kernel is used to compute similarity or distance measures between data points. This transformation maps the data into a higher-dimensional space where nonlinear relationships can be better captured. Also, we need to center the test kernel matrix according to the training centers.

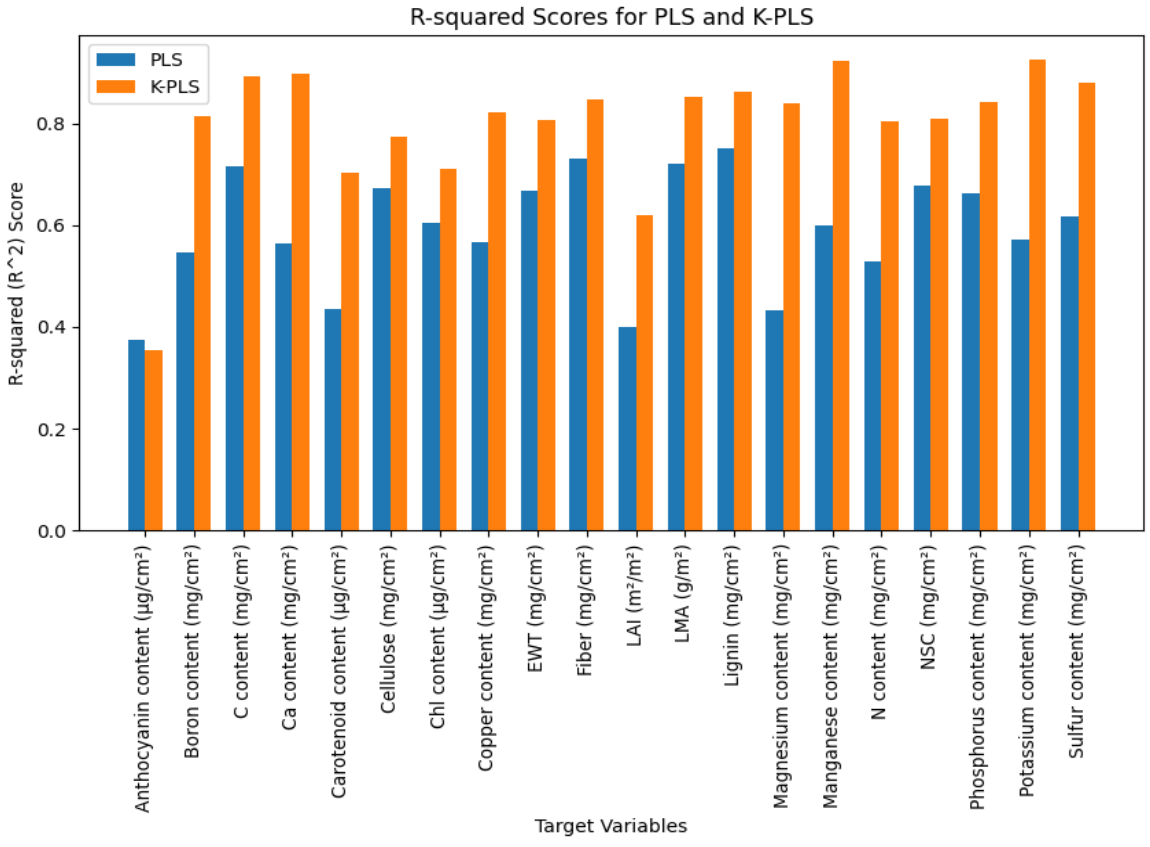
2. Partial Least Squares: Similar to standard PLS, K-PLS aims to find latent variables (components) that explain the maximum covariance

between the transformed predictor variables (X) and the response variable (Y). However, it performs this in the transformed kernel space.

3. Nonlinear Modeling: The use of a Gaussian kernel allows K-PLS to capture complex, nonlinear relationships between X and Y. It's particularly useful when the relationship between variables is not adequately modeled by linear methods.

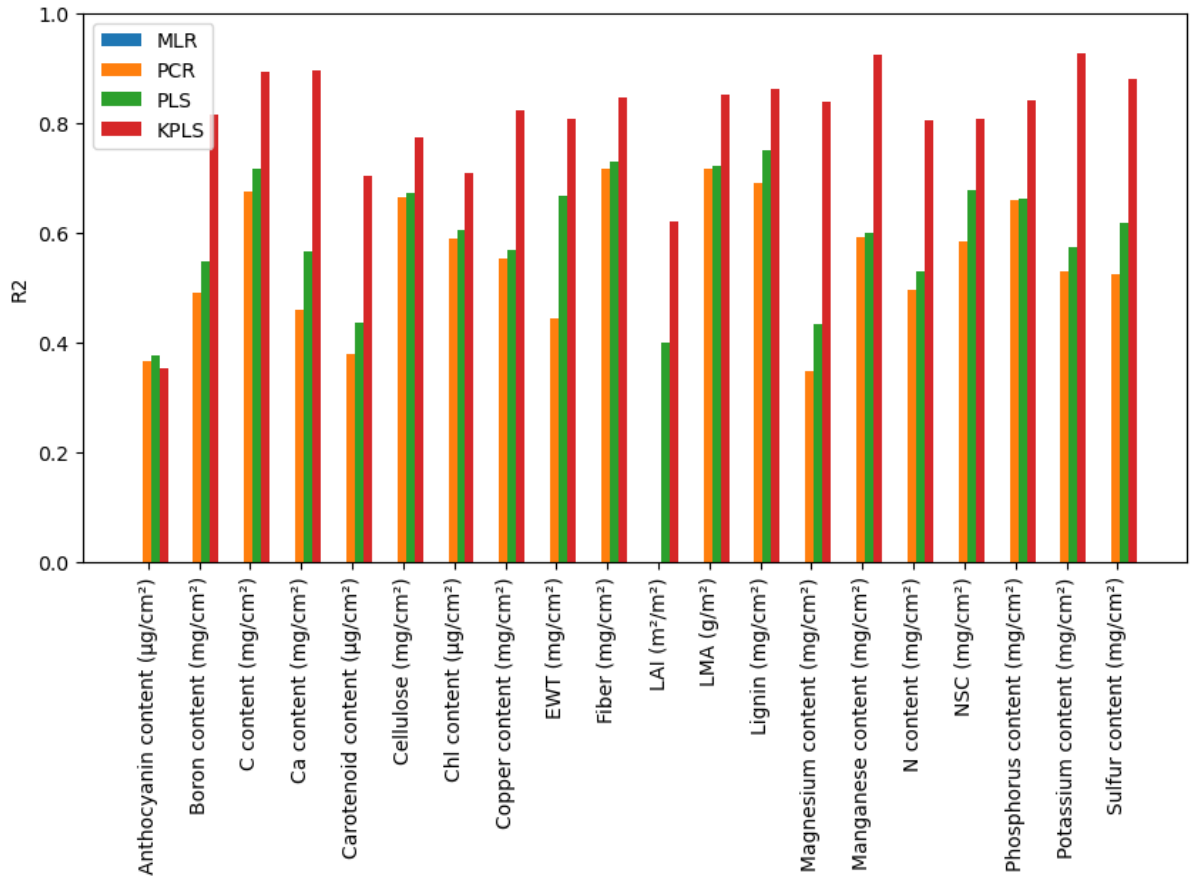
4. Optimal Component Selection: Like PLS, we will need to find the optimal number of components for each of our models.

The following plot displays the R-squared performance of PLS and K-PLS. It is evident that K-PLS outperforms PLS for nearly all the traits. The highest K-PLS performance was observed for Magnesium content, achieving an R-squared value of approximately 0.86, while the lowest performance was found for Anthocyanin content.



Results

We will have a different performance on each trait as not all traits have the same amount of available data for training nor the same ability to be explained by wavelengths, thus we need to be careful and plot the results for each trait individually.



Overall, we can see that KPLS outperforms all other models by far (not considering Anthocyanin), being able to double its contenders in traits like Magnesium Content.

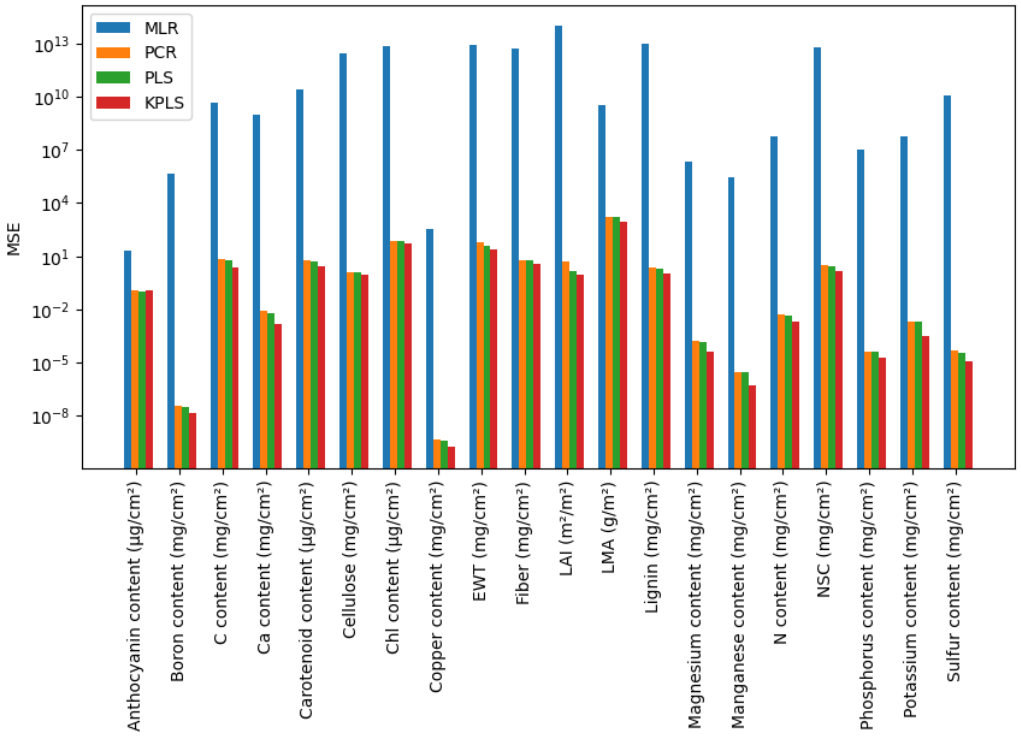
PLS and PCR models seem to perform very similarly in most of the traits, with PLS slightly outperforming PCR. However, in traits like EWT, PLS outperforms by almost 0.2 the PCR model.

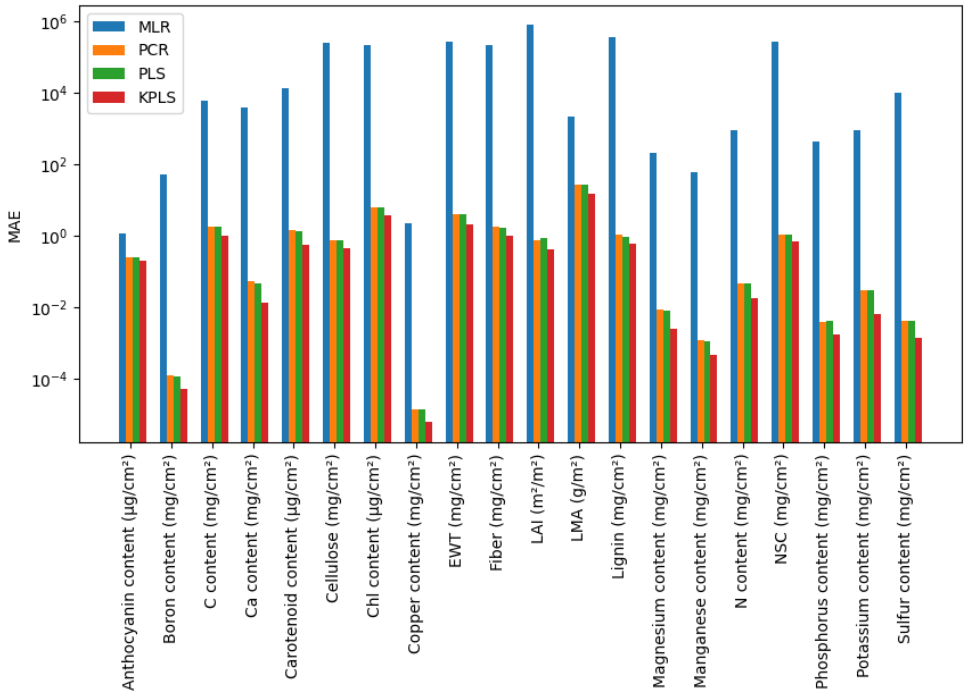
Note that MLR does not have any positive R2. This might be due to the fact that the input data does not have a linear correlation with the traits or with the high overfitting produced.

Some traits are more easy to predict for some models than others. For instance, Manganese Content seems to be the easiest model to predict for K-PLS (and actually the model with the best score, with more than 0.9 R2), while for PLS the best trait is Lignin with almost 0.8.

The worst trait to predict seems to be Anthocyanin, without even reaching 0.4 R2, which looks normal as it is the trait with less number of samples,

thus the model did not have enough samples to train. Surprisingly, on this trait KPLS is slightly outperformed by the other two models.



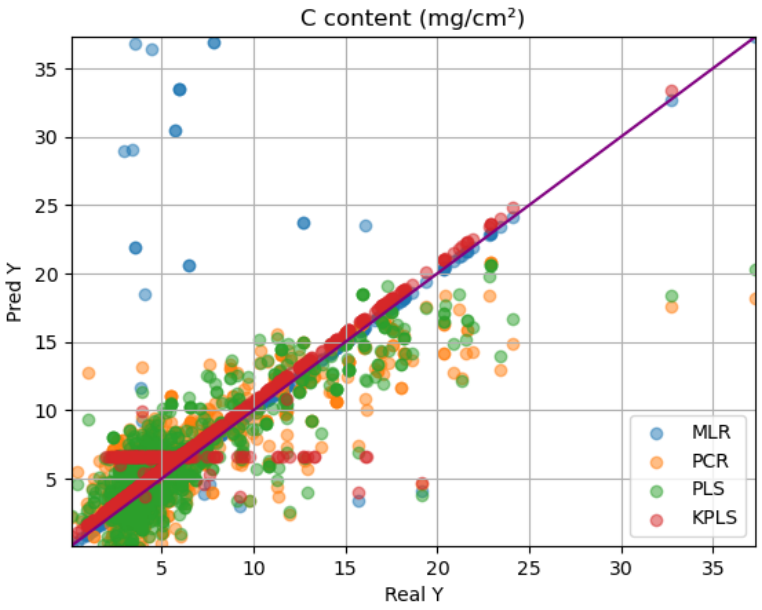


The errors made by the MLR model are outstanding, as the y-axis scale is logarithm, even reaching 10^13 MSE and a MAE of 10^5. This means that when predicting those specific traits, on average our model will have an

error of 10^5 on the prediction, which is a number too large to consider it a decent model.

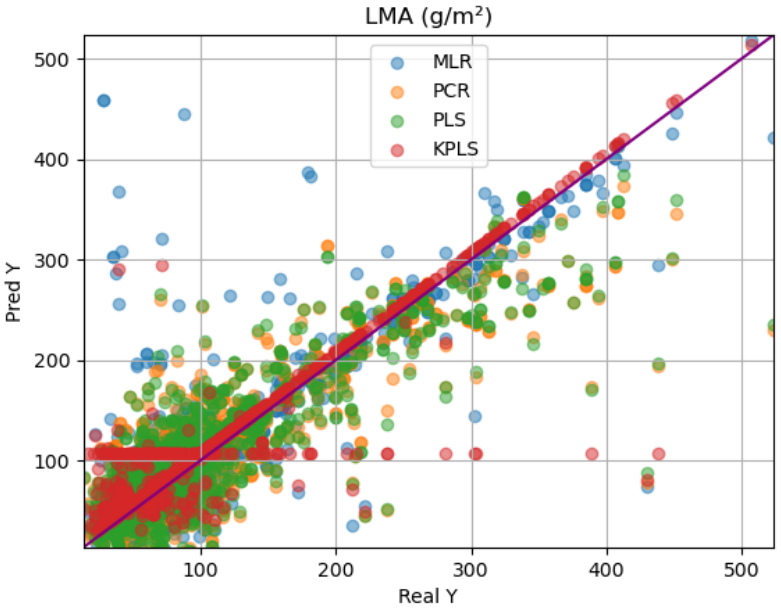
However, all other models seem to perform in a similar range, following the trends mentioned in the R2 plot. Notice that most of the models have a mean absolute error of less than 1, which indicates that when making a group of predictions, we will in average have an absolute error of 1. This might be a good indicator that our models have a good overall performance. Also, the worst performing trait does not even reach a mean absolute error of 10, which gives up an upper bound of the expected error.

We have also plotted the real trait value versus the predicted one to check in which range do the models have a better performance. Note that plots for traits that were not displayed in the report are available in the github code.



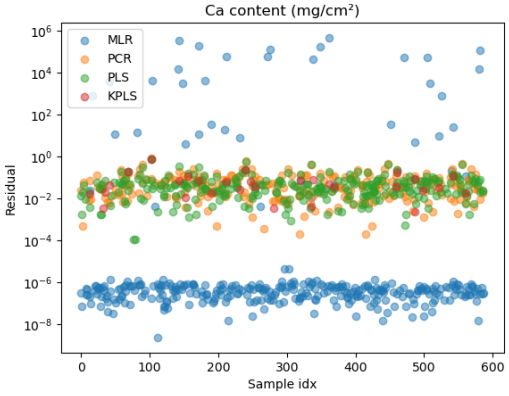
As we can see, MLR predictions follow two trends, either they are very accurate on the perfection line or they are very different from the real value. Then PLS and PCR seem to have similar predictions, being very close to the real value. Finally, KPLS also seems to follow two trends: either it yields a perfect prediction or it just predicts a value of around 6.5 (because it might be the mean of the target values?).

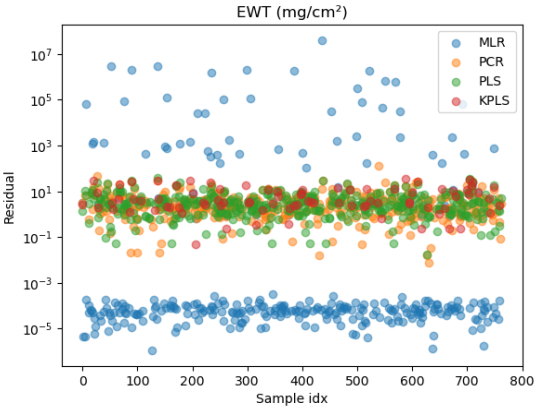
Overall, predictions seem to get worse as the real target value increases, at least for this trait.



The previous behavior and patterns can be observed in multiple traits such as this one.

Another way of visualizing our predictions is to plot the residuals, as we can see around which values to the residuals stay.





All the residuals plots seem to follow a clear structure: a cloud of points with an extremely low error corresponding to some of the MLR residuals, a point cloud of the other three models that cover more or less the same area, and a point could made of more MLR predictions with a significantly

high error. This might be due to the fact that MLR memorized the training set and those samples from the test set that were very close to the samples on the training set got a very good prediction, while those that were a bit more distant got an extremely worse prediction.

Conclusions

Discussing why MLR does not achieve good results, we have to check the low dimensionality of our dataset. We have noted that while all datasets have a large number of columns (more than 1700), most of them do not have a representative number of samples (even some of them having less samples than the number of columns). This results in a high overfitting done in MLR as we do not have a method to control it (we actually do, which are L1 and L2 penalties, but we wanted to build a plain MLR model). However, in other models used, we can control the number of components used in order to reach the specific point where we have enough features to explain our data, but not a large number which would cause overfitting.

We have then shown that using sophisticated models such as PCR, PLS and KPLS increases by far the model performance, being a great option to avoid overfitting, as we can see at which point increasing this number will just lead to increasing overfitting.

We have also shown that using a Kernel such as the gaussian before applying PLS further increases the models performance.

Overall, we were able to build solid and consistent models ensuring a MAE of less than 1 for some traits, and the others not surpassing the 10 bound.

TODO:

In the following reports, we aim to merge all reports into one, bringing the explanatory analisis into this report.

Furthermore, KPLS hyperparameters (number of components, other kernels and distance) should be further optimized to get the best possible

predictions, which could not be done this week due to the lack of time and the excessive computational cost.

Compare the models with tree-based models such as XGBoost to check if they perform as good as the state-of-the-art models.

Adjust MLR to avoid overfitting adding L2 and/or L1 penalty.