KU LEUVEN

Faculty of Science

Modeling of protein and dry matter content using NIR data of insect feed and products

Tamás BARÁTH

Promotor: Prof. Bart de Ketelaere
Mechatronics, Biostatistics and Sensors (MeBioS)

Co-promotor: Eric Schmitt
Protix

Thesis presented in
fulfillment of the requirements
for the degree of
Master of Science in Statistics

Academic Year 2019-2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Description of the task

My initial task was to develop control charts In Python for the company sponsoring the Thesis, Protix, to speed up quality control of their insect protein products. These chart were to be developed for input data in the form of near infrared spectroscopy (NIRS) measurements and for response predicted from the input, nutrient content (protein, fat, ash, and dry matter). However, over time the goals evolved. Measurements were only available in sufficient quantity for dry matter and protein. Moreover, it became clear, that prediction based on the available data was only possible for dry matter content. Therefore, the final goals of Thesis are:

1. Developing tools necessary for analyzing the samples in Python, as most specialized tools are currently only available in Matlab, while the preferred language of Protix is Python.

2. Comparing different pre-processing and modeling methods for the purpose of predicting dry matter and protein content.

3. Investigating the feasibility of predictive modeling with the current measurement device.

4. Identifying potential future investment avenues for Protix, in order to improve their spectroscopic nutrient modeling.

Chapter 2

Literature review

In the following I will give an overview of the concepts I used for tackling the different goals of the Thesis. I will start with explaining the physical background of how the input data is generated and how it relates to different chemical concentrations on a theoretical level, and what it implies for a statistical application. Then an overview will be given of practical techniques used to transform the data to be more suitable for statistical modeling. Finally, different mathematical techniques will be detailed, that can be used to connect the input variables to the output by concentrations.

2.1 Infrared (IR) Spectroscopy (IRS)

Based on the textbook by Stuart (2008)[23] IRS is founded on the physical phenomenon of absorption. Absorption in general means that a photon's energy is taken up by molecules. If the energy of a photon matches exactly a certain energy quantum of a molecule in the medium, then this photon will be absorbed. Absorption of the infrared waves specifically occurs from the vibration of the covalent bonds in molecules, changing the dipole moment of the molecule. For example, molecules with symmetric bonds such as N_2, O_2, or F_2 do not absorb in the infrared since bond vibration does not change their dipole moment.

There are two primary modes of this vibration. With only two atoms the only vibration is stretching, and with three or more atoms involved the bonds can also bend. Of these two primary modes of vibration stretching vibrations are typically of higher energy and can be symmetric or asymmetric. Bending vibrations tend to be lower energy and can be in-plane and out-of-plane. These two bending modes can be further subdivided into scissoring/rocking and wagging/twisting vibrations respectively. The energy of the stretch decreases as the mass of the atoms is increased. For example the absorption due to C-H stretching vibration occurs at $\sim 3,333.3$ nm, but for C-I at $\sim 20,000$ nm. Additionally, hybridization also affects the wavelength, with a C-H bond in an sp hybridized molecule like C_2H_4 corresponding to $\sim 3,030.3$ nm, while in an sp3 hybridized molecule like CH_4 orbital the same bond vibrates at $\sim 3,333 - 3508$ nm.

As only photons of specific wavelengths are absorbed for every molecule depending on its structure and chemical bonds, identification of specific molecules and compounds is possible based on absorption data over a wide spectrum of IR waves. The aforementioned primary vibration modes are typically observed in the mid infrared (MIR) range of 8,500 nm to 12,500 nm, which is often considered the "fingerprint region".

Following the description of Davies [3], beyond the primary vibration modes, there are overtones and combinations of the primary vibrational motions, which show up at higher energy levels, more characteristic of near infrared (NIR). Overtones can be thought of as harmonics. Every primary vibration will produce a series of absorptions at (approximately integer) multiples of the frequency (frequency is the reciprocal of wavelength). Combinations arise from the sharing of energy between two or more primary vibrations. While the number of possible overtones from a group of primary vibrations in a molecule are limited to a few, a very large number of combinations will be observed. The effect of all these absorptions combine to make

many NIR spectra to look rather uninteresting and to consist of only a few rather broad peaks when compared to lower energy IR spectra with primary vibrations.

2.1.1 Near Infrared (NIR) Spectroscopy (NIRS)

NIR spectroscopy (NIRS) is a type of high-energy vibrational spectroscopy performed in the wavelength range from 750 to 2,500 nm. The advantage of NIRS comes from simplicity when compared to lower energy IRS. Specifically, the measurement devices are cheaper and the samples require less preparation, making NIRS suitable for on-line use in production. The ease of use comes at a cost however. In the NIR region, absorption bands come from overtones (principally O-H, N-H, and C-H stretching modes), combinations of overtones and/or combinations of fundamental vibrational motions. The overtone bands and combination bands are much less intense and are broader than the corresponding fundamental absorption bands, which makes NIR data much harder to utilize when compared to MIR, necessitating more advanced mathematical tools. As summarized by Pasquini (2018)[18], NIR data is not promptly available for analytical purpose because physical properties, such as particle-size distribution of powdered samples also impart significant, and usually unwanted (not informative or deteriorative), effects over the NIR spectrum. Most of the time the information is dispersed all over the spectrum. That is because, usually, most of the sample matrix constituents will also provide some contribution to the NIR absorption/reflectance spectrum in addition to the spectral information due the analyte itself.

2.1.2 Transmission spectroscopy

Transmission spectroscopy is the oldest and most basic technique for analyzing IR samples, but it is applicable on different wavelength ranges as well. Light from a source passes through a sample to be registered by the detector, and some of the photons get absorbed on the way at specific wavelengths. The analyte (especially for solids) needs to be presented in a thin film, so that not too much of the light gets absorbed. Even for liquids and gases, the sample thickness needs to be controlled for the light to have a known path length. This requirement for sample preparation is the biggest disadvantage of transmission spectroscopy. The advantage of this method is that the measurement device tends to be cheaper and have superior noise-to-signal ratio than alternatives, specifically diffuse reflectance.[2]

The measured ratio between the light intensity at the detector relative to the source is recorded as transmittance and can be linked to absorbance and analyte concentration using the Beer-Lambert law, will be presented under preprocessing techniques.

2.1.3 Diffuse reflectance spectroscopy

Following the description of Blitz (1998), diffuse reflectance is commonly used in the UV-visible, near-infrared and mid-infrared regions to obtain molecular spectroscopic information. It is usually used to obtain spectra of powders with minimum sample preparation. To understand

what the diffuse part means, one needs to take a step back. A reflectance spectrum is obtained by the collection and analysis of surface-reflected electromagnetic radiation as a function of frequency or wavelength (in the Thesis I exclusively use wavelengths). Two different types of reflection can occur: regular or specular reflection usually associated with reflection from smooth, polished surfaces like mirrors, and diffuse reflection associated with reflection from so-called mat or dull surfaces textured like powders. Techniques such as external reflectance and total internal reflectance spectroscopy use the phenomenon of specular reflection to obtain spectroscopic information. More relevantly, in diffuse reflectance spectroscopy, electromagnetic radiation reflected from dull surfaces is collected and analyzed. The difference between specular and diffuse reflection can be illustrated by considering a mirror and a white wall illuminated by sunlight. When a mirror is illuminated by sunlight, at most angles of observation, the sunlight will not be visible to the observer. However, at a very specific angle, the reflected sunlight will provide a direct image of the source (the sun). In contract, the wall appears equally bright at any angle, because the wall's surface reflects the incident sunlight at angles independent of the angle of incidence. In other words, the surface of the wall scatters the incoming radiation. Ideal diffuse reflection requires that the angular distribution of the reflected radiation be independent of the angle of incidence.

Whereas specular reflectance can be rigorously treated theoretically using the Fresnel equations, diffuse reflection is produced by many complex processes (a combination of reflection, refraction, and diffraction). The most often used theory to describe and analyze diffuse reflectance spectra is the Kubelka-Munk theory. The two-constant Kubelka-Munk theory leads to conclusions that are qualitatively confirmed by experiment and can be used for quantitative work in many cases. Details of the theory will be presented under preprocessing techniques.

2.2 Preprocessing

As previously mentioned regarding NIRS, the measured data is not promptly available for analysis, due to a number of issues. Firstly, much, if not most of the variance in the data tends to be due to physical and not chemical properties, ie noise. Secondly, the relevant information is non-linearly linked to the concentrations, which requires more complex mathematical modeling than a linear relationship would. Fortunately, the theory behind the nature of both the form of non-linearity and the effect of physical noises is well understood, therefore a number of techniques were developed to preprocess NIR measurements.

2.2.1 Conversion to absorbance

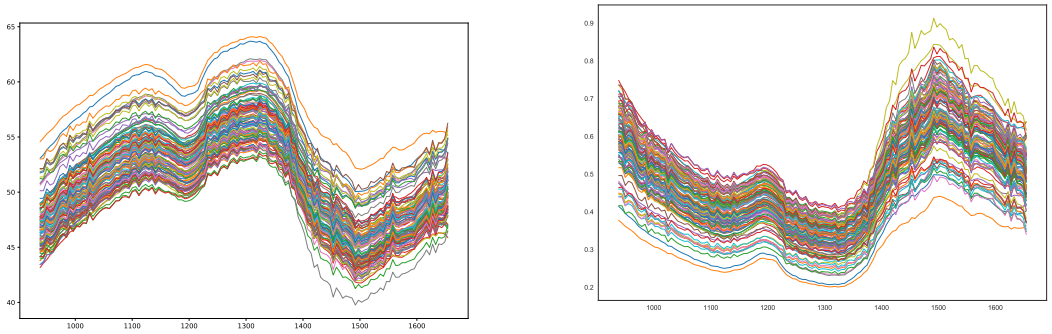
In the first step, to improve linearity the measured reflectance R or transmittance T percentage is normally converted to A absorbance, due to the assumption that A is a linear function of analyte concentration. For transmission spectroscopy The Beer-Lambert law describes the conversion, and goes as follows: If A is the absorbance of the medium and T the transmittance,

then:

$$A = log_{10}\left(\frac{1}{T}\right)$$

The analogous function for diffuse reflectance was derived by Kubelka and Munk. Without going into details, it is suffice to say that it is merely an approximation with many assumptions, the biggest one being no Fresnel reflection. In practice usually the reflectance is not entirely diffuse, but rather a mix of specular and diffuse. This can become especially problematic with spectra of samples with high absorptivity, but is usually not an issue with NIRS. Additional assumptions are that particles in the sample are much smaller than the thickness of the entire sample, the sample thickness is greater than the beam penetration depth (i.e. no transmission), sample diameter is much greater than the focus of the incident beam, and the sample area detected is smaller than the area illuminated. With these assumptions the conversion is the following. If A is the absorbance of the medium and R the reflectance, then:

$$A = \frac{(1-R)^2}{R}$$



(a) Raw reflectance spectra (horizontal axis in nm, vertical axis in %)

(b) Raw reflectance in absorbance (horizontal axis in nm, vertical axis in %)

Figure 2.1: Effect of spectral transformation on spectra

As stated, the absorbance is assumed to be a linear function of analyte concentration, but this only holds under optimal conditions. In practice however, the presence of instrument, physical and chemical noise can cause this linearity to break down. As a result, a number of methods were developed to filter out noise and irrelevant information in order to linearize the relationship between the spectra and the response before analysis.

2.2.2 Baseline correction and detrending

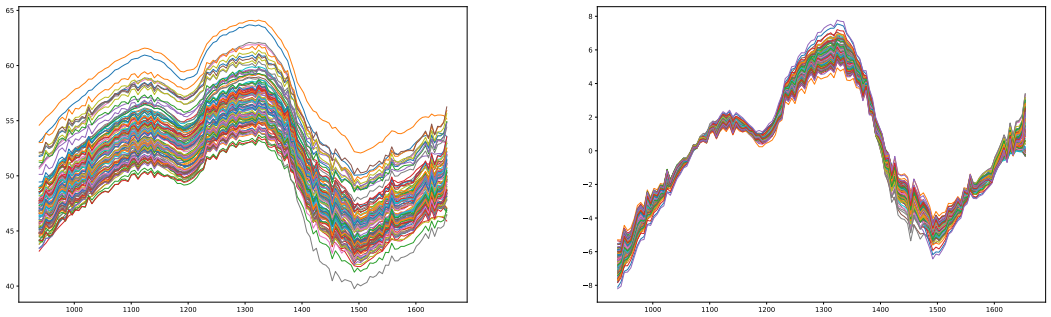
Scattering induces additive (offset) and/ or multiplicative (slope) effects, thus scattering contributes to the baseline of the spectrum. By fitting a baseline to the spectrum and subtracting this line from it, the influence of the background is removed. (Wouter, 2019)[30]

2.2.2.1 Baseline correction

To perform baseline correction, first the baseline needs to be found. The first step is selecting two point where the signal is only due to baseline and not due to the component of interest. In the second step a baseline is fitted to each spectrum by Weighted Least Squares (WLS). After the baseline is thus found, the correction is performed by taking the variables above the baseline as the signal, and those below as the background (Wouter, 2019)[30]. Unfortunately, NIR spectra tends to come from the sum of multiple absorption bands spreading over the whole spectrum, instead of clear peaks separated by flat baselines, making this simple technique unfeasible.

2.2.2.2 Detrending

A more viable approach for NIR data is using detrending, which can remove a constant, linear or curved offset. A polynomial of given order is fitted to every observation separately, then this polynomial is subtracted from each. While simple, this method only works well when the largest source of signal in each sample is background interference. (Wouter, 2019)[30] This means, that since the actual baseline is unknown (as opposed to baseline correction) the entire sample is being used to fit this polynomial, and variance of all sources is removed, not just the actual baseline. If the variance in the data mostly comes in the form of absorption peaks and not spectral shifts due to physical properties, more useful information will be removed, than noise.



(a) Raw spectra (horizontal axis in nm, vertical axis in %)

(b) Spectra after linear detrending (horizontal axis in nm)

Figure 2.2: Effect of detrending on spectra (reflectance)

2.2.3 Smoothing and derivatives

Derivatives are useful for removing additive (first derivative) or additive and multiplicative effects (2nd derivative). The first derivative returns the slope at each point of the spectrum, assigning zero values to local minima and maxima. Negative values mean diminishing absorption and positive values correspond to increasing absorption. There are two common ways to

calculate these derivatives based on discrete measurements, and both rely on a form of smoothing, because derivation can amplify certain noise (fluctuations around the main spectra curve) in the measured spectral curves, especially in finite differences.

The premise of data smoothing is that the measured variable is both slowly varying and also corrupted by random noise. In this case, the signal, the underlying sum of the spectral peaks, is supposed to be smooth, with small changes over any interval. When white noise is added to the smooth line, it gets rough, with the finite differences randomly fluctuating around zero with a much higher magnitude, than the underlying signal's change. If that is the case, it could be useful to replace each data point by a some form of local average based on surrounding data points, as nearby points measure very nearly the same underlying value, and averaging can reduce the level of noise without adding too much bias. The first way is the Norris-Williams method, which approximates the value by taking the differences of averages over several measured points.The second way is the Savitzky-Golay method, which involves taking the derivative of a locally fitted polynomial curve on the spectrum. Both methods use a symmetric window smoothing, requiring the number of data points on each side of a central point to be the same, which leads to the loss of some points at both ends of the spectrum. The window size needs to be selected by the user based on performance of the consequent modeling step.

2.2.3.1 Norris-Williams (NW) derivatives

NW derivation was developed by Norris and Williams in 1984[17] to avoid the previously mentioned noise inflation in finite differences. The smoothing approach is a simple moving window average. The NW derivation consists of two steps:

1. Smoothing of the spectra by averaging over a given number of points. If n is the number of points in the smoothing window centered around the current measurement y_i (ie the total number of points in the window is $2n + 1$), then:

$$y_{\text{smooth},i} = \frac{\sum_{j=-n}^{n} y_{i+j}}{2n+1}$$

2. The first-order derivatives with a given gap size (larger then zero) between them will be:

$$y'_i = y_{\text{smooth},i+\text{gap}} - y_{\text{smooth},i-\text{gap}}$$

and the second-order derivatives will be:

$$y''_i = y_{\text{smooth},i-\text{gap}} - 2 \cdot y_{\text{smooth},i} + y_{\text{smooth},i+\text{gap}}$$

NW derivation is often followed by normalization of the corrected spectra.

2.2.3.2 Savitzky-Golay (SG) derivatives

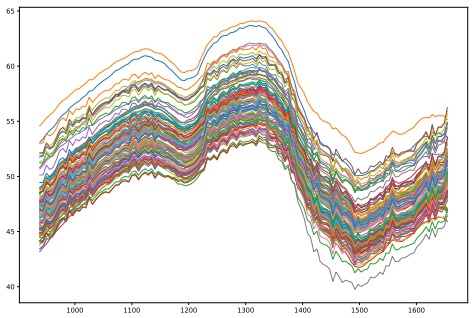
A serious problem with the previously shown NW approach is that a simple moving window average is going to bias any function with a non-zero second derivative, ie the local maxima will always decrease. This is especially problematic for spectroscopy, where narrow local peaks corresponding to absorption bands are sought. To circumvent this problem Savitzky and Golay (1964)[20] suggested a data smoothing method, where a polynomial is fitted in a symmetric window on the raw data in order to find the derivative at centre point y_j.

A moving window average for the point y_j (the jth measurement) can be defined as a weighted sum of m (an odd number) points: $y_{\text{smooth},j} = \sum_{\frac{1-m}{2}}^{\frac{m-1}{2}} c_i \cdot y_{j+i}$, with c_i the weights assigned to the m points in the window, and where the indexing for i goes from the negative index $\frac{1-m}{2}$ to $\frac{1-m}{2}$. In the NW case the c_i weights were equal constants summing up to 1, therefore $c_i = \frac{1}{m}$. SG smoothing is about finding different weights based on a polynomial, in order to reduce bias for nonlinear functions.

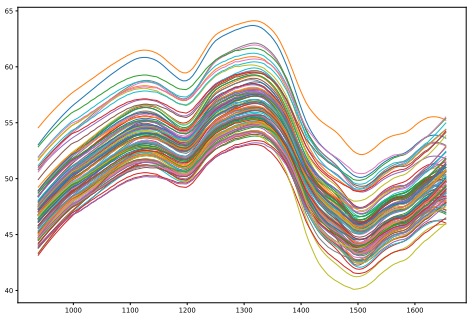
Specifically, the goal is to least squares fit a polynomial on the m points in the windows centered on each point y_j, then take the non-parametric estimates of the center points and use them as the new $y_{\text{smooth},j}$. This idea does not seem to make use of a weighted sum, but in practice there is no need to actually fit a least squares approximation on all windows separately. A more efficient approach is pre-calculating a set of weights that are identical in all windows and lead to the same solution. In fact, any polynomial function can be reproduced perfectly non-parametrically as a weighted moving window average with the right choice of weights, and these weights only depend on the window length and the polynomial degree, but not the actual function. These so called convolution weights are available in a table by the original authors. Their derivation is as follows.

First, for the y_j points in the window a polynomial a set of x_j are needed as predictors. For spectra, these are the wavelengths and are normally equally spaced. In case of unequal intervals the least squares fitting described above needs to be done for every window separately. Then, a unit vector e of m dimensions is needed that is zero everywhere, except for the middle (corresponding to the center of the window), where it is 1. Next a Vandermonde matrix $\boldsymbol{X}$ is created with the rows corresponding to x_j and the columns being powers of x from 0 to the desired polynomial degree, so $\boldsymbol{X}_{j,c} = x_j^c$. This is the same procedure one would follow when creating a design matrix for a linear regression with polynomial expansion of a single predictor. Next the $\boldsymbol{X}$ matrix is regressed on the unit vector e by least squares, and the estimated values $\hat{e}$ will be the weights c, the vector containing the m convolution coefficients, therefore $\hat{e} = c = \boldsymbol{X} \cdot (\boldsymbol{X}' \cdot \boldsymbol{X})^{-1} \cdot \boldsymbol{X}' \cdot e$.

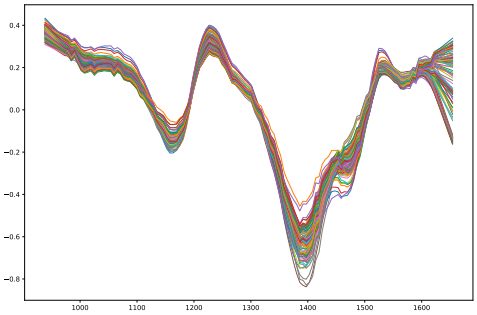
As for the derivatives, simply different sets of convolution weights c need to be calculated depending on the derivative, and these are also available in a tabular form in the original publication. Naturally, the highest derivative that can be determined depends on the degree of the fitted polynomial.



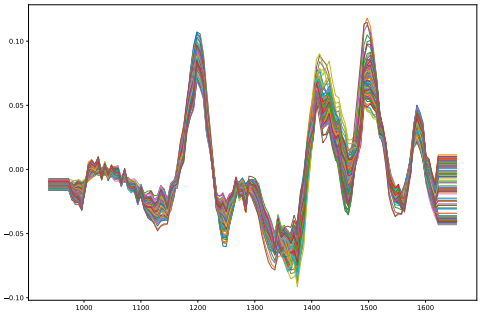
(a) Raw spectra



(b) Spectra after SG smoothing



(c) First derivative of spectra after SG smoothing



(d) Second derivative of spectra after SG smoothing

Figure 2.3: Effect of SG smoothing and derivatives on spectra (reflectance) using window length of 13 and second polynomial approximation (horizontal axis in nm, vertical axis in %)

2.2.4 Multiplicative scattering correction (MSC)

MSC in its basic form was first introduced by Martens et al. in 1983[15] and further developed by Geladi et al. in 1985[8]. MSC has two steps:

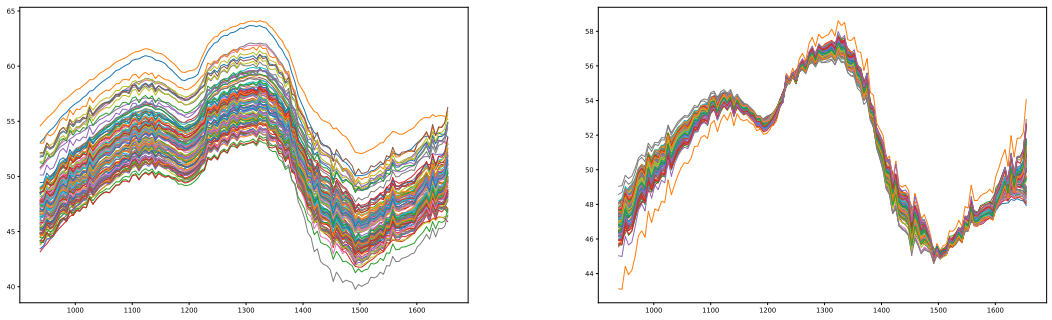
1. Estimating the correction coefficients (additive and multiplicative contributions). If $\mathbf{x}_{org}$ is one sample from the original raw spectra, $\mathbf{x}_{ref}$ the reference spectrum used for preprocessing the entire dataset, $\mathbf{e}$ the un-modeled part of $\mathbf{x}_{org}$, and b_0 and $b_{ref,1}$ are sample specific scalar parameters, then:

$$\mathbf{x}_{org} = b_0 + b_{ref,1} \cdot \mathbf{x}_{ref} + \mathbf{e}$$

2. Correcting the raw spectra. If $\mathbf{x}_{corr}$ is the corrected spectra, then:

$$\mathbf{x}_{corr} = \frac{\mathbf{x}_{org} - b_0}{b_{ref,1}} = \mathbf{x}_{ref} + \frac{\mathbf{e}}{b_{ref,1}}$$

Usually, simply the average spectrum of the calibration set is used as $\mathbf{x}_{ref}$, but when available, a chosen reference spectrum should be applied. Since MSC divides all measurements with the same reference, it can potentially amplify outliers, eg a large absorbance gets divided by a small



(a) Raw spectra (horizontal axis in nm, vertical axis in %)

(b) Spectra after MSC (horizontal axis in nm)

Figure 2.4: Effect of MSC using average spectrum on spectra (reflectance)

one. Using the mean spectrum as reference can mitigate this effect, as the mean is sensitive to outliers itself.

2.2.5 Standard normal variate (SNV) and normalization

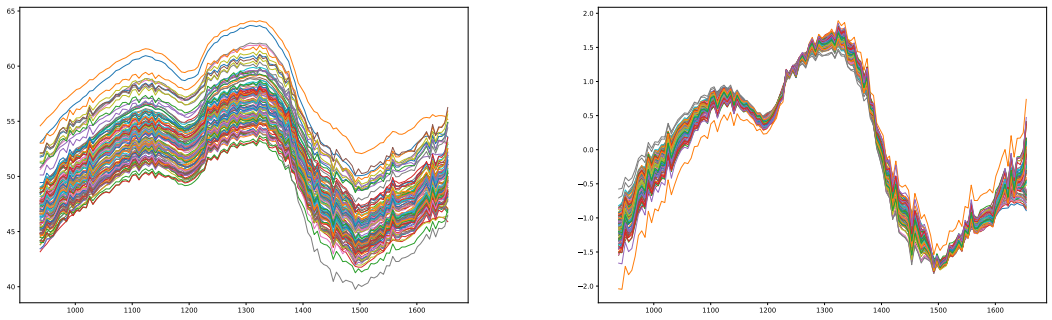
The idea behind SNV and normalization is the same as for MSC, but instead of using a common reference signal each observation is processed on its own, isolated from the remainder of the set (Barnes, Dhanoa, & Lister, 1989)[1], which can be advantageous when one has outliers, as the normalization constant will be scaled with it. Additionally, they do not involve a least squares fitting in their parameter estimation. The process is therefore similar to the one in MSC:

$$\mathbf{x}_{corr} = \frac{\mathbf{x}_{org} - a_0}{a_1}$$

For SNV, a_0 is the average and a_1 the standard deviation of the sample spectrum to be corrected. If a_0=0, then it is normalization. For normalization different vector-norms can be used for the scaling factor a_1. The most common ones are the Euclidean norm (square root of the sum of the squared elements) or the total sum of the absolute values of the elements. Other less common options are normalizing to the maximum absorbance variable or normalizing towards a single selected wavelength. For both the SNV and normalization approach instead of using the average and the standard deviation as the correction parameters, one might consider using the more robust equivalents of these statistical moments, like the median or the mean of the inner quartile range and the standard deviation of the inner quartile (Rinnan, Berg, & Engelsen, 2009)[19].

2.3 Dimensionality reduction

The common way to handle the multicollinearity problem with NIR regression is by reducing the dimensionality of the regressors. In spectroscopy it is usually not easy to find selective wavelengths for the chemical constituents in the samples, which makes it unfeasible to solve the



(a) Raw spectra (horizontal axis in nm, vertical axis in %)

(b) Spectra after SNV (horizontal axis in nm)

Figure 2.5: Effect of SNV on spectra (absorbance)

multicollinearity by selecting only a subset of the predictor variables. Instead, either principal component analysis (PCA) or partial least squares (PLS) regression methods can be used. PCA tries to reduce the dimensionality of the data while preserving as much of the variance in the predictor variables as possible by orthogonal transformation. While this works very well with NIR data, as most of the variance can usually be represented with only a few orthogonal variables (called principal components), there is no guarantee that the variance of the regressors always contains useful information instead of noise. The PLS approach aims to avoid this pitfall by considering the covariance structure with the dependent variables as well.

2.3.1 Principal Component Analysis

While for the purposes of the Thesis there are many more efficient techniques available, PCA is important to mention, as later methods, specifically Partial least squares and its extensions can be understood as a refinement of the principles of PCA. As such, I will only give a brief introduction to the idea and workings of this method, and not go into detail over its practical application.

According to Jolliffe (2002)[14] "the central idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables." The first principal component is derived as a linear combination of the original variables (also called a principal component score) that has maximal variance with the constraint that weight vector w_1 (also called the loadings vector) has to have unit length. For the constrained maximization the technique of Lagrangian multipliers is used. The Lagrange multiplier λ_1 will correspond to the largest eigenvalue of the covariance matrix of the original variables with the corresponding eigenvector being w_1. Once w_1 is known, the first PC can be calculated. In general, the jth component can be found by subtracting the first j - 1 principal components from the original variables and repeating the constrained maximization (finding

the unit weight vector w_j which extracts the maximum variance from this new data matrix). As it turns out, the jth component is given by the jth largest eigenvalue and corresponding eigenvector of the original covariance matrix.

Formally the process can be described as follows: Let X be a data matrix, with k columns (variables) and column-wise zero mean (every variable has been mean centered) and t_j be the PC with the jth largest variance. The X data can also be rescaled, but no scaling is needed when all the variables are measured in the same units, as in spectrometry. Then $t_j = X \cdot w_j$ with w_j being the unit weight vector corresponding to the jth PC. The eigendecomposition of $X' \cdot X$ square matrix gives k eigenvalues and eigenvectors. As the eigenvalues are proportional to the variance explained by their corresponding PC, sorting the eigenvalues in order from largest to smallest gives the order of the corresponding eigenvectors which are the k weight vectors. Therefore the jth PC corresponds to the jth largest eigenvalue.

2.3.2 Partial Least Squares (PLS) Regression

The goal of PLS is to extract a number of latent variables ($\boldsymbol{T}$ and $\boldsymbol{U}$) from sampled predictors and responses, respectively. The extracted factors $\boldsymbol{T}$ (also referred to as $\boldsymbol{X}$-scores) are used to predict $\boldsymbol{U}$ (the $\boldsymbol{Y}$-scores), and then the predicted $\boldsymbol{Y}$-scores are used to construct predictions for the responses. $\boldsymbol{T}$ and $\boldsymbol{U}$ are chosen so that the relationship between successive pairs of scores is as strong as possible (the covariance is maximized), therefore this is not the same as doing PCA on the predictor and response variables.

Geladi and Kowalski (1986)[9] describes a PLS regression on NIR data as follows. First, the data (both the regressor and dependent variables) are mean-centered, that is the mean of each variable is subtracted from the variable itself. The data can also be rescaled, but no scaling should be needed when all the variables in a block are measured in the same units, as is the case in spectrometry. Then the predictor and response variables are decomposed as products of sets of orthogonal factors (the $\boldsymbol{X}$- and $\boldsymbol{Y}$-scores) and of sets of specific loadings. The latent $\boldsymbol{T}$ and $\boldsymbol{U}$ matrices are obtained vector-by-vector, in the first iteration only the first pair of t and u latent vectors is obtained, containing the most covariance. The second iteration extracts a pair with the second most covariance and so on.

There multiple algorithms to fit a PLS model, the two arguably most popular being non-linear iterative PLS (NIPALS) and statistically inspired modification of the PLS (SIMPLS). The practical difference between them is mostly in their computational speed (SIMPLS being faster). If the results differ between the two algorithms, it is only due to rounding errors during computation. Since OSC also relies on NIPALS as described in 2.3.5.1, and later on maximal correntropy-weighted PLS relies on weighted SIMPLS in 2.3.3.1 I will describe both variants in the following.

2.3.2.1 NIPALS Algorithm

The process to calculate the ath component by NIPALS, based on Geladi and Kowalski's prevously mentioned tutorial[9], goes as follows:

1. Select one of the response variables from Y_a as an initial estimate of the u_a vector.
2. Project the predictor variables X_a on u_a by OLS, obtaining w_a, a weight vector.
$$w_a = \frac{X_a' \cdot u_a}{u_a' \cdot u_a}$$
3. Normalize the euclidean length of w_a to be equal to 1.
$$w_a = \frac{w_a}{\sqrt{w_a' \cdot w_a}}$$
4. Take the weighted sum of the variables in X_a using w_a, obtaining t_a, a latent vector of X-scores.
$$t_a = X_a \cdot w_a$$
5. If response variable is univariate skip to step 10, for multivariate response continue.
6. Project all response variables in Y_a on t_a by OLS, obtaining q_a, a weight vector.
$$q_a = \frac{Y_a' \cdot t_a}{t_a' \cdot t_a}$$
7. Normalize the euclidean length of q_a to be equal to 1.
$$q_a = \frac{q_a}{\sqrt{q_a' \cdot q_a}}$$
8. Take the weighted sum of the variables in Y_a using q_a, obtaining u_a, a latent vector of Y-scores.
$$u_a = Y_a \cdot q_a$$
9. Check convergence of u_a. If not converged go to step 2.
10. Calculate the loading vector p_a by projecting X_a on w_a by OLS.
$$p_a = \frac{X_a' \cdot t_a}{t_a' \cdot t_a}$$
11. Deflate X_a and Y_a by the ath component.
$$X_{a+1} = \boldsymbol{X}_a - t_a \cdot p_a'$$
$$Y_{a+1} = Y_a - u_a \cdot q_a'$$

12. Repeat from step 1 for $a + 1$.

Once we have extracted all the latent components needed, we can regress the $\boldsymbol{T}$ variables on the $\boldsymbol{U}$ variables, obtaining the vector β which contains the coefficients for the nutrient model.

2.3.2.2 (Weighted) SIMPLS Algorithm

In the following the weighted version of SIMPLS will be described, as this is the one that will be used later for MCW-PLS in 2.3.3.1. The only difference between the vanilla and weighted version is that for the former the diagonal weight matrix $\boldsymbol{A}$ is an identity matrix, therefore $\boldsymbol{A} = \boldsymbol{I}_{n \times n}$. The process to calculate the ath component, based on the description of Fonseca Diaz et al.[6], is as follows:

1. Center the predictor variables in $\boldsymbol{X}$ and the response variables in $\boldsymbol{Y}$ using their weighted means by $\boldsymbol{A}$.

$$\boldsymbol{X}_c = \text{trace}(\boldsymbol{A})^{-1} \cdot \boldsymbol{1} \cdot \boldsymbol{A} \cdot \boldsymbol{X}$$
$$\boldsymbol{Y}_c = \text{trace}(\boldsymbol{A})^{-1} \cdot \boldsymbol{1} \cdot \boldsymbol{A} \cdot \boldsymbol{Y}$$

2. Calculate starting covariance matrix $\boldsymbol{S}_{a=1}$ using the weight matrix $\boldsymbol{A}$ and the centered matrices $\boldsymbol{X}_c$ and $\boldsymbol{Y}_c$.

$$\boldsymbol{S}_1 = \boldsymbol{X}_c' \cdot \boldsymbol{A} \cdot \boldsymbol{Y}_c$$

3. Calculate the singular value decomposition of $\boldsymbol{S}_a$ and take the first left-singular vector r_a, which is the first column of the left-singular vector matrix $\boldsymbol{U}_a$.

4. Calculate the ath score vector t_a as the weighted sum of the predictor variables $\boldsymbol{X}_c$ by r_a.

$$t_a = \boldsymbol{X}_c \cdot r_a$$

5. Standardize t_a using the weight matrix $\boldsymbol{A}$.

$$t_a = \frac{\text{trace}(\boldsymbol{A})^{-1} \cdot \boldsymbol{1} \cdot \boldsymbol{A} \cdot t_a}{\sqrt{t_a' \cdot \boldsymbol{A} \cdot t_a}}$$

6. Calculate the factor loadings p_a for the ath component by projecting $\boldsymbol{X}_c$ on t_a by weighted OLS with weights $\boldsymbol{A}$.

$$p_a = \boldsymbol{X}_c' \cdot \boldsymbol{A} \cdot t_a$$

7. If $a = 1$, then normalize $p_{a=1}$ and store it as the first column of $\boldsymbol{V}$, a matrix of orthonormal loading vectors. $\boldsymbol{V} = \frac{p_1}{\|p_1\|}$

8. If the current component is not the first, then v_a, a Gram-Schmidt orthonormalization of p_a needs to be calculated, using $\boldsymbol{V}$, a matrix containing all the orthonormal factor

loadings calculated so far. After v_a is calculated, normalize it to unit length and append it to $\boldsymbol{V}$ by column.

$$\begin{aligned} v_a &= p_a - \boldsymbol{V} \cdot \boldsymbol{V}' \cdot p_a \\ v &= \frac{v_a}{\|v_a\|} \\ \boldsymbol{V} &= [\boldsymbol{V}, v_a] \end{aligned}$$

9. Deflate the covariance matrix with the first component using loading vector v_a for the next iteration

$$\boldsymbol{S}_{a+1} = \boldsymbol{S}_a - v_a \cdot v_a' \cdot \boldsymbol{S}_a$$

10. Repeat from step 3 until enough components are extracted.

2.3.3 Maximum Correntropy-Weighted Partial Least Squares (MCW-PLS)

This approach was only recently developed by Fonseca Diaz et al. in 2020[6] to control the robustness of the vanilla PLS model. Since the dataset at hand does not seem to suffer from outliers, it might seem counter-intuitive to include this approach in the analysis. However, MCW-PLS is not exclusively aimed at making the model outlier resistant, which can be seen when compared to another popular robust PLS approach, RSIMPLS, introduced by Hubert et al. in 2003[11]. RSIMPLS is less applicable in the current case because it identifies outliers and discards them completely. Since the available dataset is already quite small and there is no obvious sign of outliers, removing samples would probably cause more harm than good. While the authors entertain the possibility of applying continuous weights between zero and one, they only developed a hard rejection algorithm. In contrast MCW-PLS iteratively assigns weights to different samples in the training set similar to the well known iteratively reweighted least squares algorithm, with the weights given by a radial basis function (RBF) kernel applied to the residuals. The algorithm allows for controlling the free parameter of the RBF kernel, for different degrees of soft rejection. This makes it applicable for problems where there are no clear outliers, but there might be heteroscedastic samples with different levels of noise, even after preprocessing.

The MCW-PLS method was specifically designed for use on data with outliers that are outlying in the sense, that they do not come from a normal distribution. Violation of the normality assumption is problematic, as PLS models rely heavily on OLS regression. This is true regardless of the algorithm used to fit them, as irrespective of how the latent variables are calculated, the final steps involve linking the score vectors to the response variables by linear regression. The name "maximum correntropy" comes from the author deriving a solution to the problem using an information theoretical approach. Namely, she shows that maximizing correntropy is equivalent to minimizing the quadratic entropy Rényi entropy in the generic

linear regression model, with correntropy defined non-parametrically using an RBF kernel.

MCW-PLS has the convenient property of having the same regression model output with the same interpretation as PLS. Additionally, it can even reduce to a vanilla PLS model with a large enough hyperparameter setting.

2.3.3.1 MCW-PLS Algorithm

1. Initialize weights as an identity matrix $\boldsymbol{A} = \boldsymbol{I}_{n \times n}$

2. Fit a weighted PLS model using SIMPLS on $\boldsymbol{X}$ and $\boldsymbol{Y}$ using $\boldsymbol{A}$ with a chosen number of components to get $\boldsymbol{R}$, a matrix of factor loadings.

3. Center $\boldsymbol{X}$ using the weighted mean by $\boldsymbol{A}$.

$$\boldsymbol{X}_c = \text{trace}(\boldsymbol{A})^{-1} \cdot \boldsymbol{1} \cdot \boldsymbol{A} \cdot \boldsymbol{X}$$

4. Calculate the score vectors $\boldsymbol{T}$ using $\boldsymbol{X}_c$ and $\boldsymbol{R}$ and augment it with a column of constants.

$$\boldsymbol{T}_{\text{aug}} = [\boldsymbol{1}, \boldsymbol{X}_c \cdot \boldsymbol{R}]$$

5. Calculate the vector of regression coefficients β by weighted OLS using the augmented scores $\boldsymbol{T}_{\text{aug}}$, response $\boldsymbol{Y}$ and weights $\boldsymbol{A}$.

$$\beta = (\boldsymbol{T}'_{\text{aug}} \cdot \boldsymbol{A} \cdot \boldsymbol{T}_{\text{aug}})^{-1} \cdot \boldsymbol{T}'_{\text{aug}} \cdot \boldsymbol{A} \cdot \boldsymbol{Y}$$

6. Calculate the vector of residuals e_i for every i sample using the augmented scores $t_{\text{aug},i}$ and regression coefficients β.

$$e_i = Y_i - t_{\text{aug},i} \cdot \beta$$

7. Calculate the free parameter of the RBF kernel ρ^2 using the residuals e_i, σ, a hyperparameter that needs to be tuned by CV and n, the number of samples

$$\rho^2 = \frac{\sigma^2}{n} \cdot \sum_{i=1}^{n} e_i$$

8. Calculate the new weights for the diagonal weight matrix $\boldsymbol{A}$ using the residuals e_i, ρ^2 and an RBF kernel.

$$A_{ii} = \frac{\exp\left(\frac{-\|e_i\|^2}{2\rho^2}\right)}{2\rho^2}$$

9. Repeat from step 2 for a desired number of iterations, or until ρ^2 gets too small (the model fit is good enough).

2.3.4 Interval Partial Least Squares (IPLS)

The method was introduced by Nørgaard et al. (2000)[16] in order to improve the interpretability of PLS models. Namely, for spectroscopists it can be of use to know the solution in the original variables as opposed to latent ones extracted by PLS. IPLS also aims at improving accuracy on noisy data by selecting only regions with strong predictive power.

An interval PLS model is a simple extension of the base PLS. The input spectra is divided into m equal intervals, and then a separate PLS model is fit on every interval. The ideal number of components to extract is expected to be different for all of these local models, but the selected model dimension has to be common in order to make a comparison possible. Next, all the local models are evaluated using cross-validation and in case of a forward-stepwise approach the best one is selected, for a backward one the worst one is discarded. In the following iterations the leftover intervals are again one-by-one combined with the selected ones and a PLS model is fit for the combined intervals to extract a number of latent variables. For a forward selection process the output of IPLS could be interpreted as a sequence of PLS models on increasingly larger spectra with every iteration, as opposed to a combination of more and more of local PLS models on different intervals. Ideally, since the spectra is reduced, the number of extracted components should also be the same or reduced.

2.3.5 Orthogonal Signal Correction (OSC)

Orthogonal signal correction was first introduced in 1998 by Wold et al.[29] and Sjöblom et al.[22] in the same journal issue describing the same concept with two similar but different approaches. Unlike the PLS and MCW-PLS techniques which are used for bi-linear modeling, OSC is a preprocessing technique. However, I decided to include it with the dimensionality reduction rather than the other preprocessing methods, as it is very different from them; the main difference being that it is a supervised approach. It also involves reduction in dimensionality, albeit in the underlying data structure space, not in input space. That is, it removes latent variables from the data, but the number of input variables stays the same, however the data points move closer in input space, their variance reduces.

As with the other preprocessing approaches, due to Beer-Lambert law it is assumed that the relevant information is linearly correlated with concentration, therefore a preprocessing technique should aim at removing information that is not. OSC aims to explicitly filter out information from the data that is orthogonal to the dependent concentration variables. Since the filtering is done using both X (spectra) and Y (concentration) variables, it is a supervised learning technique. Consequently, it requires cross-validation (as laid out in 2.5) when tuned, as Y is not available after the training stage. The way OSC works is closely related to PLS, both in being "reverse PLS" but also in the nonlinear iterative partial least squares (NIPALS) algorithm used to perform it.

It might be logical to ask how using OSC leads to a different result from using PLS, considering they both aim at the same goal from a different angle. Indeed, a 2002 paper by Svensson[25]

et al. analyzing different OSC algorithms notes that this method is more suited to reducing the number of components used for a later PLS step, rather than improve the modeling accuracy. The advantage is in concentrating the useful information in fewer components, improving interpretability, not added predictive performance.

2.3.5.1 OSC Algorithm

In the following I will use the notation introduced for PLS. The vectors t, w and p are the factor scores and weights, and final loadings. There are many OSC algorithms, the one I will use is the first one introduced by Wold et al.[29]. The idea is similar to the ordinary PLS algorithm except, instead of calculating loadings that maximize covariance between $\boldsymbol{X}$ and Y, here they will instead be calculated as to minimize it, i.e., to get as close to orthogonality between t and Y as possible.

The algorithm to calculate the first latent variable to be removed goes as follows:

1. Calculate the first principal component of the current $\boldsymbol{X}$ as described in 2.3.1 and set t_1 equal to its score vector. This ensures that the starting score t_1 is an optimal linear summary of $\boldsymbol{X}$.

2. Orthogonalize t_1 against Y to get t_1^*. This t_1^* is no longer a linear combination of $\boldsymbol{X}$.

$$t_1^* = (\mathbf{1} - Y \cdot (Y' \cdot Y)^{-1} \cdot Y') \cdot t_1$$

3. Use t_1^* as response variable and calibrate a PLS model with a given number of components with $\boldsymbol{X}$ as predictors using the algorithm in 2.3.2.1 (any PLS algorithm works). The number of these internal components to extract need to be tuned by cross-validation. Use the slope coefficients from the PLS model as loadings vector w.

4. Recalculate t_1 using $\boldsymbol{X}$ and w.

$$t_1 = \boldsymbol{X} \cdot w$$

5. Check convergence of t_1 to t_1^*. If not yet converged, repeat from step 2.

6. Project the predictor variables $\boldsymbol{X}$ on t_1^*, obtaining p_1, a final loadings vector.

$$p_1 = \frac{\boldsymbol{X}' \cdot t_1^*}{t_1^{*\prime} \cdot t_1^*}$$

7. Remove the component from $\boldsymbol{X}$.

$$\boldsymbol{X}_{\text{osc}} = \boldsymbol{X} - t_1^* \cdot p_1'$$

While the original papers by Wold et al. [29] and Sjöblom et al. [22] both recommend removing multiple components, in practice this is usually not necessary for the above algorithm, because

the number of components in the internal PLS step already controls the complexity. In essence, this method pre-summarizes the orthogonal information in the component before removal, instead of doing it component-by-component.

2.4 Support Vector Machines (SVMs)

The previously mentioned preprocessing and modeling methods all stem from the same paradigm derived from Beer-Lambert law. The underlying model connecting spectra to concentrations should be linear, therefore the modeling stage has to use some form of linear regression. Should the connection in the data be non-linear, it needs to be linearized through preprocessing. While this is a reasonable way to solve the problem, there are two drawbacks. One, the preprocessing methods are not very flexible and make strong assumptions about both the nature of the noise that needs to be filtered and the way the data should be transformed to get linearity, with no way to verify if any of them is actually correct. Two, both the preprocessing and linear regression methods with their hyperparameters need to be tuned by cross validation. This means up to 5 different hyperparameters in total beyond choosing the methods to use (3 parameters for SG filtering and 2 for MCW-PLS), while training data are very limited.

An alternative could be using non-linear modeling, which would do both steps in one. Nowadays the most popular non-linear modeling framework is neural networks, and especially deep learning. Unfortunately, deep learning is very data greedy, making it unfeasible with the data set available. One could do transfer learning, pretraining a neural network on a different data set, but I also do not have an applicable transfer training set available of the required size.

Support vector machines (SVMs) are another popular non-linear modeling approach. While SVMs do not represent state of the art performance compared to deep learning, they are still very powerful models and were actually designed for smaller sample sizes. In fact, as it will be explained later on, SVMs mainly scale in computational difficulty with the number of samples, not the input dimensions and have no trouble if the number of variables exceeds the number of samples. The reason why a feasible SVM requires less samples than a neural network (and especially deep learning) comes down to having less free parameters to cross-validate, orders of magnitude less model parameters, and a unique global optimum for any set of free parameters that can be found by quadratic programming.

Naturally, choosing a model and tuning some hyperparameters is still required but for a support vector machine regression one only needs to tune the ϵ-tube, penalty term c and one or two kernel paramaters by cross-validation. Unfortunately, this approach is not without drawbacks either. While an SVM works better on smaller datasets than neural networks, the data at hand is perhaps still too small. Additionally, an SVM regression has far lower interpretability than a PLS model due to the way it normally operates as instance based learning in its dual form, modeling on datapoints as opposed to variables. This, however, should not be a big issue, as the primary purpose is prediction, not inference.

SVM models get their name from a subset of training points, the support vectors (SVs),

they are using for prediction. As it will be shown below, in the dual form an SVM can be seen as instance based learning. For classification problems (for which SVMs were originally formulated) the SVs are the data points that either lie on the decision margins, or are misclassified (have non-zero error), as the other points are unnecessary for defining the boundary between classes. Using only the SVs for the model makes SVMs sparse, helping interpretation and controlling model complexity. For regression problems with a continous response the problem is more complicated. In the following I will rely on the description by Suykens et al. (2002)[24].

Skipping the introduction for the more simple classification case, an SVM regression model is generally defined as follows. The initial goal is fitting a linear trend of the form $w' \cdot x_k + b$ with minimal ξ error, where w is a vector of coefficients giving the slope, b a constant term and x_i a single predictor vector. This gives us the objective function of the primal formulation of an SVM regression.

$$\min \frac{1}{2} w' \cdot w + c \sum_{k=1}^{N} (\xi_k + \xi_k^*)$$

The formula has two parts. Firstly, $\min \frac{1}{2} w' \cdot w$ means that the slope coefficients should be minimal. This is the same regularization idea that is used for many regression methods, preventing overfitting. Indeed, the primal form of an SVM with absolute errors is that of a Lasso regression! Secondly, $c \sum_{k=1}^{N} (\xi_k + \xi_k^*)$ means that model errors (positive and negative both) should be minimized, and the ratio of penalization between the magnitude of w and the errors is defined by the parameter c, which is one of the parameters controlling model complexity. The errors for the kth observation, ξ_k, are defined with Vapnik's ϵ-insensitive error function:

$$|y - f(x)|_\epsilon = \begin{cases} 0 & \text{if } |y - f(x)| \leq \epsilon \\ |y - f(x)| - \epsilon & \text{, otherwise} \end{cases}$$

$$y - f(x) = \begin{matrix} \xi & \text{if } y - f(x) \geq 0 \\ \xi^* & \text{, otherwise} \end{matrix}$$

The ϵ threshold for the error function is necessary for model sparsity. In classification sparsity arises from discarding points correctly classified and not on the margin as they do not contribute to the model. With a linear trend regression any noise would result in all points except two becoming support vectors, as they all have an error associated. By creating an ϵ tube based on the noise level of the data, the number of points becoming SVs can be controlled, effectively controlling model complexity through sparseness (a model with higher ϵ is more "flat").

Previously I stated that the goal is to fit a linear trend $w' \cdot x_k + b$. Naturally, the regression problem is not linear, otherwise an SVM would not be required. In the next step the goal needs to be modified in such a way that it can accommodate a non-linear fit in a linear regression. For this purpose the non-linear mapping φ is introduced. This function can be a wide range of non-linear functions, or in the simplest case a linear kernel. By transforming x with φ a very

flexible regression model can be achieved. In the next step some constraints are added that define the relationship between the regression coefficients (w and b) and the errors (ξ_k and ξ_k^*). Using the previous error definition the objective function is subject to the below constraints. Note that there are two constraints in order separate positive and negative errors, which is the same as using an absolute value on the errors.

$$\begin{aligned} y_k - w' \cdot \varphi(x_k) - b &\leq \epsilon + \xi_k \\ w' \cdot \varphi(x_k) + b - y_k &\leq \epsilon + \xi_k^* \\ \xi_k, \xi_k^* &\geq 0 \end{aligned}$$

To solve the primal problem, first a Lagrangian needs to be constructed.

$$\begin{aligned} &\mathcal{L}(w, b, \xi, \xi^*; \alpha, \alpha^*, \eta, \eta^*) = \\ &\frac{1}{2} w' \cdot w + c \sum_{k=1}^{N} (\xi_k + \xi_k^*) - \sum_{k=1}^{N} \alpha_k (\epsilon + \xi_k - y_k + w' \cdot \varphi(x_k) + b) \\ &- \sum_{k=1}^{N} \alpha_k^* (\epsilon + \xi_k^* + y_k - w' \cdot \varphi(x_k) - b) - \sum_{k=1}^{N} (\eta_k \xi_k + \eta_k^* \xi_k^*) \end{aligned}$$

with Lagrange multipliers $\alpha_k, \alpha_k^*, \eta_k, \eta_k^* \geq 0$ giving the following saddle point:

$$\max_{\alpha, \alpha^*, \eta, \eta^*} \min_{w, b, \xi, \xi^*} \mathcal{L}(w, b, \xi, \xi^*; \alpha, \alpha^*, \eta, \eta^*)$$

and conditions for optimality:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0 & \rightarrow w = \sum_{k=1}^{N} (\alpha_k - \alpha_k^*) \varphi(x_k) \\ \frac{\partial \mathcal{L}}{\partial b} = 0 & \rightarrow \sum_{k=1}^{N} (\alpha_k - \alpha_k^*) = 0 \\ \frac{\partial \mathcal{L}}{\partial \xi_k} = 0 & \rightarrow c - \alpha_k - \eta_k = 0 \\ \frac{\partial \mathcal{L}}{\partial \xi_k^*} = 0 & \rightarrow c - \alpha_k^* - \eta_k^* = 0 \end{cases}$$

This constrained optimization problem cannot be directly solved in the above primal form due to the kernel function. If non-linear mapping φ is a polynomial function, then the primal regularized regression problem could be solvable, but the computational complexity scales with the input dimensions, which blow up from the polynomial terms. However, for an RBF kernel φ maps the input space into an infinite dimensional feature space, making the primal problem not just impractical, but unsolvable. In order to circumvent this obstacle, the constrained optimization needs to be solved in the dual formulation using the so called "kernel trick". First, the above conditions for optimality can be reformulated into the following dual quadratic

programming (QP) problem:

$$\max_{\alpha,\alpha*} \mathcal{Q}(\alpha,\alpha_*) = -\frac{1}{2}\sum_{k,l=1}^{N}(\alpha_k-\alpha_k^*)(\alpha_l-\alpha_l^*)\,\varphi(x_k)'\cdot\varphi(x_l) - \epsilon\sum_{k=1}^{N}(\alpha_k+\alpha_k^*)+\sum_{k=1}^{N}y_k(\alpha_k-\alpha_k^*)$$

subject to constraint:

$$\sum_{k=1}^{N}(\alpha_k-\alpha_k^*)=0$$

Then the kernel trick is applied based on the Mercer theorem. This allows for replacing the expression $\varphi(x_k)'\cdot\varphi(x_l)$ using:

$$K(x_k,x_l)=\varphi(x_k)'\cdot\varphi(x_l)$$

This K kernel function (with its free parameters) is the last ingredient defining an SVM. Specifically, for an SVM model a kernel function is chosen in advance such that the Mercer theorem can be applied, rather than trying to find the kernel function for a chosen non-linear mapping φ. This also means that φ itself does not need to be explicitly defined. K can be interpreted as a similarity measure between two points, which means that an SVM is a non-parametric estimator in the dual form.

It needs to be emphasized, that I will focus on SVM models with an RBF kernel in the Thesis, to the exclusion of linear, polynomial and all other kernels. The rationale behind the decision is that the RBF kernel is considered to be the default approach for non-linear problems and has a proven track record over many different applications. In fact, Wang et al. (2004)[27] proved that an RBF kernel SVM is a universal approximator to arbitrary functions on a compact set. Since I specifically aim for non-linear modeling with SVMs, it does not make sense to use a linear kernel, and I do not have any reason to believe that a polynomial kernel could be more useful than an RBF one. Additionally, properly tuning SVM models is computationally expensive, and investigating additional kernels for completeness's sake is infeasible on my desktop.

Once K is substituted in, the QP can be solved without calculating w or $\varphi(x_k)$, giving the unique and global solution support vectors α, α_* and constant b, with the following estimator:

$$y(x)=\sum_{k=1}^{N}(\alpha_k-\alpha_k^*)\,K(x_k,x)+b$$

When using an ϵ-insensitive absolute loss function the solution is also sparse. One could however, use a squared loss instead, which is standard in most regression models, leading to a so called least squares support vector machine (LS-SVM). This turns the primal formulation into a ridge regression and the dual solvable by a linear Karush-Kuhn-Tucker (KKT) system as

opposed to a QP problem. This is very advantageous, as the algorithm to solve the QP problem involves iteratively solving KKT systems, equivalent to solving multiple LS-SVM problems. Unfortunately, using such an approach comes at the cost of loss of sparsity, but it can be restored by pruning. Pruning sorts the solution support vectors by size, removes an increasing number of data points corresponding to the smallest SVs and re-estimates the model. This process is iterated until some performance index degrades. However, for the purposes of the Thesis I am relying on an absolute loss based SVM formulation due to preexisting fast C++ implementations wrapped in Python, even if an LS-SVM could, if properly implemented, work faster.

2.5 Cross-validation

During model building a decision has to be made on the number of explanatory variables (latent variables in this case) to retain. Additionally, another choice has to be made between different preprocessing methods, and for some of them there are also parameters to tune. These decisions should be based on the prediction accuracy of the resulting models. In order to evaluate the predictive performance, the data needs to be split into calibration (also called training) and validation sets, so that the competing models are built using the calibration data and evaluated on the validation data. This is necessary in order to avoid overfitting, the phenomenon of models always improving their predictive performance on the training data when using more degrees of freedom by capturing more variability, even if that variability is noise. There are many different ways of splitting the data into training and validation sets, the most popular ones being:

- exhaustive methods which learn and test on all possible ways to divide the original data (e.g. leave-one-out cross-validation);
- k-fold cross-validation, where the original sample is first randomly partitioned into k equal sized subsamples and then of the k subsamples, a single subsample is retained as the validation set, and the remaining $k-1$ subsamples are used as training set;
- repeated random sub-sampling validation, where the original data is randomly split into training and validation sets multiple times.

For spectroscopy sometimes another method, cross-validation across measurement day (MD CV) is necessary, because spectroscopic measurements are usually collected over several different moments in time. As demonstrated by Kemps et al. (2010)[12] NIR spectra recorded at different times can have variation not related to the relevant chemical or physical properties of the samples (eg temperature), which a predictive model trained on NIR data might pick up, when these variations are also correlated with concentrations. For example if one day samples with above average protein content are investigated, and that day temperature and humidity are also higher than usual, a statistical model might simply learn to connect spectral changes caused by higher temperature to protein content. This is especially likely in smaller samples, as there

are going to be less examples available for all the different factors connected to experimental conditions. A classical CV approach is not going to help with this problem, as with a random split both the training and validation sets will contain samples with higher concentrations and the correlated spectral fingerprints of the unusually high temperature, allowing for lower validation error.

Since in general it is not possible to control for all experimental conditions or to remove these time specific effects from the spectra, the cross validation needs to be performed across measurement days. This means generating the CV folds by iteratively taking one time-group of samples as validation set, and using the rest as training set. This equivalent to a k-fold strategy, but the subsamples are the different time-groups, not randomly chosen.

Initial tests performed on the dataset suggested that this phenomenon is likely to be true for the data at hand, as both random subset and k-fold CV tend to give somewhat lower prediction error for any choice of modeling framework. Consequently, in the following I will exclusively focus on MD cross-validation.

Chapter 3

Models and Methods

3.1 Measurement device

The device used to take the measurements is an Fourier-transform infrared spectrometer (FTIR) device. While explaining the principle of FTIR is beyond the scope of the Thesis, it is suffice to say that it is the standard method for IR spectroscopy due to its affordable cost, high accuracy and fast speed. It works in the 950 to 1,650 nm range, as described in its product sheet [7], which means that even within the NIR region it only covers the middle part. The measurements are diffuse reflectance values, as is usually the case for solid organic samples. Following Pasquini (2018)[18] the method can be described as follows. In this type of measurement the light source and the detector are at the same side of the sample. The material is placed in a glass petri dish to avoid plastic noise. The NIR radiation is cast on the samples, the interaction between the radiation and the sample constituents takes place and the radiation leaving the sample is collected and analyzed. The absorption and the scattering of the radiation are the main phenomena affecting the reflectance spectrum. As explained earlier, an advantage of this presentation mode is that it works well with highly absorbing or highly scattering samples. In practice this means that the technician can simply pour a large amount of powder into a container, as opposed to depositing only a thin layer, which is consistent for all samples. The downside of diffuse reflectance is that the noise to signal ratio tends to be worse than for transmission, as less of the energy emanating from the source reaches the detector. Additionally, it is also more expensive than transmission spectroscopy [2].

3.2 Material

The material analyzed is a protein powder feed product produced from black soldier fly larvae. It has a homogeneous consistency similar to dry dirt when it comes to particle size, which shows up as an additional noise factor in measurements. The predictor variables in the dataset of interest consists of 116 NIR diffuse reflectance measurements for dry matter content and 115 for protein content at 128 different wavelengths ranging from 938.367 nm to 1,654.281 nm at equidistant intervals, measured on 4 different days. The response variables come from lab measurements of dry matter and protein content for the same samples. Dry matter can intuitively best predicted as dry matter% $= 1 -$ water%, instead of estimating all the different chemicals in the sample, therefore in the following I will refer to the dry matter response as water content interchangeably.

3.2.1 Dry Matter / Water

In terms of practicality water is very easy to detect and analyze because it absorbs very strongly in the IR region, when compared to organic molecules. Water is a simple molecule with a single type of bond, O-H. Due to this fact, and that it absorbs very well in IR, it is possible to measure water content from even a single wavelength. According to Azo Materials, a manufacturer of NIR measurement devices [10], the two main regions of absorption are at $\sim 1,430$ nm for the

first overtone and at $\sim 1,900$ for a stretching + bending combination, with the latter having multiple times higher rate, giving the best accuracy. Unfortunately, the measurement device used for the dataset only measures up to 1,650 nm, which leaves variables $\sim 1,430$ nm as the only viable predictors. As organic compounds present in the analyte likely contain O-H bonds themselves, whose first overtone is also expected to be close by, only limited accuracy is expected from modeling, which could likely be further improved by investing in a NIR device capable of measuring around $\sim 1,900$ nm. In fact, once the exact wavelength is tuned in based on temperature and pressure, even a simple photometer (a device measuring only at a single wavelength) could be used for prediction, as opposed to a spectrometer. Additionally, a MIR measurement device could provide predictors based on primary vibrations.

3.2.2 Protein

Protein is not a specific chemical compound, rather a group of large bio-molecules consisting of one or more long chains of amino acids. As such, the specific amino acids themselves need to be investigated in order to predict ex-ante which wavelengths should be measured to model protein content. The amino acid profile of black soldier fly larvae can vary considerably based on the type of feed used for rearing [21], therefore no general composition can be assumed, and there was no detailed profile available from Protix either. As a result, I was unable to identify key regions of the spectra where absorption peaks are expected. However, even with knowledge of the amino acid profile of the samples it would be a very complex task to list all the absorbing wavelengths due to the large molecule size and number of bonds. As such, even if predictive modeling is possible on the available spectra, the mathematical model linking spectra to protein content is expected to be complex and based on a large number of wavelengths.

3.3 Methodology

The modeling can be separated into two stages. In the first stage one or more unsupervised preprocessing techniques are applied on the data, and in the second a supervised model is tuned for prediction using cross-validation. The performance of preprocessing and the predictive model are scored together, based on the CV error of the best parameter setup found in the second stage. This two stage process is performed for every combination of preprocessing and predictive modeling techniques considered for the Thesis, over a range of potential free parameter values. The only notable exception is when using OSC for preprocessing, because it is a supervised technique itself. Therefore, in that case the preprocessing and modeling are CV tuned together in one stage, though this only makes a difference for the actual Python implementation, not the theoretical process.

From all methods outlined in the literature review, I decided not to use Norris-Williams derivatives due to the expected low performance from a simple moving average flattening spectral peaks, which is avoided by Savitzky-Golay smoothing and derivatives. Similarly, I will

not use PCA for dimensionality reduction, because PLS is implementing the same idea, but with considering the correlations with the concentrations, so there is little reason to expect a competitive performance from PCA. I will also not use a simple baseline correction, as there is no clear baseline in the data. Still, mentioning these more simple techniques was necessary to better illustrate the advantages of models actually used in the Thesis, and also helps better understand them.

On the other hand I will also try a forward stepwise selection linear regression (FSSR) model to get a sort of baseline performance, which can give a better idea about the benefit of using more sophisticated and domain specific methods. As readers should be familiar with simple ordinary least squares regression and since I will also not analyze the FSSR results, I did not detail it in the literature review. It is suffice to say that the selection criteria for including variables during model fitting is based on a 5-fold cross-validation error. More details will be given with the specific software implementation in 3.4.2.

The list of methods investigated in the following is summarized in Table 3.1.

Preprocessing	Modeling
Conversion to absorbance	PLS
Detrending	MCW-PLS
MSC	IPLS
SNV	FSSR
SG smoothing	SVM
SG derivatives	
OSC	

Table 3.1: Methods investigated

Model performance is evaluated exclusively using cross-validated metrics, predictive accuracy will not be reported on training data. The training and validation sets are generated based on measurement day groups as explained in 2.5. As mentioned before, the reason for using MD CV is due to generally higher scores using other CV methods. Namely, k-fold approaches with sequential folds only give slightly better results, as the data is sorted by date by default, but random folds are definitely more optimistic, especially for protein content. This can be explained by a simple analysis of variance (ANOVA) on measurement day and concentration (see Tables 3.2 and 3.3). For protein the null hypothesis of no difference between group means can be confidently rejected with $p = 0.00039$. This is especially significant, as models on protein turned out to have rather low predictive power using spectra in the first place.

With MD CV the number of samples in each fold differs depending on the number of measurements for each day. The error over all the folds is calculated as a simple unweighted average of the individual fold errors, as I do not want to make the model focus on any measurement day in particular. This means that aggregating all the predictions and evaluating them against the observed concentrations would give a somewhat different accuracy metric.

Unfortunately there was no independent test set available for a final evaluation. A possibil-

Groups	Count	Mean	Variance
17/01/2020	28	96.358	0.831
24/01/2020	30	96.612	1.071
31/01/2020	18	97.155	1.414
11/02/2020	40	96.400	1.940

Source of Variation	SS	df	MS	F	P-value
Between Groups	8.633	3	2.878	2.104	0.104
Within Groups	153.158	112	1.367		
Total	161.790	115			

Table 3.2: ANOVA of dry matter content with measurement day groups

Groups	Count	Mean	Variance
17/01/2020	28	56.363	6.357
24/01/2020	29	55.326	5.501
31/01/2020	18	55.501	1.323
11/02/2020	40	54.275	1.736

Source of Variation	SS	df	MS	F	P-value
Between Groups	73.987	3	24.662	6.583	0.0004
Within Groups	415.867	111	3.747		
Total	489.854	114			

Table 3.3: ANOVA of protein content with measurement day groups

ity would be withholding a portion of the data, but due to the measurement day CV scheme the minimum amount would be 1 fold out of the 4 available, which I deemed too expensive, especially as the sample size is already very low even without considering the CV groups. As a result, there will be no test set evaluation. It must be emphasized that error on validation sets is likely to be biased downwards compared to the true error, as models are tuned to minimize CV error and can thus indirectly overfit the data, especially with such a small sample.

The first metric I used to evaluate performance is the root mean squared error (RMSE), calculated as $\sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)}{n}}$, where y and $\hat{y}$ stand for observed and estimated outcomes respectively. The other metric is the residual prediction deviation (RPD), which is the standard deviation of the sample outcomes divided by RMSE. While RPD is going to be inversely proportional to RMSE, and thus lead to the same ranking between models, it has the advantage of giving a dimensionless measure of accuracy that is comparable across different applications. As such, it is used as the standard way to report the quality of a model in chemometrics. According to a tutorial on RPD by Williams P. (2014)[28], in NIR spectroscopy due to complications with sample preparation, sample presentation to the instrument, difficulties with reference testing and inherently low variance in the sample set, RPD values of as high as even 3.0 may be difficult to obtain. As a result he recommends using the interpretation in Table 3.4 for NIR spectra of more complex materials. That being said, in the end it will be entirely up to Protix to decide what is or is not a good performance, namely what tolerance they have for prediction error.

Even an otherwise high RPD could be too low for their specific processes. Referring to RPD is mostly useful for a neutral reader. A big downside of RPD is that it is very sensitive to outliers, namely a high RPD can be achieved by having even a single extremely high or low concentration, inflating sample standard deviation. To avoid such a pitfall, the data at hand, and the resulting models' residuals will be inspected visually.

RPD	Classification	Application
0.0-1.9	Very poor	Not recommended
2.0-2.4	Poor	Rough screening
2.5-2.9	Fair	Screening
3.0-3.4	Good	Quality control
3.5-4.0	Very good	Process control
4.1+	Excellent	Any application

Table 3.4: Interpreting the RPD for complex NIR applications

Besides these metrics, I will also look at some diagnostics for the models with the best performance based on their RMSE/RPD:

- observed vs predicted concentrations on validation sets
- predicted concentrations vs residuals on validation sets
- Q-Q plot of validation residuals
- (externally) studentized residuals on entire dataset
- studentized difference in fits (DFFITS) on entire dataset.

A studentized residual is the quotient resulting from the division of a residual by an estimate of its standard deviation. This is useful, because it standardizes the size of the residuals, and allows the use of rule of thumb cutoffs to identify relatively large values. External studentization means that for the estimation of residual variance, the residual in question is excluded from the estimate. The rationale behind external studentization is that an outlier is easier to detect if it is not allowed to inflate the residual variance. The formula for calculating externally studentized residual e_i^* is the following:

$$e_i^* = e_i \sqrt{\frac{n-m-1}{\mathrm{SSE}\,(1-h_{ii}) - e_i^2}}$$

where e_i are the residuals calculated as $\hat{y}_i - y_i$, SSE is the residual sum of squares $\sum_{i=1}^{n} (\hat{y}_i - y_i)^2$, n the number of samples, m the number of parameters and h_{ii} the diagonal of the hat matrix $\mathbf{H}$ calculated as the following in a weighted least squares model:

$$\boldsymbol{H} = \boldsymbol{X} \cdot (\boldsymbol{X}' \cdot \boldsymbol{A} \cdot \boldsymbol{X})^{-1} \cdot \boldsymbol{X}' \cdot \boldsymbol{A}$$

Detection of outliers is done by observing that if the majority of the data points follows a linear model with normal errors, these studentized residuals should approximately lie in the interval ± 2 with a confidence of 95% and in ± 2.5 with a confidence of 99%.

DFFITS measures the leverage the ith observation has on the ith fitted value. However, a large DFFITS does not necessarily imply a problem, as it can belong to a good leverage too. It is calculated from the studentized residual e_i^* by multiplying it with the leverage factor $\sqrt{\frac{h_{ii}}{1-h_{ii}}}$. Points with DFFITS outside the interval $\pm 2\sqrt{\frac{m}{n}}$ are considered influential.

3.4 Python Implementation

The modeling will be done exclusively in Python at the request of Protix, as their internal systems already run in Python, and unlike the more standard Matlab tools, it is open source. Since much of the chemometrics tools are not yet publicly available in Python, manual development was necessary. In the following I will detail the specific implementations used for the Thesis. All the modeling was performed on an Intel® Core™ i5-7300HQ Processor (4 cores, 4 threads, 2.50Ghz) with 8GB of RAM. The Python distribution used was the 3.7.4 64bit version, with the following list of general libraries:

- sys
- math
- random
- numpy
- pandas
- matplotlib, mpl_toolkits and seaborn
- scipy
- sklearn

3.4.1 Preprocessing

For Savitzky-Golay filters I could rely on the existing signal processing tools available in the SciPy library (scipy.signal), as these techniques are used widely and beyond the context of chemometrics. On the other hand multiplicative scattering correction and standard normal variate are more domain specific and I could find no public Python version, but due to their simplicity could be self-implemented easily, same as the conversion to absorbance from reflectance. For polynomial detrending there were existing solutions, but the process is simple enough that I opted to write my own version.

In the case of orthogonal signal correction there were no public implementations either, but a yet-unpublished script by Valeria Fonseca Diaz was shared with me, which uses the Sjöblom

algorithm[22] as it is also the one used in the PLS_Toolbox for Matlab by Eigenvector Research, a popular library for chemometrics. However, I preferred to use the approach of Wold et al.[29] based on NIPALS, so it had to be programmed manually as well. The difference between the OSC algorithms is minimal, but the Sjöblom version is not guaranteed to always remove orthogonal components[25].

3.4.2 Forward Stepwise Selection Regression

While FSSR is one of the most basic statistical processes, none of the main Python statistics or machine learning libraries seemed to have an implementation of it. As the technique is rather simple, I manually coded it for the purposes of the Thesis. In every step the algorithm goes through all variables and calculates how much the RMSE is when the variable is added to the model. Next, the variable that gives the lowest RMSE is selected. Then, the process repeats, and the error is calculated for all remaining variables, when used with the already selected ones. This continues as long as the error keeps decreasing. The RMSE is calculated by using a 5-fold CV on the given training sample. In effect this means that when evaluating model performance, every MD based training set will be further sub-split into training and validation sets. An alternative to CV could be using an information criterion.

3.4.3 Partial Least Squares

For partial least squares Sci-kit Learn has an implementation of the NIPALS algorithm, the class sklearn.cross_decomposition.PLSRegression(). There is no Python code publicly available for SIMPLS, which is considered the default algorithm by most practitioners due to its faster computational speed. As a result, I also developed a SIMPLS based class, that follows the general syntax and logic of the Sci-kit Learn library.

3.4.4 Maximum Correntropy-Weighted Partial Least Squares

For MCW-PLS there was no public Python code available, but Valeria Fonseca Diaz, the author of the method, did develop a Python implementation, along with one for weighted SIMPLS, and she allowed me to use her scripts for the purposes of the Thesis. While the number of components are easy to tune due to the small search space (integers between from 1 to 10) and uniformity across different preprocessing scenarios, the σ is more challenging as it can be any real number and does vary across the scenarios. Still, since it is a single free parameter it can be tuned with a brute force grid search even on my desktop, but for efficiency and a more fair comparison I used the same grid search heuristic I implemented for the SVM models. For more details see 3.4.7.

3.4.5 Interval Partial Least Squares

There were no public Python implementations for IPLS, but there is one in the R package mdatools, one in PLS_Toolbox for Matlab by Eigenvector Research and also in the free Matlab iToolbox by the original authors. While, the original publication [16] did not contain a concrete description of the algorithm, I could reverse engineer the process based on the mdatools and iToolbox code.

3.4.6 Support Vector Machine Regression

For the SVM models I relied on Sci-kit Learn's sklearn.svm.SVR() class combined with another class, sklearn.pipeline.Pipeline(), for standardizing the training and validation data. This SVM class is essentially a Python wrapper for code from the C++ LIBSVM library. It should be noted that while in the theoretic introduction in 2.4 I referred to the free parameter of the RBF kernel as σ, the authors of the Python class use an alternative γ parametrization with $\gamma = \frac{1}{2\sigma^2}$.

Since parameter tuning for an SVM model is more difficult than for the other more classical statistical approaches due to the number, mutual dependence and sensitivity of the free parameters, my desktop did not have the computing power for an exhaustive grid search. To circumvent this problem I implemented a simple heuristic adaptive grid search algorithm.

3.4.7 Grid search heuristic

As mentioned before, I had a large number of different preprocessing and model combinations to evaluate, and for SVMs tuning a good model is not trivial. The main problem is the fact that the 3 free parameters of an RBF SVM regression cannot be tuned independently of the others, as all of them control model complexity in some form. Additionally, accuracy was very sensitive to even small changes in the parameters, and there seemed to be problems with many local optima. This meant that neither a rough grid search, nor some simple hill climbing optimization strategy gave good results. The heuristic algorithm I used to tune my model parameters was the following:

1. Take a vector of starting values p_0 for all $p_{0,i}$ parameters and use them to create a tensor $\mathbf{P}_0$ of values to evaluate. The order of the tensor equals the number of parameters, and every axis is defined by 10 values at equal distances in the closed interval $[p_{0,i}/C, p_{0,i} \cdot C]$, with C being a chosen scale constant, equal to 5. For an RBF SVM this means a 10 by 10 by 10 3rd order tensor. This gives a relatively wide initial search area.

2. Every parameter combination in the tensor $\mathbf{P}_0$ is evaluated by CV, and the one with the lowest RMSE is selected. This new vector of parameters is p_1.

3. If any of the $p_{1,i}$ parameters in p_1 fall outside the middle 6 deciles along their corresponding axis, then the parameter search continues and a new tensor $\mathbf{P}_1$ is created. This time the axes are defined by 10 equidistant units on the closed interval $[p_{1,i} - 5 \cdot M, p_{1,i} + 5 \cdot M]$,

with M being a scale constant equal to the order of magnitude of $p_{1,i}$ in decimals minus 1, ie $M = \lfloor \log_{10} p_{1,i} \rfloor - 1$. What this does, is that if the optimal parameter setup has any value that is at the border region along an axis, a new relatively narrow search interval is created centered around that value to refine it.

4. If any of the best parameters in p_1 are at the minimum or maximum along an axis, then the minimum/maximum of the axis is doubled/halved in that direction. This speeds up the climb towards better solutions.

5. This process is repeated until all the best parameters are in the middle 6 deciles of their respective interval, or until there have been no improvements in error for 10 iterations. At that point the best p vector is selected over all the iterations.

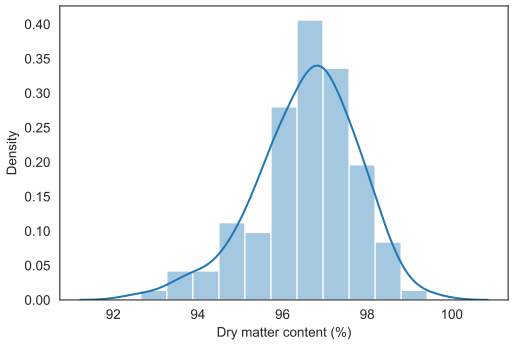
This strategy is very simple, but it has managed to reduce computation times to a feasible duration on my desktop, and has in virtually all cases resulted in better models than those tuned by simulated annealing, a single larger scale grid search or multiple iterations over manually defined search grids.

Chapter 4

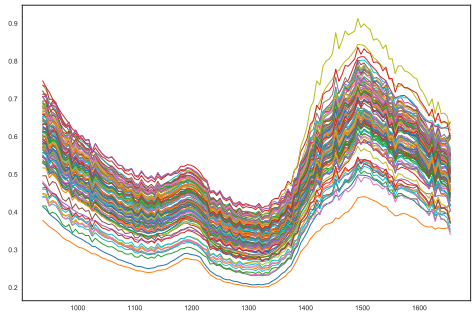
Results and Discussion

4.1 Dry Matter / Water

Looking at Figure 4.1a the distribution of concentrations is nothing out of the ordinary, it is unimodal, slightly left-skewed and most values are concentrated around the mean/mode/median. There are also no outliers, the smallest value is 92.65% and the maximum 99.42%, with standard deviation of 1.181%. This means that the sample variance is not inflated and usual RPD interpretations are applicable.



(a) Distribution of dry matter concentrations



(b) Raw spectra in absorbance (horizontal axis in nm, vertical axis in %)

Figure 4.1: Modeling inputs for dry matter content

As for the predictors, it is hard to tell if the samples contain outliers due to the high dimensionality. For certain models, that reduce the input dimensions, it might be possible to take a second look after model fitting, but the best ones turned out to still have too many predictors for simple diagnostics. The most important features in 4.1b are the roughness of the spectral lines, the lack of a clear baseline and the presence of additive/multiplicative noise, however, all of these issues can be corrected. Since not all preprocessing techniques can correct for roughness on their own, for the unprocessed spectra, detrending, SNV and MSC I will also apply quadratic smoothing afterwards, which will be done with the Savitzy-Golay method with no derivative taken. To simplify the process, the window length will be chosen based on which parameter works best for the SG derivatives. I did not test non-smoothed preprocessing for dry matter (besides the completely unprocessed base model) as both preliminary trials and theory suggests that retaining roughness is detrimental, especially for such a simple molecule as water.

	Reflectance					Absorbance				
	FSSR	PLS	IPLS	MCW PLS	SVM	FSSR	PLS	IPLS	MCW PLS	SVM
None	1.453	1.471	1.851	1.431	1.837	1.163	1.418	1.788	1.484	1.798
Smooth.	1.318	1.606	1.835	1.556	1.881	1.371	1.450	1.762	1.484	1.842
Dtr.1+sm.	1.705	1.863	1.903	1.876	2.014	1.526	1.737	1.855	1.729	2.025
Dtr.2+sm.	1.608	1.788	1.916	1.768	1.904	1.515	1.767	1.742	1.749	2.010
Detr.3+sm.	1.689	1.816	1.908	1.816	1.916	1.278	1.756	1.703	1.762	2.032
SNV+sm.	1.721	1.969	1.955	2.011	2.041	1.639	1.904	1.895	1.962	2.025
MSC+sm.	1.782	1.963	1.924	1.939	2.039	1.814	1.855	1.929	1.936	1.995

Table 4.1: CV RPD values of different preprocessing-modeling combinations when predicting dry matter content. The parameter of detrending is the polynomial order. All values were smoothed with quadratic SG smoothing, using window size of 13 for reflectance and 9 for absorbance.

The results for all the tested models are summarized in Tables 4.1, 4.2, 4.3 and 4.8, due to the large number of models, using a single table was not possible. Similarly, a table reporting the mean CV RMSE values is not included for conciseness, as transforming to RMSE from RPD preserves the relative scores. For the simple smoothing I used a window length of 13 on reflectance and 9 on absorbance, as SG filtering in 4.2 and 4.3 seemed to work best with these settings and trying more would have multiplied the scenarios to test. First, without focusing on any particular method the key takeaways are the following:

	Reflectance					Absorbance				
	FSSR	PLS	IPLS	MCW PLS	SVM	FSSR	PLS	IPLS	MCW PLS	SVM
SG 5,1,1	1.283	1.785	1.877	1.935	1.826	1.525	1.819	1.766	1.858	1.909
SG 9,1,1	1.905	1.857	1.938	1.973	1.977	1.122	1.810	1.677	1.853	2.036
SG 13,1,1	1.418	1.766	1.888	1.967	2.022	1.485	1.778	1.712	1.827	1.972
SG 17,1,1	1.556	1.783	1.848	1.937	1.963	1.466	1.726	1.658	1.878	1.939
SG 5,2,1	1.461	1.815	1.870	1.947	1.843	1.525	1.819	1.769	1.924	1.956
SG 9,2,1	1.905	1.861	1.931	1.980	1.989	1.122	1.819	1.671	1.865	2.017
SG 13,2,1	1.359	1.863	1.838	2.001	2.044	1.485	1.781	1.699	1.870	1.995
SG 17,2,1	1.556	1.829	1.867	1.982	1.920	1.395	1.612	1.677	1.840	1.958
SG 5,3,1	1.372	1.692	1.712	1.825	1.771	1.377	1.659	1.759	1.794	1.764
SG 9,3,1	1.432	1.784	1.838	1.918	1.849	1.393	1.830	1.793	1.930	1.978
SG 13,3,1	1.718	1.837	1.918	1.949	1.947	1.128	1.820	1.784	1.845	2.052
SG 17,3,1	1.450	1.863	1.834	1.979	2.021	1.351	1.823	1.685	1.935	2.027

Table 4.2: CV RPD values of different first derivative SG filtering-modeling combinations when predicting dry matter content. The parameters for the SG models are in the following order: window size, polynomial order and derivative order.

1. Transforming from reflectance to absorbance is detrimental for the linear models. SVMs seem to work better after conversion to absorbance in many cases, but not always. However, the best absorbance SVM model is better than the best reflectance one. This leads me to believe that the Kubelka and Munk transformations in fact de-linearize the data, and SVMs are simply less bothered as they can fit non-linear functions too. Interestingly, when using the similar transformation function defined for transmittance values (not included in the model comparison), the resulting absorbance spectra seems to work better for the linear models, but still worse than reflectance.

2. All linear models struggle on the unprocessed raw spectra, but SVMs still work reasonably well, which was expected as they are designed for non-linear modeling. Even so, linearizing the data seems to be beneficial even for SVMs.

3. Generally SVMs and MCW-PLS have the highest scores.

4. While the score of the linear models tends to be similar regardless of preprocessing method used, SVMs do tend to struggle for some. This is most likely due to failure to find a good local optimum, not the nonexistence of one. To save time, I did not try to get the best possible result for every scenario, as long as the search heuristic managed to get good

results for some.

5. The ranking between the modeling techniques is clear. SVMs are the most accurate, followed by MCW-PLS, IPLS, PLS and FSSR. This is hardly surprising, as performance simply corresponds to model sophistication. That being said, even the FSS regression can under ideal conditions select a good set of variables for dry matter (due to the simplicity of water's absorption band) and give performance competitive with at least PLS.

	Reflectance					Absorbance				
	FSSR	PLS	IPLS	MCW PLS	SVM	FSSR	PLS	IPLS	MCW PLS	SVM
SG 5,2,2	0.973	1.506	1.651	1.708	1.547	1.218	1.524	1.674	1.705	1.508
SG 9,2,2	1.442	1.837	1.846	1.894	1.909	1.380	1.751	1.651	1.777	1.790
SG 13,2,2	1.635	1.941	1.942	1.863	2.000	1.225	1.756	1.712	1.789	1.983
SG 17,2,2	1.396	1.949	1.953	1.895	1.952	1.505	1.758	1.664	1.809	2.036
SG 5,3,2	0.973	1.543	1.651	1.668	1.506	1.218	1.568	1.685	1.686	1.503
SG 9,3,2	1.442	1.837	1.776	1.894	1.909	1.380	1.751	1.720	1.777	1.790
SG 13,3,2	1.650	1.890	1.908	1.936	1.989	1.189	1.796	1.655	1.815	1.957
SG 17,3,2	1.405	1.921	1.954	1.913	1.980	1.488	1.723	1.616	1.801	1.979

Table 4.3: CV RPD values of different second derivative SG filtering-modeling combinations when predicting dry matter content. The parameters for the SG models are in the following order: window size, polynomial order and derivative order.

6. The overall best RPD belongs to an SVM, in fact there are more than a dozen SVM models better than the best MCW-PLS one, and in most cases the SVM model tends to be slightly better than the MCW-PLS one. Where SVMs underperform, it is most likely due to failure to converge to a good solution with the simple grid search heuristic. Despite this, the difference between SVM and MCW-PLS is rather small, and the sample size was also limited, therefore the difference in validation scores in not substantial enough to declare one better than the other.

7. OSC is by far the worst preprocessing technique. In fact, the less variance is removed with OSC, the better the results, and even in the best case it works worse than not processing the raw data at all. This is not unexpected, as in the introduction of the method it was highlighted, that it is more aimed at concentrating the useful information in fewer components for a subsequent PLS step, than improving prediction quality. An additional issue with OSC is that it assumes that any variance orthogonal to the concentrations is noise, which is true under ideal conditions, but not necessarily in practice. Should there be any non-linear information in the data, an SVM could potentially extract it.

8. The ranking between preprocessing methods is unclear (besides OSC). Namely, the highest scoring SG filters give the best RPDs, but they are closely followed by SNV, MSC and detrending with smoothing, which perform better than most SG filters. The best method also depends on specific model used afterwards. Quadratic and cubic detrending seems to be unnecessary, as it either under performs the linear variant, or gives identical results.

In the following I will select and analyze the best performing models in details, namely SVM regression and MCW-PLS.

	Reflectance					Absorbance				
	FSSR	PLS	IPLS	MCW PLS	SVM	FSSR	PLS	IPLS	MCW PLS	SVM
OSC 1,1	1.527	1.656	1.659	1.697	1.683	1.467	1.586	1.593	1.632	1.657
OSC 1,2	0.448	1.606	0.396	1.548	1.021	0.328	1.549	0.436	1.530	1.010
OSC 1,3	0.147	1.295	0.556	1.119	1.035	0.157	1.143	0.687	1.126	1.042
OSC 5,1	1.591	1.651	1.642	1.708	1.682	1.521	1.617	1.632	1.641	1.679
OSC 5,2	0.336	0.943	0.842	1.708	1.201	0.604	0.951	0.856	1.124	1.179
OSC 5,3	0.075	0.642	0.655	0.712	1.105	0.218	0.713	0.482	0.970	1.150
OSC 10,1	1.451	1.612	1.574	1.661	1.614	1.476	1.560	1.569	1.683	1.605
OSC 10,2	0.168	0.706	0.272	1.025	1.012	0.241	0.854	0.757	1.080	1.228
OSC 10,3	0.238	0.567	0.435	0.765	1.132	0.228	0.529	0.306	0.775	1.119
OSC 20,1	1.324	1.619	1.607	1.613	1.629	1.146	1.504	1.553	1.562	1.546
OSC 20,2	0.059	0.444	0.255	1.368	0.995	0.088	0.613	0.410	0.877	0.998
OSC 20,3	0.157	0.445	0.455	0.757	1.095	0.382	0.461	0.400	0.877	1.104
OSC 40,1	0.980	1.544	1.543	1.580	1.492	0.967	1.453	1.493	1.531	1.421
OSC 40,2	0.112	0.369	0.374	0.591	0.960	0.047	0.470	0.276	0.717	1.010
OSC 40,3	0.328	0.469	0.460	0.477	1.094	0.451	0.462	0.454	0.484	1.151
OSC 80,1	0.962	1.546	1.545	1.581	1.492	1.016	1.453	1.500	1.530	1.412
OSC 80,2	0.095	0.369	0.374	0.588	1.160	0.052	0.475	0.380	0.715	1.143
OSC 80,3	0.293	0.480	0.474	0.505	1.040	0.446	0.494	0.494	0.503	1.057

Table 4.4: CV RPD values of different orthogonal signal correction processed models on dry matter. The parameters for OSC are in the following order: number of internal components, number of extracted components.

4.1.1 Support Vector Machine Regression

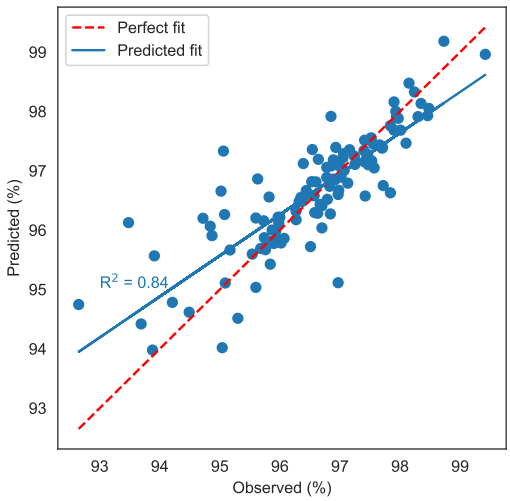
The SVM approach gave the highest RPD value of all competing models with 2.052, using conversion to absorbance, applying SG smoothing with a window size of 13, cubic polynomial approximation and first derivative. The tuned parameters are the following:

- the width of the RBF kernel, $\sigma = \sqrt{\frac{1}{2 \cdot 0.000418888888888888}} = 34.5490079778659$,
- the penalty term for the errors, $c = 25.452222222222225$,
- the width of the ϵ-tube, $\epsilon = 0.14994$.

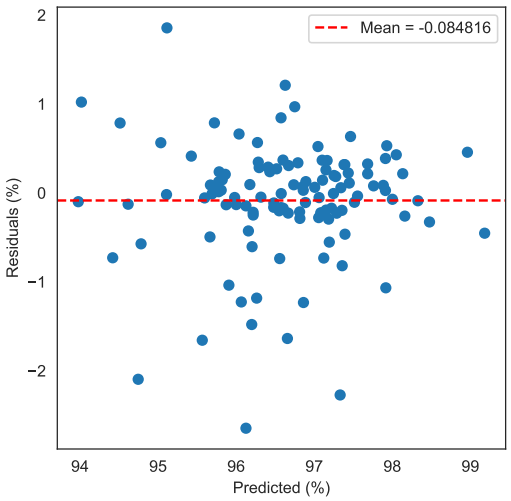
An RPD value slightly above 2 corresponds to poor performance suitable only for rough screening according to Table 3.4. Even so, this table was designed as more lenient scoring for complex materials, which certainly applies for ground larvae powder, but not necessarily to its water content. In terms of RMSE the accuracy seems somewhat better, being equal to 0.576. To put this in more practical terms, the standard deviation of the errors is 0.576% on a total concentration range of 6.769% with standard deviation of 1.181%. The biggest problem with this approach is, that it is not possible to tell what wavelengths it is actually using for prediction.

Unlike traditional least squares regression, support vector machine regression with an RBF kernel is a non-parametric approach and does not have a strong set of assumptions. The most important thing to check is the presence of outliers. Even then, the SVM model I used is based on absolute loss, and is therefore more robust. When fitted on the entire data set, the model uses 74 out of 116 sample elements as support vectors, so the solution is rather sparse, see Figure 4.2d. The majority of these support vectors (49 of the 74) has the maximal magnitude α of either 25.45 or -25.45, the rest being much closer to zero. The absolute α of a support vector is the measure of how influential the point is for prediction, which means that out of the 116 measurements 49 have equally large importance and the rest less or zero. This implies that there are no points with an obvious outlying influence on the model, just a large set of points with no or low influence. Due to how the model is formulated, this sparsity is usually expected so points having small influence is natural and even healthy.

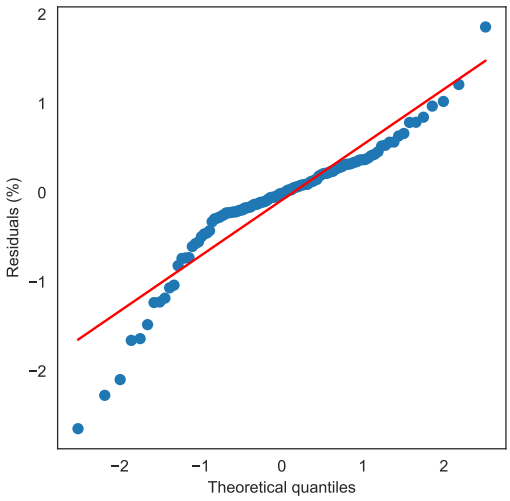
Based on Figure 4.2b there are some residuals with an overly large value (close to ± 2), but there is no clear way to tell how much leverage they actually have. The mean of the residuals is very close to zero, therefore the predictions seem mostly unbiased. Being unbiased is not a trivial property for an SVM, especially not on a validation set. This can be easily seen by the primal formulation's objective function, which is that of a lasso regression trading bias for variance. What also stands out looking at Figures 4.2a and 4.2b is that the model seems to perform better on higher dry matter (or lower water) concentrations, implying heteroscedastic errors. While this does not violate any assumptions, is definitely not ideal. Finally, based on the Q-Q plot in Figure 4.2c the residuals are slightly left-skewed (similar to the observed concentrations) when compared to a normal distribution and more heavy-tailed but also more peaked.



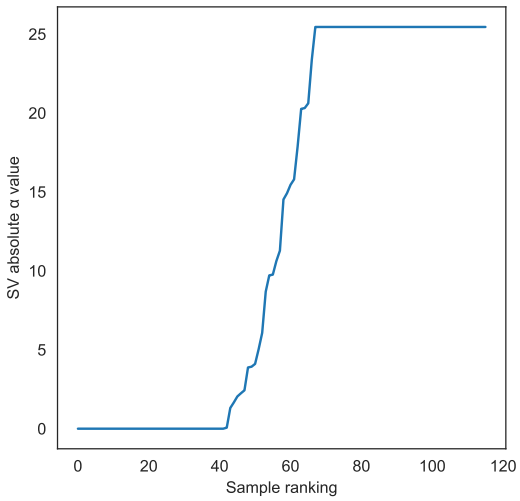
(a) Observed vs predicted based on CV values



(b) Residuals vs predicted based on CV values



(c) Normal Q-Q plot of the CV residuals



(d) Sorted absolute α values (when fitted on the entire data set)

Figure 4.2: Residual diagnostic plots for SVM regression on dry matter

4.1.2 MCW-PLS

The second best modeling technique was MCW-PLS, fitted on spectra preprocessed with SNV and smoothed with quadratic SG approximation with a window width of 13, closely followed by the one fitted on the SG filtered spectra with the same window length and polynomial degree, but a first derivative. Its RPD is 2.011, corresponding to an RMSE of 0.604%. The tuned model parameters are $\sigma = 0.10685823754789273$ and 4 latent components.

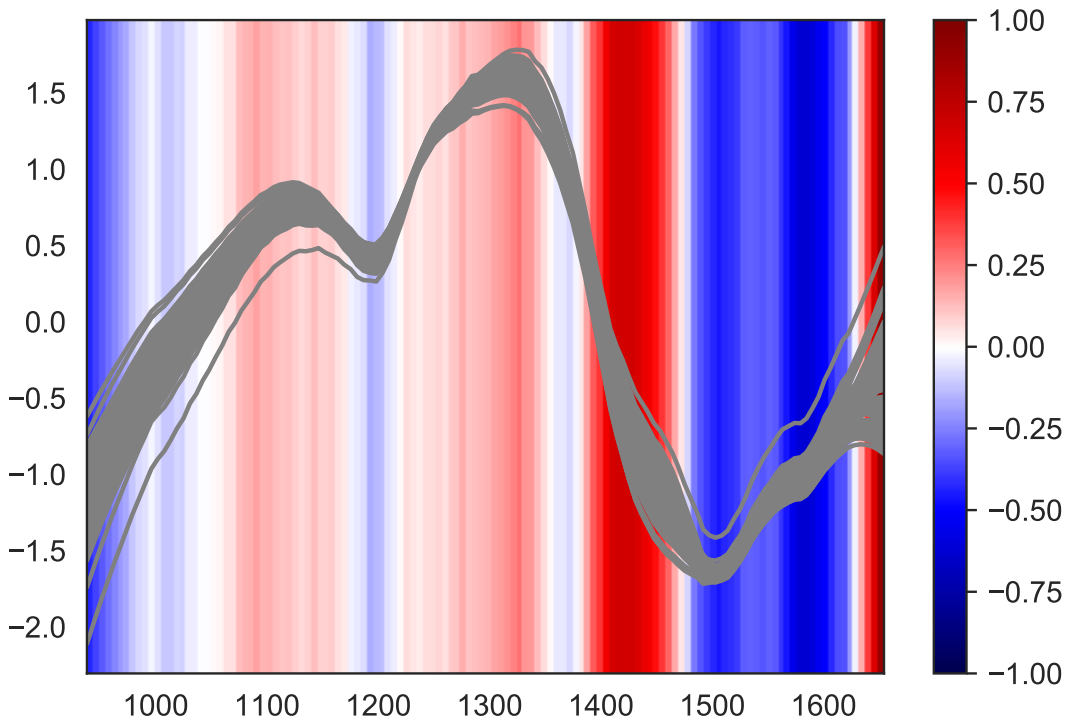
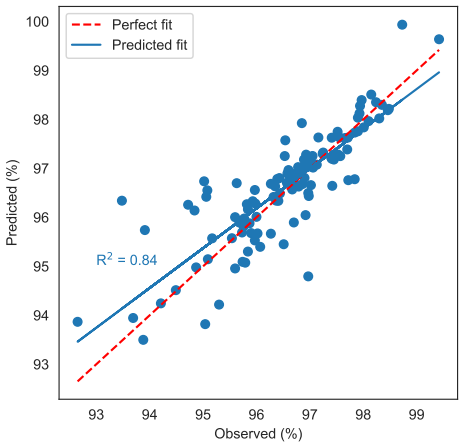


Figure 4.3: SNV processed and SG smoothed spectra in reflectance (horizontal axis in nm) colored grey, wavelengths colored according to the regression coefficients assigned by MCW-PLS

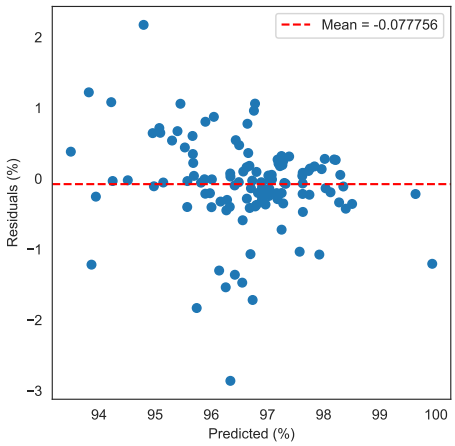
While the RPD is slightly lower than for the best SVM models, MCW-PLS has a big advantage in giving more insight into how it makes the predictions. By looking at the regression coefficients calculated for the processed input spectra for different wavelengths in Figure 4.3 the picture is quite clear. The model is using the band from 1,397.25 nm to 1,475.31 nm with positive coefficient, but also the very high end of the spectrum. Since the number of variables are multiple times less in the latter, it effectively has a smaller effect than the former region despite having larger coefficients. Moreover, there are negative coefficients for the region in-between these two positive sections, and also at the very bottom of the spectrum. Since the predicted variable corresponds to dry matter %, or 1 minus water %, but the measurements are also in reflectance with more water content implying more absorption and less reflectance, there should be a positive linear connection between dry matter % and the reflectance values. Thus the direct predictions of dry matter % come from the positive coefficients, with the negative ones interpretable more as a correction to the positive coefficients to remove the confounding effect of molecules other than water, whose absorptions bands can easily overlap with water's first O-H overtone. While the emphasis on the band around 1,430 nm is expected due to chemical properties of water detailed in 3.2.1, the ends of the spectrum are distorted both by the moving window smoothing and larger measurement noise, so ideally should not be good predictors.

The diagnostic plots in Figures 4.4a, 4.4b and 4.4c look almost the same as for the SVM

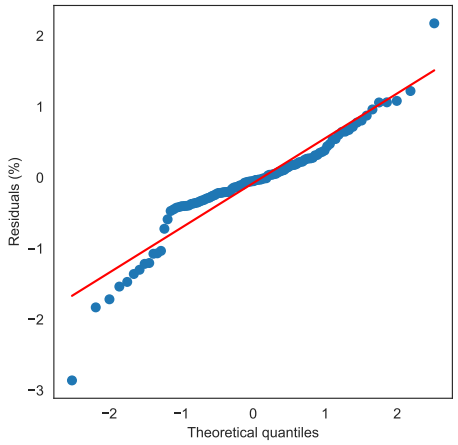
solution. Despite being a very different model, it also has similar sparsity. The extent of samples getting lower weights can be seen in Figure 4.4d. The biggest difference seems to be on the Q-Q plot in Figure 4.4c, with the residuals generally following a normal distribution more closely than for the SVM, except for the largest residual in the heavy right tail. This is probably due to the fact that MCW-PLS was specifically designed to deal with non-normal errors. However, the residuals seem heteroscedastic in this case as well looking at Figure 4.4b. A big advantage of MCW-PLS compared to the SVM model is the availability of more sophisticated diagnostics, like studentized residuals and DFFITS for detecting outliers, as the final model is simply a weighted least squares with the PLS scores and sample weights. For these values the model is fitted on the entire sample available and the training residuals are examined. In Figure 4.4e none of the residuals qualifies as truly outlying, and in Figure 4.4f there are no leverage points visible either.



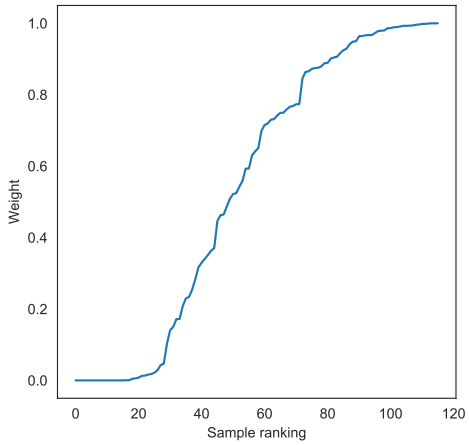
(a) Observed vs predicted values based on CV



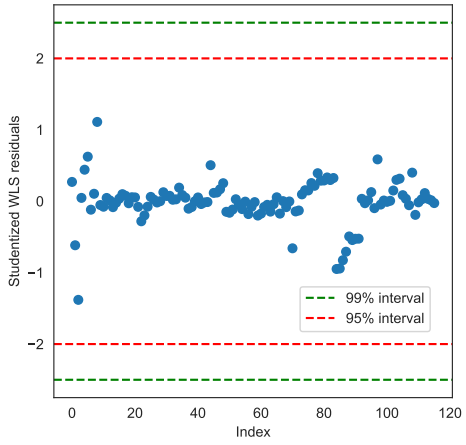
(b) Residuals vs predicted values based on CV



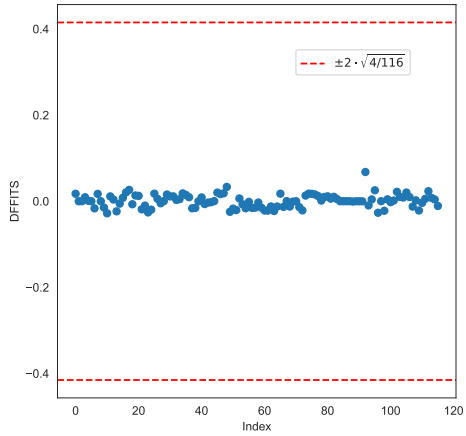
(c) Normal Q-Q plot of the CV residuals



(d) Sorted sample weights (when fitted on the entire data set)



(e) Studentized residuals (when fitted on the entire data set)

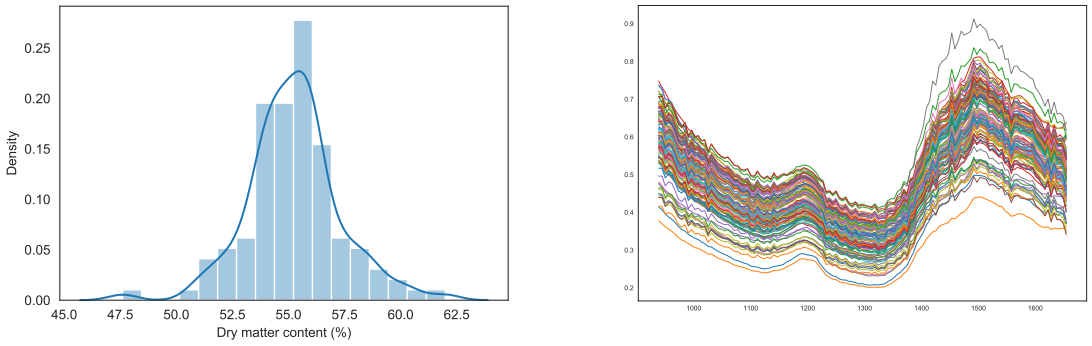


(f) DFFITS (when fitted on the entire data set)

Figure 4.4: Residual diagnostic plots for MCW-PLS on dry matter

4.2 Protein

In Figure 4.5a nothing stands out about the distribution of concentrations, it is uni-modal with most values concentrated around the mean/mode/median. While the concentrations are more dispersed than for dry matter, there are also no outliers, the standard deviation is 2.073%, the smallest value is 61.99% and the maximum 47.61%. The standard deviation is much higher than for water, but still not inflated by extreme values. As before, it is hard to tell if the predictors contain outliers due to the high dimensionality. Since the spectra is identical for the one before, expect for one less observation, it has the same issues with the roughness, the lack of a clear baseline and the presence of additive/multiplicative noise. What is different, is that initial tests suggested that smoothing does not improve predictive accuracy, except for SVMs. While theory would suggest that the roughness is a sign of measurement noise, it might actually contain useful information for protein content, as amino acids are complex chemicals with a large number of spectral peaks expected. As a result I will only perform the extra smoothing step in the SVM scenarios for detrending, SNV and MSC, using the quadratic Savitzy-Golay method with no derivative taken. To simplify the process, the window length will be chosen based on which parameter works best for the SG derivatives.



(a) Distribution of protein concentrations

(b) Raw spectra in absorbance (horizontal axis in nm, vertical axis in %)

Figure 4.5: Modeling inputs for dry matter content

	Reflectance					Absorbance				
	FSSR	PLS	IPLS	MCW PLS	SVM	FSSR	PLS	IPLS	MCW PLS	SVM
None	1.029	1.061	1.172	1.274	1.100	1.026	1.033	1.196	1.234	1.111
Detr. 1	1.014	1.136	1.196	1.158	1.343	1.013	1.120	1.183	1.123	1.209
Detr. 2	0.848	1.154	1.184	1.136	1.270	0.930	1.140	1.173	1.135	1.235
Detr. 3	0.931	1.164	1.164	1.136	1.243	0.971	1.142	1.165	1.142	1.293
SNV	0.961	1.140	1.213	1.307	1.375	0.917	1.131	1.245	1.243	1.323
MSC	1.060	1.191	1.195	1.315	1.411	1.014	1.131	1.166	1.261	1.336

Table 4.5: CV RPD values of different preprocessing-modeling combinations when predicting protein content. The parameter of detrending is the polynomial order.
*: For SVM models the values were also smoothed with quadratic SG smoothing, using window size of 13 for reflectance and 9 for absorbance.

The results for all the tested models are summarized in Figure **??**. For the SVM models with detrending, SNV or MSC I also used quadratic SG smoothing with a window length of 13 on reflectance and 9 on absorbance, as SG filtering seemed to work best with these settings. First, without focusing on any particular method the key takeaways are the following:

1. Transforming from reflectance to absorbance is detrimental once more, especially for the linear models. SVMs still work well after conversion to absorbance, in fact it gives better scores in most cases, though the best absorbance SVM model is inferior to the best reflectance one. As before, I believe that the Kubelka and Munk transformations actually de-linearize the data. SVMs probably handle it better as they are non-linear, and when they manage to find a better local optimum for absorbance inputs, it is by chance.

2. Surprisingly, MCW-PLS models seem to work best with raw data this time, much better than even SVMs. IPLS models also work quite well without preprocessing relatively speaking.

3. Generally SVM and MCW-PLS have the highest scores. The overall best RPD belongs to an SVM once more, and there are plenty of SVM models better than the best MCW-PLS one. On the other hand, for most SG filters MCW-PLS found a better solution. These two are followed by IPLS and PLS, and as expected the worst model is FSSR.

4. The ranking between preprocessing methods is unclear, the best method depends on the model applied afterwards. Generally speaking SNV, MSC and some variant of SG filtering works well for all models, and linear detrending SVMs. Quadratic and cubic detrending is unnecessary and gives poor results.

	Reflectance					Absorbance				
	FSSR	PLS	IPLS	MCW PLS	SVM	FSSR	PLS	IPLS	MCW PLS	SVM
SG 5,1,1	0.730	1.162	1.224	1.356	1.192	0.828	1.143	1.087	1.304	1.295
SG 9,1,1	0.838	1.150	1.148	1.317	1.185	0.727	1.137	1.098	1.189	1.231
SG 13,1,1	0.861	1.139	1.106	1.263	1.182	0.858	1.123	1.204	1.274	1.361
SG 17,1,1	1.075	1.151	1.137	1.362	1.374	1.014	1.096	1.081	1.195	1.309
SG 5,2,1	0.736	1.147	1.224	1.297	1.207	0.826	1.159	1.084	1.281	1.297
SG 9,2,1	0.816	1.161	1.152	1.281	1.191	0.776	1.139	1.024	1.203	1.252
SG 13,2,1	0.861	1.160	1.154	1.267	1.392	0.741	1.128	1.177	1.213	1.242
SG 17,2,1	1.040	1.163	1.175	1.324	1.385	0.894	1.128	1.111	1.241	1.258

Table 4.6: CV RPD values of different first derivative SG filtering-modeling combinations when predicting protein content. The parameters for the SG models are in the following order: window size, polynomial order and derivative order.

	Reflectance					Absorbance				
	FSSR	PLS	IPLS	MCW PLS	SVM*	FSSR	PLS	IPLS	MCW PLS	SVM*
SG 5,2,2	0.860	1.167	1.184	1.175	1.193	0.846	1.174	1.161	1.230	1.296
SG 9,2,2	0.699	1.169	1.187	1.274	1.168	0.896	1.161	1.169	1.226	1.332
SG 13,2,2	0.896	1.183	1.189	1.285	1.208	0.730	1.164	1.195	1.260	1.263
SG 17,2,2	0.867	1.172	1.227	1.316	1.233	0.915	1.166	1.183	1.233	1.293
SG 5,3,1	0.671	1.124	1.148	1.173	1.175	0.666	1.161	1.143	1.159	1.185
SG 9,3,1	0.763	1.139	1.200	1.265	1.233	0.646	1.130	1.109	1.310	1.315
SG 13,3,1	0.967	1.156	1.154	1.260	1.208	0.783	1.137	1.176	1.290	1.256
SG 17,3,1	0.882	1.166	1.157	1.248	1.195	0.847	1.137	1.141	1.221	1.248
SG 5,3,2	0.872	1.152	1.184	1.177	1.195	0.846	1.109	1.161	1.180	1.176
SG 9,3,2	0.699	1.169	1.195	1.274	1.168	0.896	1.161	1.169	1.226	1.332
SG 13,3,2	0.890	1.135	1.195	1.271	1.239	0.730	1.122	1.158	1.183	1.279
SG 17,3,2	0.836	1.171	1.211	1.357	1.236	0.981	1.164	1.161	1.212	1.279

Table 4.7: CV RPD values of different second derivative SG filtering-modeling combinations when predicting protein content. The parameters for the SG models are in the following order: window size, polynomial order and derivative order.

	Reflectance					Absorbance				
	FSSR	PLS	IPLS	MCW PLS	SVM	FSSR	PLS	IPLS	MCW PLS	SVM
OSC 1,1	0.916	1.031	1.073	1.034	1.160	1.042	1.027	1.083	1.053	1.066
OSC 1,2	0.047	1.043	0.583	1.088	1.056	0.077	1.039	0.631	1.076	0.996
OSC 1,3	0.040	1.022	0.429	1.071	1.109	0.044	1.004	0.329	1.040	1.139
OSC 5,1	0.959	1.001	1.030	1.057	1.074	0.999	1.008	1.044	1.048	1.084
OSC 5,2	0.217	0.574	0.782	0.717	1.011	0.418	0.502	0.761	0.685	1.034
OSC 5,3	0.547	0.613	0.814	0.793	1.015	0.354	0.490	0.861	0.698	1.024
OSC 10,1	1.026	1.016	1.020	1.105	1.097	0.941	1.001	1.018	1.030	1.013
OSC 10,2	0.414	0.423	0.440	0.663	1.029	0.425	0.518	0.372	0.583	1.017
OSC 10,3	0.341	0.312	0.357	0.469	1.076	0.356	0.344	0.339	0.516	1.075
OSC 20,1	0.792	0.875	0.859	1.054	0.859	0.753	0.849	0.871	0.985	1.046
OSC 20,2	0.169	0.176	0.165	0.838	1.094	0.182	0.232	0.169	0.341	0.995
OSC 20,3	0.154	0.156	0.146	0.316	1.000	0.208	0.217	0.184	0.284	1.074
OSC 40,1	0.662	0.790	0.786	0.907	1.161	0.620	0.770	0.799	0.972	1.184
OSC 40,2	0.111	0.152	0.131	0.252	1.103	0.106	0.156	0.130	0.406	1.090
OSC 40,3	0.136	0.161	0.144	0.165	1.097	0.158	0.194	0.177	0.180	1.087
OSC 80,1	0.673	0.799	0.796	1.054	1.155	0.625	0.773	0.803	0.972	1.177
OSC 80,2	0.112	0.152	0.131	0.249	1.103	0.101	0.156	0.130	0.405	1.090
OSC 80,3	0.137	0.128	0.135	0.178	1.089	0.131	0.128	0.134	0.198	1.095

Table 4.8: CV RPD values of different orthogonal signal correction processed models on protein. The parameters for OSC are in the following order: number of internal components, number of extracted components.

In the following I will select and analyze the best performing models in details, namely SVM regression and MCW-PLS.

4.2.1 Support Vector Machine Regression

The SVM approach gave the highest RPD value of all competing models with 1.411, after applying MSC and quadratic SG smoothing with a window size of 13. The tuned parameters are the following:

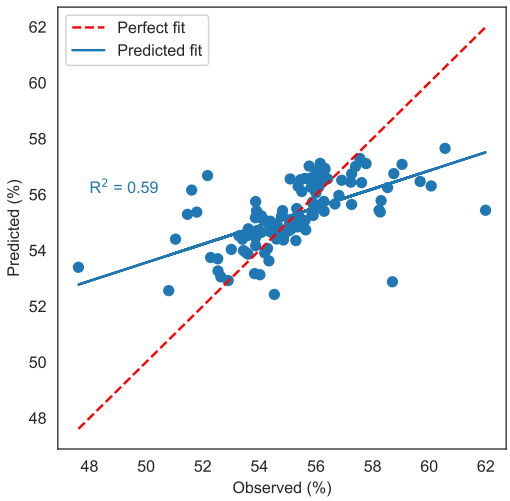
- the width of the RBF kernel, $\sigma = \sqrt{\frac{1}{2 \cdot 0.0001813}} = 52.5153301495577$,

- the penalty term for the errors, $c = 267.4024444444445$,
- the width of the ϵ-tube, $\epsilon = 0.015622222222222223$.

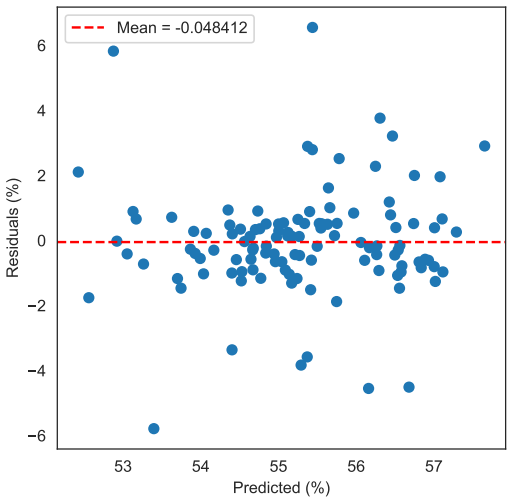
An RPD value slightly above 1.4 corresponds to very poor performance, not recommended for any application according to Table 3.4. When looking at the RMSE, the accuracy does not seem much better either, being equal to 1.463. This means that the standard deviation of the errors is 1.463%, but on a more dispersed total concentration range of 14.38% with standard deviation of 2.073%, making the RPD look more optimistic.

With such low accuracy, it is important deliberate whether the model is predicting protein content or a correlated variable. Specifically, it is natural to assume that dry matter content is going to be a predictor of protein content itself, as less water should mean more protein in protein powder. To investigate this question, I built a simple OLS model between dry matter % as regressor and protein % as regressand, using the fact that for all the 115 protein samples dry matter values are also available. The null hypothesis of the coefficient for dry matter being equal to zero has a p-value of 0.025, but even on the training set the RMSE is as high as 2.018, compared to the validation RMSE of 1.463 of the SVM model using the spectra. Therefore, it is clear, that some information is extracted from the reflectance beyond water content. Whether it is NIR absorption of amino acids, or some other correlated variable is still unclear however.

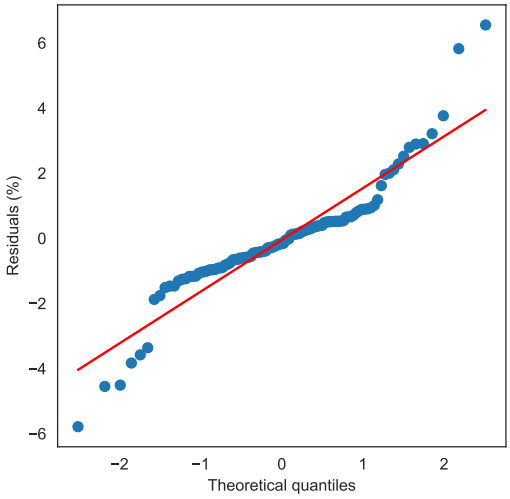
Unlike traditional least squares regression, support vector machine regression with an RBF kernel is a non-parametric approach and does not have a strong set of assumptions. The most important thing to check is the presence of outliers, though the model should have some robustness due to using absolute loss. When fitted on the entire dataset, the model uses 114 out of 115 sample elements as support vectors, with the majority (96 of the 114) having an α of either 267.402 or -267.402, the rest spread more or less uniformly in-between, see Figure 4.6d. The absolute α of a support vector is the measure of how influential the point is for prediction, which means that out of the 115 measurements 96 have equally large importance and the rest less or zero. While there are no points with outlying influence on the model, the model is also not sparse. In fact, when taking into account the rather large σ parameter (area of influence for support vectors), this lack of sparsity explains why the model is rather flat with low predictive power.



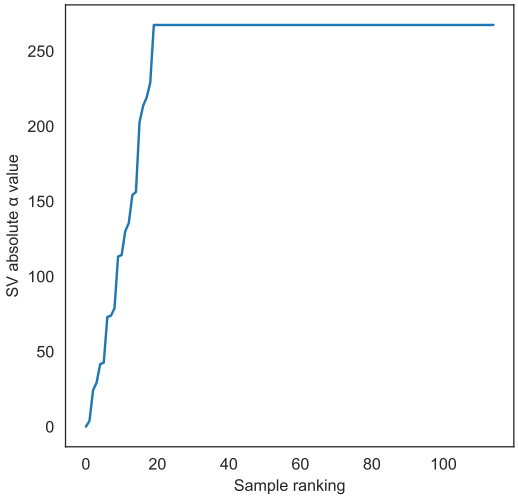
(a) Observed vs predicted based on CV values



(b) Residuals vs predicted based on CV values



(c) Normal Q-Q plot of the CV residuals



(d) Sorted absolute α values (when fitted on the entire data set)

Figure 4.6: Residual diagnostic plots for SVM regression on protein

Based on Figure 4.6b there are some residuals with a rather large value, but more due to poor accuracy, than outliers. The mean of the validation residuals is very close to zero, therefore the predictions seem unbiased. Looking at Figure 4.6a the low predictive power is evident, the predictions generally falling close to the mean concentration. This also manifests in Figure 4.6b with the points further from the mean having larger residuals. Finally, based on the Q-Q plot in Figure 4.6c the residuals are more more heavy tailed than a normal distribution, but also more concentrated in the center. While none of this looks promising, the overall picture is that of poor predictive power, rather than model misspecification.

4.2.2 MCW-PLS

The second best modeling approach was MCW-PLS, fitted on spectra preprocessed with linear SG approximation using window width of 17 and first derivative. The model parameters are $\sigma = 0.01$ and 2 latent components, giving an RPD of 1.362 or equivalently an RMSE of 1.515%.

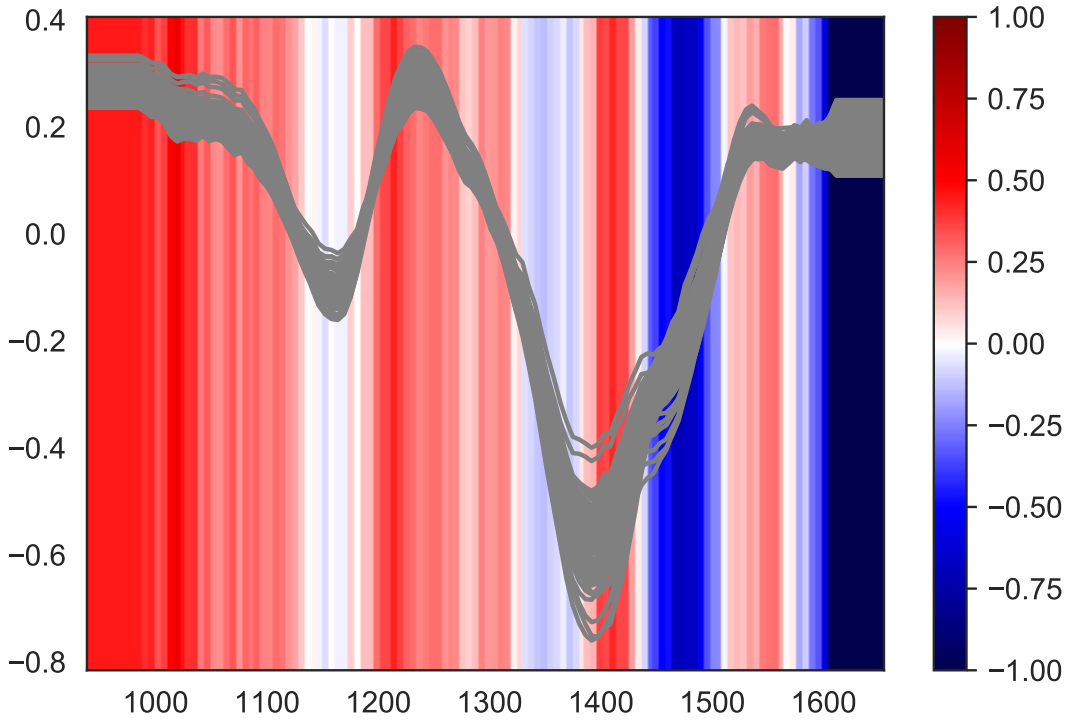


Figure 4.7: Linear SG smoothed spectra with first derivative in reflectance (horizontal axis in nm) colored grey, wavelengths colored according to the regression coefficients assigned by MCW-PLS

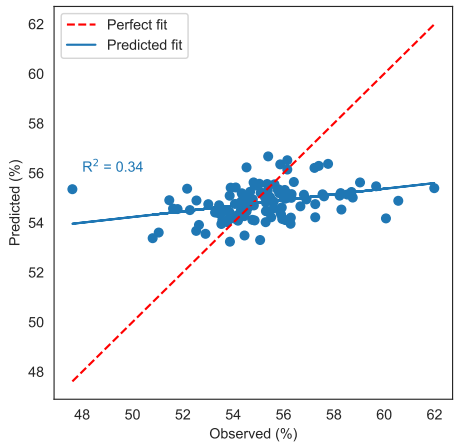
Unfortunately the regression coefficients plotted in Figure 4.7 do not give as much insight, as they did for dry matter. Even though there are clear positive and negative effect regions, the predictive power of the model is very limited, therefore drawing conclusions from the model's regression weights should be done carefully. What can be read from the plot is that the model gives wavelengths in the following intervals a positive coefficient: [938.37 nm, 1,134.93 nm], [1,192.50 nm, 1,323.89 nm] and also to the band corresponding to water's absorption region, which stands out around 1,430 nm. Additionally, there is a narrow band with negative coefficients from 1,453.12 nm to 1,519.40 nm and at the top end of the spectrum, flattened by window smoothing and derivation. This time the dependent variable is protein % but the predictors are in reflectance, therefore bands with high absorption corresponding to amino acids should have low reflectance, and a negative linear relation to the predicted %.

This leads to the interpretation that protein content is primarily estimated from the region around 1,486 nm and especially from the top part of the spectrum. The constant part not only has more negative coefficients, but is also wider (having more variables) and seems to have

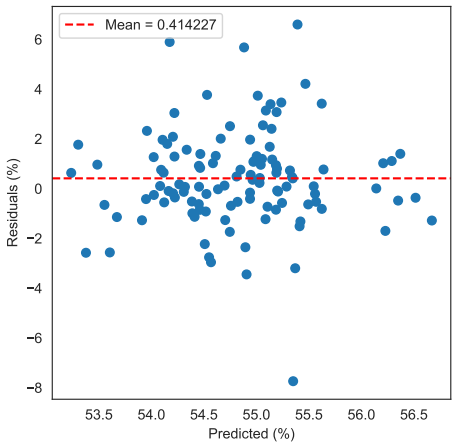
more variance over the observations. The only information this flat part should be capturing, is the general reflectance level of the spectrum in that general area. In other words, observations that reflect more at 1,500 nm, also tend to reflect more all the way up to 1,650 nm, so any wavelength in the interval could capture this information. The reason why the extreme end at 1650 nm is chosen by the model, is exactly because of amplification of variance by both the measurement device and the SG smoothing. For example if 1,500 nm were to have the same correlation with protein content as 1,650 nm, the latter will have higher variance and thus higher covariance due to the previously mentioned inflation factors, and will be favored by PLS.

The wavelengths having positive coefficients in the linear model mean decreasing the protein estimate based on how little /much samples reflect/absorb at that wavelength. Ideally this could be interpreted as absorption lines of chemicals in the analyte whose peaks overlap with that of protein. For instance, the absorption band of water around 1430 nm shows up strongly with positive coefficients on the plot, because naturally the more water, the less protein in a sample. Unfortunately, I do not know what molecules these spectral bands could correspond to, and due to the model's low accuracy these coefficients could be meaningless.

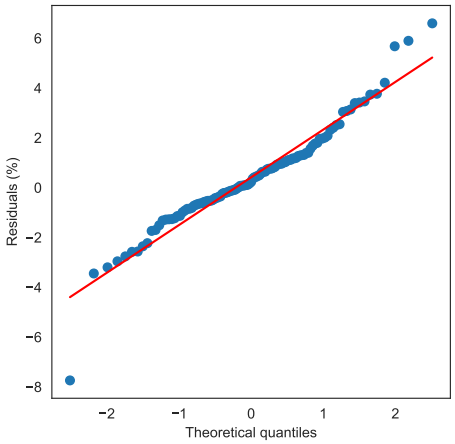
The solution for protein is much more sparse (see Figure 4.8d) than for the dry matter problem, due to σ being one order of magnitude smaller and the residuals being much larger as well. The diagnostic plots in Figures 4.4a, 4.4b and 4.4c once again look similar to the SVM solution's, the biggest difference being that the residuals in Figure 4.4c are much closer to normal. Another difference is that the residuals seem to have a small negative bias and a much worse R^2 when compared to the SVM solution, even though their mean RPD and RMSE is very close. It should be noted however, that the models were selected to minimize the unweighted mean RMSE over measurement days, not the total validation squared error, hence there can be a discrepancy between performance. Even so, the fact that the SVM model holds up much better on this different validation score should be kept in mind for the final judgment. The residuals are largest around the middle of the predicted concentrations, which is probably due to the fact that the model is rather flat and in general tends to predict close to the mean response. As a result, the predictions with a large residual will be the ones that are estimated to be close to the mean but belong to a true concentration close to the minimum/maximum. This is completely normal for low predictive power and not a sign of model misspecification. In Figure 4.8e none of the residuals qualifies as truly outlying relative to the others, though there is one close to the lower 95% interval. However, this should not be a big concern, as in Figure 4.8f all leverages are very small, so that a single large residual hardly makes a difference for the model, which is expected as MCW-PLS was designed to be resistant to such outliers.



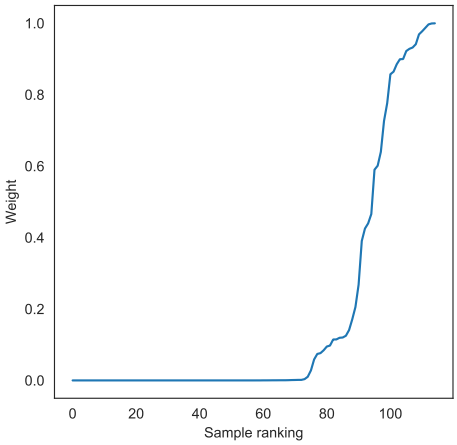
(a) Observed vs predicted values based on CV



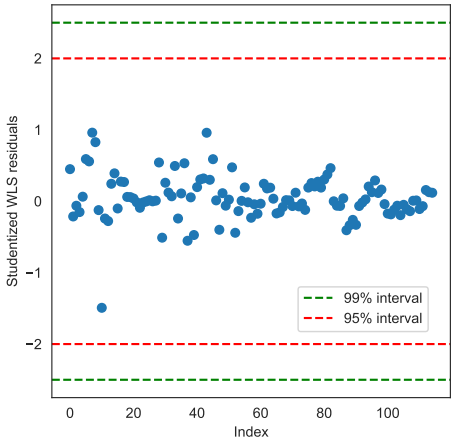
(b) Residuals vs predicted values based on CV



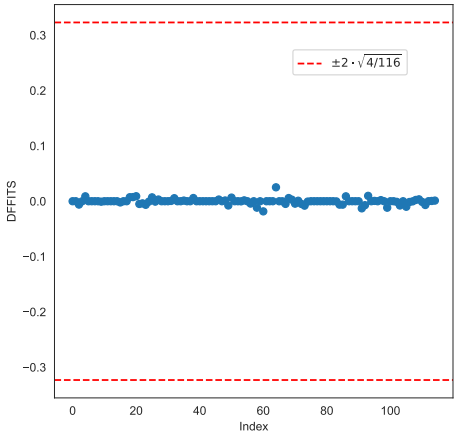
(c) Normal Q-Q plot of the CV residuals



(d) Sorted sample weights (when fitted on the entire data set)



(e) Studentized residuals (when fitted on the entire data set)



(f) DFFITS (when fitted on the entire data set)

Figure 4.8: Residual diagnostic plots for MCW-PLS on dry matter

4.3 Summary

The results obtained on dry matter and protein content share many characteristics. Most importantly, the predictive accuracy on both models were underwhelming. While it might seem that modeling of dry matter has been more successful, determining water content of a sample should be a trivial matter, with its well known absorption lines in the NIR region. It should also be highlighted, that only validation errors were investigated, which almost surely underestimate/overestimate the true error/RPD. For preprocessing MSC, SNV and SG filtering are most appropriate, the former two requiring additional SG smoothing in certain cases, for modeling SVM regression or perhaps MCW-PLS. Of special not is that while OSC is a popular technique, it seemed to remove useful information from the data, rather than noise. Similarly, according to theory conversion to absorbance units from reflectance is essential, yet it degraded performance in this case. Diagnostics do not suggest the presence of any serious misspecification, and the findings were consistent across a wide range of methods.

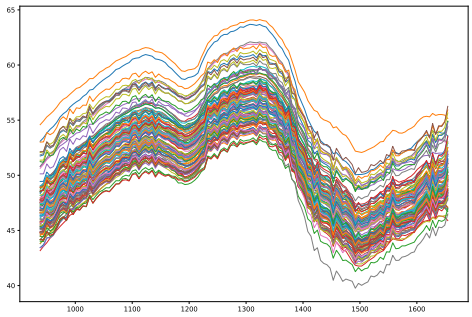
4.4 Discussion

For modeling MCW-PLS and SVM regression are the best choices. SVMs give lower errors than any other investigated technique, most likely due to their ability to deal with non-linearity. This property also allowed them to give decent performance even in scenarios where the linear alternatives had drastically lower RPD. In some cases I did not manage to tune a good SVM regression, but I believe that given enough computational time a similar score can be achieved in all scenarios except for OSC, as it appears to remove useful information. This flexibility makes SVMs not just the most accurate model, but perhaps also the safest one, as it gives the most consistent performance over different conditions.

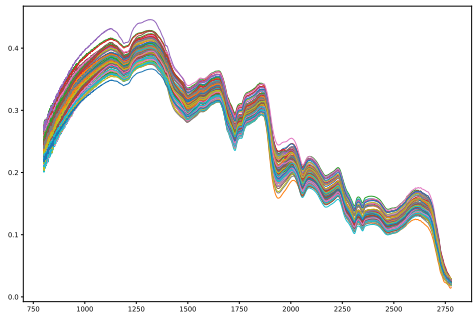
Despite all of these advantages of SVMs, I would still recommend using MCW-PLS. While performance is only marginally lower, and also relatively consistent over the different conditions, it has some useful properties that SVMs do not. The most important is interpretability. The output of the model contains linear regression coefficients for the original input variables, which enables the user to understand how the predictions are made and identify key spectral regions used for prediction. This can be very useful in a practical application, for example, once the important wavelengths are discovered with measurements from a cheaper wide spectrum device, investment into higher grade instruments operating on more narrow intervals (possibly on a single wavelength) is possible. The other advantage is the availability of well established diagnostics that have been developed for OLS regression and much faster model tuning.

Of more importance however, is the failure to achieve useful predictive performance. This problem is consistent across all preprocessing and modeling methods, which cover both the traditional approaches in the field and some more advanced techniques, like MCW-PLS, which was only published recently. In my opinion the problem lies in the predictors lacking the required information to make good predictions, not choice of preprocessing or modeling. Beyond

the fact that none of the attempted methods seemed to work, there seems to be two clear issues with the spectra. Firstly, the range is limited even for NIR, covering only the middle part. This could very easily explain the low RPD for water, as it was explained in 3.2.1, that the best band in NIR happens to lie outside the available interval. Secondly, the spectral lines were unusually rough, much more so than what is common in publications in NIR spectroscopy. These most likely come from the technical limitations of the Flame NIR device (it is a relatively inexpensive model), not from bad experimental conditions. These issues, along with the low accuracy quickly became clear once measurements first became available for analysis. To acquire data with better quality, and to be able to identify useful spectral regions currently outside Protix's device's capabilities a number of samples were shipped to KU Leuven's laboratory, where I have re-measured the reflectance of some samples using the more high-end spectroscope of the university. These measurements were both clearly smoother and had much better range and resolution. Unfortunately when I tried to use the data for modeling concentrations it had no predictive power at all. The source of the problem could never be cleared up, as I could no longer access the laboratories due to the restrictions that came with the SARS-CoV-2 crisis.



(a) Raw reflectance spectra as measured by Protix (horizontal axis in nm, vertical axis in %)



(b) Raw reflectance spectra as measured by KU Leuven's device (horizontal axis in nm, vertical axis in %)

Figure 4.9: Protix's spectra compared to the one measured with KU Leuven's device

Chapter 5

Conclusions

The original purpose of the Thesis was to develop the statistical tools for Protix to implement a process control system based on a NIR spectroscope for their insect protein products. The ideal outcome would have been an on-line NIR monitoring solution for multiple nutrients, perhaps even beyond protein and dry matter content, using the optimal models identified and implemented in Python in the Thesis.

After measurements first became available for analysis, it quickly became clear that data quality might be problematic. Even after extensive modeling, the accuracy of predictions was far off both from Protix's expectations and the standards in publications in the field.

Bibliography

[1] Barnes, R. J., Dhanoa, M. S., & Lister, S. J. (1989). Standard Normal Variate Transformation and De-Trending of Near-Infrared Diffuse Reflectance Spectra. Applied Spectroscopy, 43(5), 772–777. doi: 10.1366/0003702894202201

[2] Blitz, J. P. (1998). "Diffuse Reflectance Spectroscopy," in Modern Techniques in Mirabella, F. M. (Ed.). Applied Molecular Spectroscopy: Techniques in Analytical Chemistry (pp. 185–219). New York: John Wiley & Sons.

[3] Davies, A. M. C. (n.d.). An introduction to near infrared (NIR) spectroscopy. Retrieved July 30, 2020, from https://www.impopen.com/introduction-near-infrared-nir-spectroscopy

[4] De Jong, S. (1993). SIMPLS: An alternative approach to partial least squares regression. Chemometrics and Intelligent Laboratory Systems, 18(3), 251-263.

[5] Fabian P., Gaël V., Alexandre G., Vincent M., Bertrand T., Olivier G., Mathieu B., Peter P., Ron W., Vincent D., Jake V., Alexandre P., David C., Matthieu B., Matthieu P., Édouard D. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

[6] Fonseca Diaz, V., De Ketelaere, B., Aernouts, B., & Saeys, W. (2020). Robustness control in bilinear modeling based on maximum correntropy. Journal of Chemometrics, 34(4), N/a.

[7] Flame-NIR Product Sheet. (2019). Flame-NIR Product Sheet. Retrieved from https://oceanoptics.com/wp-content/uploads/Flame_NIR_Product_Sheet_web.pdf

[8] Geladi, P., Macdougall, D., & Martens, H. (1985). Linearization and Scatter-Correction for Near-Infrared Reflectance Spectra of Meat. Applied Spectroscopy, 39(3), 491–500. doi: 10.1366/0003702854248656

[9] Geladi, P., & Kowalski, B. (1986). Partial least-squares regression: A tutorial. Analytica Chimica Acta, 185(C), 1-17.

[10] Guided Wave. (2019, October 17). An Introduction to Online NIR Water Measurements in Liquid Samples. AZoM. Retrieved on July 30, 2020 from https://www.azom.com/article.aspx?ArticleID=17511.

[11] Hubert, M, & Branden, K. Vanden. (2003). Robust methods for partial least squares regression. Journal of Chemometrics, 17(10), 537-549.

[12] Kemps, B. J., Saeys, W., Mertens, K., Darius, P., Baerdemaeker, J. G. D., & Ketelaere, B. D. (2010). The Importance of Choosing the Right Validation Strategy in Inverse Modelling. Journal of Near Infrared Spectroscopy, 18(4), 231–237. doi: 10.1255/jnirs.882

[13] Jin, Y., & Ikawa, S. (2003). Near-infrared spectroscopic study of water at high temperatures and pressures. The Journal of Chemical Physics, 119(23), 12432-12438.

[14] Jolliffe, I.T. Principal Component Analysis. 2nd Ed. Springer-Verlag, 2002.

[15] Martens, H., Jensen, S. A., & Geladi, P. (1983). Multivariate linearity transformations for near infrared reflectance spectroscopy. Applied Statistics, 205–234.

[16] Nørgaard, L., Saudland, A., Wagner, J., Nielsen, J. P., Munck, L., & Engelsen, S. B. (2000). Interval Partial Least-Squares Regression (iPLS): A Comparative Chemometric Study with an Example from Near-Infrared Spectroscopy. Applied Spectroscopy, 54(3), 413-419. doi:10.1366/0003702001949500

[17] Norris, K. H., & Williams, P. C. (1984). Optimization of Mathematical Treatments of Raw Near-Infrared Signal in the Measurement of Protein in Hard Red Spring Wheat. I. Influence of Particle Size. Cereal Chemistry, 61, 158–165.

[18] Pasquini, C. (2018). Near infrared spectroscopy: A mature analytical technique with new perspectives – A review. Analytica Chimica Acta, 1026, 8-36.

[19] Rinnan, Å., Berg, F. V. D., & Engelsen, S. B. (2009). Review of the most common pre-processing techniques for near-infrared spectra. TrAC Trends in Analytical Chemistry, 28(10), 1201–1222. doi: 10.1016/j.trac.2009.07.007

[20] Savitzky, A., & Golay, M. J. E. (1964). Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Analytical Chemistry, 36(8), 1627–1639. doi: 10.1021/ac60214a047

[21] Shumo, M., Osuga, I. M., Khamis, F. M., Tanga, C. M., Fiaboe, K. K. M., Subramanian, S., Ekesi, S., Van Huis, A. & Borgemeister, C. (2019). The nutritive value of black soldier fly larvae reared on common organic waste streams in Kenya. Scientific Reports, 9(1), 10110-13.

[22] Sjöblom, J., Svensson, O., Josefson, M., Kullberg, H., & Wold, S. (1998). An evaluation of orthogonal signal correction applied to calibration transfer of near infrared spectra. Chemometrics and Intelligent Laboratory Systems, 44(1-2), 229-244.

[23] Stuart, B. H. (2008). Infrared spectroscopy: Fundamentals and applications. Chichester, West Sussex: Wiley.

[24] Suykens, J. A.K., Van Gestel, T., & De Brabanter, J. (2002). Least squares support vector machines. Singapore: World scientific.

[25] Svensson, O, Kourti, T, & MacGregor, J. F. (2002). An investigation of orthogonal signal correction algorithms and their characteristics. Journal of Chemometrics, 16(4), 176-188.

[26] Tran, U., & Formann, A. (2009). Performance of Parallel Analysis in Retrieving Unidimensionality in the Presence of Binary Data. Educational and Psychological Measurement, 69(1), 50-61.

[27] Wang J., Chen Q., Chen Y. (2004) RBF Kernel Based Support Vector Machine with Universal Approximation and Its Application. In: Yin FL., Wang J., Guo C. (eds) Advances in Neural Networks

[28] Williams, P. (2014). The RPD Statistic: A Tutorial Note. NIR News, 25(1), 22-26.

[29] Wold, S., Antti, H., Lindgren, F., & Öhman, J. (1998). Orthogonal signal correction of near-infrared spectra. Chemometrics and Intelligent Laboratory Systems, 44(1-2), 175-185.

[30] Wouter, S. (2019). B-KUL-G0B70A: Chemometrics, session on Pre-processing and Linearization

AFDELING
Straat nr bus 0000
3000 LEUVEN, BELGIË
tel. + 32 16 00 00 00
fax + 32 16 00 00 00
www.kuleuven.be