Multivariate Data Analysis and Data Fusion Techniques for Modeling Spectroscopic Data During CO_2 Capture with Amines

Master Thesis in Process Technology

Adam Joshua Armstrong

Department of Chemistry

UNIVERSITY OF BERGEN

December 30th 2018

Foreword

This thesis is written at the University of Bergen in accordance with the last year of an MSc in process technology, with a specialization in chemometrics. This project was planned and provided by CO_2 Technology Center Mongstad and Bjørn Grung with the Department of Chemistry at the University of Bergen.

I would like to give my heartfelt thanks to my supervisor, Bjørn Grung, for his amazing patience and understanding during this project. I would also like to thank Egil Nodland with the Department of Chemistry at the University in Bergen for always providing assistance when requested.

And finally, I would like to thank my family for supporting me through this entire journey, and listening to my endless ramblings about things they don't understand, and are certainly not interested in. Chief among them, my father, who did not live to see this project completed, but who I dedicate this thesis to.

Thank you,

Adam Joshua Armstrong

Abstract

This master thesis is a collaboration between CO_2 Technology Center Mongstad and the University of Bergen. The goal of the project was to use multivariate data analysis methods to generate predictive models for the total alkalinity (TOT_ALK), total inorganic carbon (TIC) and density of anime solvent samples. This was to be done using three different instrument types, namely ATR-FTIR, NIR and Raman spectrometers.

An individual predictive model was to be created for each instrument type, and then data fusion techniques were to be employed in order to create fused models which combine traits from all of the different instruments. The goal of this was to determine which individual instrument provided measurement data which was most useful in creating predictive models, as well as to determine if the use of multiple instruments in tandem is a worthwhile endeavor.

In order to accomplish this, a partial least squares regression (PLS-R) algorithm was programmed in MATLAB, as well as multiple subroutines for data preprocessing and analysis, including Monte-Carlo cross-validation (MCCV), variable selection, outlier detection, extended multiplicative signal correction (EMSC) and Savitzky-Golay filtering.

The models that were generated indicate that the use of Raman spectroscopy is not advised for the stated intent, and that a combination of ATR-FTIR and NIR instruments provide the best and most consistent overall results.

List of Abbreviations

ATR-FTIR	Attenuated total reflectance Fourier transform infrared
EMSC	Extended multiplicative signal correction
IR	Infrared radiation
LLDF	Low-level data fusion
MEA	Monoethanolamine
MCCV	Monte-Carlo cross-validation
MIR	Mid-infrared
MLDF	Mid-level data fusion
MSC	Multiplicative signal correction
NIR	Near-infrared
OPD	Optical path difference
PCA	Principal component analysis
PLSR	Partial least squares regression
RMSEP	Root mean square error of prediction
RMSECV	Root mean square of cross-validation
RSD	Residual standard deviation
SR	Selectivity ratio
TIC	Total inorganic carbon
TOT_ALK	Total alkalinity
VIP	Variable importance in prediction

Table of Contents

1 Introduction

Climate change is an ever looming threat on a global scale that has the potential to wipe out all life on the planet. A large contributor to climate change is the massive amount of CO_2 emissions being released into the atmosphere on a daily basis. This contributes to what is known as the greenhouse effect, and gasses that contribute to the effect are known as greenhouse gases. The refining and burning of fossil fuels generates a significant portion of the CO_2 that is released into the atmosphere, and there is therefore a need to reduce these emissions with haste.

In an effort to reduce CO_2 emissions, the CO_2 that would normally be released is instead captured, and then stored or used for other purposes. This can be done in many ways, but a promising technique is that of amine based capture, as seen in the pilot plant at CO_2 Technology Center Mongstad. This technique involves using amine as a solvent to absorb CO_2 from the flue gas produced by, for example, a refinery. This is done by providing large areas of contact between the flue gas and the amine solvent in what is known as an absorber tower, or a gas scrubber tower [1].

The low CO_2 concentration solvent, known as the lean solvent, is sent into the tower, mixed with the gas, and then removed from the tower with a higher CO_2 concentration, known as the rich solvent. Meanwhile, the flue gas, with a substantially reduced CO_2 concentration is removed from the tower and either released or further processed. The rich solvent is then sent forward to have the CO_2 removed from it, known as regeneration, so that it may be turned back into lean solvent, and recycled back into the system, heavily reducing waste. This removed CO_2 is then ready to be stored, transported or used [1].

A problem that arises is that the regeneration process is not perfect, and over time the amine solvent degrades, and not only becomes less effective at absorption, but also creates solid precipitate byproducts which can damage machinery, or even clog it completely. For this reason it is important to monitor the health, if you will, of the solvent, so that it can be replaced or cleaned before this damage or clogging can take place.

A common way of monitoring processes is through the usage of spectrometers to determine characteristics of the bulk fluid flowing through the system at various points. This allows information about the chemical composition of the fluid to be analyzed, and predictions to be made about the fluid [2]. Models for these predictions are based on multivariate data analysis, and can be based on a wide range of spectrometer types, such as ATR-FTIR, NIR and Raman, each bringing their own unique perspective on the properties of the fluid.

As each type of spectrometer has a different perspective on the properties, it is not difficult to extrapolate that their combined perspectives may bring about more accurate predictions of said properties, by working together in tandem. Combining the datasets from multiple instrument types is known as data fusion, and is an emerging technique for the increased prediction accuracy in many fields and applications across the globe. With these fused datasets, techniques such as PLS-R can be used to make predictions, and hopefully give the prediction that most closely represents the actual properties of the given substance.

2 Spectroscopy

2.1 The EM spectrum

Electromagnetic (EM) radiation is energy that is radiated out from the radiation’s source. As such, the EM spectrum is the range of all of those types of EM waves, which are divided up into sub-types based on their energy, or wavelengths, as seen in figure 1.

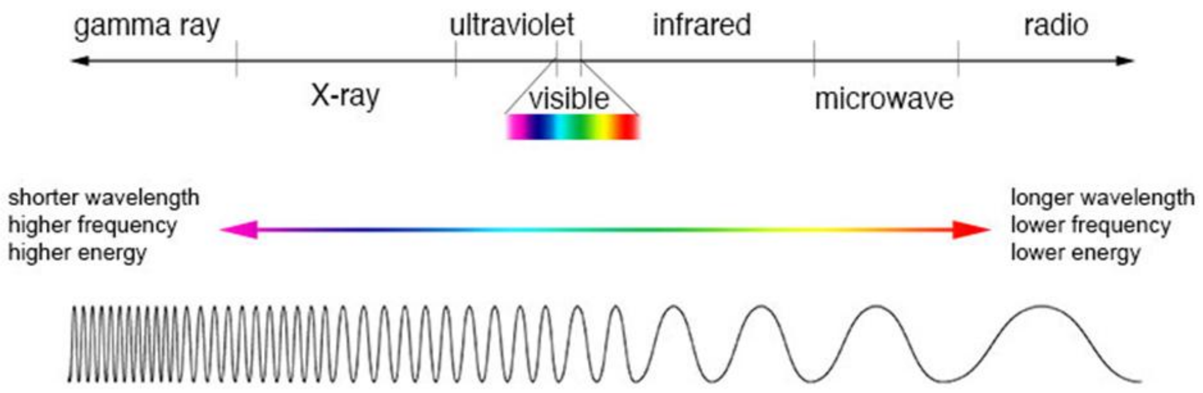

Figure 1: Diagram of the electromagnetic spectrum [52].

The common name for EM radiation is light. Light exists in a duality, being both a particle and a wave at once, with the particles of light being known as photons [3, p. 1081]. Although the word light is typically associated only with the visible light section of the spectrum, the entire spectrum is comprised of light of differing energy levels, which then affects the wavelength of the propagated waves, with higher energy giving shorter wavelengths, and therefore a higher frequency [3, pp. 1051-1052]. This is due to the fact that the speed of light is a constant for a given propagation medium. This relationship is defined in equation 1, known as the Planck-Einstein relation, and equation 2.

$$E = h * f \qquad \textbf{\textit{Equation 1}}$$

where E is energy in joules (J), h is Planck’s constant in joules by seconds (J*s) and f is the frequency of the wave, in waves per second (s^{-1}).

$$f = c * \lambda \qquad \textbf{\textit{Equation 2}}$$

where f is frequency (s^{-1}), c is the speed of light in meters per second (m/s) and lambda is the wavelength in meters (m).

As both c and h are constant, one needs only know the wavelength, energy, or frequency of the light to know the others. This is especially useful for detectors which can easily determine the energy of a photon striking it, and can then determine the frequency and wavelengths of the light. The wavelength, measured from peak to peak from the waves, is a common unit for the EM spectrum to be displayed and graphed. The wavenumber, a transformation of the wavelength to units in cm^{-1} is also a very common unit notation.

The EM spectrum is divided up into seven basic groupings, which, as seen in figure 1, are gamma rays, X-rays, ultraviolet rays, visible light, infrared light, microwaves, and radio waves [3, pp. 1054-1055]. Visible light is perhaps the grouping which most people are most familiar with, comprising of a relatively small section of the EM spectrum, ranging from around 400 nanometers (nm) to 700 nm. The wavelength of the photons then determine the color of the light that our eyes perceive, with the longest wavelengths being seen as red, and the shortest wavelengths being violet. This then lends to the nomenclature of the areas of the EM spectrum directly adjacent to them, namely the infrared spectrum and ultraviolet spectrum.

2.2 General Spectroscopic Principles

Spectroscopy is a technique for dividing up all of the incoming photons into their respective energy levels, or wavelengths. The light which you are seeing right now is a collage of uncountable different wavelengths, bombarding you in a constant stream that allows you to interpret the world around you. In a similar fashion, radio waves from various wireless devices and radio towers are all broadcasting at different frequencies, and therefore different wavelengths. In order for something like a radio to tune into a specific radio station, it must fine tune, and listen to a specific frequency.

In the same way, someone wishing to break down any of the EM radiation around them must have some way to focus into a small section of the frequencies. This is where spectroscopy comes in [4]. Although there are many ways of crudely breaking up the EM spectrum, for example with a prism on a desk, turning seemingly white light into a rainbow of colors, spectrometers are pieces of equipment that are typically used when attempting to observe and log various sources of light by breaking it down into its component wavelengths.

This becomes especially useful when attempting to determine information about the EM radiation source, as all matter above absolute zero gives off EM radiation, and at specific energy levels. In addition, whenever a photon approaches the substance, be it an atom, or a molecule, there are many possibilities for what can happen. The atom or molecule will have various excited states that electrons can jump to, which have specific energy levels. If the photon in question matches one of these energy requirements, the photon will be absorbed into it, and the electron(s) will move to the given excited state. These excited states, however, are unstable, and this extra energy is eventually kicked out of the atom or molecule as either one or multiple photons, the total energy of which equals the total energy absorbed. This kicking out is known as emission [5, pp. 85-86]. A visual representation of this phenomena can be seen in figure 2.

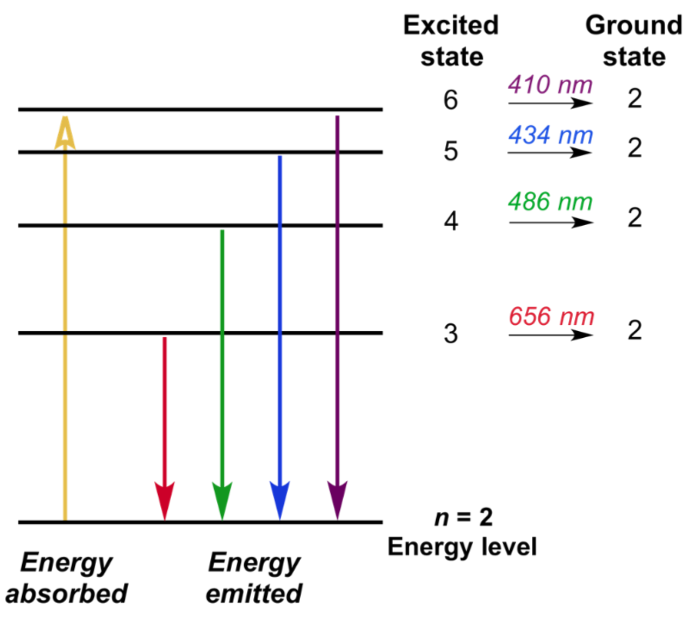

Figure 2: Diagram depicting absorption of photons and emission possibilities of an example atom. [6]

These emitted photons are easily detected by a spectrometer, and their energy levels are characteristic of the substance in which it was emitted from, possibly allowing for identification of the substance.

However, not all approaching photons can be absorbed, as not all photons carry an energy that matches an excited state. These photons then can then either be reflected, transmitted, or scattered. In reflection, the photon simply approaches, and is deflected away without absorption and emission. Transmission of the photon means that the photon passes through without interacting with the substance at all. Scattering, however, is when the photon is highly energetic, and is unable to be absorbed completely, but that still gives some of its energy to the substance in an incident collision. As with absorption, this excitation is highly unstable, and will be emitted. The key difference between scattering and absorption is that with scattering, the whole photon is not absorbed into the substance. It simply loses, or in some cases, gains, some energy in the collision, and then continues on [7].

All of these changes can be detected by various spectrometers, and can be seen in figure 3. With a known photon source, it is relatively easy to selectively determine various characteristics of the subject in question.

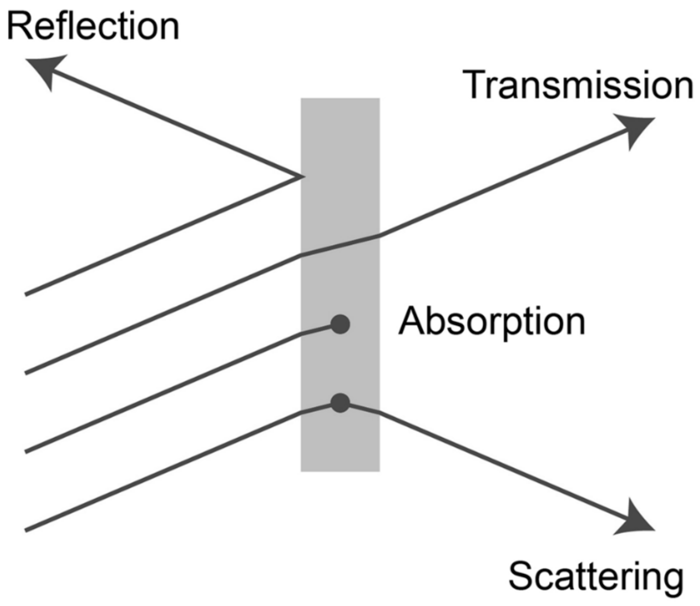

Figure 3: Types of interactions that can occur when a substance interacts with photons [7].

2.3 Infrared Spectroscopy

Infrared (IR) spectroscopy is the measurement of the interactions of IR radiation with a substance. The substance is bombarded with IR radiation of varying energy levels, and those photons are either absorbed, transmitted, or reflected by the substance, which can then be measured by a detector [8]. The IR spectrum is typically measured in three different groupings, known as near-, mid- and far-infrared.

An IR spectrometer is comprised of a radiation source, some sort of dispersion element (E.g. a diffraction grating, or a prism) and a detector. The dispersion element is also commonly referred to as a monochromator. In some machines, a laser source is used, and there may not be a need for a dispersion element. The radiation source, often an incandescent, light emitting diode, or halogen light source, sends a beam of energy at the dispersion element, which focuses a very narrow band (commonly described as monochromatic light) of the incoming light at the substance being examined. A basic diagram of this can be seen in figure 4.

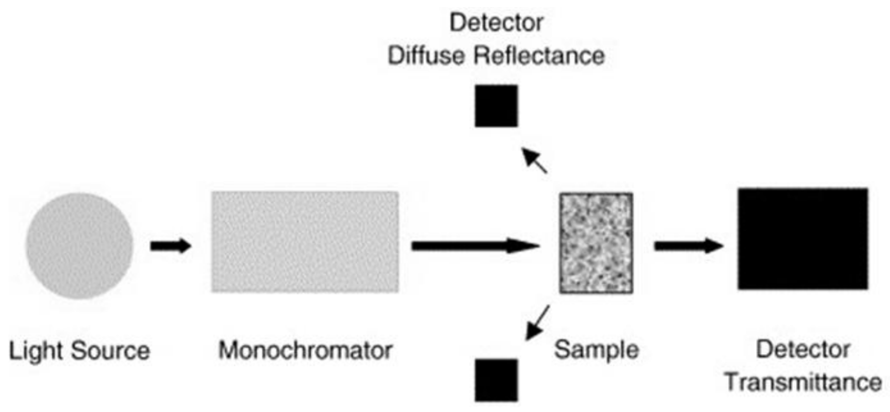

Figure 4: General diagram of the inner workings of an IR spectrometer [9].

Once the monochromatic light interacts with the substance, a certain amount of the radiation will transmit through, unhindered, and will be detected by the transmittance detector, while peripheral detectors will measure diffuse reflectance. By determining what amount of the original light was transmitted through the sample, it can also be inversely related to the absorption of the light, and this can then be graphed for a visual representation of the data, as seen in figure 5.

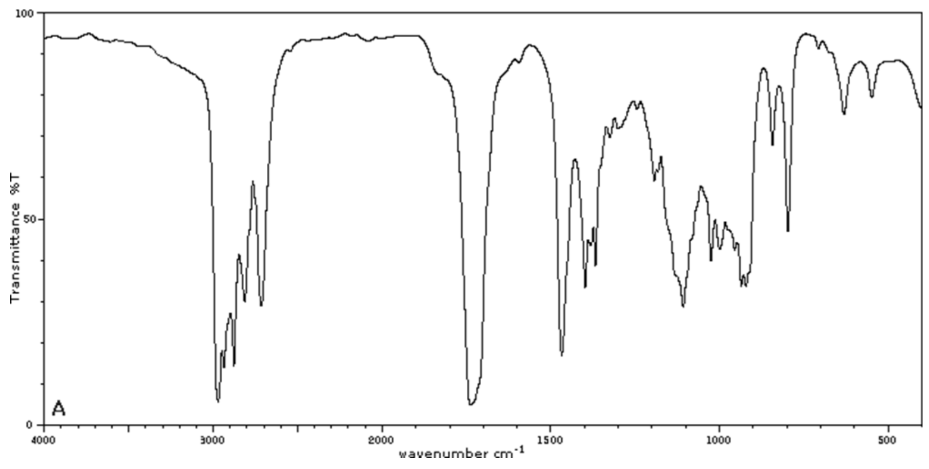

Figure 5: Transmittance plot of the mid-IR spectrum for a given substance [8].

The absorbed IR radiation is absorbed at specific energy levels, which correspond to the energy levels of either bending, or stretching of the bonds between atoms within a molecule [10]. This requires a dipole moment to be exploited, and thus only solids, liquids and gases can be examined. The type, and amount, of possible bends or stretches a molecule can have varies depending on the types of bonds as well as the structure, but a basic overview can be seen in figure 6.

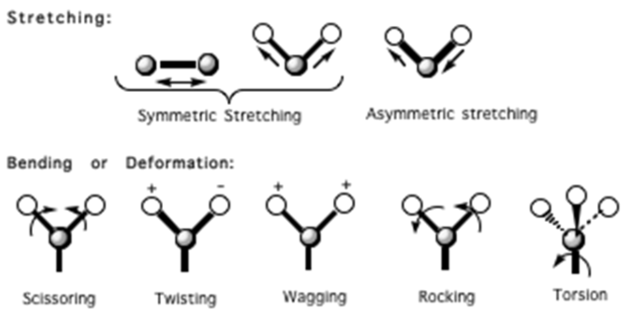

Figure 6: Example of bending and stretching that molecules are capable of [10].

The three groupings of the infrared spectrum, near-, mid- and far-infrared, are all examined and measured in different ways, and therefore by different types of machines, with different devices and methods.

2.4 Fourier Transform Infrared Spectroscopy

Within the scope of IR spectroscopy, there are more advanced mechanisms within the spectrometer which can aid in collecting data. One such variation gives us the Fourier Transform Infrared (FTIR) spectrometer, often used for the mid-IR (MIR) spectrum, ranging from wavenumbers 4000–400 cm^{-1} (2500-25000 nm) [11]. Rather than using monochromatic light to test the sample, a wide range of wavelengths are contained within the beam that will bombard the sample, and the detector will take in data from all of these energy levels. The beam will then be changed to contain a different combination of wavelengths, and a new reading will be made, and so on until enough data is collected so that a computer can work backwards and determine the transmittance of each wavelength from the total spectrum being used.

This is done by using a source beam which contains all of the wavelengths to be examined, and shining it onto a device known as a Michelson interferometer. The interferometer splits the beam of light into two beams of light, each with a different path length to the sample [11]. This is done by using a beam splitter to split the beam of light, and two mirrors, one stationary, and one mobile, as shown in figure 7.

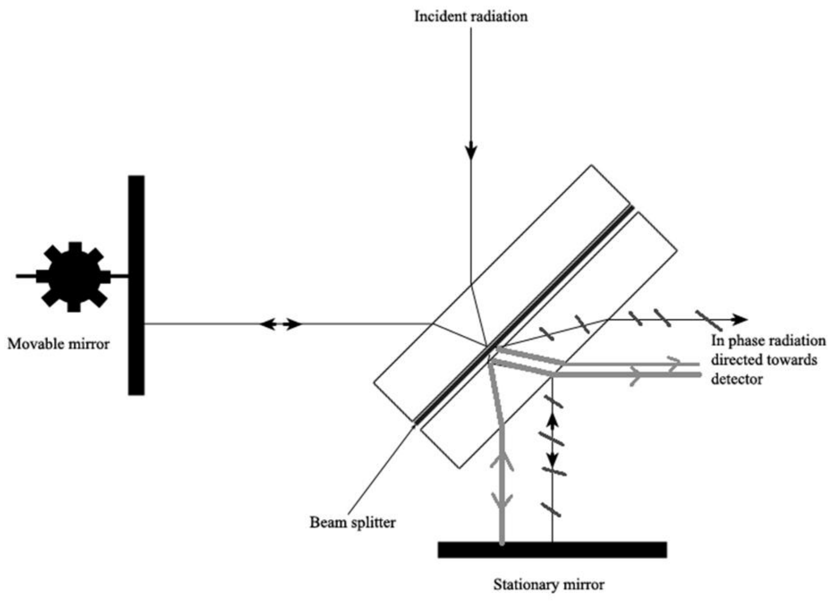

Figure 7: Diagram of the inner workings of an interferometer from an FTIR spectrometer [11].

The beam splitter directs half of the light to each mirror, and by adjusting the distance of the mobile mirror from the beam splitter, one pathway varies, while the other stays constant. Once each beam is reflected from its respective mirror, the beams are recombined at the beam splitter, and continue on to the sample as one beam. The waves within the beams, due to their difference in path length, known as optical path difference (OPD), are no longer in phase with one another, and therefore each wavelength is subjected to constructive and destructive interference [11] .

When the mirrors are exactly the same distance away, the OPD is zero and is referred to as zero path difference (ZPD). At this point, the two beams are perfectly in phase, which leads to maximum constructive interference, giving maximum intensity. When the OPD is exactly half of the wavelength, or a multiple thereof, there is maximum deconstructive interference, and the intensity of the beam at that wavelength becomes zero, as the two combined waves cancel each other out [11]. This phenomena leads to a different sort of measurement plot, known as an interferogram, as seen in figure 8.

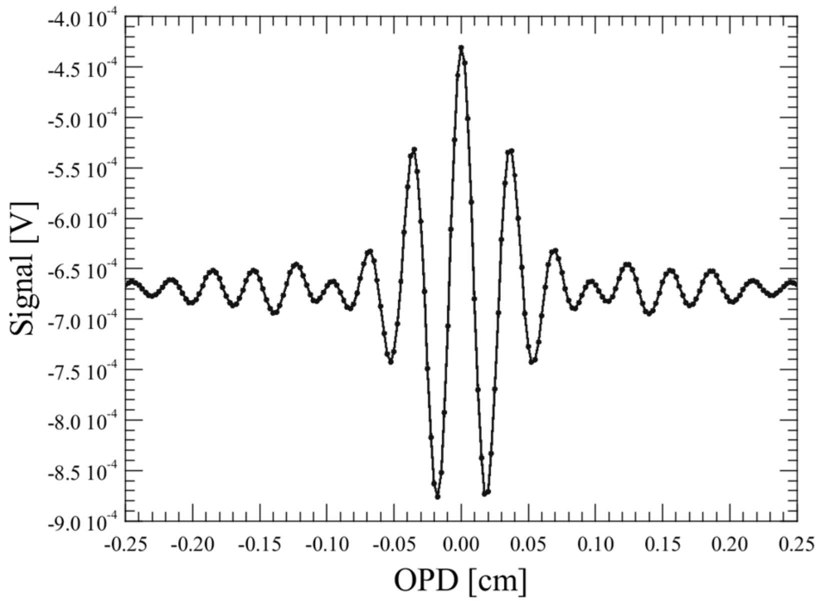

Figure 8: An example of an interferogram, the signal produced by an interferometer [12].

The interferogram dataset must then be transformed for each wavelength to give the final spectrum, using a mathematical technique known as a Fourier transformation, hence the name of the device. In essence, the interferogram is a function of time, and this time domain is transformed with the Fourier transformation to the frequency domain, which is then used to create the spectrum [11]. The main advantage of this technique is that since it can examine many wavelengths at once, it is typically significantly faster at analyzing samples than a standard IR spectrometer.

2.5 Attenuated Total Reflection IR Spectroscopy

Sample preparation can be both time consuming and inconvenient, if not entirely impossible for use in various IR spectroscopy techniques. This can be especially true with samples which cannot, or should not, be diluted, altered or destroyed. For this reason, a new device was needed, and it came in the form of the Attenuated Total Reflection (ATR) technique. An ATR crystal is used, which is a crystal that is IR transparent, but with a high refractive index [13].

The sample to be tested is placed in direct contact with the ATR crystal, and an IR beam is fired at an angle, typically 45° relative to the crystal's surface, at the crystal. At the crystal/sample interface, the IR beam is completely reflected off of the sample after a certain amount of penetration, and then reflected a number of times internally within the crystal until the beam leaves the opposite side of the ATR crystal, and travels to the detector, as seen in figure 9.

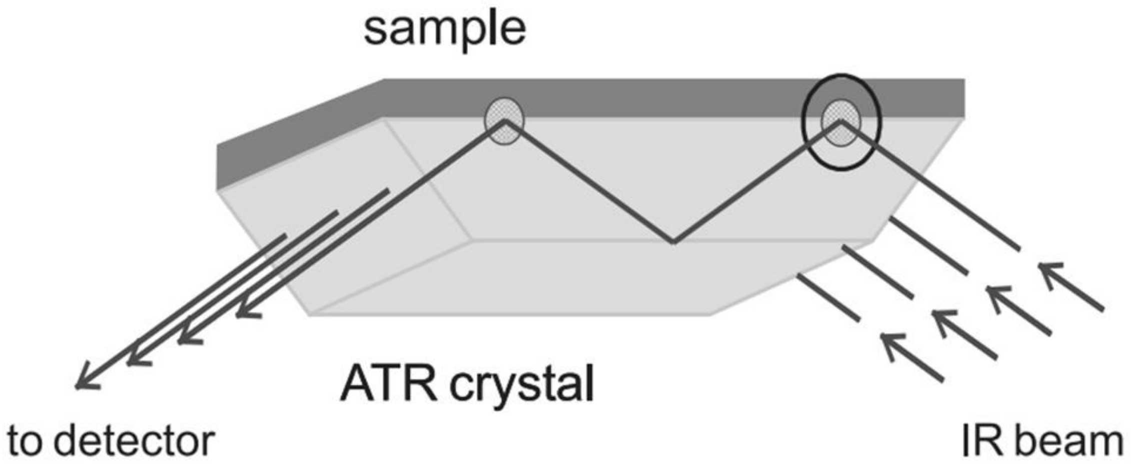

Figure 9: Diagram depicting how an ATR crystal reflects incoming light against a sample on its surface [13].

Only a fraction of the light waves in the IR beam reach the sample, and this fraction is known as the evanescent wave. The exact penetration depth within the sample is determined by the wavelength of the light, the reflective index of the crystal and sample, and the angle of the IR beam relative to the crystal. The penetration depth is quite small, often between 0.5 and 3.0 µm. If the sample absorbs energy at the wavelength of the evanescent wave, the wave is attenuated, which can then be measured and calculated by a computer [13].

Although this method allows fast and easy sample measurement, the primary downside of the method is that only the attenuated evanescent waves are used for measurement, making the intensities significantly lower than that of other methods, which makes the overall sensitivity worse. Additionally, only the substances at or near the surface interface are measured, which can be problematic if the interface area is not representative of the bulk sample [13].

2.6 Near-Infrared Spectroscopy

Near-Infrared (NIR) is the highest energy IR radiation, typically ranging from 14,000-4,000 cm^{-1} (~700-2500 nm). NIR spectrometers operate on the same basic principles and components as MIR spectrometers, with some NIR spectrometers even employing interferometers to make FT-NIR spectrometers. NIR spectrometers, however, are typically only dispersive.

NIR spectroscopy is capable of measuring either transmittance (and by extension, absorption) or reflection, and much like in ATR-FTIR, sample preparation is fast and simple, often without the need for altering or destroying the sample in any way. Samples can be measured either internally, typically using a cuvette of a specific dimension, or externally via a fiber optic probe. This probe can be found in many different configurations, and can be set up to measure the surface of a solid directly, or through a cuvette. Additionally, a mirror system is often employed allowing the probe to measure a liquid or even a gas without the need for a cuvette. In this case, the probe is positioned at a specific length away from the mirror, and the space between the mirror and the probe is filled with the liquid or gas which is to be examined. Some probes contain a built in mirroring system, allowing the probe to simply be submerged in the fluid, without the need to do any other preparations.

2.7 MIR versus NIR Spectroscopy

Although MIR and NIR spectroscopy are very similar in their operation and data collection, the ranges of the IR spectrum that they measure have very different properties. As previously mentioned, IR radiation is absorbed by molecules due to the bonds between atoms, and the overall structure of the molecule, leading to bending and stretching. As such, different functional groups on a molecule will have different energy levels that they absorb and emit, and this is based on what sort of bond is present and the atomic makeup of the group [14].

As an example, if pure carbon dioxide is measured in the MIR area, two primary peaks will appear in the spectra, as the only bond types are double bonded carbon and oxygen molecules, which are only capable of stretching and bending. The absorbed wavelength for the C=O bond bending is approximately 550 cm^{-1}, while the wavelength corresponding to the C=O bond stretching is found at approximately 2400 cm^{-1}, as seen in figure 10.

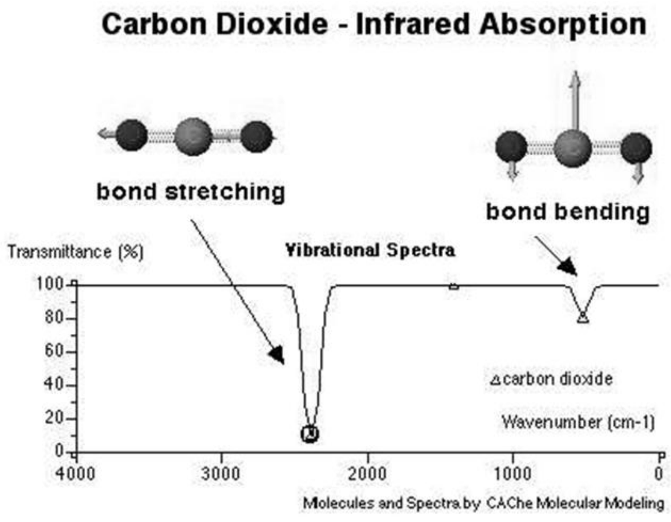

Figure 10: MIR spectrum for pure CO_2 gas [15].

Each functional group can have bending and stretching, and often multiple different types, each with its own associated wavenumber. In this way, it is possible to examine the spectrum and determine what sorts of functional groups are likely present within the sample, and make educated guesses as to what the compound may be.

In some cases, especially in relatively simple molecules, it is easy to determine, while more complex compounds, or a mixture of multiple compounds can prove troublesome. Samples in a solvent can be particularly tricky to examine, as the solvent will often dominate various functional group identification areas, making it difficult to determine if the sample itself is contributing to absorption or not. For example, aqueous solutions are very common, and the hydrogen-oxygen single bonds found in the water blanket a large range of wavelengths, as can be seen in figure 11, and will make determining certain functional groups of the sample more difficult [16]. The use of, for example, an ATR based IR spectrometer can prove useful for this purpose, as the low penetration into the sample limits the amount of absorption that will take place over an instrument that is passing a beam through the entire length of the sample.

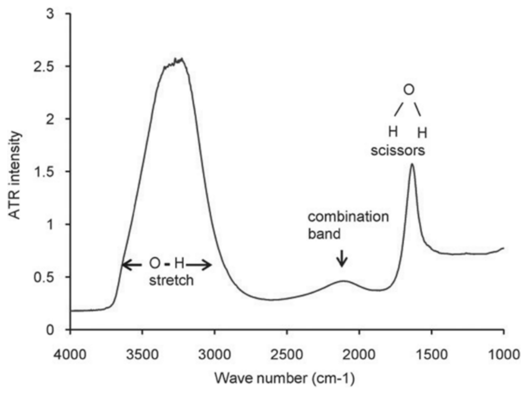

Figure 11: ATR-FTIR produced spectrum of pure water [16].

The MIR area of the IR spectrum is often the de facto area for identifying molecules, as it contains a significant amount of the common functional groups, especially for organic compounds, making for multiple routes of identification and categorization. In addition to this, functional groups in this range tend to have sharp, easy to identify peaks, which in turn also makes calibration of the spectrometer easier and less instrument specific [17].

MIR, however, has several downsides which must be taken into consideration. High absorption of common solvents has been mentioned, but this extends to many samples as well. Due to this, very short pathways are often key in limiting the total absorption in certain areas of the spectrum, which can easily make low concentration sample components more difficult to identify, if not outright impossible due to being blanketed or scaled down too heavy handedly [17].

The NIR area of the spectrum mostly contains the overtone and combination areas from the fundamental structures often identified in the MIR area. This can be used in tandem to strengthen the argument for certain functional groups being present, but is not always helpful due to the broader, less defined peaks found in NIR as compared to MIR. NIR, however, has several positives, the main one of which being the fact that the strong absorption of common solvents such as water are not present in the NIR spectrum as compared to MIR. This allows water and others to be used more readily, and path lengths for aqueous solutions to be longer, allowing for more sensitive detection of certain low concentration compounds when compared to MIR. NIR spectrometers are often more economically priced, which may be a determining factor in which type a group can afford to implement [17].

The primary negative of using NIR is that the afore mentioned overtones are relatively weak when compared to the primary peaks in MIR, and also not always well separated, which can muddle results in some applications, as can be seen in figure 12. Many NIR spectra also exhibit a strong upward trending in intensity with increasing wavelength, as can also be seen in figure 12. This also makes NIR spectrometers notoriously difficult and time consuming to calibrate, and each machine much be calibrated individually, rather than applying one calibration to multiple machines with MIR [17].

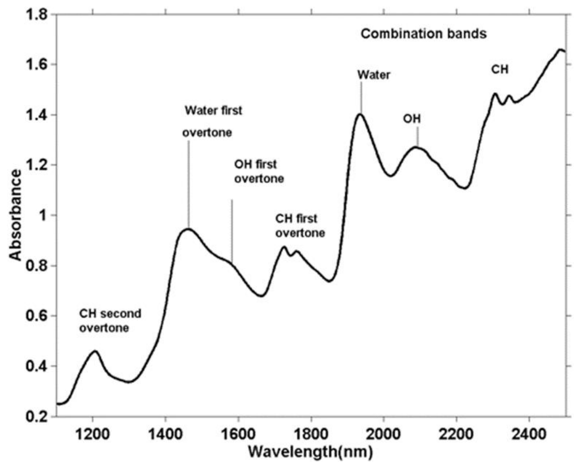

Figure 12: Spectrum of an aqueous solution taken with a NIR spectrometer [18].

2.8 Raman Spectroscopy

Raman spectroscopy is a spectroscopic method based on the Raman scattering phenomena primarily due to the inelastic vibrations and rotations in a system [19]. In Raman spectroscopy, high energy electromagnetic energy such as ultraviolet (UV), NIR or visible light is used from high powered laser systems. The photons from these energy sources are much too energetic to be absorbed directly, and therefore the molecule is excited into a high energy vibrational state from incident light. As with the absorption of photons, this increase in energy state is highly unstable, and the energy is emitted as scattering. If the scattering is elastic, the scattered light will have exactly the same amount of energy as the differences in energy levels between vibrational states from the incident light. This type of scattering is known as Rayleigh scattering, and is by far the most common type of scattering [20].

However, some of the scattered light loses energy during emission, leaving the molecule in a higher energy state than prior to the excitation. This is known as Stokes scattering. If a molecule is in an energetic state after Stokes scattering when it is then acted upon by incident light again, the system may lose additional energy when it emits again, leaving it in a lower energy state than when it began the vibration. This is known as anti-Stokes scattering. These two types of scattering are a subset of inelastic vibrations known as Raman scattering [20].

Raman scattering occurs in approximately $1*10^{-7}$ of the total scattered photons [21]. Due to this, the elastic scattering must be filtered out prior to photon detection or the effects of Raman scattering would be almost undetectable. It is worth noting that while Raman scattering is very similar to fluorescence, it differs in that it uses incident light which is never fully absorbed, unlike fluorescence.

A Raman spectrophotometer is made up of a light source, typically a monochromatic laser in the UV, visible or NIR spectrums, a series of laser rejection filters, a spectrometer and a detector. The laser provides the high energy incident light, while the laser rejection filters filter away the Rayleigh scattering so that nearly all of the light which arrives at the detector is from Raman scattering. The scattered light is then broken up into various wavenumbers, known as the Raman shift, and plotted against the intensity detected at that wavenumber. These wavenumbers are commonly in cm^{-1} units, while the intensity is entirely arbitrary, and useful only as a ratio compared to the other wavenumbers.

At a Raman shift of 0.0 cm^{-1}, no change in energy is detected, and is therefore known as the Rayleigh line due to this area being from Rayleigh scattering. There is often a sharp peak surrounding this point due to bleed over from errant Rayleigh scattering. Loss in energy is shown as negative wavenumbers, and therefore in the anti-Stokes area, while gained energy is shown as positive wavenumbers in the Stokes area, accordingly [22]. As Stokes scattering is typically much more intense, positive wavenumbers are often the focus for a Raman spectrum, an example of which can be seen in figure 13.

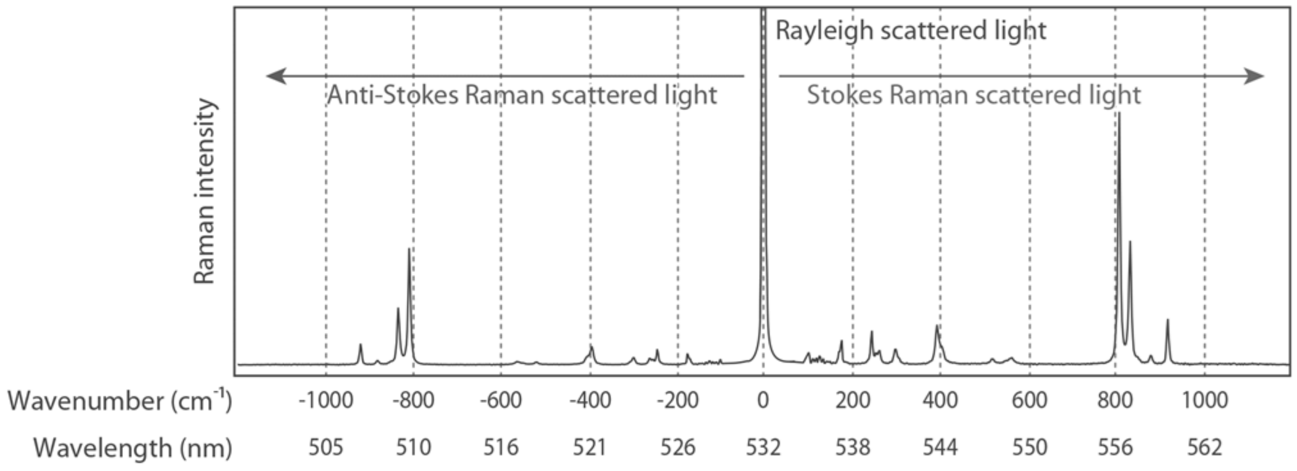

Figure 13: An example spectrum produced by a Raman spectrometer showing the various areas of the Raman spectrum [23].

Raman spectroscopy has several advantages over other forms of spectroscopy. Unlike some types of IR spectroscopy, little or no sample preparation is needed. Water in both liquid and gaseous phases scatter very little incident light, as does carbon dioxide, meaning that the chamber does not need to be evacuated, and both aqueous and solid samples can be analyzed. The process is also non-destructive, and can be carried out through glass or polymer sample containers. Raman spectroscopy can be used on both organic and inorganic samples, although metals and alloys cannot produce Raman bands, as it is not possible for them to have a change in polarization [20]. A major advantage for chemists is that Raman spectroscopy is extremely sensitive to the geometric structure of the system being examined. This allows for the differentiation between various allotropes of an element [20].

A downside of Raman spectroscopy, however, is that it is very sensitive to any fluorescence from the sample, and the rather weak signals from Raman scattering can easily be lost in a sea of fluorescent signal depending on the properties of the samples being tested. This also makes fluorescent lighting in the area incompatible with external testing of samples. Although there are many ways to reduce, or mitigate the damage done to the output spectra from fluorescence, it is difficult to remove it completely, often at the cost of additional noise within the spectra. This is compounded significantly if the sample peaks are much smaller than the fluorescence present, as the noise level within the fluorescence could possibly be as high or higher than the peaks themselves, drowning out the useful information completely [24].

3 Carbon Capture

Carbon capture is an idea that focuses on mitigating the amount of carbon dioxide that is being released into the atmosphere by collecting, storing, and if necessary, transporting CO_2 from the atmosphere or from flue gases, among others. Large producers of CO_2, such as fossil fuel power plants and refineries, are under constant pressure to reduce emission levels, as CO_2 is a well-documented greenhouse gas, acting as a thermal insulator for the planet, which has a complex impact upon the weather [25]. Rising concentrations of carbon dioxide in the atmosphere have been a global issue for decades and, as such, a concerted global effort to reduce CO_2 levels has taken off, especially in the last decade.

Common capture techniques include post-combustion gas scrubbing of the flue gas, pre-combustion techniques and the use of oxyfuel to create a nearly pure carbon dioxide exhaust that can be captured directly without the need for separation techniques [25]. An amine based capture method, using gas scrubbing, can be seen at CO_2 Technology Center Mongstad, outside of Bergen, Norway.

3.1 Amine based capture

Amine based carbon capture is a carbon capture technique which is relatively mature in industrial use. Despite that fact, the usage of amine based capture is still relatively young when applied to flue gases, which can be more complex due to their high oxygen concentrations, and the large scale at which most post-combustion plants are. A diagram showing the amine based pilot plant at CO_2 Technology Center Mongstad (TCM) can be seen in figure 14.

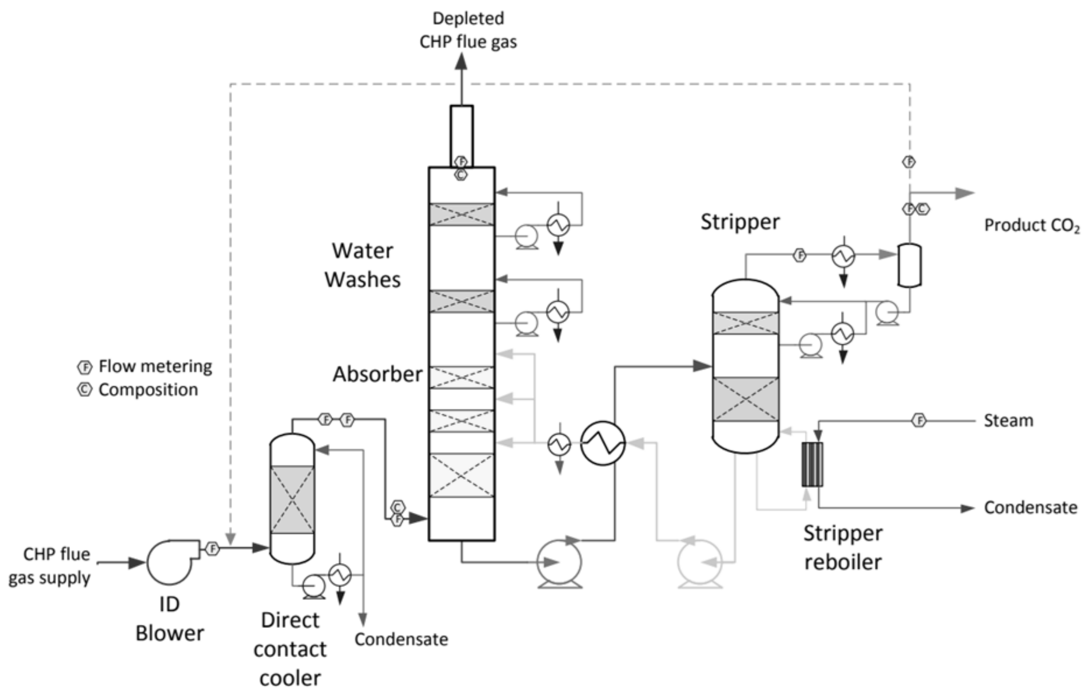

Figure 14: Diagram of the amine based pilot plant at TCM [1].

The plant at TCM uses a monoethanolamine (MEA) solvent for capturing carbon dioxide. The raw flue gas travels into the direct contact cooler where it is mixed with a wash water, allowing for a controlled temperature input into the absorber tower. The flue gas then rises through the absorber tower, where it contacts the amine solvent as it travels through multiple beds of packing material. The amine solution absorbs the CO_2 present in the flue gas, and the depleted flue gas is then removed through the top of the tower, while the rich amine solvent is removed through the bottom of the tower [1].

The rich amine solvent is then pumped into a stripper tower where it is heated in a steam reboiler which provides the energy needed to drive the endothermic regeneration cycle where the CO_2 is removed from the rich solvent. The lean amine solvent is then removed through the bottom, and recycled back into the absorber tower to absorb more CO_2, while the previously captured CO_2 travels up the stripper tower, and is removed along with water vapor, which is then condensed and recycled back into the system, allowing the whole process to be operated water neutral [1].

An issue with this process is that during regeneration, some of the solution degrades, which in turn inhibits the solvent's ability to absorb CO_2, as well as creating byproducts that are damaging to the equipment itself, including the formation of solid precipitate carbamate, which is capable of building up in the equipment and pipes, leading to equipment failure and blockages. Due to this, there is a need for a system that can monitor the buildup of these degraded compounds so that the system can be cleaned out and the amine solvent replaced before costly damages are sustained.

It is common for various spectroscopic methods to be used in order to determine concentrations of various compounds in a given sample. This is especially common in chemical process plants where one or more spectrometer is used to monitor the concentrations of a flowing fluid in order to verify that the processes are working correctly and that the product is being produced at the quality that it should be [2].

Figure 15 shows the various methods that can be employed. In-line sensors monitor the bulk flow as it travels uninterrupted through the system, while online sensors work very similarly, but only analyze a portion of the flow that is temporarily diverted from the bulk flow. Off-line methods typically involve a built-in tapping location where manual sampling can take place. A small amount of the bulk flow is completely removed from the system, and taken to a different area for analysis and evaluation. Off-line solutions are more time intensive, and therefore do not give real time data and, due to this, are not especially suited for processes that may be subject to rapid changes in quality and/or concentrations [26].

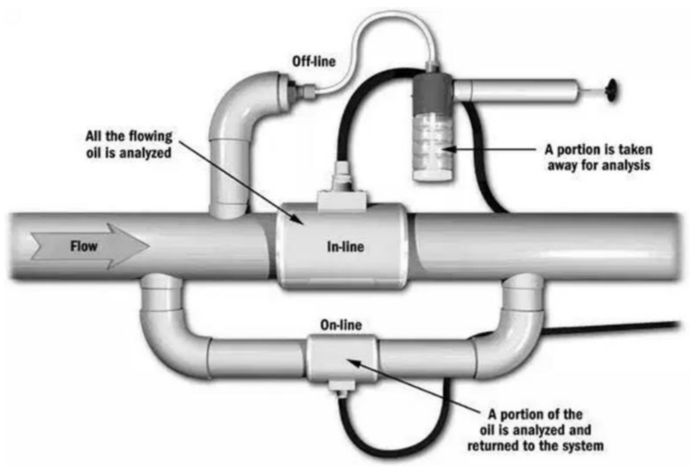
Off-line
All the flowing oil is analyzed
A portion is taken away for analysis
Flow
In-line
On-line
A portion of the oil is analyzed and returned to the system

Figure 15: Example of multiple techniques for process monitoring [26].

4 Data Manipulation and Analysis

When a given dataset is to be analyzed, especially in the case of instrument measurement datasets, the data must often be manipulated in various ways before it can then be analyzed. The type of data being examined, as well as what sort of dataset it is, will often play a role in what sort of manipulation and analysis will take place, and the techniques that will be employed.

4.1 Preprocessing

A key area of data manipulation is called preprocessing. Preprocessing is, as the name may imply, the processing of the data prior to the data being analyzed. Although this holds true for many techniques, this is not always set in stone, and some preprocessing techniques may take place after certain analysis techniques have be applied to the dataset. Regardless, preprocessing always takes place prior to the final analysis of the data in order to help the analysis, either by speeding up the process, reducing noise, or removing unwanted or disturbing data from the set. This is typically achieved by cleaning, transforming, and/or reducing the data [27].

During data cleaning, several steps can take place. Typical steps include removing, or filling in, empty data cells, removing outliers from the set, smoothing nominal noise within the set via filtering, or applying knowledge of the instrument or samples to correct for known issues, among others [27].

Data transformation techniques typically include sweeping changes to the overall makeup of the dataset, which can include normalization or standardization of the dataset to correct for data entry size issues or deviations, changing the overall scale or shape of the data, or various other methods which aim to make calculated changes to the entire dataset for whatever reason [27].

Data reduction techniques are rather straightforward in that they work to reduce the amount of data that must be processed in later analytic steps. This is typically done using knowledge of the data to remove data that is known to be useless or harmful, but also through variable reduction techniques, by either directly removing variables from the dataset, or removing unnecessary or redundant variables from the dataset [27].

4.1.1 Normalization

Normalization is a data transformation technique that is used to make variables comparable with one another. This is an important step in the overall preprocessing of the data, as statistically important variables that are relatively small when compared to other variables in the dataset would otherwise be ignored or marginalized during analysis, possibly even against variables that have no viable information. This is especially important when the variables within the dataset are not measured on the same scale, or in the same units. As an example, if there is one variable in meters, and another in kilometers, it would appear as though 5 meters is a larger value than 2 kilometers, which does not represent reality, and a variation of 1 meter will have drastic effects on one, but not the other [28].

There are many different ways to compute normalization on a dataset. One common way, known as feature scaling, or unity-based normalization, is shown in equation 3.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad \textit{\textbf{Equation 3}}$$

where X_{norm} is the normalized dataset, X is the original dataset, X_{min} is the minimum value within the set, and X_{max} is the maximum value within the set.

What this does is it scales all of the data into ranges between 0 and 1, no matter what the data was originally. The smallest value in the original dataset then becomes 0, and the largest becomes 1. This then allows each variable to more accurately display its variation, which in turn allows the variables to be compared to each other more fairly in further data analysis [29]. As can be seen in figure 16, the plot on the left makes it difficult to make out how much variation is taking place in two of the variables, while the middle range variable has a large amount of variation. During data analysis, the variations in the top (near 1000 on the Y axis) variable, and the variations in the bottom (near 0 on the Y axis) would almost certainly be completely ignored, and it would appear as though only the middle variable has any useful data in the variation. However, when normalized, as seen in the plot to the right, the variation becomes much more pronounced, and each variable will now contribute fairly to the analysis.

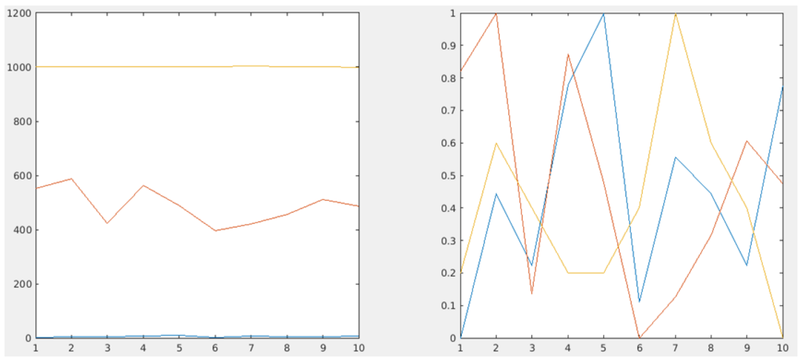

Figure 16: Plot of three raw datasets (left) and plot of the three corresponding normalized datasets (right).

The downside of normalization is that it can be extremely sensitive to outliers. As it makes the smallest value equal to zero, and the largest equal to one, an outlier that is significantly larger or smaller than the rest of the data will set the scale boundaries, and can heavily influence later analysis. Due to this, it is often wise to remove potential outliers before normalization if at all possible [29].

4.1.2 Signal Smoothing

Signal smoothing involves smoothing out false peaks in a continuous spectrum due to relatively minor noise within the system, improving the signal to noise ratio. Most prominent techniques are filter based, such as the commonly used Savitzky-Golay filter [30] [31].

The Savitzky-Golay filter works by gradually moving from one end of the spectrum to the other, selecting data points in a user specified window size. This window, always an odd number, of data points is used to generate a user specified degree polynomial, often from first to third, which best fits the window's data. This polynomial then calculates a predicted midpoint of the window and this predicted value becomes the first data point of the filtered spectrum. The window then moves one data point further along, repeats the process, and continues on until the entire spectrum is finished. Due to the fact that the window has a width larger than one, some variable loss is incurred, at the rate of half the window size, rounded down, per end of the variable dataset. So a window size of 11 would cause the first 5 and last 5 variables in the spectrum to be removed. This slight variable loss is not typically an issue in datasets with a large amount of data, such as spectra [30] [31].

The larger the window that is used in the filter, the broader the peaks that will be harshly smoothed. Due to this, care must be taken to choose a window size that will smooth out unwanted noise, but not destroy or blunt peaks that hold valuable information. In some applications, multi-pass filters are employed with smaller windows which repeat the smoothing algorithm to create the more representative smoothing you would get from a wider window, without the added risk of removing valuable information. Figure 17 shows a comparison of a spectrum before and after the Savitzky-Golay filter has been applied, demonstrating the power the filter has at removing noise.

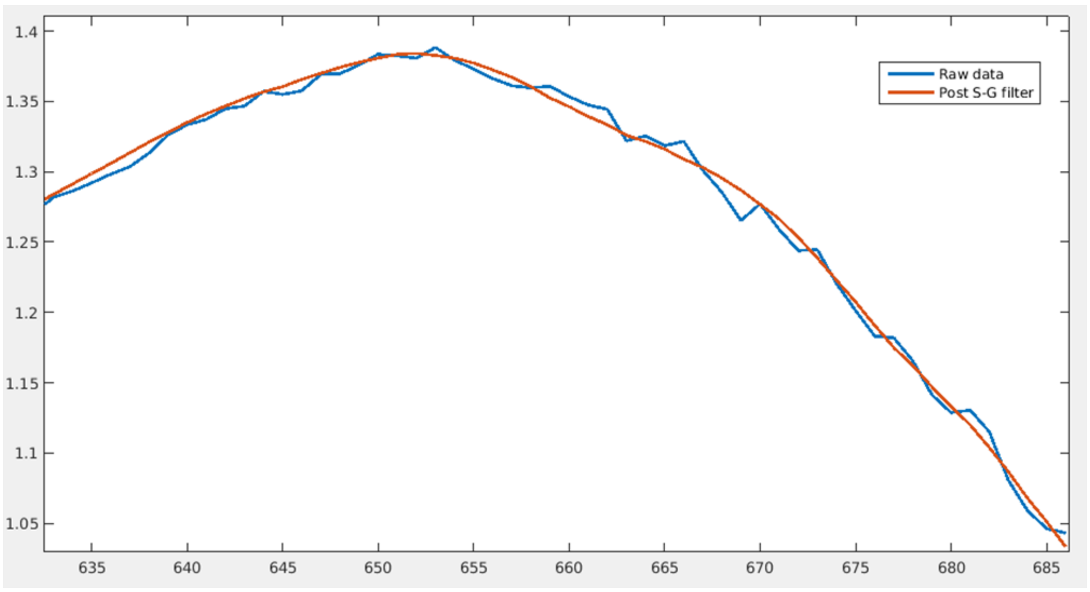

Figure 17: Example of the smoothing effect Savitzky-Golay filtering provides to noisy data, run with a window size of 15, and a 3rd degree filtering polynomial. Axis values are arbitrary numbers.

The Savitzky-Golay filter is often used for data transformation as well as smoothing, by outputting the filtered points as the derivative of the midpoint, rather than the midpoint itself. Calculating the derivative of a spectrum is incredibly difficult, as there is no direct formula to be derived. However, the polynomials created by the filter's window are trivial to derive, making it quick and efficient at outputting derived spectra. This has the added benefit of significantly reducing false local maxima and minima that noise would generally create.

4.1.3 Multiplicative Scatter/Signal Correction

Multiplicative signal correction (MSC), also known by its original name, multiplicative scatter correction, is a technique used to remove or reduce additive or multiplicative variation in related spectra which is not due to the properties of the sample itself. Due to varying path lengths and other phenomena, the detected signal within a spectrometer may vary slightly up or down without representing the chemical properties of the sample. This will cause a separation of the curves from multiple samples without this separation being due to a difference in concentration or chemical makeup, for example [32]. This can lead to analytical techniques being applied later that take this variation into account when creating a predictive model, when that particular area of the spectrum offers no predictive qualities.

MSC works to remove these variations by attempting to fit the spectra as best it can over the top of each other, so that non-predictive areas overlap, rather than vary. Once this is accomplished, it is also much easier to visually determine areas which are of importance, and areas that are not. Figure 18 shows a comparison of Raman spectral data before and after MSC has been applied.

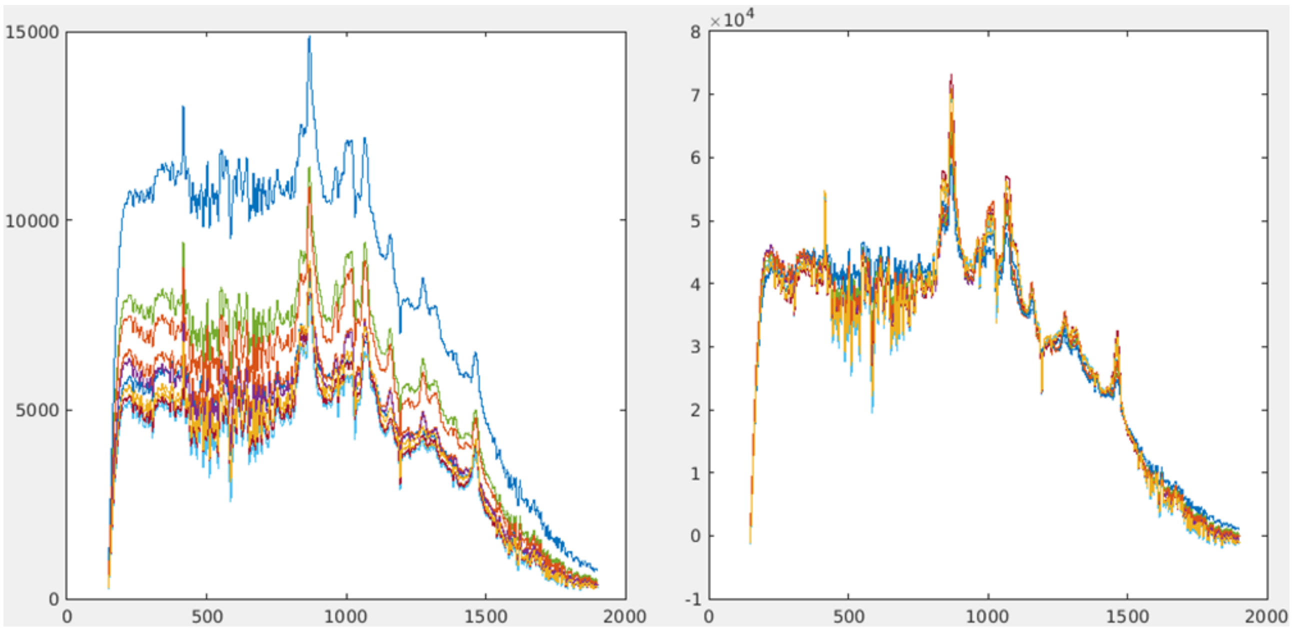

Figure 18: Plots comparing raw Raman spectral data (left) to MSC processed versions of the same spectra (right).

MSC accomplishes this by plotting a reference spectrum, often simply the average of all the sample spectra, against each sample spectra, and then running least squares linear regression on it to determine the regression coefficients of the plot, as shown in equation 4 [32].

$$X_0 = a + X_r * b + e$$ ***Equation 4***

where X_0 is the spectrum to be corrected, a and b are regression coefficients, X_r is the reference spectrum, and e is the residual.

These regression coefficients are then used to correct the individual sample spectra by first removing the intercept, a, and then dividing it by the slope, b, as shown in equation 5 [32].

$$X_{corr} = \frac{X_0 - a}{b}$$ ***Equation 5***

where X_{corr} is the corrected spectrum.

Although MSC is a powerful tool on its own, it can be expanded upon to correct for many different phenomena which disturb the measured signals. This expanded type of MSC is called extended multiplicative signal/scatter correction (EMSC), and is not tightly defined in what exactly it is, only that it adds extensions to MSC in one or more ways. A typical expansion on MSC is detrending, where a trend within the data is removed to correct for its presence [32]. This detrending expansion on MSC is a common form of EMSC, and involves reducing signal at a given wavelength based on a polynomial curve which is dependent on wavelength. This is typically done by adding variables to the data matrix which are wavelength dependent, as shown in equation 6.

$$X_0 = a + [X_r + \lambda + \lambda^2 + \cdots] * b + e$$ ***Equation 6***

where λ is the wavelength vector.

The detrend polynomial can be of any order, but should be chosen wisely, as it is only meant to remove a trend from the dataset, and overfitting of the polynomial will instead remove useful data from the dataset by pushing the detrend into the peaks. An example of an MSC processed dataset compared to a 2nd order detrend EMSC processed dataset can be seen in figure 19.

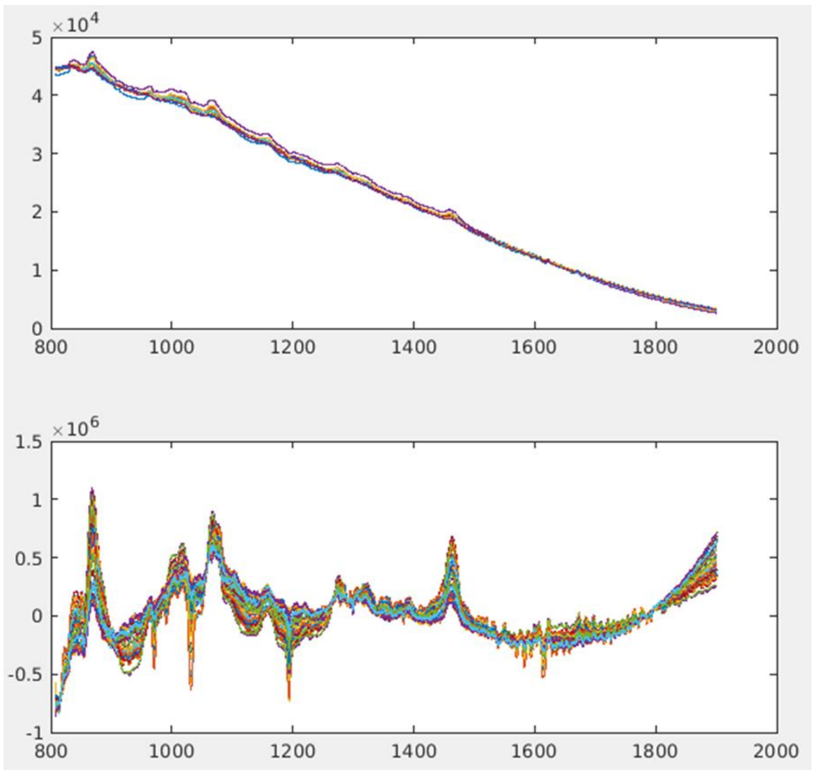

Figure 19: Example of MSC processed Raman spectra (top) compared with the same spectra after EMSC processing with a 2nd order detrend (bottom).

As shown in figure 19, peaks which were once dwarfed by a high intensity trend are now clearly identifiable and become much easier to separate from areas of noise, such as from about 1500 to 1800.

4.1.4 Data Fusion

Data fusion is a series of techniques used to combine datasets from multiple sensors in a way that creates better predictive models than would otherwise be possible with just one set of data alone. There are three main types, namely low-level, mid-level (also known as intermediate level) and high-level data fusion. Each data fusion level is separated by where in the model creation process the data is fused [33]. As such, not all types of data fusion are strictly considered preprocessing, although the most common types are.

Low-Level Data Fusion

Low-level data fusion (LLDF) is the most common, and also simplest, method of data fusion. In LLDF, the predictor datasets are concatenated in a series to create one large predictor dataset. There are several techniques that are employed to do so, one such being the running of variable selection for each dataset before concatenation, but typically the raw datasets are fused prior to variable selection. This is done so that there is a chance that linearities between variables in different datasets, that are possibly uninteresting in their own dataset, have a chance to become important in the fused dataset. However, prior to fusing the datasets, it is typically a requirement that each dataset be at least normalized individually. This is so that the scales between various datasets do not dominate or trivialize other datasets, as normalization after the fusion will not typically fix this. Other preprocessing techniques such as MSC, EMSC, signal smoothing, etc. are also often performed before LLDF, although not always [33] [34].

After the datasets are fused, further processing, and eventually modeling, takes place as would be typical with any other single dataset. A diagram depicting the basic idea and flow of LLDF before variable selection can be seen in figure 20.

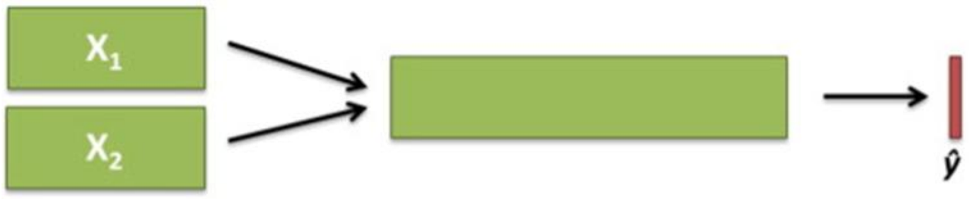

Figure 20: Basic processing flow employed by low-level data fusion [35].

Mid-Level Data Fusion

Mid-level data fusion (MLDF) is a relatively common method of data fusion, and involves fusing the data later in the processing cycle than LLDF. One method for MLDF is fusing the datasets after all preprocessing and transformations have taken place at the individual level [36]. This type of MLDF is a simpler version of MLDF, and has a lot of overlap in common with LLDF. A more complex, although more typical, MLDF technique is fusing datasets generated by the first steps of modeling. Often, these datasets are comprised of selected latent variables or scores from PCA or PLS, among others [33] [34] [35].

In a typical MLDF, some sort of modeling technique will be applied to the datasets individually, and the most important features from each model will then be fused into a single dataset, as can be seen in figure 21. This new dataset will then be used to create a new model, which in some, but not all, cases has a better predictive ability than LLDF. The most difficult part of MLDF is determining which features are most important from each initial model so that the optimal number or variables will be carried further into the final modeling stage.

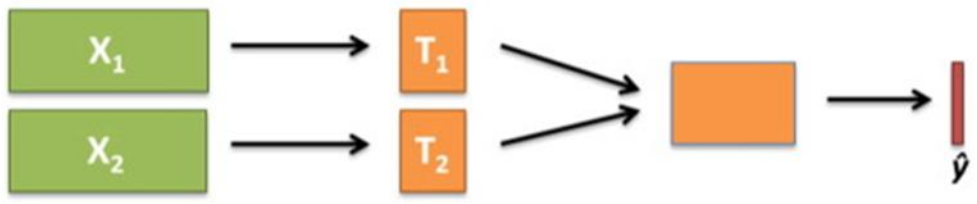

Figure 21: Basic processing flow employed by mid-level data fusion when modeling from PCA or PLS scores [35].

It should be noted that the number of features from each model does not need to be the same, and one model may only have a few features, while another model may have many. A secondary pass of preprocessing of the fused dataset may also be required in order to reduce possible scaling issues.

High-Level Data Fusion

High-level data fusion (HLDF) involves running predictive models individually on each dataset and then fusing the predictions from each dataset into a new dataset. Various methods are then used to decide which prediction is most likely correct, and then the prediction from that model is moved into a final prediction [33] [34] [35]. For this reason, HLDF is often referred to as decision level data fusion. A basic diagram of this can be seen in figure 22.

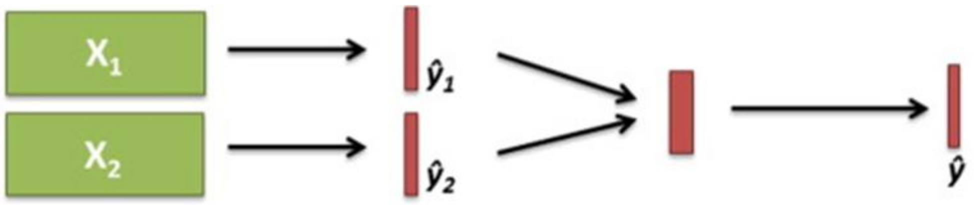

Figure 22: Basic processing flow employed by high-level data fusion [35].

Although there are many ways to go about determining the best prediction, majority vote is a common way to decide. Majority voting involves comparing the predictions for a given object that are supplied by the models, and picking the one which is most common [37]. For quantitative analysis and prediction, this is ineffective, and much more advanced mathematical methods must be applied in an attempt to determine the probability of each one being correct, and then picking the one with the highest probability, often through Bayesian inference methods [37]. For this reason, HLDF is almost exclusively reserved for qualitative, or classification prediction.

As an example of HLDF, three models are built, and two models say that the answer is category A, while one says that it is category B. The majority vote would determine that the final prediction will be category A. This can be beneficial when attempting to determine if a sample contains a substance at all, or if a certain physical characteristic is present or not, among other examples.

4.2 Partial Least Squares Regression

Partial least squares regression (PLS-R, often just PLS) is a technique used to create predictive models in situations where the variables outweigh the objects, and the variables are largely collinear. This is typical in spectroscopic datasets where there may only be a handful of samples (objects) but thousands of variables (wavelengths/wavenumbers) to work with. PLS attempts to predict the response, or responses, in an object while reducing the overall residuals as much as possible [38] [39].

PLS accomplishes this by breaking down the variable matrix and extracting latent factors, each of which explaining a certain amount of the variance present. These latent factors, or components, are projections of scores onto loadings which attempt to recreate the original predictor dataset. Subsequent components that are extracted are then added to the previous attempt at recreating the original predictor dataset, which slowly builds a predictor dataset which closely, or, given enough components, possibly exactly matches the original predictor dataset [38] [39].

For modeling a single response variable, PLS multiplies the transposed predictor matrix, containing the variables and objects, by the response vector, and then divides that by its norm. This creates a weighting vector, as seen in equation 7 [38].

$$w_i = \frac{X' * y}{||X' * y||}$$ ***Equation 7***

where w_i is the weighting vector for component i, X’ is the transposed predictor matrix and y is the response vector.

The predictor matrix is then multiplied with weighting vector, which produces a score vector, as seen in equation 8 [38].

$$t_i = X * w_i$$ ***Equation 8***

where t_i is the score vector for component i.

The transposed predictor matrix is then multiplied by the score vector, and subsequently divided by the sum of squares of the score vector, producing the loading vector, as seen in equation 9 [38].

$$p_i = \frac{X' * t_i}{t_i' * t_i}$$ ***Equation 9***

where p_i is the loading vector for component i.

By projecting the score vector onto the transposed loading vector, the predictor matrix approximation for the component is created. This approximated predictor matrix is then subtracted from the original predictor matrix, and the resulting matrix is then the residual matrix containing the unexplained variance left over, as shown in equation 10 [38].

$$X_{unexplained} = X - t_i * p_i'$$ ***Equation 10***

where $X_{unexplained}$ is the predictor matrix that is leftover after the explained variance is removed.

This cycle continues for the number of components that the user specifies, and when the cycles are complete, the resulting unexplained predictor matrix becomes the residual matrix. A prediction response is then created based upon the basic principles of linear regression, shown in equation 11.

$$Y_{predicted} = X * b + E$$ ***Equation 11***

where $Y_{predicted}$ is the predicted response, b is the regression coefficient vector and E is the residual matrix.

Although it is the predicted response that is often the primary focus of PLS, information about the system can be gleaned from the score and loading vector as well. By plotting the score or loading vector from a given component against the score or loading vector (corresponding accordingly) of a different component, structures within the data can often be observed. This can take place in the form of trends, clustering, or other organized structuring. This can often be of use for determining how similar objects are, or a certain category that they may belong to. It can also be used in some situations to determine if objects are outliers or begin to stray away from the norm.

4.2.1 Outlier Detection

Outlier detection is a technique used to determine if a given object varies significantly from what would be expected when compared to similar objects [40]. This is important due to the fact that outliers within the system create problems in several preprocessing steps, and can alter the overall model in a way that diminishes the predictive capability of the model.

There are many ways of performing outlier detection, many of which involve plotting a residual transformation against leverage. Leverage describes the influence that an object has on the model due to its distance from the average object [40]. Leverage is found in the diagonal of the Hat Matrix, and is calculated for each sample, using equation 12 [41]. A high leverage value on its own does not indicate an outlier, but simply shows that the object has the potential to create large changes in the model due to how far away it is. This is illustrated in figure 23.

$$H = T * (T' * T)^{-1} * T' + \frac{1}{n}$$

Equation 12

where H is the Hat Matrix, T is the score matrix containing the score vectors for each component, and n is the number of objects.

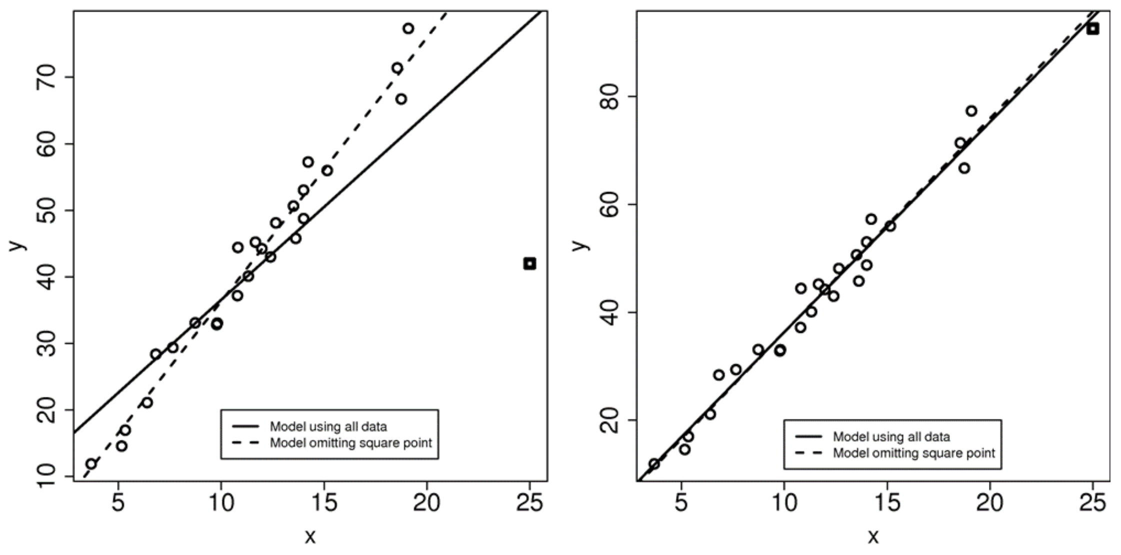

Figure 23: Plots showing points with high leverage, and how the addition of a high residual (left) or low residual (right) affect the modeling ability [42].

As a general rule of thumb, any leverage value above 2m/n, where m is the number of components, and n is the number of objects, is cause for concern, and should be examined [41]. As leverage alone is not necessarily a good indicator that the object is disturbing the model, it should be used in tandem with some sort of residual technique. This is often done by plotting the leverage against the residual standard deviation (RSD), Studentized residual, or some other form of residual standardization [40] [41].

The RSD is calculated using the residual matrix, as shown in equation 13, and represents the standard deviation found within the residual matrix.

$$RSD = \sqrt{\frac{e_i' * e_i}{m-n}}$$

Equation 13

where e_i is the residual vector for object i, m is the total number of variables, and n is the number of components used.

As RSD represents how large the residual is, and leverage represents the potential an object has to disturb a model, any object with both a high leverage and a high RSD should be removed from the dataset. However, any object with a high RSD and low leverage, or low leverage and high RSD should also be considered as possible outliers. This relationship is easily visualized in an RSD versus leverage plot, as seen in figure 24.

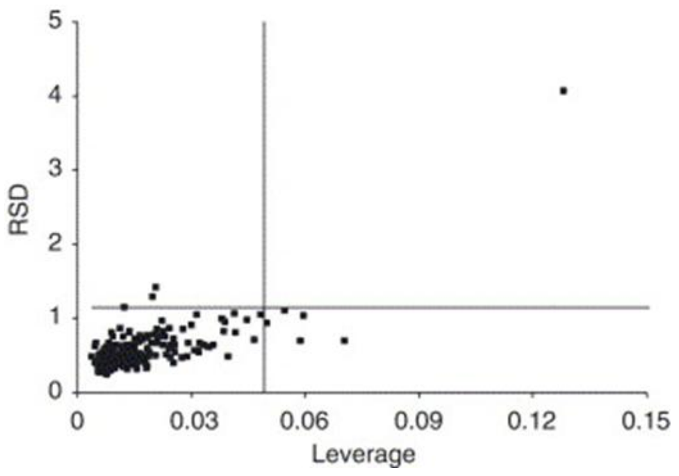

Figure 24: RSD versus leverage plot, showing the upper bound for the residual (horizontal line) and upper bound for the leverage (vertical line) [43].

The Studentized residual is often used rather than RSD as the residual to compare against leverage for outlier detection. The Studentized residual is essentially the ratio of the residual to the estimated standard deviation of the residuals. This is calculated using equation 14 [40] [42] [44]. Once calculated the Studentized residual can be plotted against the leverage, and the plot can be interpreted similarly to the RSD versus leverage plot, although there will now be an upper and lower bound for the residual, as the values are no longer strictly positive, as shown in figure 25. Common bounds for the Studentized residual include ± 2 and ± 3 [40] [44].

$$e_{student,i} = \frac{e_i}{\sqrt{MSE*(1-h_{ii})}} \qquad \textbf{\textit{Equation 14}}$$

where $e_{student,i}$ is the Studentized residual for a given object, MSE is the average of the squared prediction residuals, and h_{ii} is the i’th leverage value.

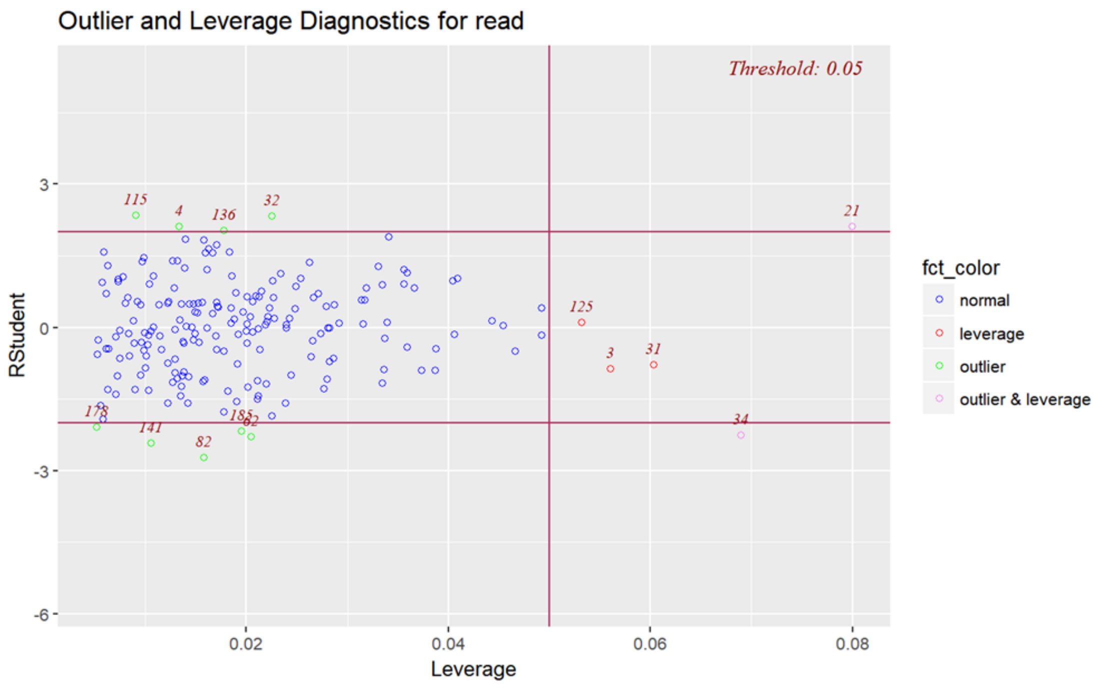

Figure 25: Studentized residual versus leverage plot, showing upper and lower bounds for the residual (horizontal lines) and upper bound for leverage (vertical line) [45].

As previously mentioned, information about outliers can also be found in the score plots, and potential outliers should be cross-referenced there, and vice versa. If a residual versus leverage plot detects a potential outlier, and a score plot shows the same object deviating significantly from the expected area, it is very likely that this object is indeed an outlier, and should be removed from the model to see if the model improves or not. It is important to note, however, that if a cluster of objects appear to be outliers from the main object structure within the score plot, it is possible that they have a categorization which has not been examined, which should be looked into.

4.2.2 Model Validation

An important step in determining the real world prediction capability of a model is by using techniques to validate the model(s) generated. A model that makes accurate predictions for objects that were used to create the model may not be able to make accurate predictions for objects that were not used to create the model. This is typically caused by either overfitting, that is, using too many components and therefore modeling components that do not describe the data properly, or underfitting, which is caused by using too few components, and not modeling enough of the useful information.

Model validation can be performed several ways, including by simply using an external dataset of objects that will not be used for modeling at all. However, cross-validation can provide insight into the model's predictive capability without the need for an external dataset [46].

4.2.3 Cross-Validation

Cross-validation is a set of techniques which use the modeling data to create smaller models, using some of the objects from the modeling data as unknown, external objects. The objects selected to do the modeling are known as the training set, and those used to validate the model are known as the test set. There are several ways to perform cross-validation, with various techniques for selecting which objects will be used in the test set.

In order to judge the validity of the model, the overall residuals between the predicted and known responses for the test set must be analyzed. This is done in several ways, but a common residual type is known as the root mean square error of prediction (RMSEP). As the name implies, it is simply the square root of the squared mean of all the prediction residuals, as shown in equation 15 [47].

$$RMSEP = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$ ***Equation 15***

where y_i is the measured response for object i, $\hat{y}_i$ is the predicted response for object i, and n is the number of objects.

If the RMSEP is low for the model compared to the size of the response, then the average residual is low, which indicates that the model was relatively accurate in its predictions of the response. This would then indicate that the model is likely a good model, and it can be inferred that it will likely do well at predicting responses from external test sets.

For cross-validation to be effective, it is typically run multiple times, using a different training and test set each time, and for a varying amount of components. To better judge how effective the model is, as well as how many components should be used to be optimal, the root mean squared error of cross-validation (RMSECV) must be calculated, as shown in equation 16. The RMSECV value is important, because an individual RMSEP value can vary quiet significantly during cross-validation repetition, so checking a single test set may give a low RMSEP, but that may not necessarily indicate that a full model will perform well. The RMSECV value gives a much better indication that the full dataset will create a model that predicts external sets well [46].

$$RMSECV_a = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_{i,a} - y_i)^2}{n}}$$

Equation 16

where $RMSECV_a$ is the RMSECV for component a, n is the number of objects, and $\hat{y}_{i,a}$ is the predicted response for object i with component a.

Not only can the RMSECV be used to give an indication of how well the final model may predict external sets, it can also be used to determine the optimal number of components that should be included in the model. This can be achieved in several different ways, but is often done by running cross-validation with a varying amount of components, and then plotting the RMSECV against the number of components. The plot will indicate at what number of components the RMSECV ceases to decrease, which is then the number of components that should be used [46]. A visual representation can be seen in figure 26.

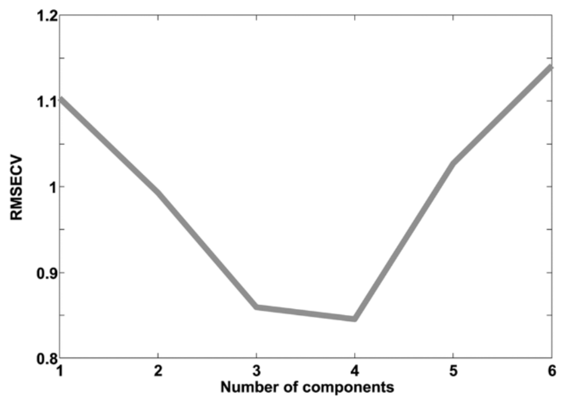

Figure 26: Plot of RMSECV versus the number of components used in PLS, showing a minimum value of 4 components [46].

As shown in figure 26, the RMSECV stops decreasing at 4 components, and actually begins to increase dramatically afterwards. This is not always the case, as the RMSECV often simply flattens out, and stops improving significantly, making an optimal amount of components a bit more difficult to determine, but in this case, adding more components than 4 will actually hurt prediction, and therefore no more than 4 components should be used.

Monte-Carlo Cross-Validation

As mentioned previously, there are many ways that the objects can be selected for use in each test and training set. Many of them include a user defining how many subsets to break the objects into, and then one subset becomes the test set, while the rest become the training set, and then cross-validation is repeated once for each subset. Perhaps a more novel approach is selecting a given test set at random from the objects. This is known as Monte-Carlo cross-validation (MCCV) [48].

Although using static subsets can give valuable information about the predictive capability of a potential model, the results will always be the same for each subset, and this can mask potentially damaging subsets. Rather than test each subset exactly once, giving each object one time in the test set, MCCV allows for objects to be selected multiple times in a user defined number of subsets, until every possible combination of objects was used for the test set. This, however, is usually not practical, as the number of possible combinations grows increasingly large with each additional object to be put into the test set. For this reason MCCV is typically run a limited amount of times, and the overall performance of all the models is determined to be approximately the same as if all of the possible combinations were used [48]. A visual representation of this random selection can be seen in figure 27.

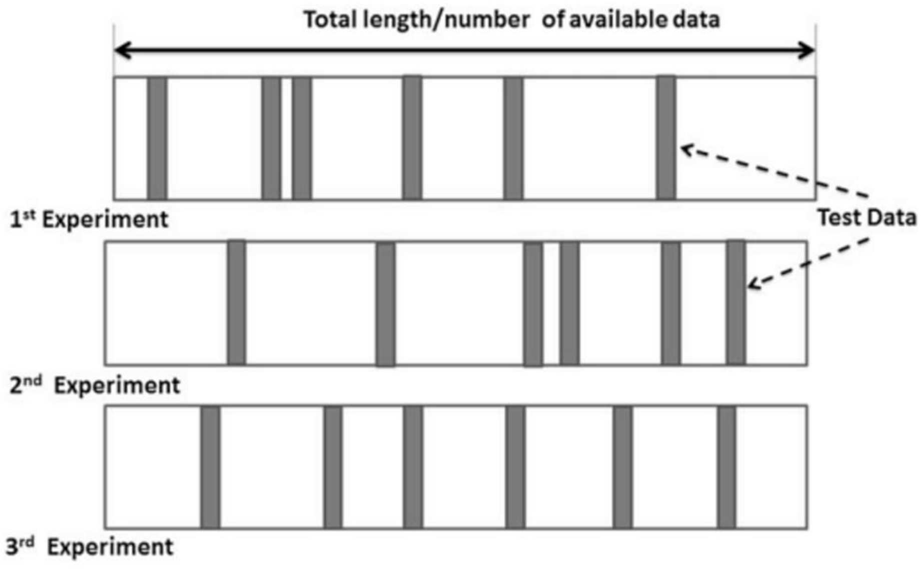

Figure 27: Diagram showing the random nature of Monte-Carlo cross-validation sampling [49].

4.2.4 Variable Selection

In many datasets, including those generated by spectroscopy, there are a very large number of variables. Many of these variables hold the same general information as other variables, as they are correlated with one another, and many others simply do not hold any information that is useful for modeling the desired response. In some cases, noise within these unnecessary variables actually degrades model performance. On top of all of this, a large number of variables makes computing and testing models much slower. For this reason, variable selection is employed to reduce the number of variables in the dataset, with the goal of improving the overall model, or at least not affecting the overall model's predictive ability [50].

Variable selection has many methods, from both manually removing variables that are known to cause problems or be unimportant, to automated techniques which attempt to identify possible candidates for removal. Two techniques that are often used are variable importance in projection (VIP) and the selectivity ratio (SR).

Variable Importance in Projection

VIP is a technique that generates scores that estimate the importance of a given variable in a PLS projection model. These scores can then be examined to determine which variables are more or less important, and the less important variables can then be candidates for removal. Although there is no clear cut score value that determines that a variable should be removed, it is generally accepted that a value greater than 1.0 can be considered important for the given model, although scores below that do not necessarily indicate that they are not important. For this reason, the threshold is often set a bit below 1.0, and models are generated with various datasets generated at differing thresholds, and then the resulting predictive errors for each threshold can be compared to make a more informed decision on which threshold should be used [50]. The scores can be calculated using equation 17.

$$VIP_k = \sqrt{\frac{K_T \sum_{a=1}^{A} w_{a,k}^2 * SS_a}{\sum_{a=1}^{A} SS_a}}$$ ***Equation 17***

where K_T is the total number of variables, $w_{a,k}$ is the weight for variable k and component a, and SS_a is the sum of squares explained by component a.

Selectivity Ratio

The selectivity ratio is the ratio between the explained variance and the residual variance from a given model, and it measures the ability of a given variable to predict a given response. A high ratio indicates that the amount of explained variance is significantly higher than that of the unexplained variance, which is an indication that the variable is useful in determining the response [51].

In order to determine the SR, a target projection must be done. This is done very similarly to standard PLS, where weights, scores and loadings must be generated. The target projection weights are created by normalizing the regression coefficients from PLS, as seen in equation 18 [51].

$$w_{TP} = \frac{b_{PLS}}{||b_{PLS}||}$$ ***Equation 18***

where w_{TP} is the target projection's weights, and b_{PLS} is the regression coefficient vector from PLS.

From this, the scores are calculated by multiplying the predictor matrix by the weight vector, as seen in equation 19.

$$t_{TP} = X * w_{TP}$$ ***Equation 19***

where t_{TP} is the score vector.

From there, the estimated predictor matrix and the residual matrix can be determined using equation 20.

$$X = t_{TP} * p'_{Tp} + E_{TP}$$ ***Equation 20***

where t_{TP} * p_{TP}’ represents the estimated predictor matrix, and E_{TP} is the residual matrix.

The SR is then determined as the ratio between the variance in the estimated predictor matrix and the variance in the residual matrix, as shown in equation 21.

$$SR_i = \frac{v_{exp,i}}{v_{res,i}}$$ ***Equation 21***

where SR_i is the selectivity ratio for variable i, $v_{exp,i}$ is the variance explained by variable i, and $v_{res,i}$ is the variance of the residual on variable i.

Similar to VIP, the vector can then be plotted to visually inspect areas of the spectrum that are important to the prediction of the response [51]. Normalizing the SR vector can often be useful when attempting to automate variable reduction, as it makes the SR score less arbitrary. Although there is no standard SR value that would lead to the variable being considered important, as there is in VIP, a case by case decision can be made by examining the SR plot.

5 Method

5.1 Software

MATLAB 2018b (The MathWorks, Natick, Massachusetts, USA) was used for all calculations made for the project, including preprocessing and multivariate data analysis.

OMNIC 9.8 Spectra Software (Thermo Scientific, Waltham, Massachusetts, USA) was used to gather and organize the MIR spectra during measurements.

FOSS NIRSystems Vision 2.11 (Metrohm AG, Ionenstrasse, Switzerland) was used to gather and organize the NIR spectra during measurements.

Kaiser Optical Systems HoloGRAMS (Endress+Hauser AG, Reinach, Switzerland) was used to gather and organize the Raman spectra during measurements.

5.2 Measurements

The MIR measurement data was provided by Helene Irgens Sjo, a master's student for the Department of Chemistry at the University of Bergen. The data was collected using a Nicolet iS 50 FTIR Spectrometer with an ATR diamond, a machine capable of a spectral range of 15-27000 cm^{-1}, with a resolution better than 0.09 cm^{-1}, making 32 scans in the area of 400-4000 cm^{-1}.

A drop of the sample liquid was placed directly onto the ATR crystal, and the measurements were made. Between each sample, the crystal was cleaned and dried, and the background measurement verified prior to the next sample being tested. A new reference background spectrum was taken every 30 minutes.

The NIR measurement data was collected in the laboratory of the Department of Chemistry at the University of Bergen using a FOSS NIRSystems Model 6500 scanning the area between 1100 and 2500 nm at an optical resolution of 2 nm.

The area between the NIR probe and a mirror, approximately a 0.3 mm distance (for a total pathway length of 0.6 mm), was filled with the liquid sample and measured. The areas was then purged with distilled water, and dried completely. The background was verified between each sample, and a new reference background spectrum was taken every 30 minutes.

The Raman measurement data was collected in the laboratory of the Department of Chemistry at the University of Bergen using a RamanRxn1 Analyzer with an acquisition time of 600 ms and 8 accumulations.

The Raman probe was inserted directly into the sample containers during measurements. After the measurement, the probe was cleaned with distilled water and dried prior to verifying the background spectrum between each sample. A new reference background spectrum was taken every 30 minutes.

279 samples were provided by CO_2 Technology Center Mongstad, of which, 48 would be used for modeling. These 48 samples consisted of 21 lean samples and 27 rich samples, with lean and rich referring to the concentration of absorbed CO_2. The MIR measurements were made on the samples while they were at approximately 5˚-10˚ C, while the NIR and Raman measurements were made when the samples had reached room temperature.

All response variable data was provided by CO_2 Technology Center Mongstad using unspecified methods. The response variables that were reliably represented in the provided datasets were total alkalinity (TOT_ALK), total inorganic carbon (TIC) and density.

5.3 Preprocessing

The measured data was first checked for outliers before preprocessing. Outlier detection was based on Studentized residuals plotted against leverage, the score plot of components 1 and 2, which were all generated using a preliminary run of PLS, and visual inspection of deviant samples during cross-validation. This was done per response variable, per instrument type, for a total of 9 times. Although several outliers were suspected, it was difficult to determine with certainty at this point, so the samples were marked for observation, and preprocessing commenced. This step would later be revisited when the outliers became more apparent.

The MIR data was smoothed with a Savitzky-Golay filter to reduce the effect from decimal rounding that had been done prior to being delivered to this project. The NIR and Raman data were processed with EMSC, while the data from MIR was processed by MSC. All three datasets were then normalized. Variable selection was run both manually and through the use of the SR, although VIP scores were also referenced. This was done for each response variable for each instrument type, for a total of 9 times. In some cases, SR limits were evaluated multiple times per response.

In all cases, an algorithm was used during variable selection which would run both VIP and SR limits in a given range, at varying step sizes, calculating the RMSECV value for each component amount, for each limit. This was then plotted to produce a graph that could be easily interpreted for determining which variable selection limit gave the lowest RMSECV for a given component. The limit with the lowest RMSECV was then used for the final variable selection to determine which variables would be removed. After this, manual adjustments were made on each set of spectra, where applicable.

5.4 Modeling

Each response variable for each instrument type was then iteratively modeled individually, resulting in 9 final models. The data sets, for each response variable, were then fused into two different LLDF datasets, one containing all three instruments, and one containing just MIR and NIR, for a total of 6 final models. The datasets, for each response variable, were then processed and fused into two different MLDF datasets, one containing all three instruments, and one containing only the MIR and NIR data, for a total of 6 final models. In total, 21 final models were produced, with 18 being showcased in the results section.

Prior to individual model generation, a second run of outlier detection was run, and several suspected outliers were removed. All models generated were done so utilizing PLS-R, specifically the nonlinear iterative partial least squares (NIPALS) algorithm. Cross-validation of the models was performed with MCCV, using an 80/20 ratio, where 80% of the samples would be used as the training set, and 20% would be used as the test set. This was iterated 200 times for each usage of MCCV to give a stable and more representative RMSECV.

The RMSECV for each component would then be plotted, and the lowest RMSECV would be used to consider the number of components that would be used in the final model. This number of components was typically the lowest RMSECV found in the range of 1-20 components. The average predicted residual percentage, as compared to the measured response, would then be plotted for a given component and examined alongside the RMSECV plot, almost always agreeing on the number of components to use for the final model. Examples of the predicted versus measured value plots from within the MCCV loop would be examined visually to look for commonly deviating samples, and inconsistencies in the overall shape of the plotted data in an attempt to strengthen the argument for a final number of components, as well as to examine possible outliers.

After the final number of components to be used was determined, the final model was created, and the predicted versus measured values were plotted together to show how well the predicted values matched the measured values. Due to the relatively low number of samples used, no external validation set was used.

For the LLDF models, the preprocessed spectra were concatenated into one large set of spectra. The spectra were then checked for outliers again, as described above, and then cross-validated and modeled as described for the individual models. The first set of LLDF models were made by concatenating MIR, NIR and Raman spectra, in that order, while the second set was made by concatenating MIR and NIR spectra, in that order.

For the MLDF models, a second run of outlier detection was performed, the detected outliers removed, and the optimal number of components and the score values from the final individual models were used. These score values were concatenated into two sets, as described for the LLDF sets. This score matrix was then used as a predictor matrix, and cross-validated as described for the individual models. Final models were then produced, again, as described for the individual models.

6 Results and Discussion

Figure 28 shows the raw ATR-FTIR spectra as measured by the spectrometer. The dataset provided was rounded to 3 decimal places, creating some rather jagged curves which would require smoothing in order to properly preprocess. Otherwise, noise in the area of wavenumbers ~1800-2400 was noted. High noise in the lower wavenumber area, below about 600, was also noted, and considered for manual removal. A clear distinction between lean samples and rich samples was observed in the areas near 500-600, 830, 870 and 900-1000.

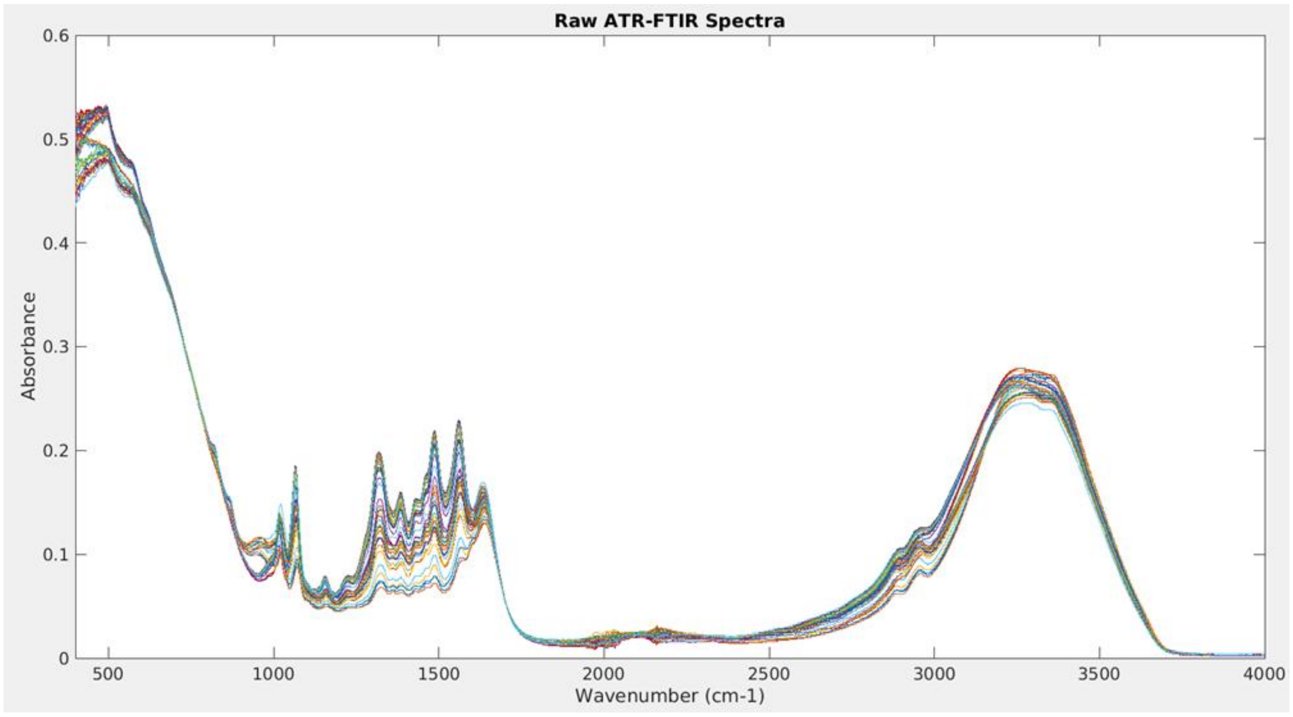

Figure 28: All 48 raw ATR-FTIR spectra.

Figure 29 shows the raw NIR spectra as measured by the spectrometer. No areas of obvious noise were seen. Again, a clear distinction between the lean and rich samples was observed, this time in the areas of ~5000 and ~6550. A possible downward trend in the data with increasing wavenumber was also noted.

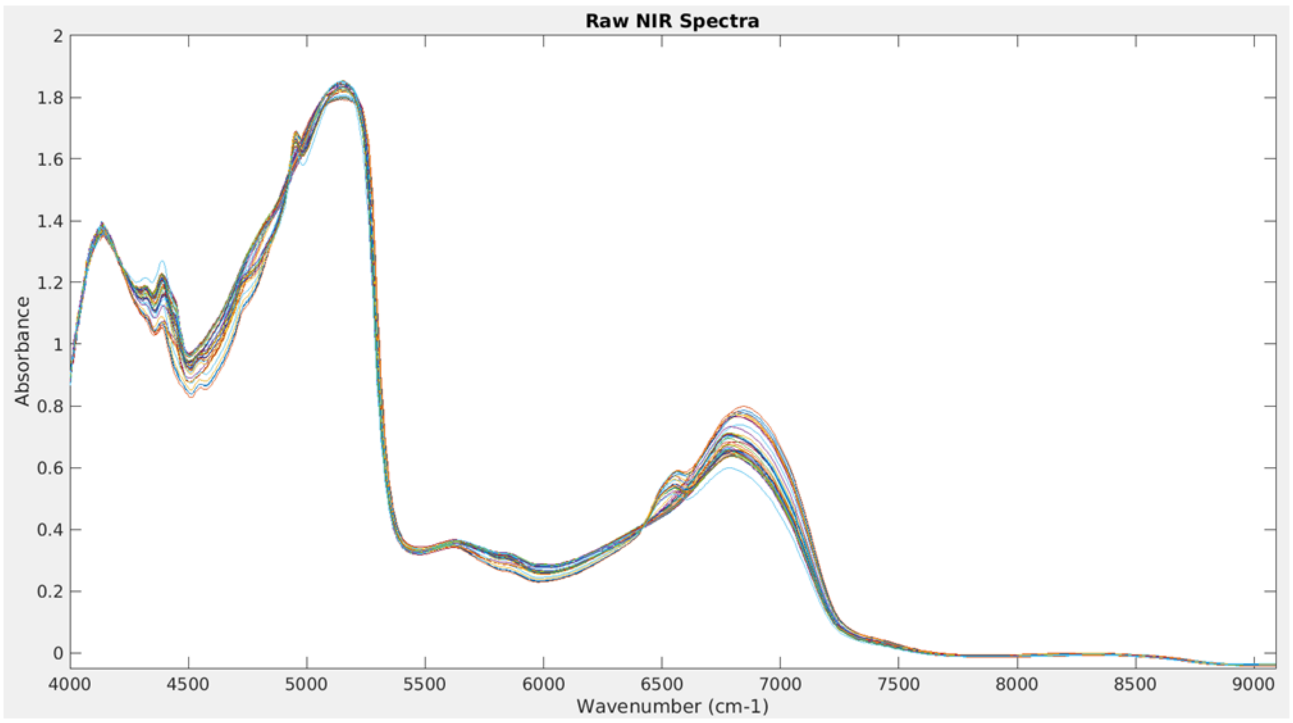

Figure 29: All 48 raw NIR spectra.

Figure 30 shows the raw Raman spectra as measured by the spectrometer. Noise was noted throughout the entire spectrum, and a strong correlation between the visible darkness of the sample and the magnitude of the "bulge" in the intensity was noted, with the lighter samples beginning at the bottom, showing almost no visible bulging, and the darker samples at the top, showing extensive bulging, to the point that the noted peaks become dwarfed in comparison. This is almost certainly due to fluorescence given off by the samples from degradation products within the samples.

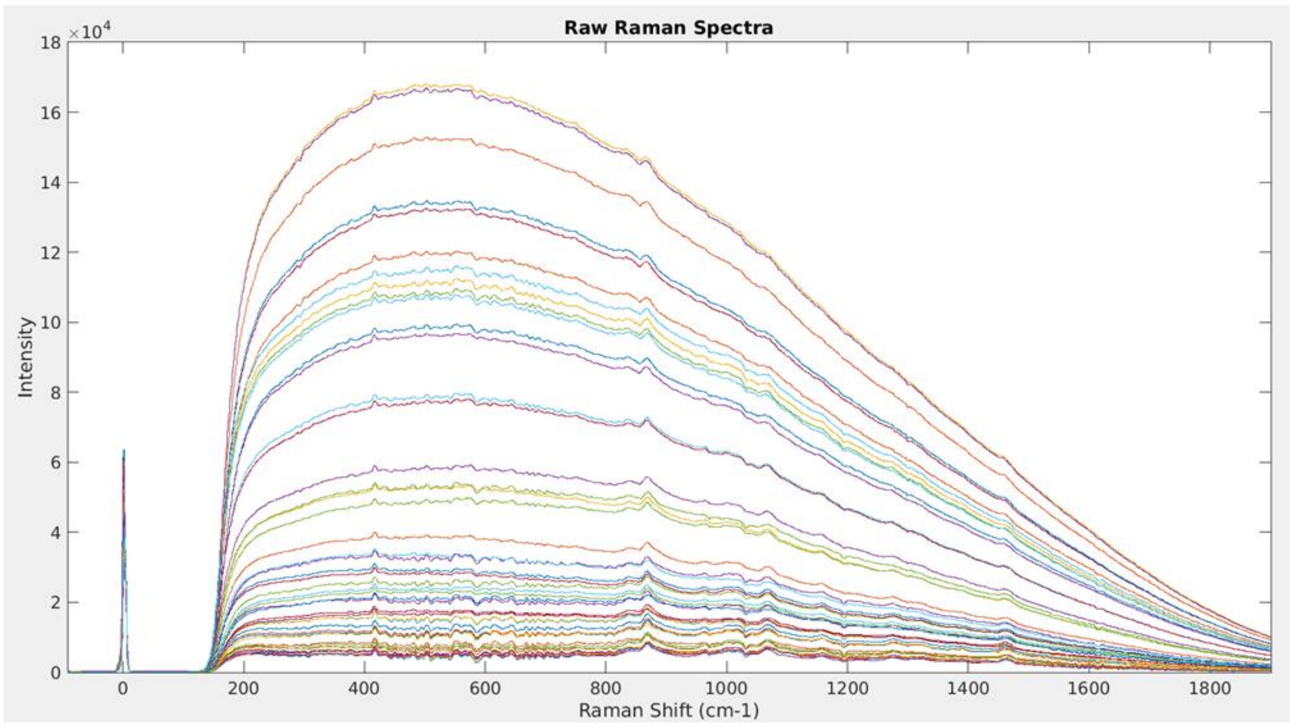

Figure 30: All 48 raw Raman spectra.

Prior to any preprocessing, the samples were checked for obvious outliers by running preliminary PLS on each of the three sets of spectra, and checking score plots as well as the Studentized residual against leverage plots.

6.1 Outlier Detection for ATR-FTIR

6.1.1 TOT_ALK

The RStudent versus leverage plot for TOT_ALK for the ATR-FTIR, shown in figure 31 showed that sample 21098 was possibly an outlier due to its high residual value, while samples 21975 and 22024 were noted as also having somewhat high residual levels. Although the leverages overall were not high, samples 21936 and 21952 were marked for closer inspection.

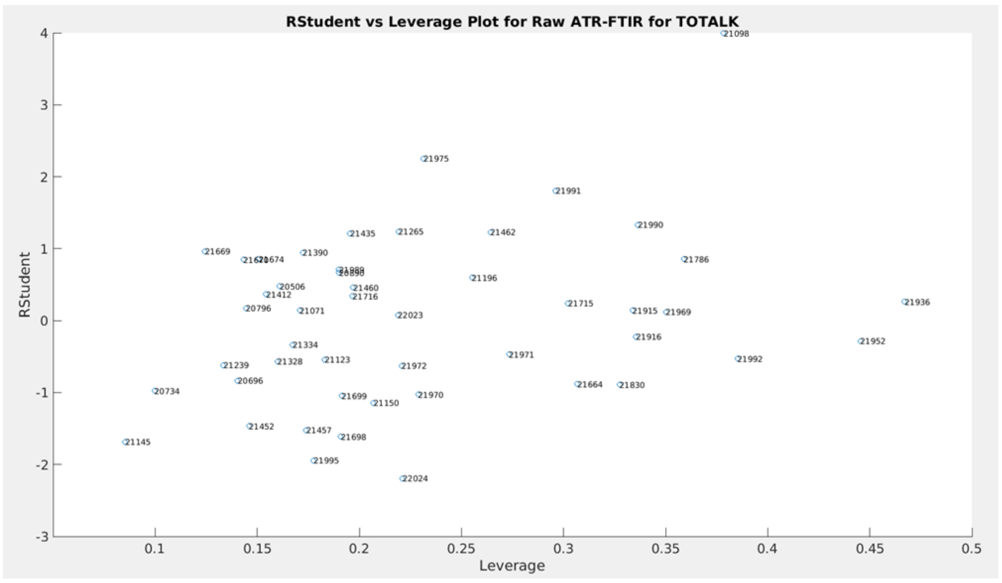

Figure 31: RStudent versus leverage plot for ATR-FTIR when modeling TOT_ALK.

The score plot of component 1 against component 2 can be seen in figure 32. The plot shows a perfect separation between lean samples and rich samples, with the one, singular, exception being sample 21786. Due to the fact that it is well separated into the rich category, and the fact that it does not display high residuals or high leverage, and the fact that this same sample would continue this trend throughout all other testing, it is assumed that this is a likely to be a clerical error, and the sample is almost certain to be a mislabeled rich sample.

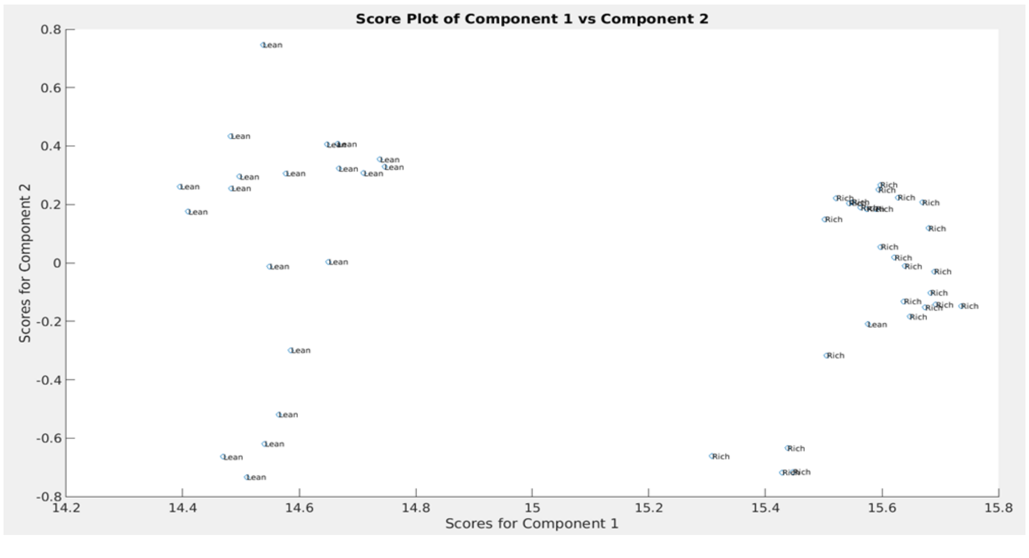

Figure 32: Score plot of components 1 and 2 for ATR-FTIR modeling of TOT_ALK.

The PLS models would be run with and without the suspected outliers, with no significant deviance in prediction errors, and as such, they were not removed from the dataset at this time.

6.1.2 TIC

The Rstudent vs leverage plot for TIC for the ATR-FTIR, shown in figure 33 showed that sample 20890 was possibly an outlier due to its high residual value, while sample 21150, 21980 and 21830 were noted as also having somewhat high residual levels. Although the leverages overall were not high, sample 20890 also contained the highest leverage, in addition to the highest residual, and was marked for closer inspection.

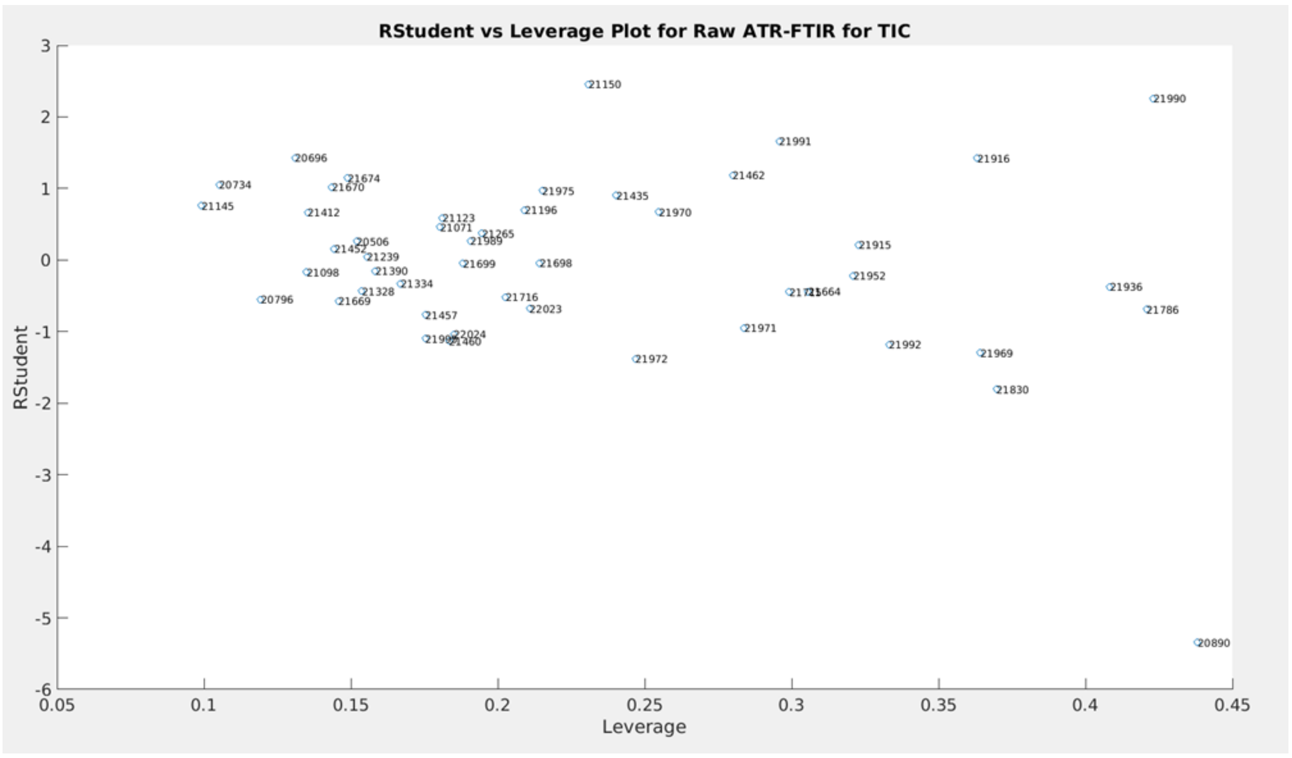

Figure 33: RStudent versus leverage plot for ATR-FTIR when modeling TIC.

The score plot of component 1 against component 2 can be seen in figure 34. The plot shows a perfect separation between lean samples and rich samples, again with the singular exception being sample 21786.

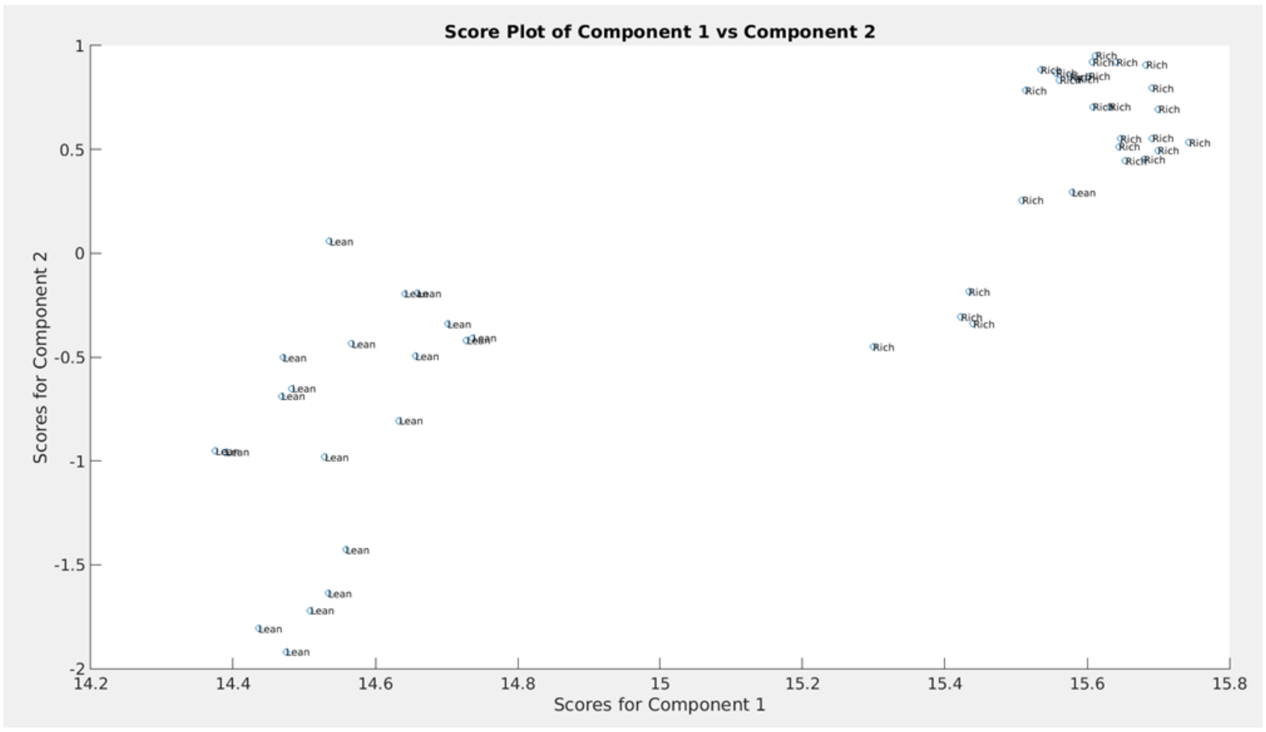

Figure 34: Score plot of components 1 and 2 for ATR-FTIR modeling of TIC.

The PLS models would be run with and without the suspected outliers, with no significant deviance in prediction errors, and as such, they were not removed from the dataset at this time.

6.1.3 Density

The Rstudent vs leverage plot for density for the ATR-FTIR, shown in figure 35 showed that samples 21457, 22023, 21952, 21462 and 20506 were noted as also having somewhat high residual levels. Sample 21990 was marked for closer examination due to its leverage standing out from the rest, and being high.

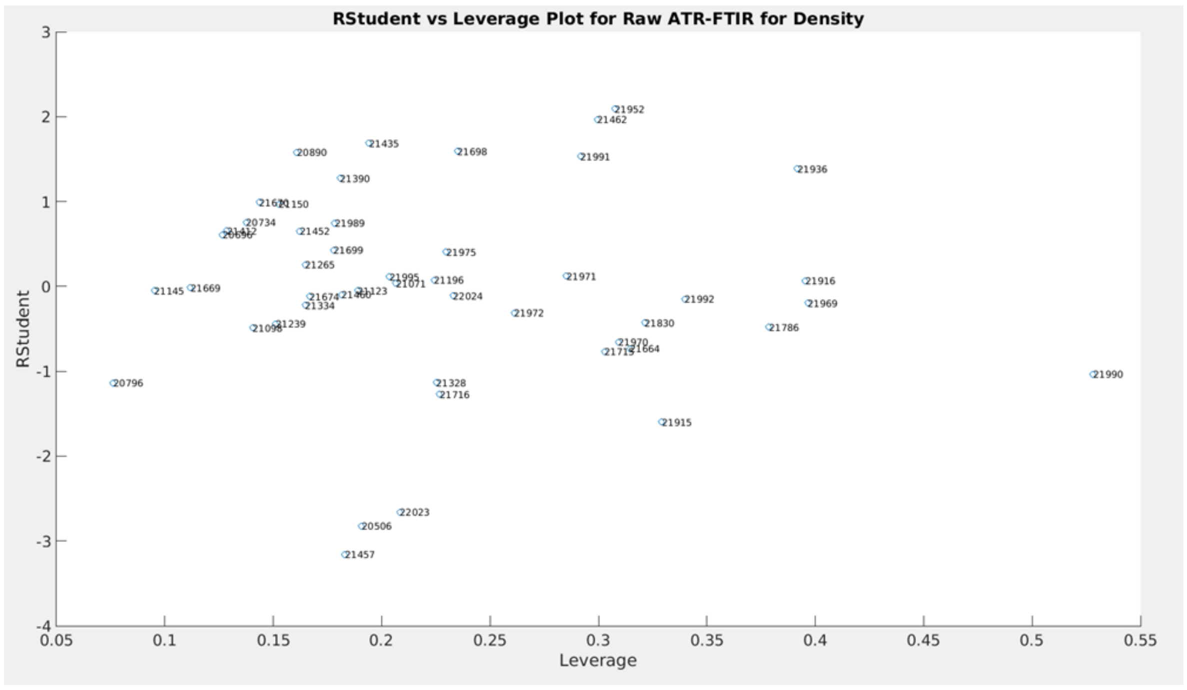

Figure 35: RStudent versus leverage plot for ATR-FTIR when modeling Density.

The score plot of component 1 against component 2 can be seen in figure 36. The plot shows a perfect separation between lean samples and rich samples, again with the singular exception being sample 21786.

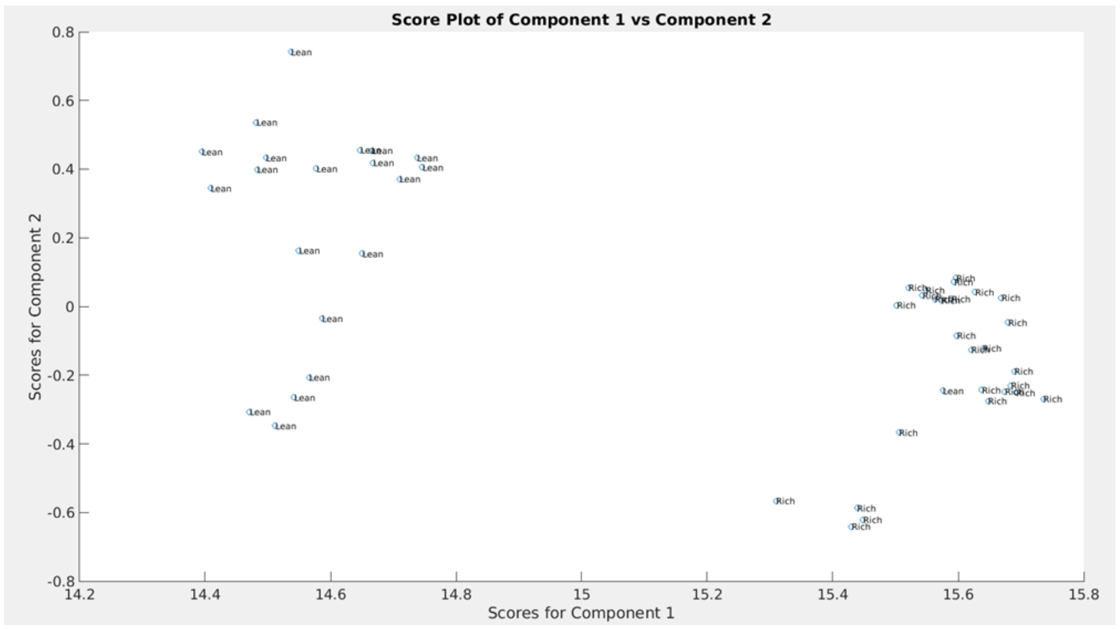

Figure 36: Score plot of components 1 and 2 for ATR-FTIR modeling of Density.

The PLS models would be run with and without the suspected outliers, with no significant deviance in prediction errors, and as such, they were not removed from the dataset at this time.

For the sake of reducing redundancy, outlier detection plotting and descriptions for NIR and Raman are not shown, as they come to the same or similar conclusions, with the same sort of separations and the same sample, 21786, being categorized as rich, rather than lean. The only deviations from this pattern came in Raman, where the rich and lean separations became quite muddled in the score plot. This is not unexpected due to the fact that the signal is dominated in most samples by fluorescence.

6.2 Preprocessing

With outlier detection completed to avoid large potential outliers causing problems during preprocessing, the spectra were preprocessed.

6.2.1 ATR-FTIR

As previously mentioned, decimal place rounding created significant issues within the dataset which would go on to highly affect various transformations. Figure 37 shows this rounding effect on the spectra, prior to removal.

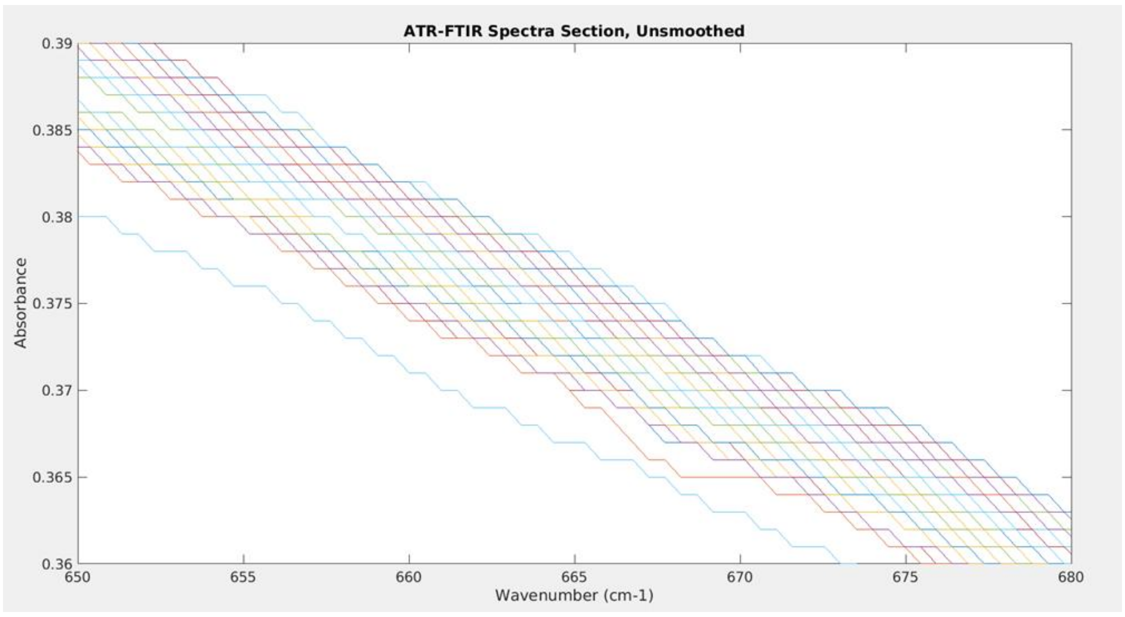

Figure 37: Jaggedness in curve caused by rounding of ATR-FTIR data.

In order to try to approximate the unrounded values, a 0th order Savitsky-Golay with a window size of 15 and 3rd order polynomial was run on the dataset. The results of this smoothing effect are seen in figure 38, and show significant improvement of the plotted data, including clear separation of all samples, although some areas of noise, namely in the wavenumber 1750-2750 area, were still rather rough.

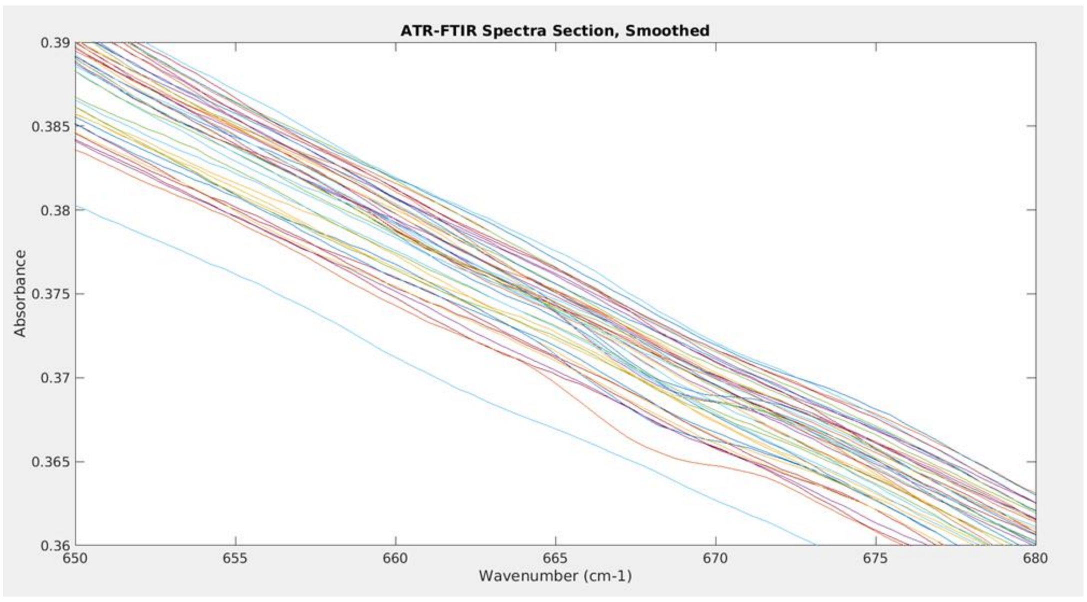

Figure 38: Plot of Savitzky-Golay filter smoothed ATR-FTIR spectral data.

The ATR-FTIR spectra had several areas where there was little additive or multiplicative scattering, so MSC was perhaps not strictly needed. However, it was applied anyway, which can be seen in figure 39, and provided a marginal increase in explained variation. EMSC was deemed unnecessary, as no clear trend was observed, and testing EMSC with 1st, 2nd and 3rd order detrends did not increase model predictions, but rather hurt them.

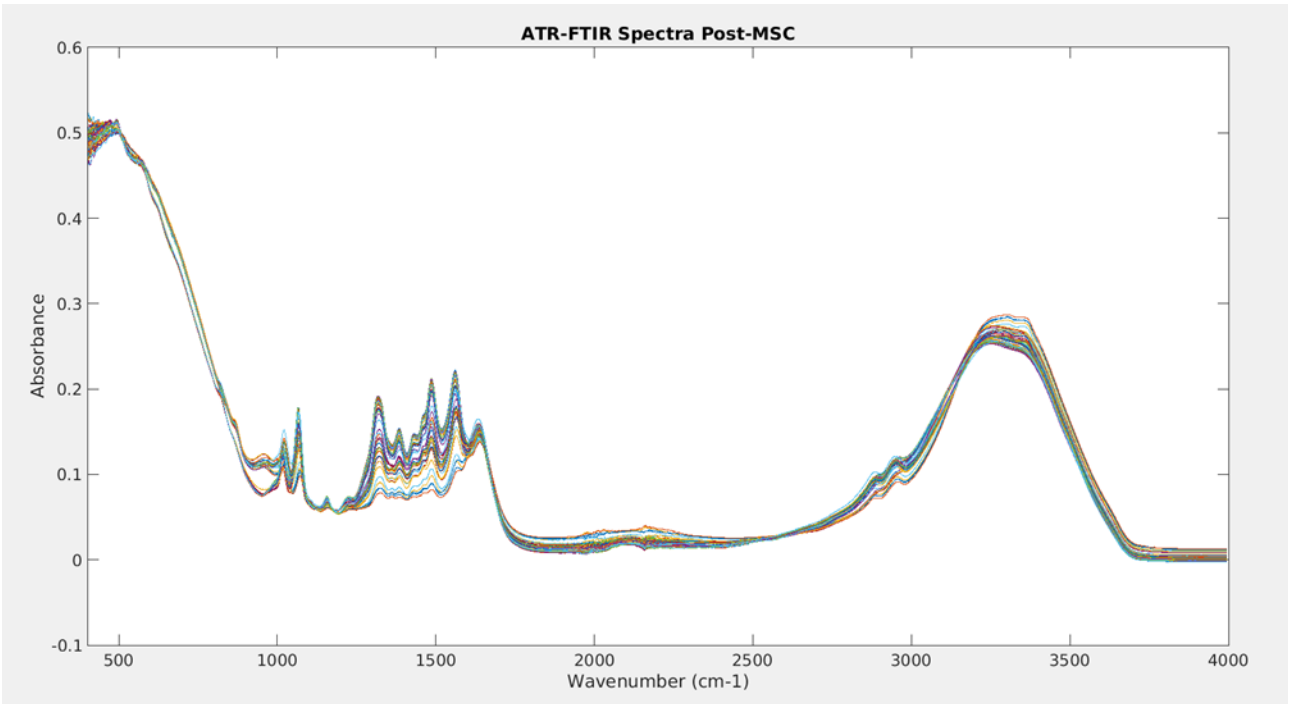

Figure 39: Plot of all ATR-FTIR spectra after MSC processing.

The spectra were then normalized, and made ready for variable selection.

6.2.2 NIR

The NIR spectra were quite noisy, and unlikely to contain useful information, below wavenumber 4240, so this area was removed. This served a second function in removing an area that would strongly influence detrending of the EMSC that was performed. A 2nd order detrend was performed on the manually reduced dataset, which seemed to provide the best balance of reducing the downward trend in the first half of the spectrum without disturbing the second half of the spectrum more than needed, as can be seen in figure 40.

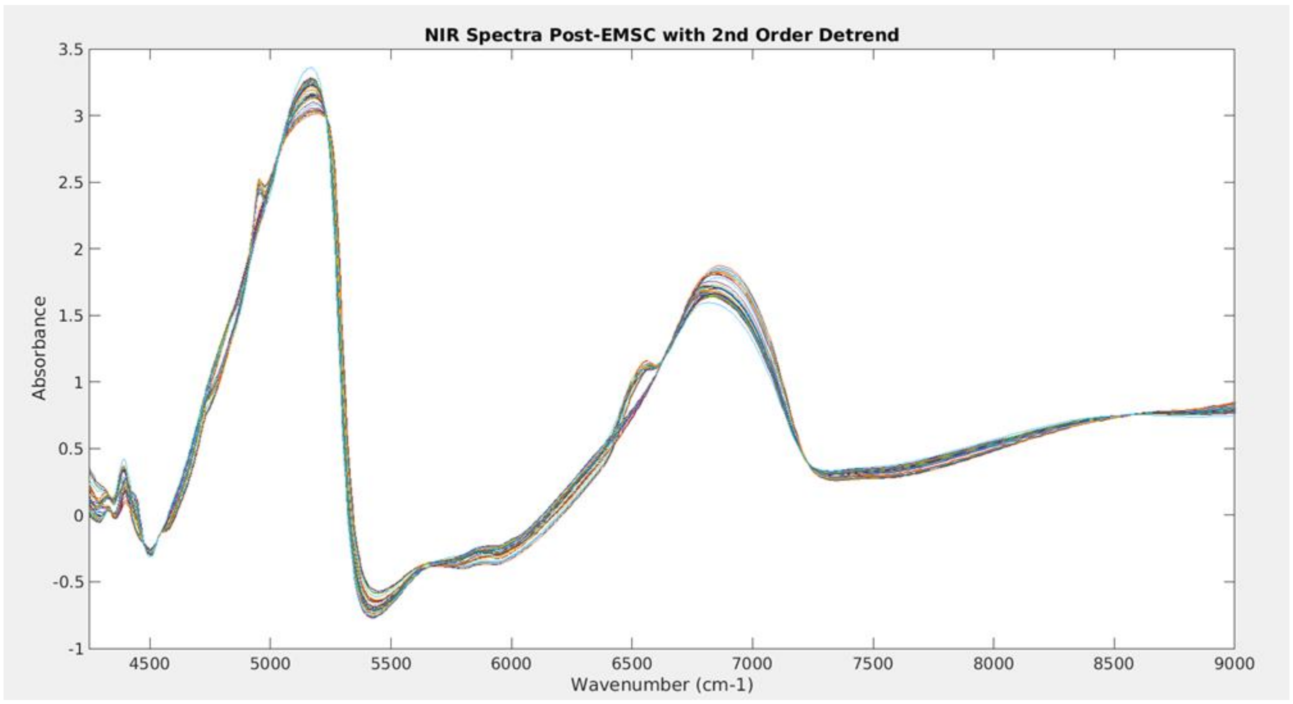

Figure 40: Plot of all NIR spectra after EMSC processing with 2nd order detrend.

The spectra were then normalized, and made ready for variable selection.

6.2.3 Raman

As no information was present in the anti-Stokes and Rayleigh scattering areas of the Raman spectra, these areas were manually removed, along with everything below Raman shift 210. Due to the magnitude of the fluorescent interference, it was difficult to determine an appropriate detrend order, with a 2nd order detrend giving a rather flat overall shape, but also disturbing the data quite a bit. A 4th order detrend did not produce as flat of an overall shape, but gave much more defined peaks. Running cross-validations on the 2nd, 3rd, 4th and 5th order detrended data showed that the 4th order detrend performed best, and by a significant margin over 2nd and 3rd order, so it was used, the results of which can be seen in figure 41.

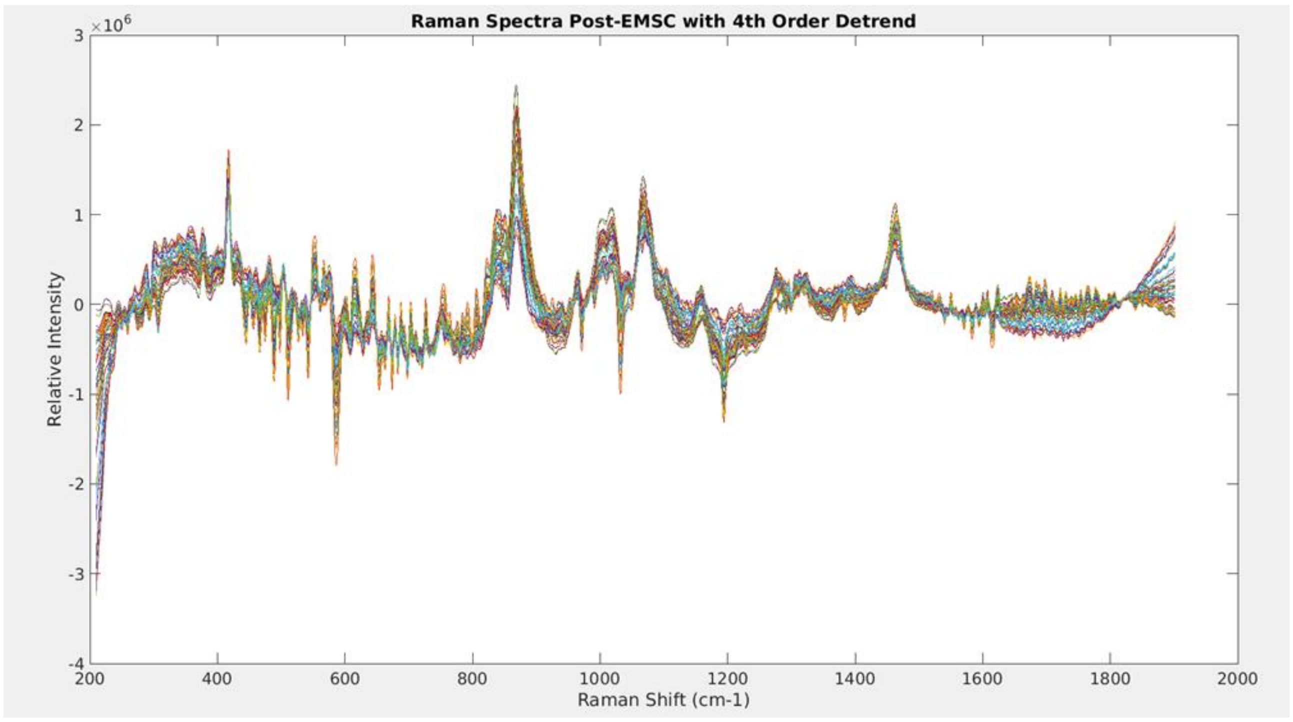

Figure 41: Plot of all Raman spectra after EMSC processing with 4th order detrend.

The spectra were then normalized, and made ready for variable selection.

6.3 Variable Selection

The two methods looked at for variable selection were VIP and SR. Although VIP is a commonly used technique in many areas, SR seemed to be consistently better at determining variables of importance, and was used in the final selection in all cases. As such, the VIP results will not be discussed in this paper. In both cases, an automated system was used to produce and cross-validate models with varying VIP and SR limits in order to determine which limit produced the models with the lowest RMSECV.

6.3.1 ATR-FTIR

For visual inspection purposes, the normalized SR values were plotted in the same space as the normalized ATR-FTIR spectra, which allowed important areas of the spectra to be identified for each response variable.

TOT_ALK

For TOT_ALK, there were several areas of importance, although perhaps the most surprising area was in the slope area after wavenumber 2500, as seen in figure 42.

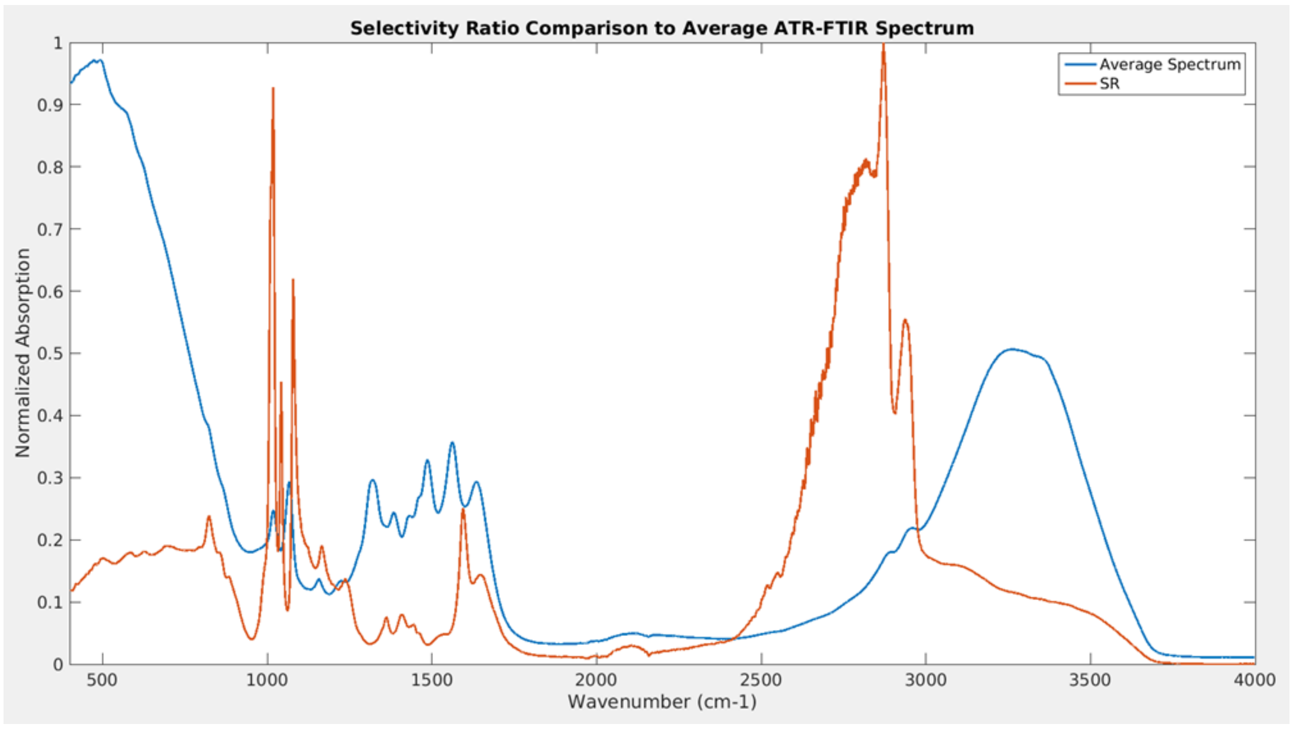

Figure 42: Plot of the average ATR-FTIR spectrum along with the SR score for each corresponding wavenumber for TOT_ALK.

The RMSECV for a given SR limit was then determined and plotted, as seen in figure 43. It shows that the best model is achieved with an SR limit of 0.18, giving an RMSECV of 0.04275. However, some other interesting observations can be made from the rest of the graph. Specifically, a decrease in RMSECV was noted from SR limit 0.5 to ~0.63. This indicates that the variables being eliminated in that area do not hurt the model, but rather help it. There are several peaks in this area, which clouds an easy fix by simply removing all of those areas as well. Trial and error led to identifying certain areas which were helping the model, allowing for manual removal of the areas that were hurting it, reducing the total number of variables from 7453 to 1275.

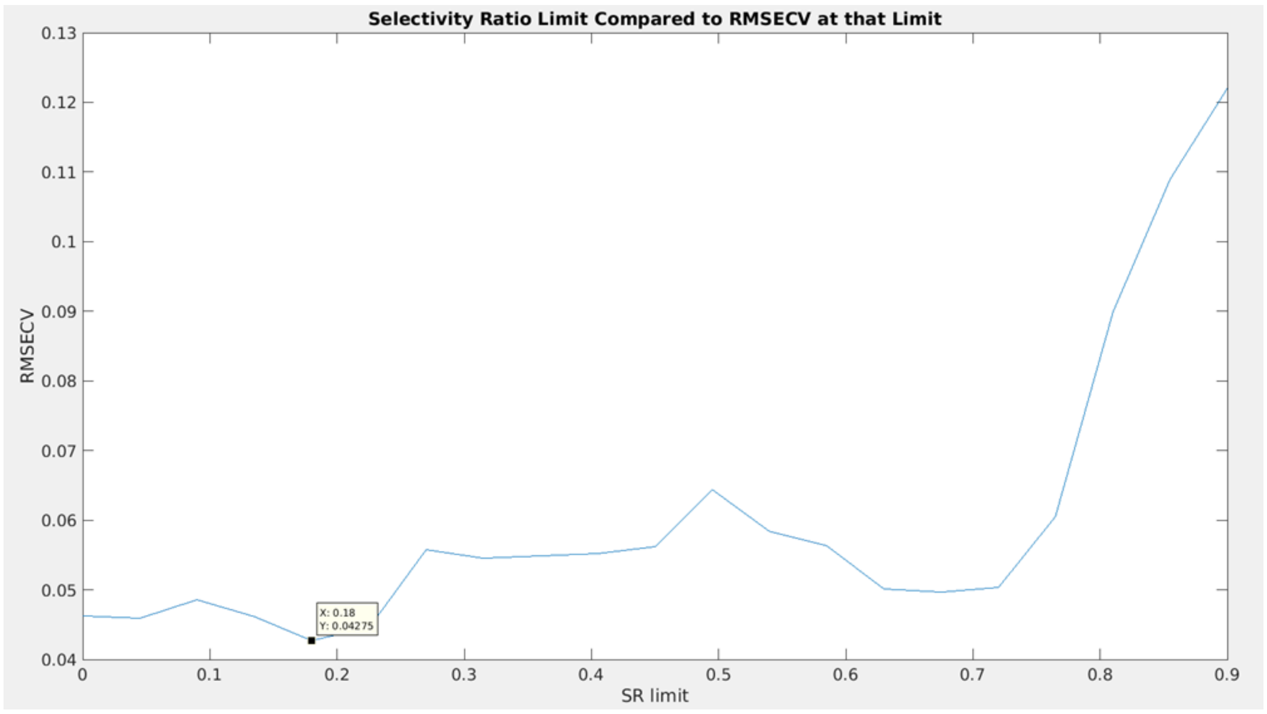

Figure 43: Plot of RMSECV for a given SR limit for ATR-FTIR for TOT_ALK.

TIC

TIC had one primary area of importance, and a couple of rather minor areas, as seen in figure 44.

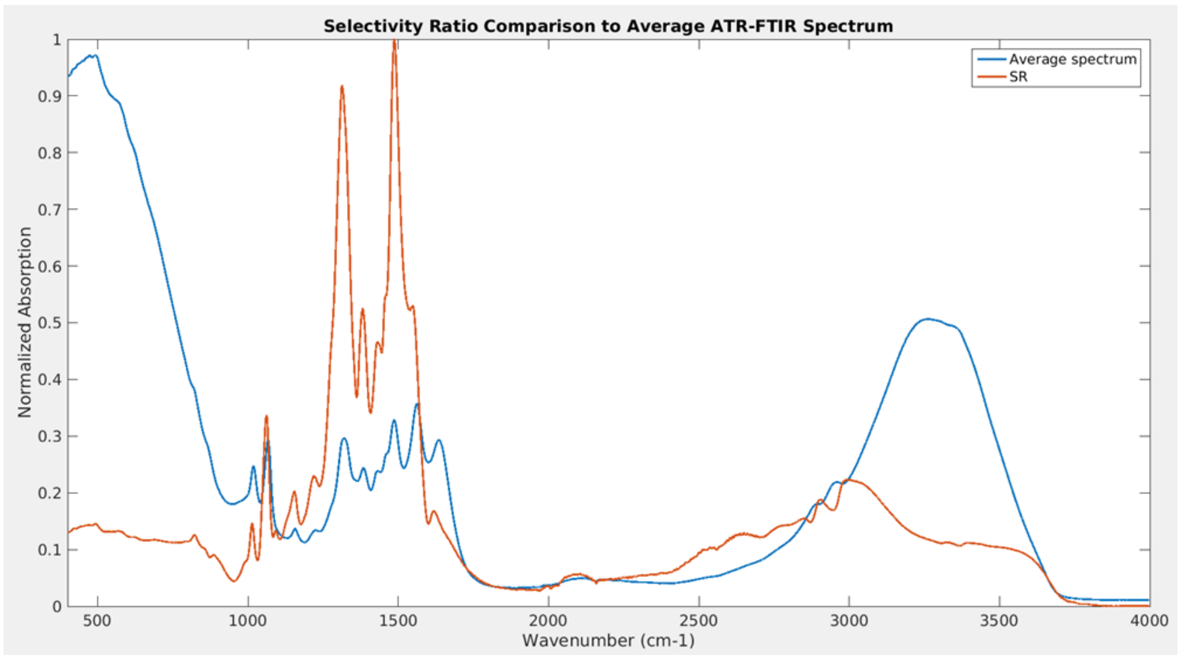

Figure 44: Plot of the average ATR-FTIR spectrum along with the SR score for each corresponding wavenumber for TIC.

The RMSECV for a given SR limit was then determined and plotted, as seen in figure 45. It shows that the best model is achieved with an SR limit of 0.2, giving an RMSECV of 0.02619. Again areas between the limits seem to give better results by cutting certain variables, but in this case, the windows were rather small, and reliable reductions were too difficult to pull off manually. The dataset was therefore stripped exactly as the SR limit of 0.2 suggested, reducing the number of variables from 7453 to 1085.

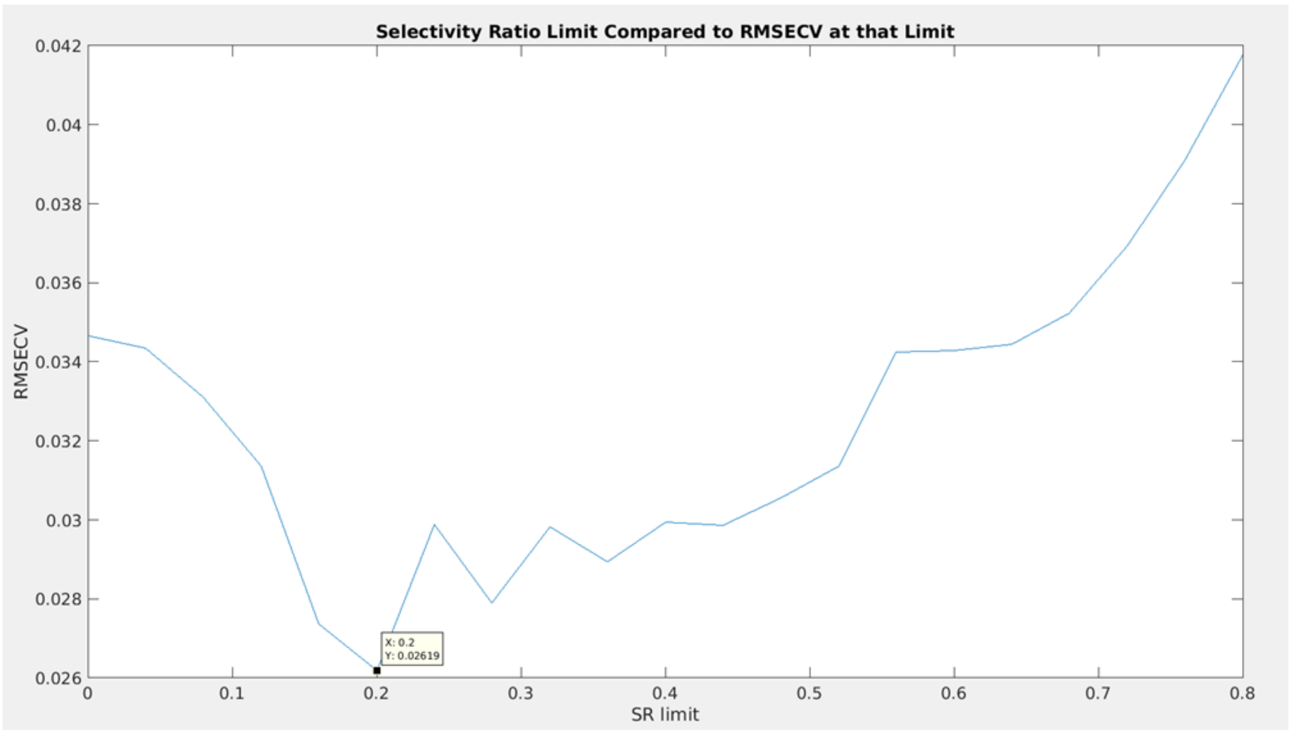

Figure 45: Plot of RMSECV for a given SR limit for ATR-FTIR for TIC.

Density

Density had one primary area of importance, and a couple of minor areas, as seen in figure 46.

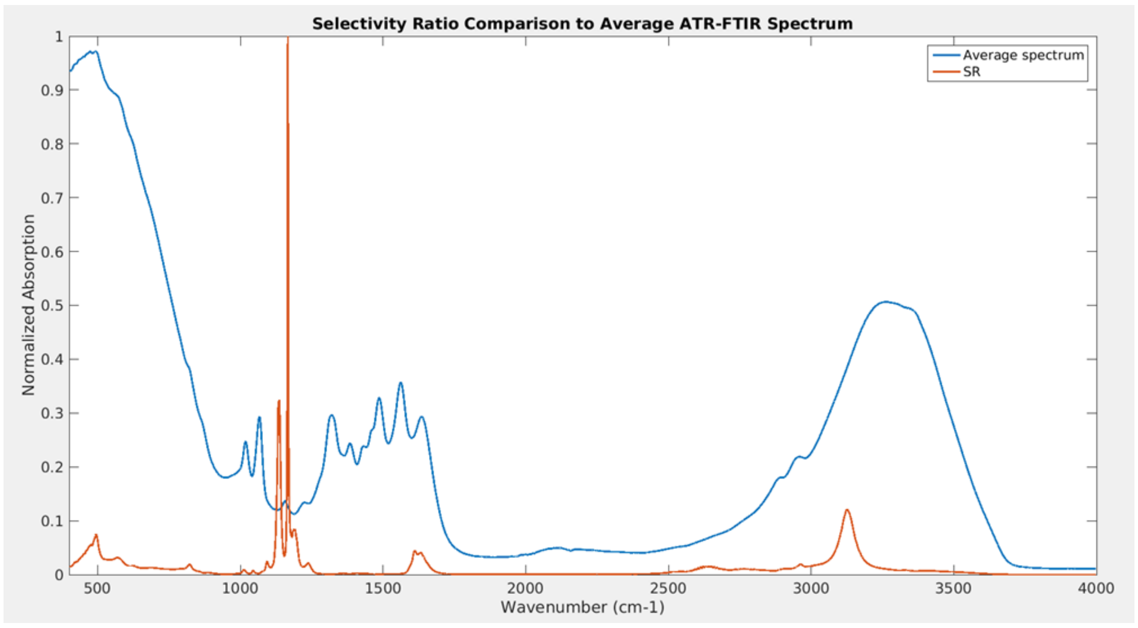

Figure 46: Plot of the average ATR-FTIR spectrum along with the SR score for each corresponding wavenumber for density.

Upon running the battery of SR tests, an oddity occurred. Due to the dominance of one section of the SR plot, it was known that the SR level would need to be low, and it was run from 0.0 to 0.2, as shown in figure 47. Strangely, though, the RMSECV increased whenever variables were removed.

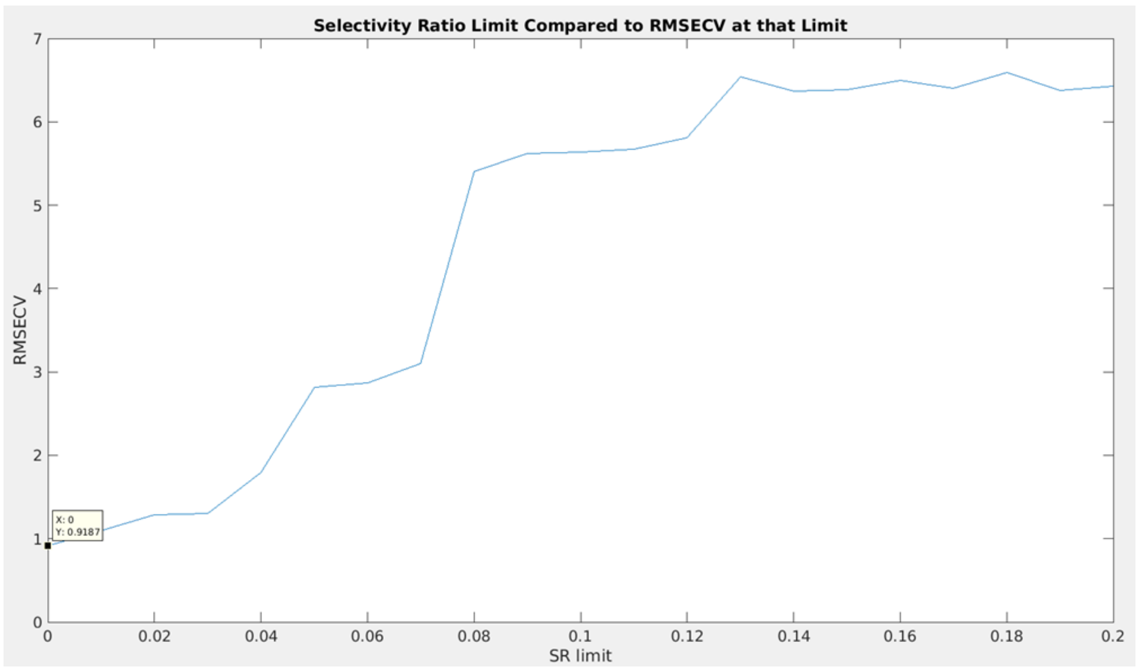

Figure 47: Plot of RMSECV for a given SR limit for ATR-FTIR for density.

It was assumed that the step size between SR limits was hopping over information on the low end of the scale, so it was reduced to step between 0.0 and 0.02, as shown in figure 48. After this, it was rather clear that reducing the amount of variables based on SR was not likely to yield results as far as increased prediction of the model. However, SR limits of 0.002 and 0.006 did not hurt the RMSECV, and were quite close to one another, at 0.8989 and 0.9007, respectively. Stripping the datatset at 0.002 reduced the variables from 7453 to 4336, while stripping at 0.006 reduced it to 3384. In order to reduce the number of variables further, higher RMSECV levels could be tolerated, as 0.9007 is only ~0.086% of the average density value, leaving a lot of room before the prediction error became significant. That said, the models produced after this point were based upon an SR limit of 0.006.

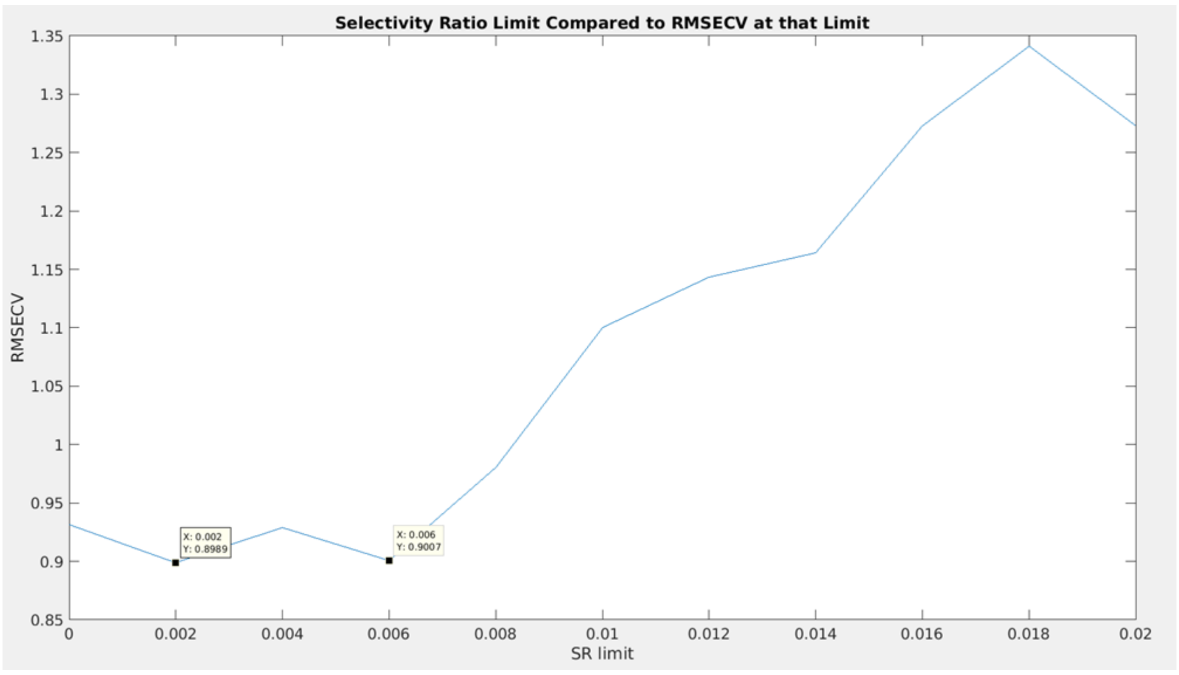

Figure 48: Plot of RMSECV for a given SR limit for ATR-FTIR for density.

6.3.2 NIR

TOT_ALK

For TOT_ALK, there were many areas of importance, although the inflated SR value from 7200 and above seemed out of place, as seen in figure 49. Although the increase from 7200 onwards seems out of place, removing it manually actually decreased overall RMSECV by nearly 10%, in some cases even more, so it was kept.

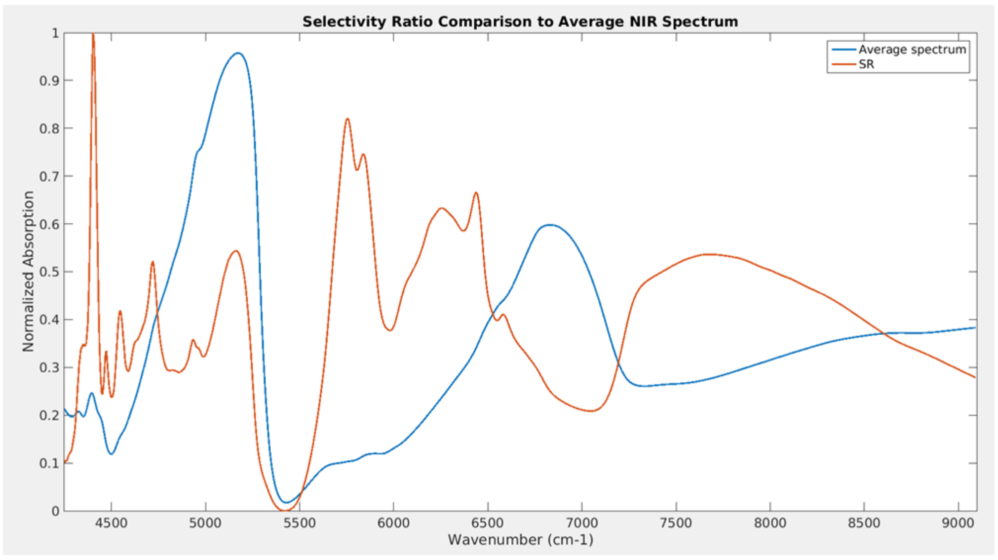

Figure 49: Plot of the average NIR spectrum along with the SR score for each corresponding wavenumber for TOT_ALK.

The RMSECV for a given SR limit was then determined and plotted, as seen in figure 50. It shows that the best model is achieved with an SR limit of 0.16, giving an RMSECV of 0.05097. A significant jump in RMSECV is observed at an SR limit of ~0.72 and above, which corresponds with a rather subtle shoulder around wavelength 4900, which is interesting, as it was also observed as one of the key indicators of the sample being rich or lean during initial visual inspection.

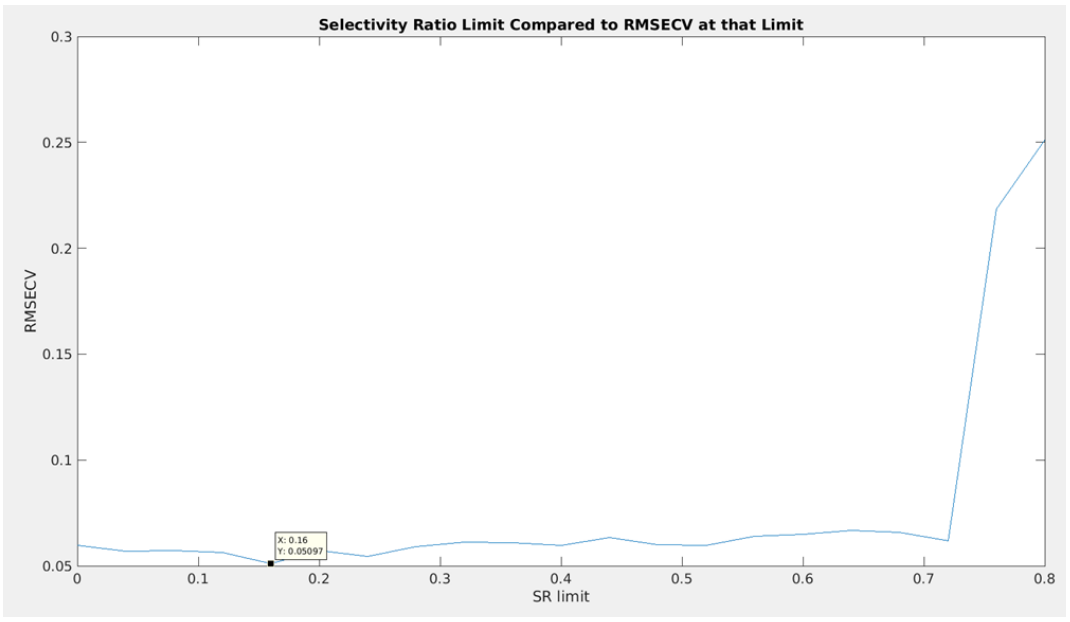

Figure 50: Plot of RMSECV for a given SR limit for NIR for TOT_ALK.

Looking at the up and down nature of the plot until 0.72, it is clear that some subtly important information is being removed during higher limits, and model disturbing information being kept at lower limits. Exhaustive manual searching was unsuccessful at producing consistently better results, however. In the end, the 0.16 SR limit was kept, reducing the amount of variables from 630 to 560.

TIC

For TIC, there were many areas that seemed to be of high importance, as seen in figure 51.

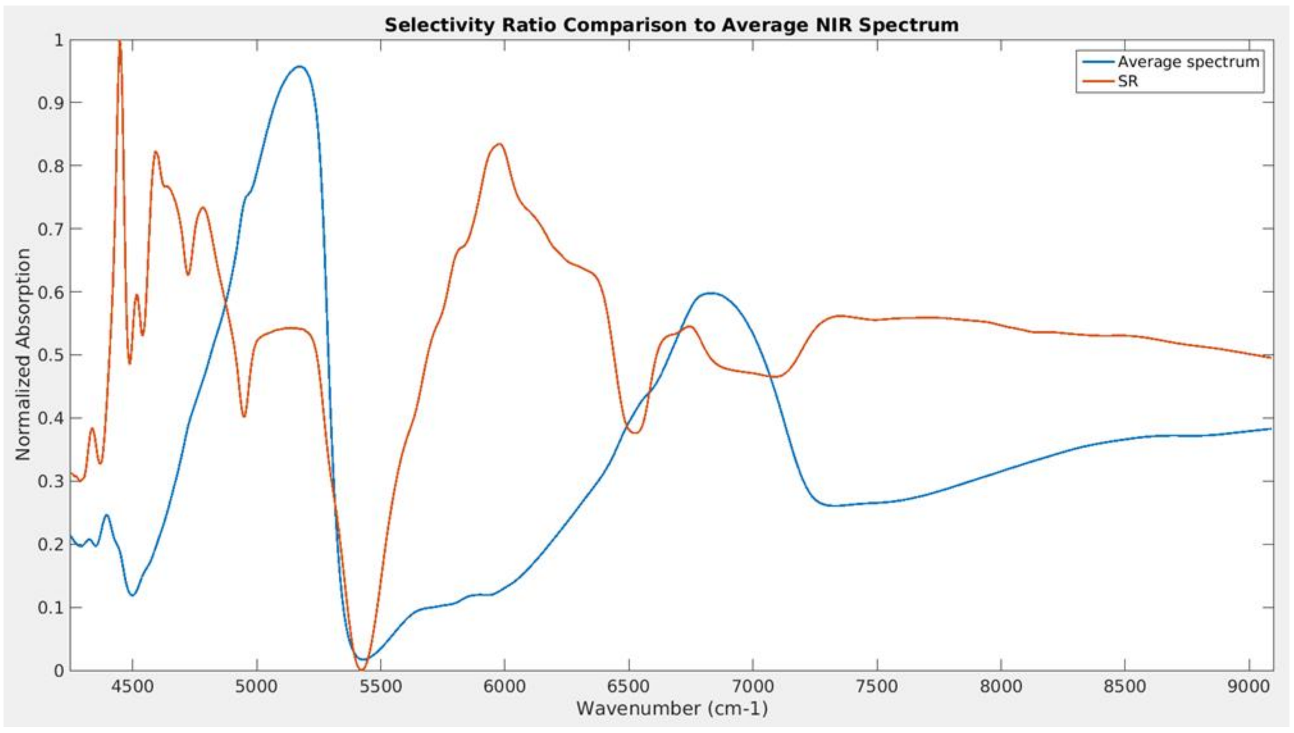

Figure 51: Plot of the average NIR spectrum along with the SR score for each corresponding wavenumber for TIC.

The RMSECV for a given SR limit was then determined and plotted, as seen in figure 52. It shows that the best model is achieved with an SR limit of 0.34, giving an RMSECV of 0.03436. An SR of 0.34 reduced the variables from 630 to 552, while an SR limit of 0.45 reduced it to 496. Although an argument could be made for using a higher SR limit of, say, 0.45, the NIR dataset was already rather small compared to the other datasets, and the variable reduction was not deemed worthy of the loss in predictive ability.

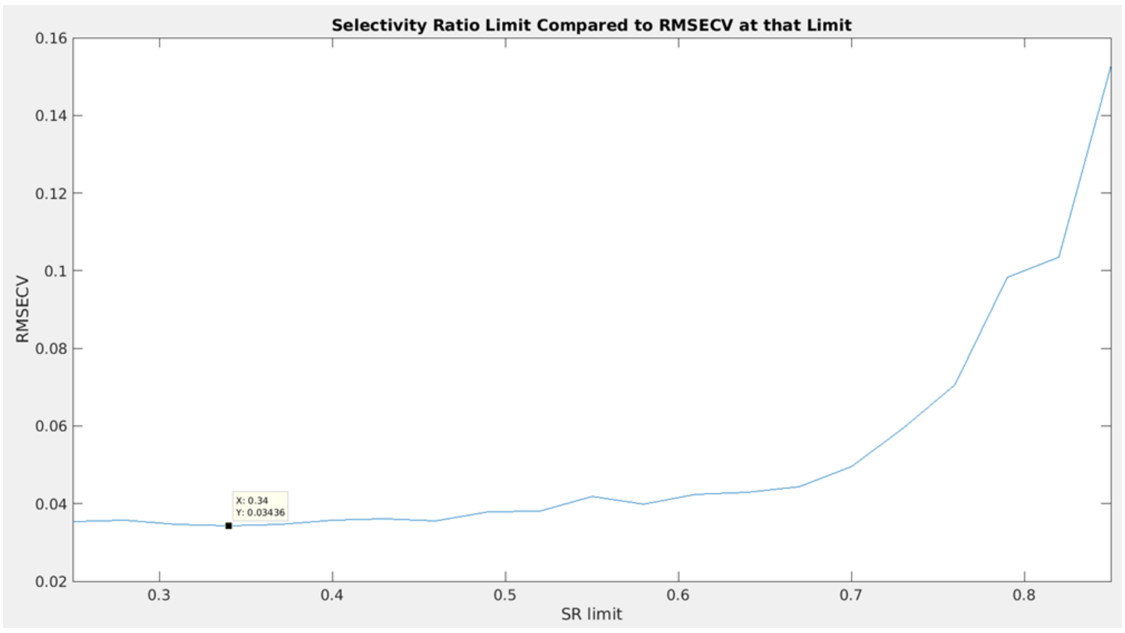

Figure 52: Plot of RMSECV for a given SR limit for NIR for TIC.

Density

For density, as it was with ATR-FTIR, there were small areas of high importance, with a few smaller areas of lower importance, as seen in figure 53.

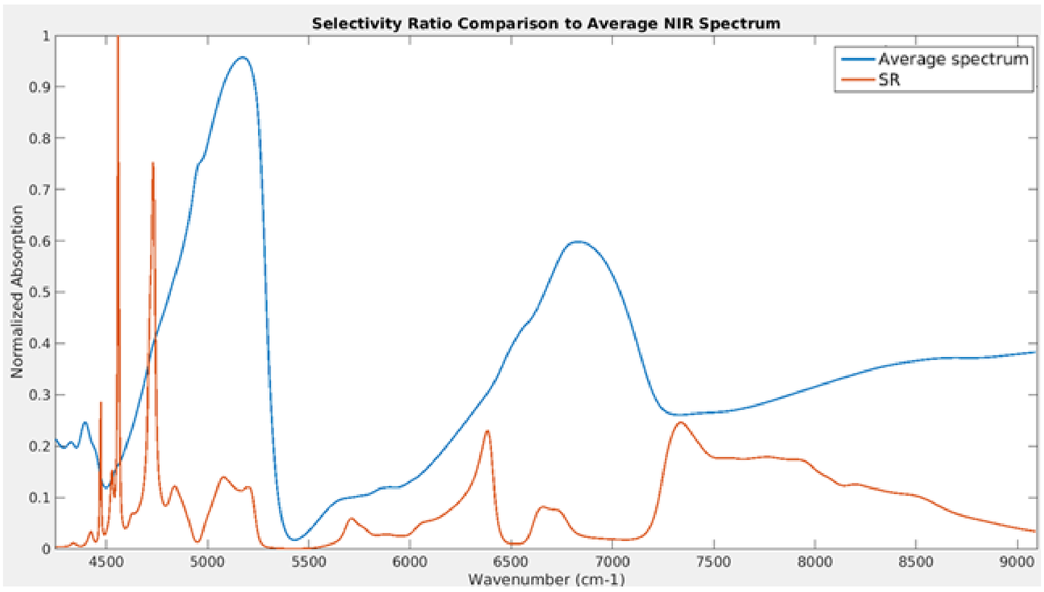

Figure 53: Plot of the average NIR spectrum along with the SR score for each corresponding wavenumber for density.

The RMSECV for a given SR limit was then determined and plotted, as seen in figure 54. It shows that the best model is achieved with an SR limit of 0.015, giving an RMSECV of 0.5293. Again, similar to when modeling ATR-FTIR data, density seems to do best when most of the data is included. Again it should be noted that if the aim was to reduce data processing volume, an upper RMSECV limit should be employed so that an acceptable level can be maintained while also removing as many variables as possible. However, the NIR dataset is so small that not much is saved by increasing the SR limit moderately. Even so, with an SR limit of 0.015, the dataset was reduced the most of the three response types, reducing the variables from 630 to 503.

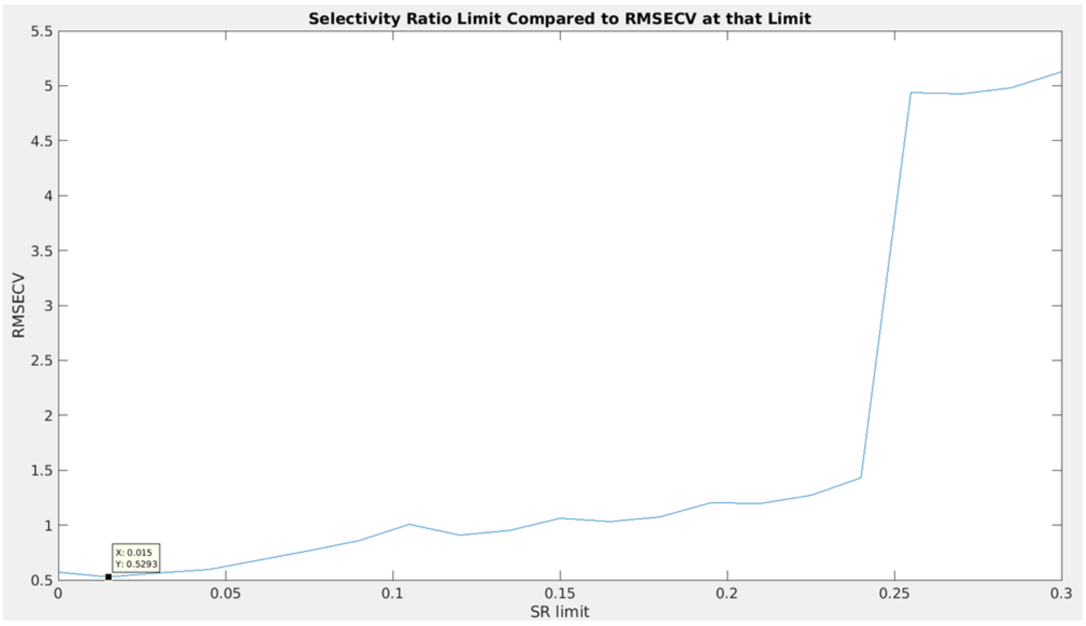

Figure 54: Plot of RMSECV for a given SR limit for NIR for density.

6.3.3 Raman

As a general theme, the Raman spectra were difficult to work with, and variable selection was no exception. The high amounts of noise, and the size of the noise in comparison to the size of the peaks made for much poorer predictions, and attempting to manually select variables was a near impossibility.

TOT_ALK

TOT_ALK produced an SR plot that was incredibly noisy, but not without areas that seemed more or less likely to be informative, as seen in figure 55.

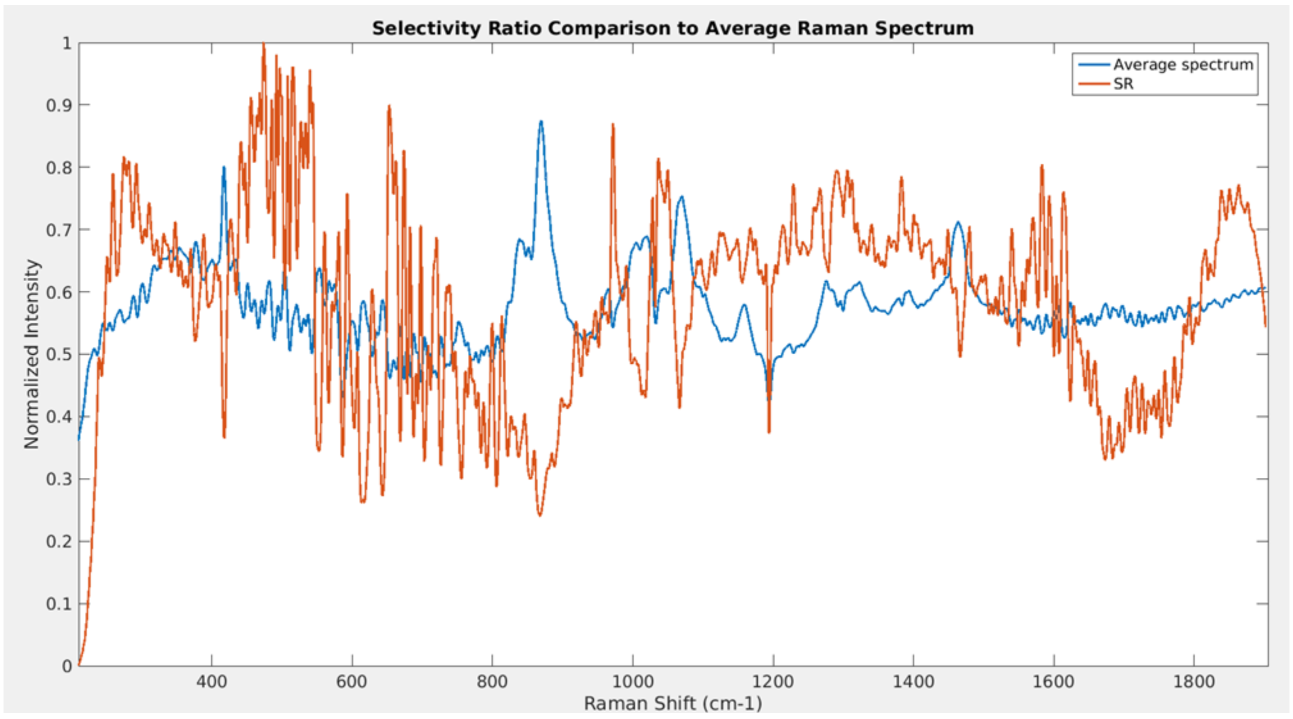

Figure 55: Plot of the average Raman spectrum along with the SR score for each corresponding wavenumber for TOT_ALK.

The RMSECV for a given SR limit was then determined and plotted, as seen in figure 56. It shows that the best model is achieved with an SR limit of 0.3625, giving an RMSECV of 0.1277. This RMSECV is quite high compared with those from ATR-FTIR and NIR, but in comparison to the average response value of 4.5835, it is only a 2.79% residual, which, although not amazing, is not bad considering the amount of noise and issues with the spectra. Multiple attempts were made to eliminate additional areas of the spectra manually, but none were ever successful in bringing the average RMSECV up. Using the SR limit of 0.3625, the number of variables were reduced from 5640 to 5117. Ideally this would be reduced further, as this is still a large number of variables, but it was simply too difficult to safely remove variables due to all of the noise. To note, doubling the RMSCEV allowed reduced the number of variables to just 1978. However, this brings the residual to around 5.5%, which is rather high.

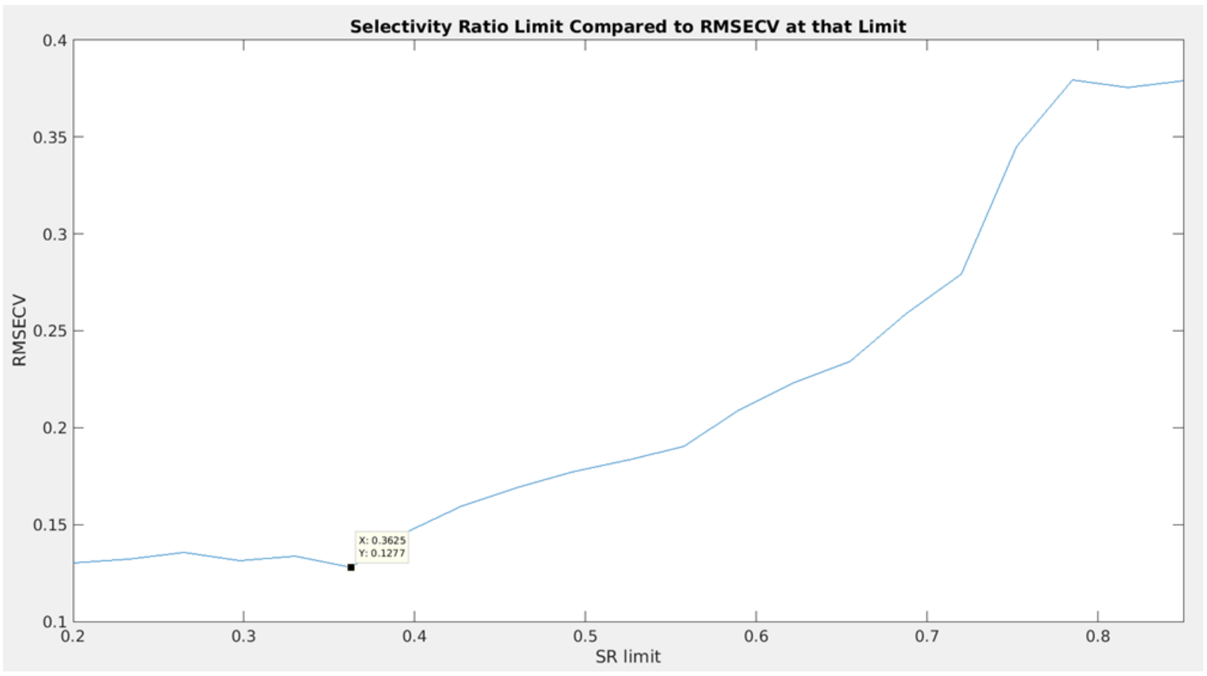

Figure 56: Plot of RMSECV for a given SR limit for Raman for TOT_ALK.

TIC

TIC produced an SR plot that was also quite noisy, but not nearly as bad as for TOT_ALK. Relatively clear peaks of important areas are possible to make out, as seen in figure 57.

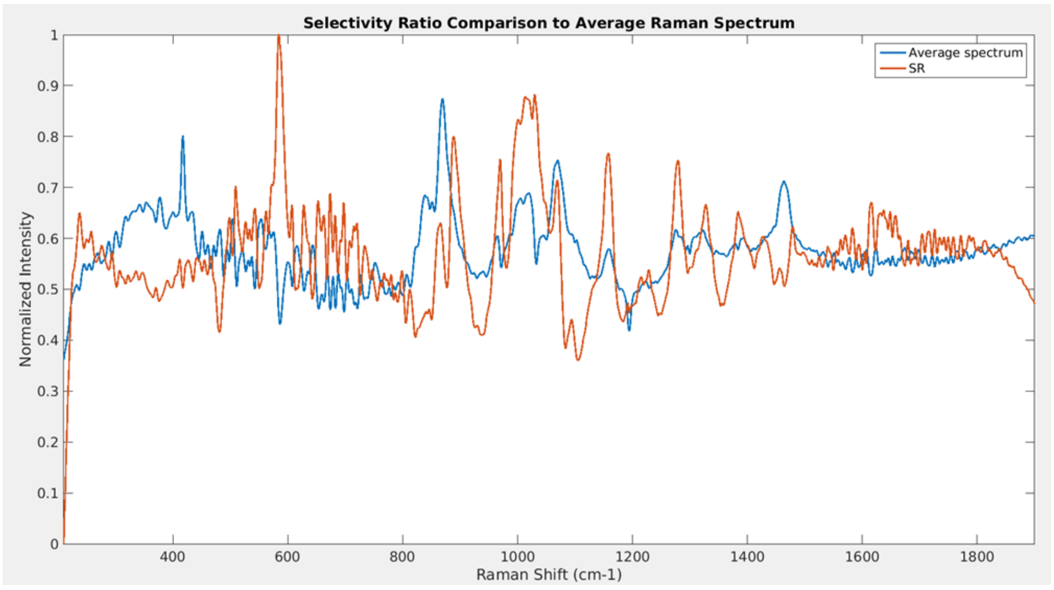

Figure 57: Plot of the average Raman spectrum along with the SR score for each corresponding wavenumber for TIC.

The RMSECV for a given SR limit was then determined and plotted, as seen in figure 58. It shows that the best model is achieved with an SR limit of 0.515, giving an RMSECV of 0.2974. In comparison to the average response of 1.6938, this is a residual of 17.89%, which is a very large amount. However, the areas from ~500-570, ~610-810 and 1500 onwards were identified as being mostly noise, and were manually removed, in addition to the removals from an SR limit of 0.515. Removing 500-570 led to no significant change in the RMSECV, while both the 610-810 block and 1500 onward block lead ~7 and ~11% increases in RMSECV, respectively. The 500-570 block was removed, as was everything after 1800, as no significant change was caused by these removals. It should be noted that a second pass of the SR limit tester was done on the stripped data, and even then no additional SR based removals gave better predictions. The combination of SR based and manual removals reduced the total variables from 5640 to 3694.

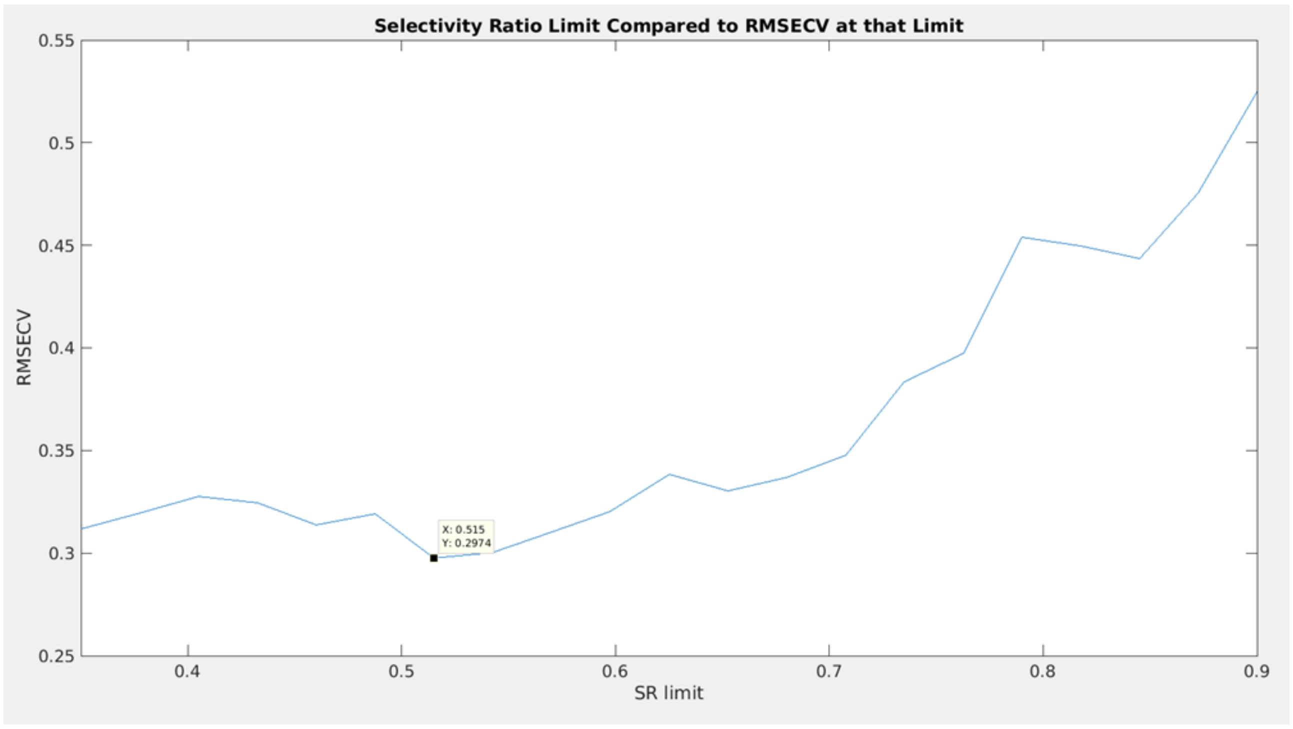

Figure 58: Plot of RMSECV for a given SR limit for Raman for TIC.

Density

Density produced an SR plot that was likely the nicest of all for Raman, just as it was for ATR-FTIR and NIR. Although still relatively noisy in the amount of peaks, the baseline is way down, and the peaks are much taller and sharper, which allowed for better SR selection, as seen in figure 59.

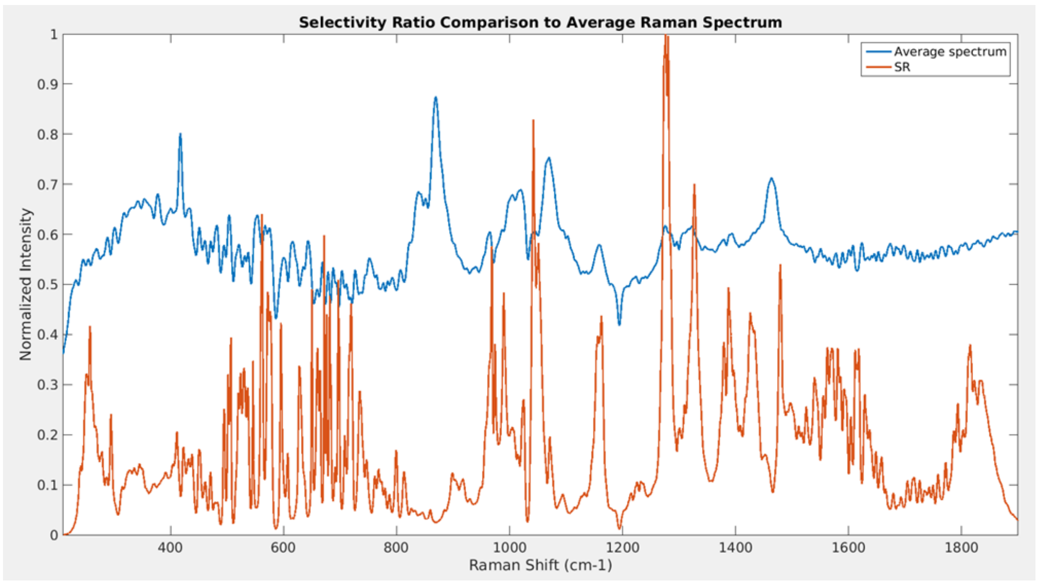

Figure 59: Plot of the average Raman spectrum along with the SR score for each corresponding wavenumber for density.

The RMSECV for a given SR limit was then determined and plotted, as seen in figure 60. It shows that the best model is achieved with an SR limit of 0.075, giving an RMSECV of 13.97, which compared to the average response value gives it a residual of ~1.33%, which is surprisingly good considering the rather chaotic structure of both the spectra and the SR plot.

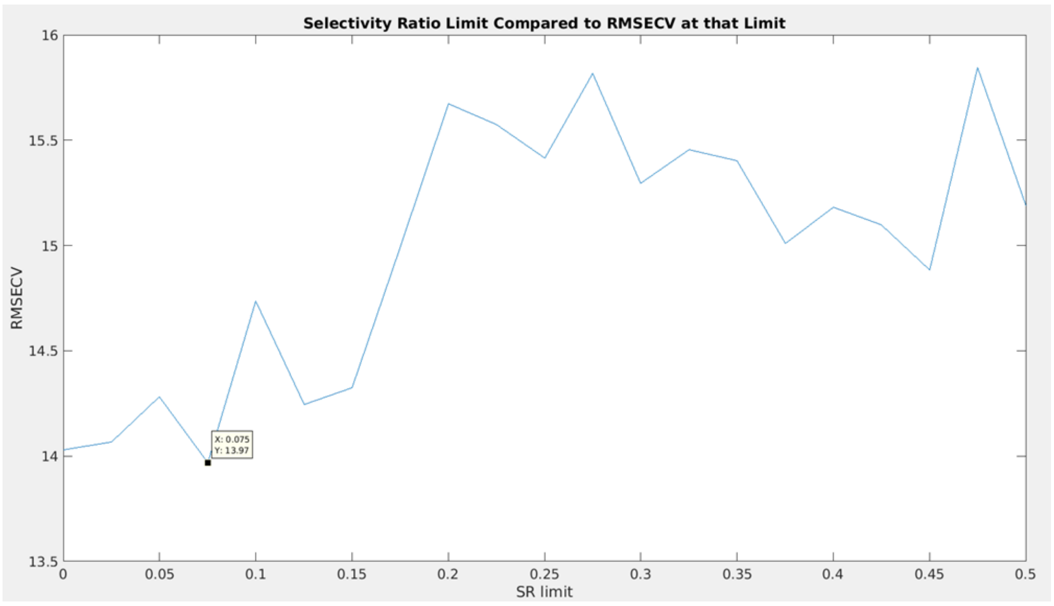

Figure 60: Plot of RMSECV for a given SR limit for Raman for density.

It is clear from the plot that the range of SR limits from ~0.27-0.45 remove something that is hurting the model. This spans a lot of peaks, and bulk removing them did not improve the model significantly. That said, bulk removing all areas below 900 and above 1500, while rather heavy handed, and then running the SR limit tester again produced similar overall results, as shown in figure 61.

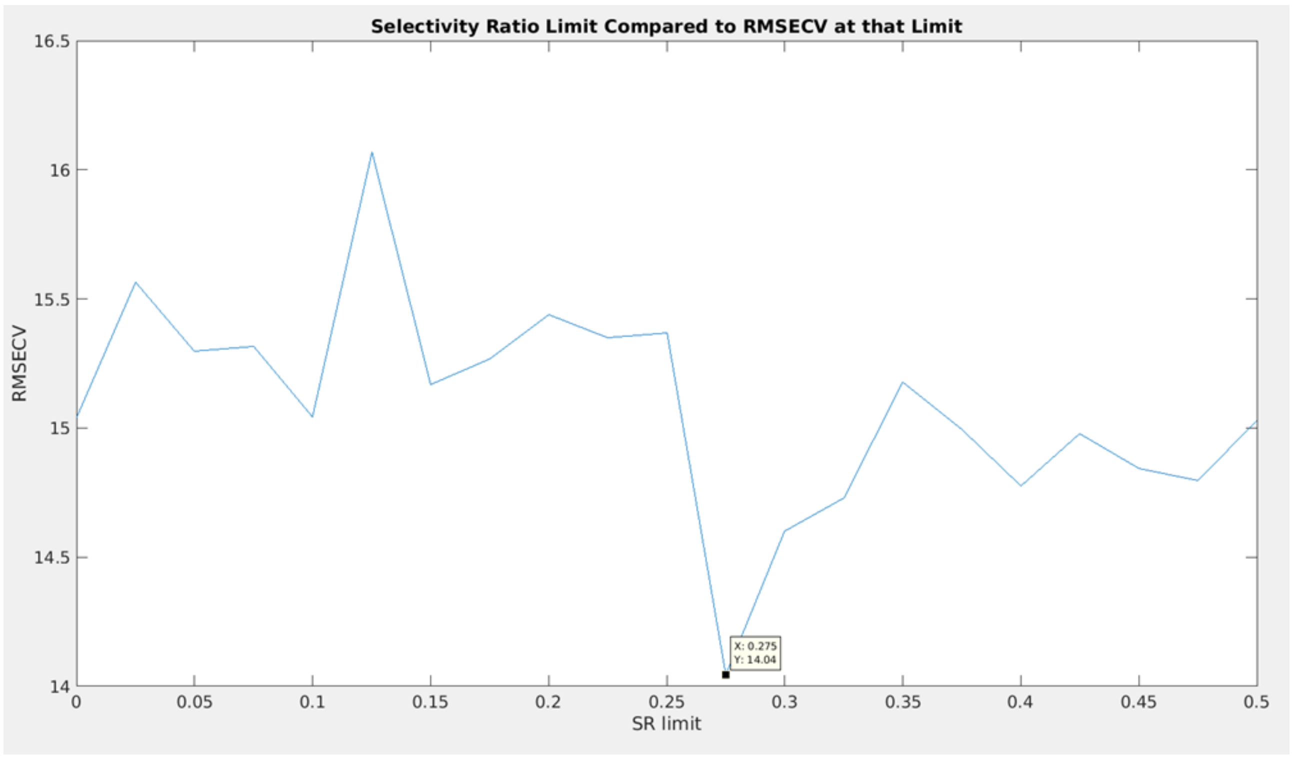

Figure 61: Plot of RMSECV for a given SR limit for Raman for density.

With a new SR limit of 0.275, the RMSECV is increased to 14.04, which is not significantly different from the previous RMSECV, but allows for a much larger reduction in overall variables, taking the total from 5640 to just 490, which is by far the best variable reduction of the set.

6.4 Individual Modeling

To get a baseline of model performance, each instrument's spectra were modeled individually for each of the three responses. This gives some frame of reference that the fused models can be compared to later in the process.

6.4.1 ATR-FTIR

The individual models for ATR-FTIR performed well in all instances, and would likely be the best choice for instruments if only a single instrument type was to be employed.

TOT_ALK

The cross-validation results for TOT_ALK were promising, but upon a second run of outlier detection, samples 21098 and 21071 were deemed to be outliers. Once removed, the RMSEP of the final model, as well as the RMSECV of the cross-validation improved by a significant amount. In the end, a final cross-validation was run, showing a minimum RMSECV at 8 components, for a value of 0.03343, as seen in figure 62, which is a residual of 0.73% compared to the average size of the total responses.

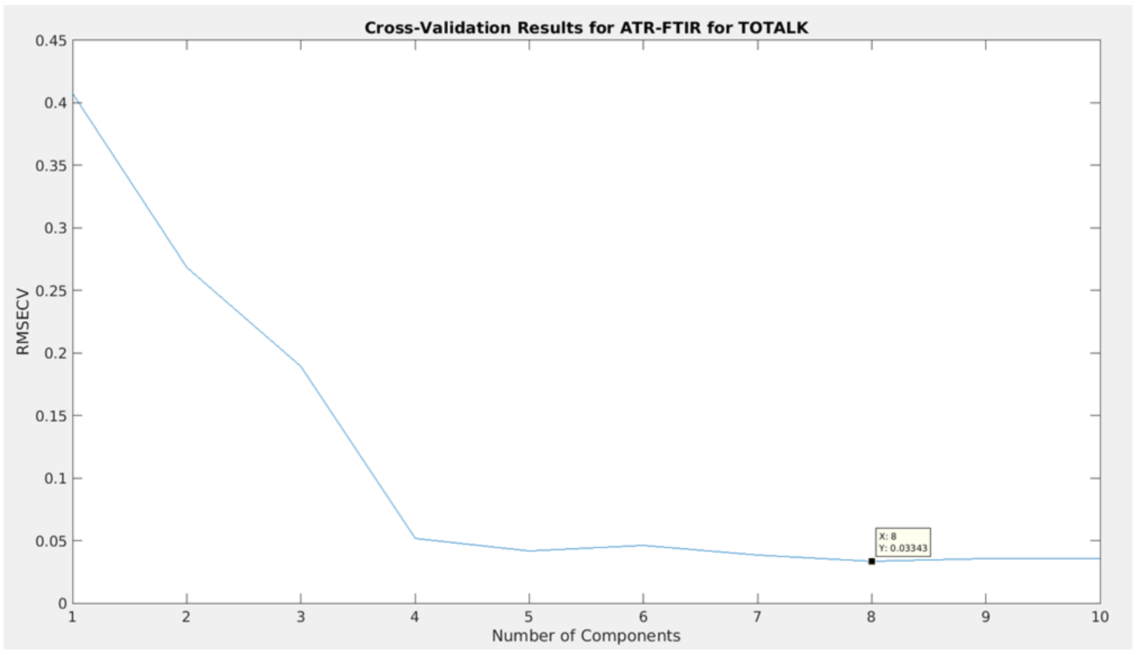

Figure 62: RMSECV versus components used for ATR-FTIR for TOT_ALK.

Although the suggested number of components is 8, and it is 8 that would be used from this point on, the difference in RMSECV between 4 components and 8 components is relatively minor, and should be considered to reduce calculation times if needed. In the average prediction residual percentage (that is, the average magnitude of the difference between the predicted response and the measured response, divided by the measured response.) against number of components plot this is reflected as well, as seen in figure 63.

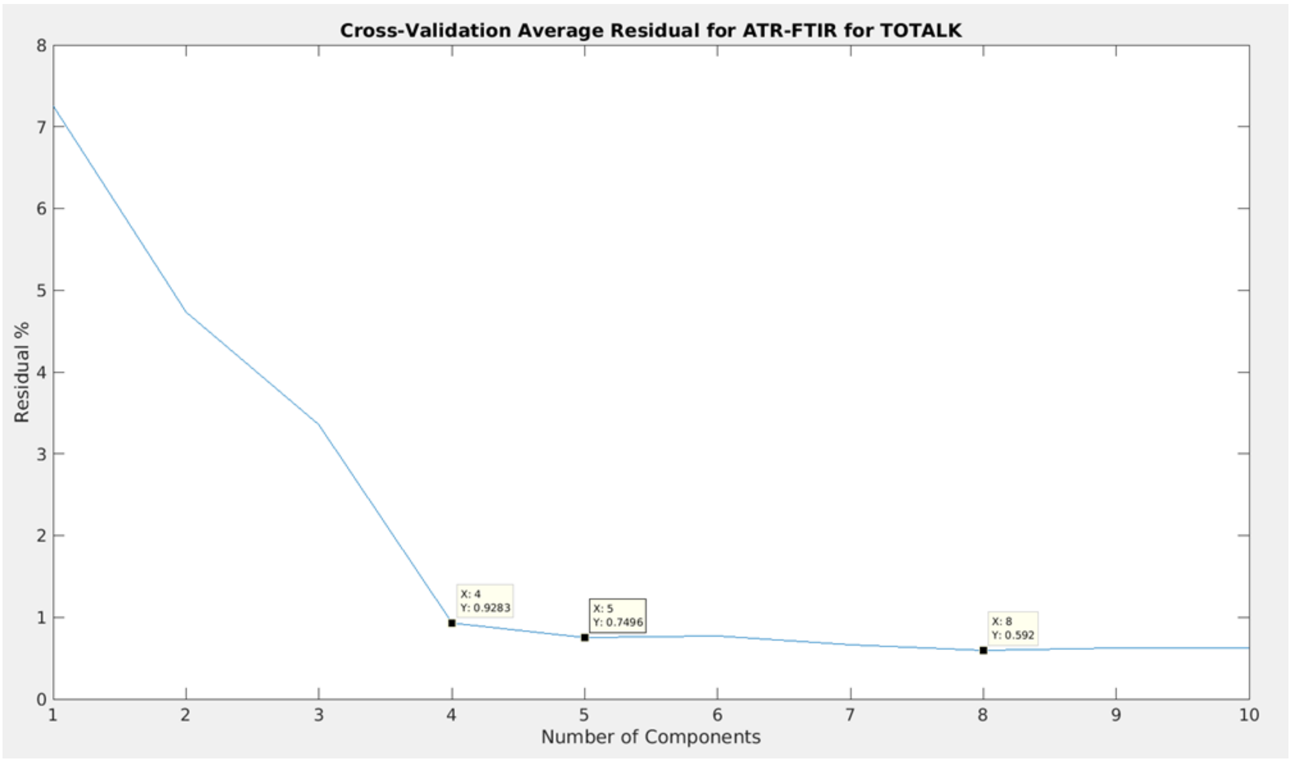

Figure 63: Plot of average residual percentage for a given amount of components for ATR-FTIR for TOT_ALK.

As seen, 8 components gives the lowest prediction residual percentage at 0.592%, while 4 components still gives a more than reasonable 0.9283%. As a compromise, 5 components offers significant computational savings, while still maintaining a 0.7496% residual. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 64.

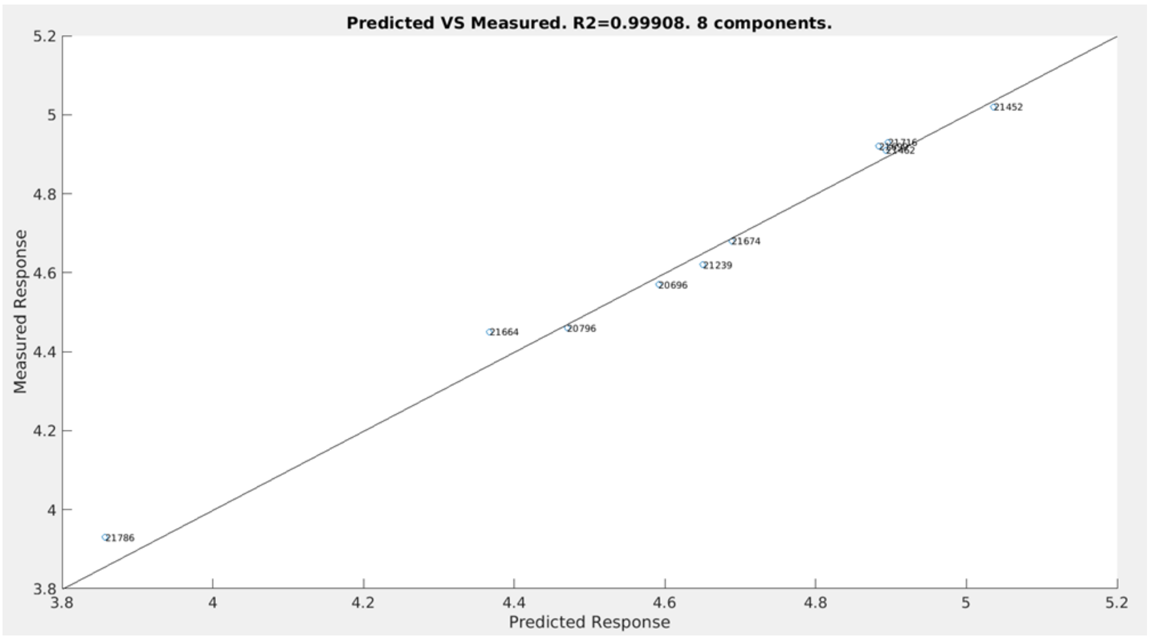

Figure 64: Predicted versus measured response for the ATR-FTIR model for TOT_ALK.

With good marks from the cross-validation, the final model, containing 8 components, was created. The predicted results superimposed onto the measured results can be seen in figure 65.

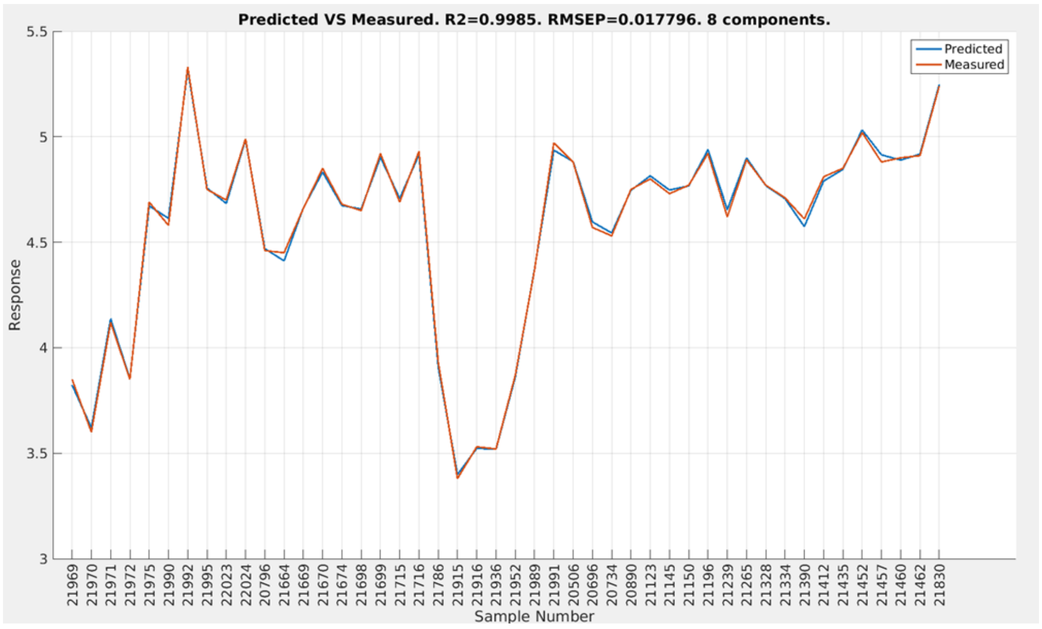

Figure 65: Plot of predicted and measured responses by sample number for ATR-FTIR for TOT_ALK.

TIC

The cross-validation results for TIC were promising, but upon a second run of outlier detection, samples 21890 and 21071 were deemed to be outliers. Once removed, the RMSEP of the final model, as well as the RMSECV improved by a significant amount. In the end, a final cross-validation was run, showing a minimum RMSECV at 6 components, for a value of 0.01411, as seen in figure 66, which is a residual of 0.83% compared to the average size of the total responses.

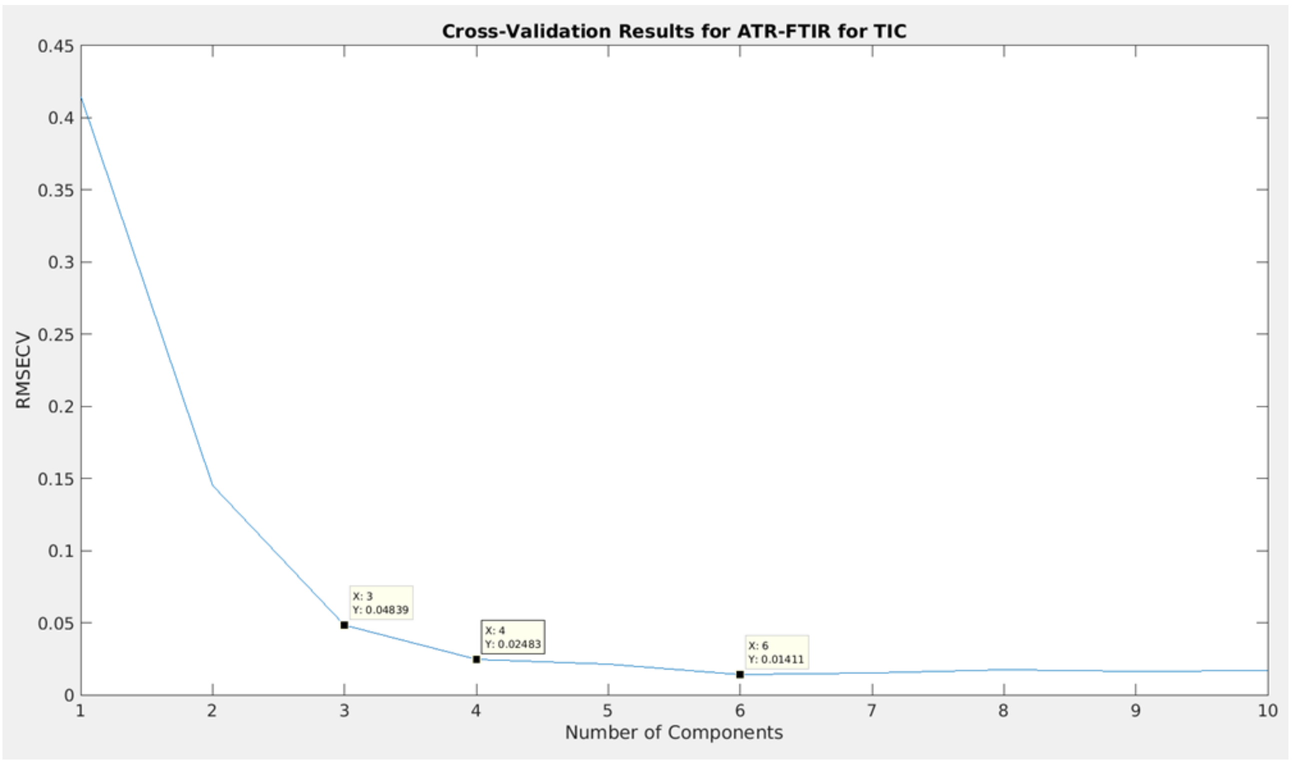

Figure 66: RMSECV versus components used for ATR-FTIR for TIC.

Although the suggested number of components is 6, and it is 6 that would be used from this point on, the difference in RMSECV between 4 components and 6 components is relatively minor, and should be considered to reduce calculation times if needed. In the average prediction residual percentage against number of components plot this is reflected as well, as seen in figure 67.

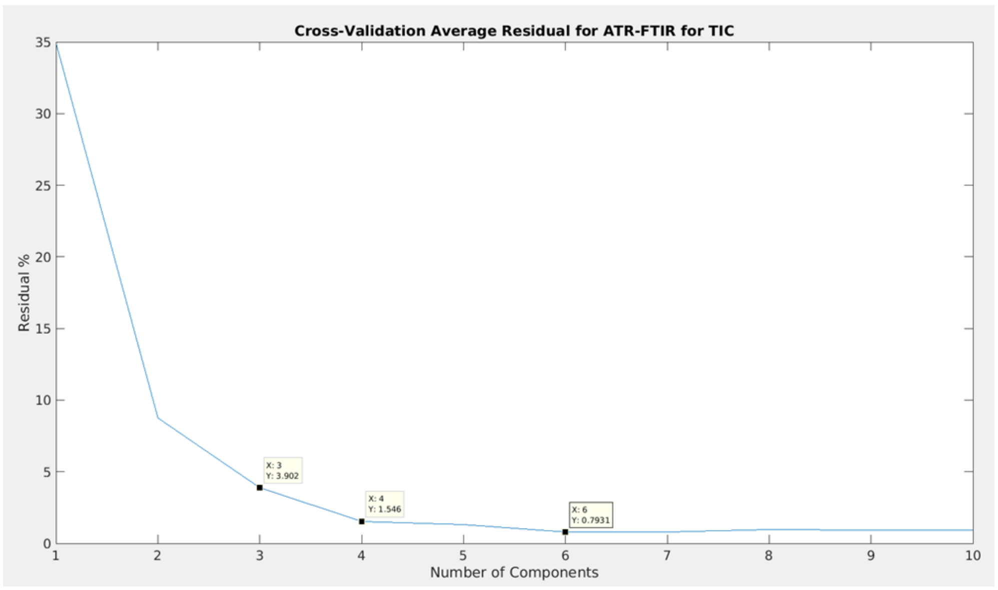

Figure 67: Plot of average residual percentage for a given amount of components for ATR-FTIR for TIC.

As seen, 6 components gives the lowest prediction residual percentage at 0.7931%, while 4 components still gives a somewhat reasonable 1.546%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 68.

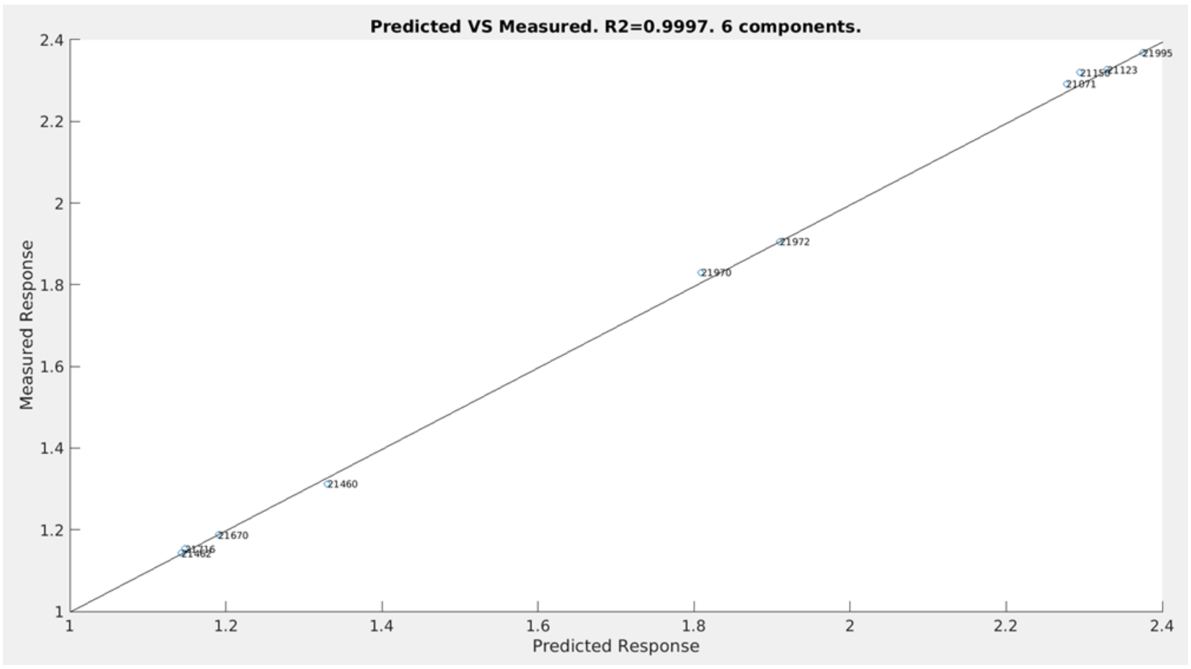

Figure 68: Predicted versus measured response for the ATR-FTIR model for TIC.

With good marks from the cross-validation, the final model, containing 6 components, was created. The predicted results superimposed onto the measured results can be seen in figure 69.

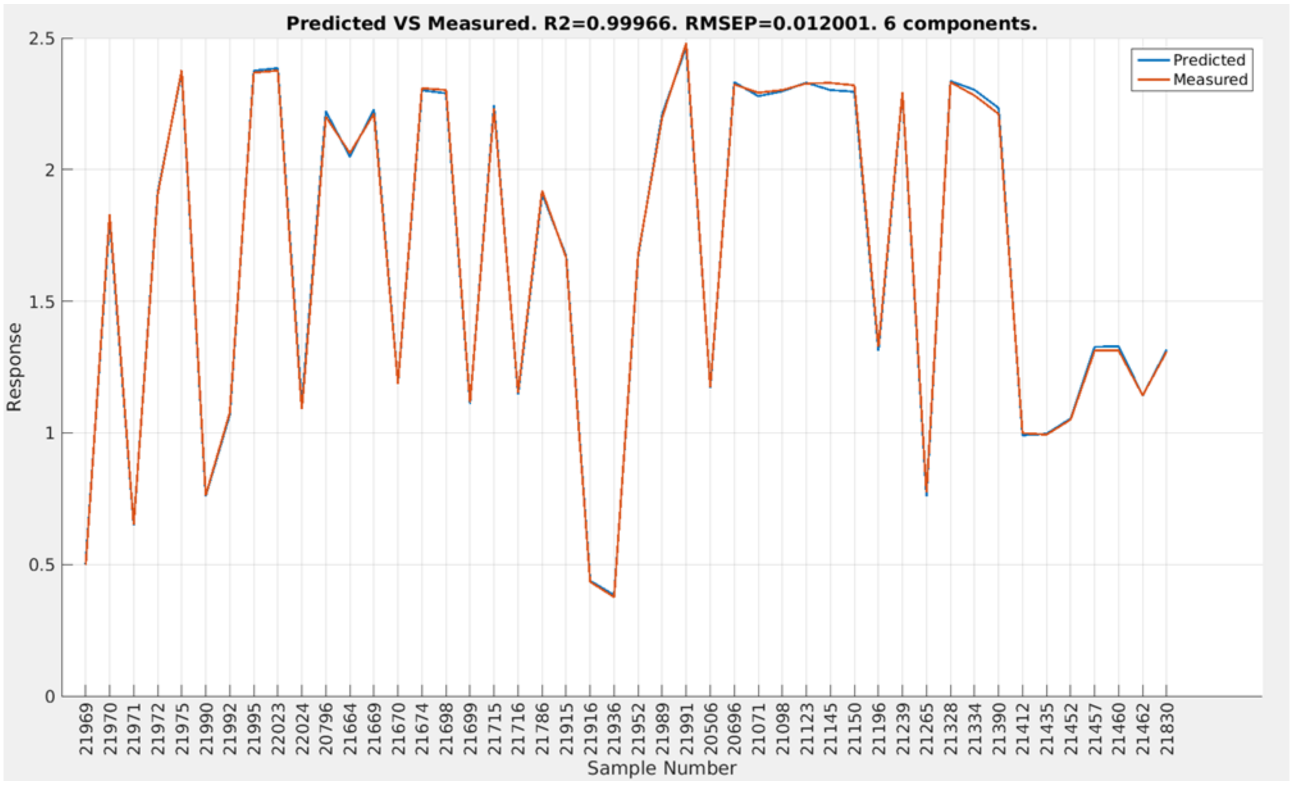

Figure 69: Plot of predicted and measured responses by sample number for ATR-FTIR for TIC.

Density

The cross-validation results for density were good, and no additional outliers were detected. In the end, a final cross-validation was run, showing a minimum RMSECV at 9 components, for a value of 0.8913, as seen in figure 70, which is a residual of 0.08% compared to the average size of the total responses.

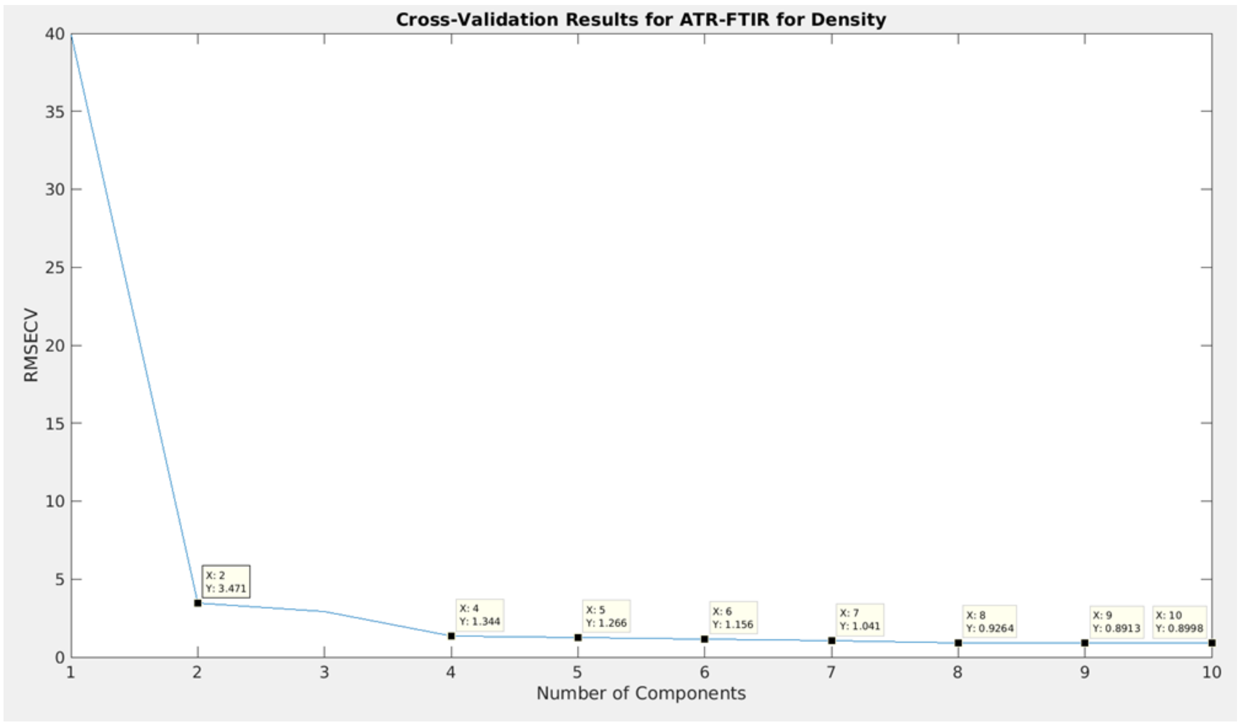

Figure 70: RMSECV versus components used for ATR-FTIR for density.

Although the suggested number of components is 9, and it is 9 that would be used from this point on, the difference in RMSECV between 4 components and 9 components is relatively minor, and should be considered to reduce calculation times if needed. Even the RMSECV at 2 components, being 3.471, is only 0.33% of the average measured response, well within reason. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 71.

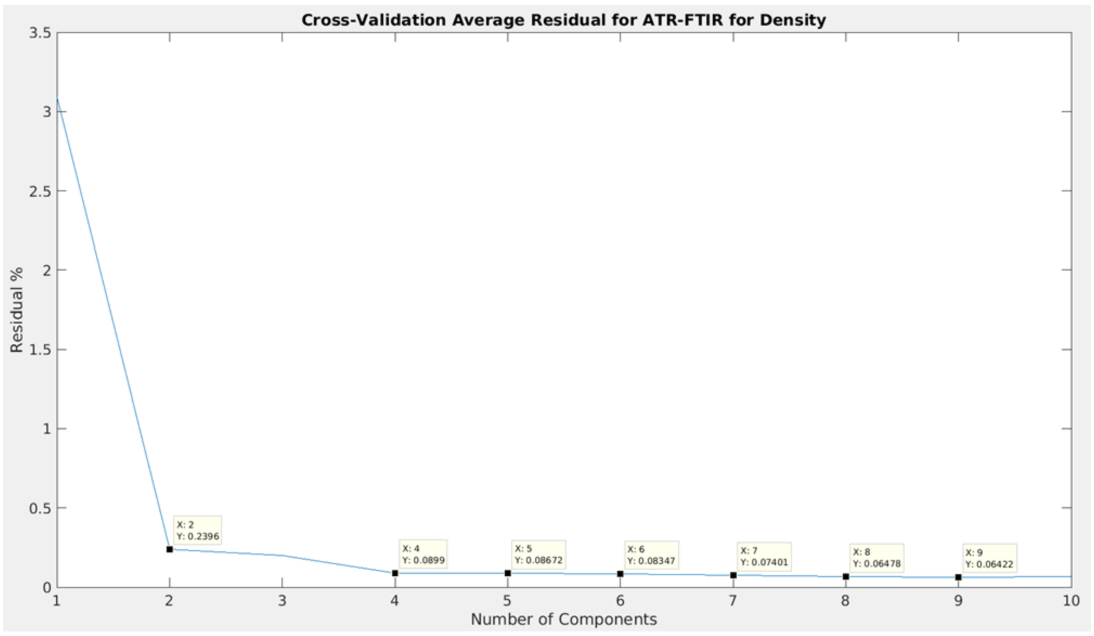

Figure 71: Plot of average residual percentage for a given amount of components for ATR-FTIR for density.

As seen, 9 components gives the lowest predicted residual percentage at 0.06422%, while 2 components still gives a more than reasonable 0.2396%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 72.

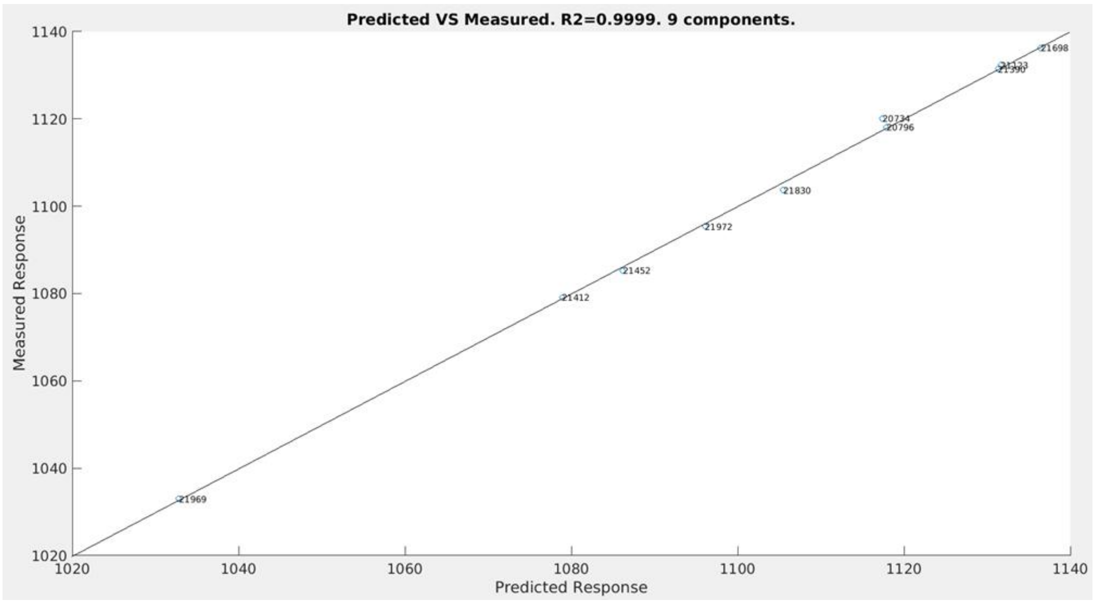

Figure 72: Predicted versus measured response for the ATR-FTIR model for density.

With good marks from the cross-validation, the final model, containing 9 components, was created. The predicted results superimposed onto the measured results can be seen in figure 73.

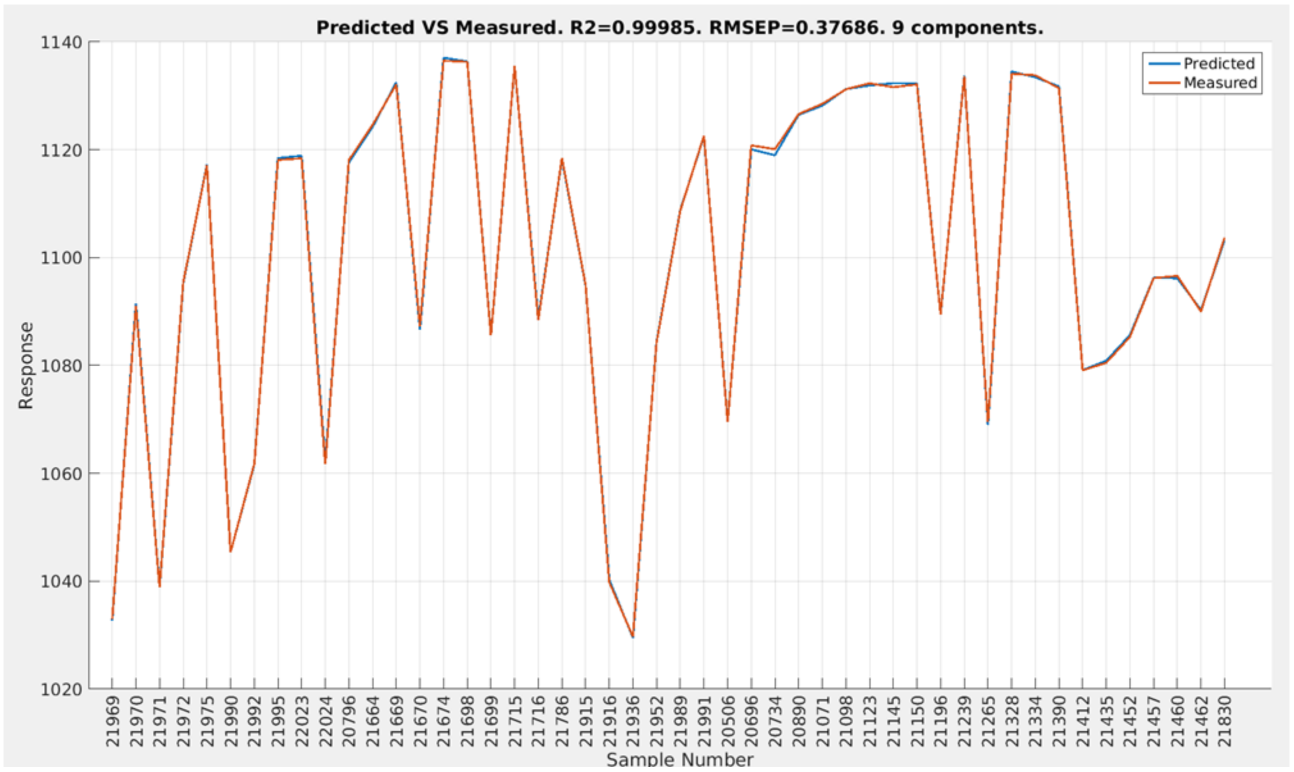

Figure 73: Plot of predicted and measured responses by sample number for ATR-FTIR for density.

6.4.2 NIR

The individual models for NIR performed reasonably well in all instances, and would still be a good choice for instruments if only a single instrument type was to be employed, although ATR-FTIR still provides overall better predictions.

TOT_ALK

The cross-validation results for TOT_ALK were very promising, but upon a second run of outlier detection, samples 21098, 21071 and 21664 were deemed to be outliers. Once removed, the RMSEP of the final model, as well as the RMSECV improved by a significant amount. In the end, a final cross-validation was run, showing a minimum RMSECV at 8 components, for a value of 0.05239, as seen in figure 74, which is a residual of 1.14% compared to the average size of the total responses.

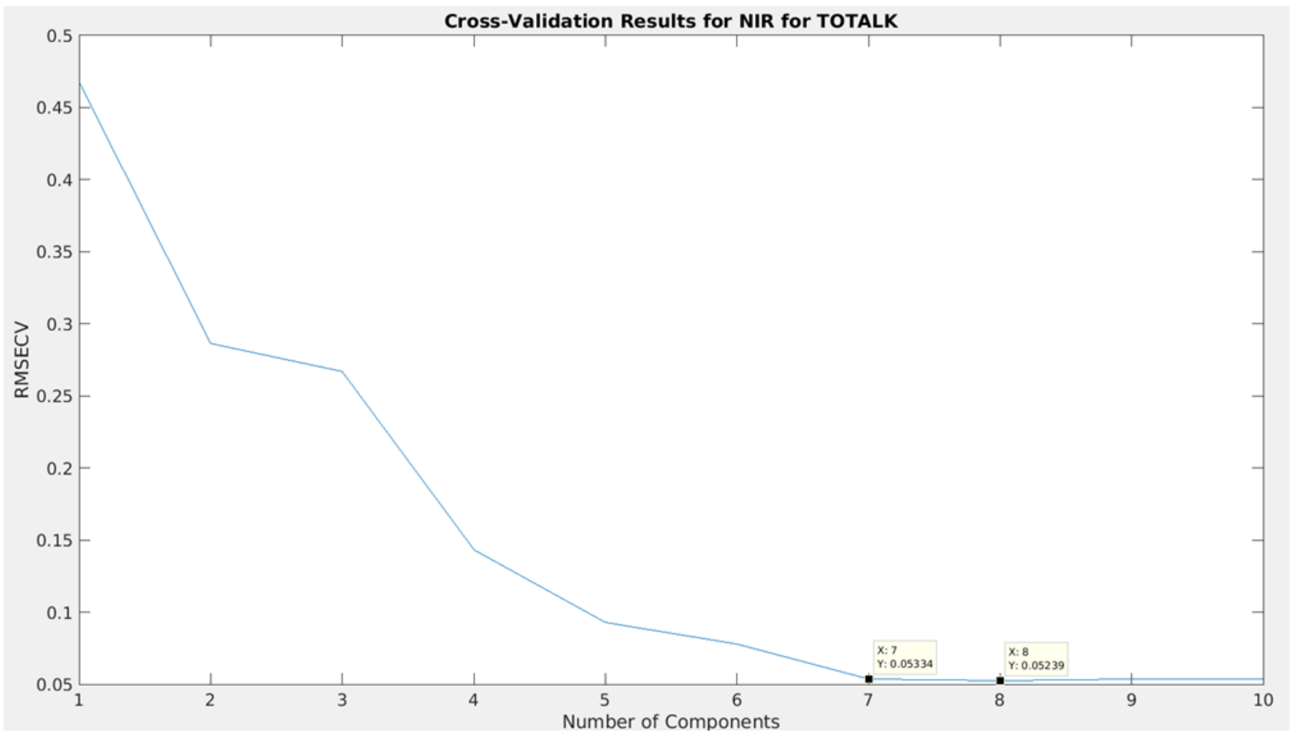

Figure 74: RMSECV versus components used for NIR for TOT_ALK.

Although the suggested number of components is 8, and it is 8 that would be used from this point on, the difference in RMSECV between 7 components and 8 components is relatively minor, and even 5 components is not too bad, and should be considered to reduce calculation times if needed. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 75.

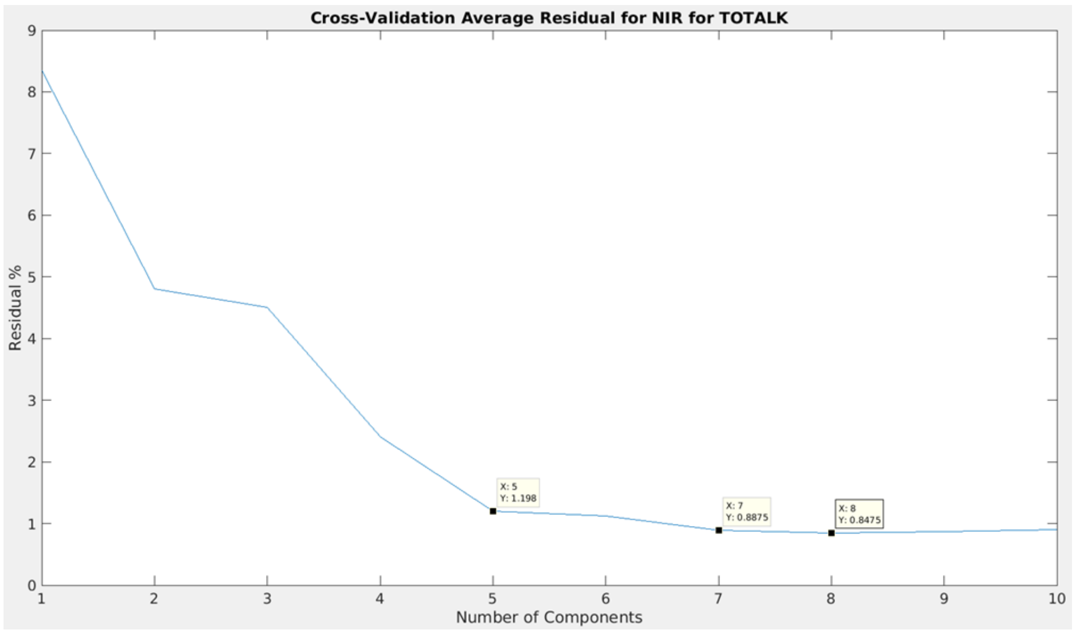

Figure 75: Plot of average residual percentage for a given amount of components for NIR for TOT_ALK.

As seen, 8 components gives the lowest predicted residual percentage at 0.8475%, while 5 components still gives a more than reasonable 1.198%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 76.

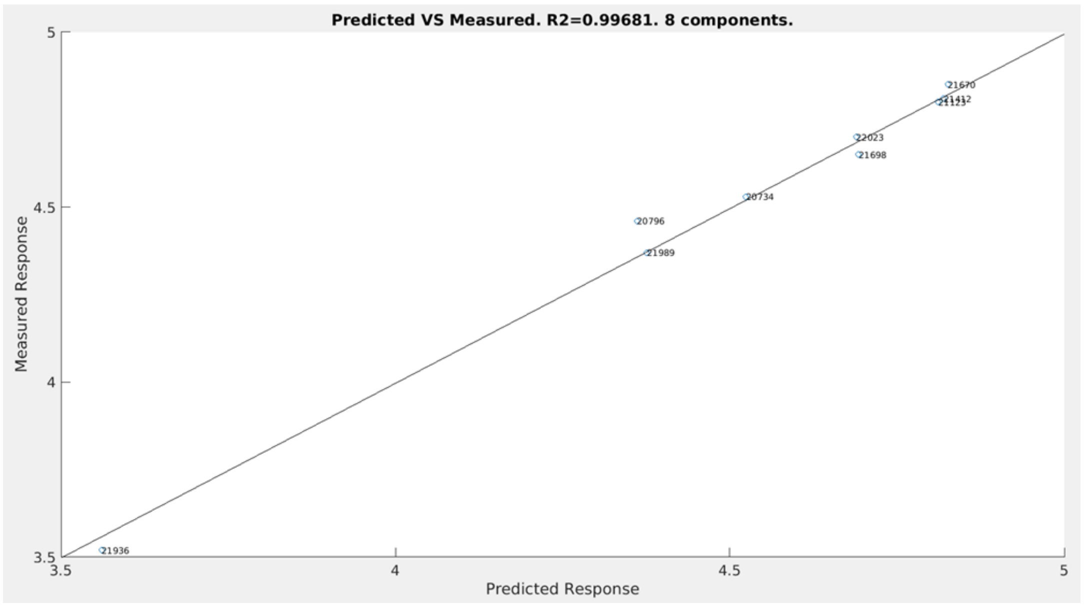

Figure 76: Predicted versus measured response for the NIR model for TOT_ALK.

With good marks from the cross-validation, the final model, containing 8 components, was created. The predicted results superimposed onto the measured results can be seen in figure 77.

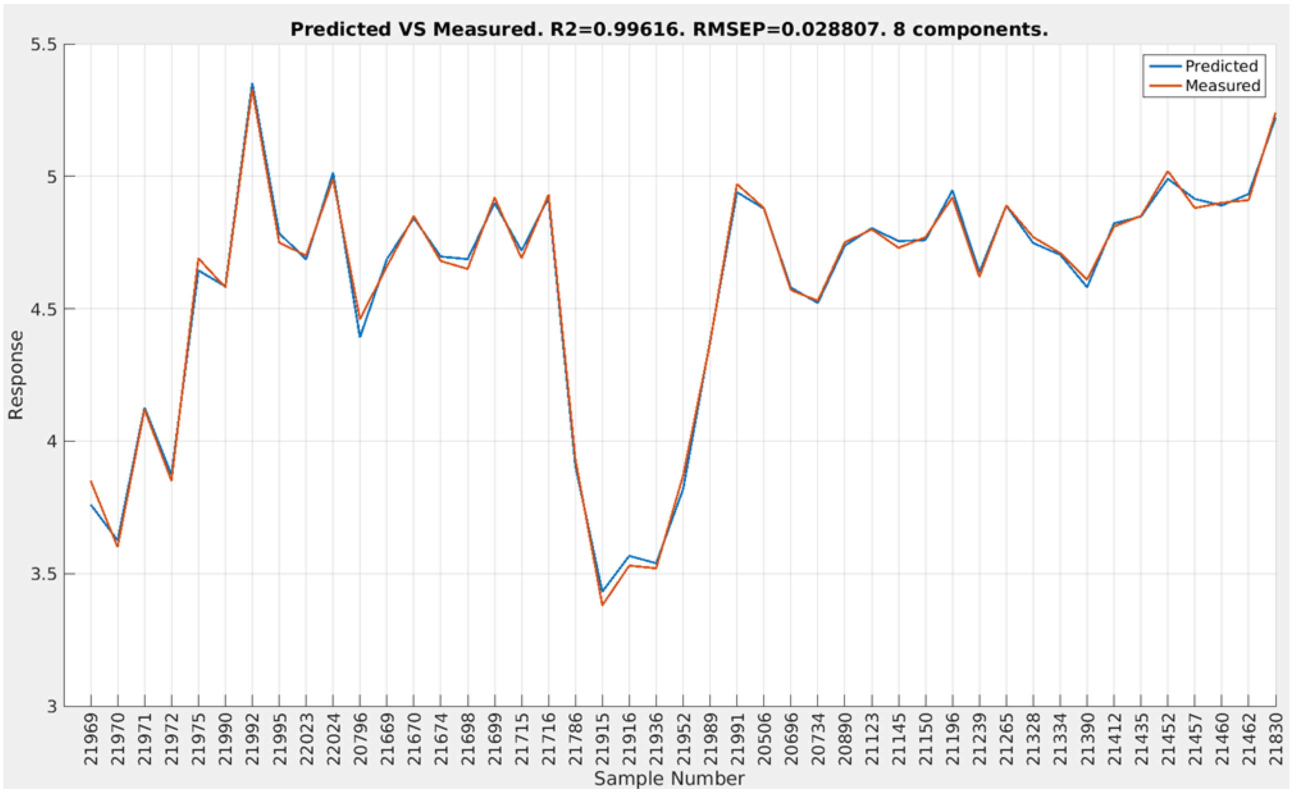

Figure 77: Plot of predicted and measured responses by sample number for NIR for TOT_ALK.

TIC

The cross-validation results for TIC were good, but upon a second run of outlier detection, samples 20734 and 20890 were deemed to be outliers. Once removed, the RMSEP of the final model, as well as the RMSECV improved by a significant amount. In the end, a final cross-validation was run, showing a minimum RMSECV at 8 components, for a value of 0.02214, as seen in figure 78, which is a residual of 1.31% compared to the average size of the total responses.

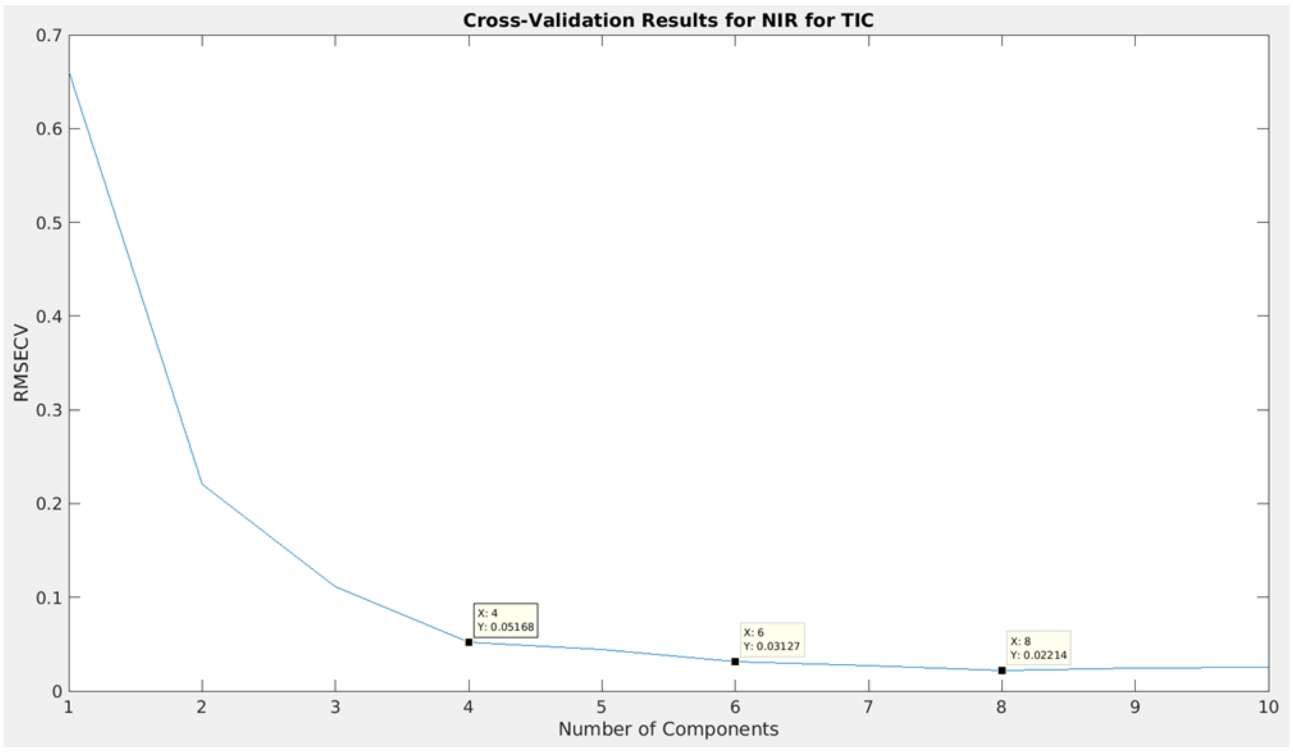

Figure 78: RMSECV versus components used for NIR for TIC.

Although the suggested number of components is 8, and it is 8 that would be used from this point on, the difference in RMSECV between 6 components and 8 components is relatively minor, and even 4 components is good, and should be considered to reduce calculation times if needed. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 79.

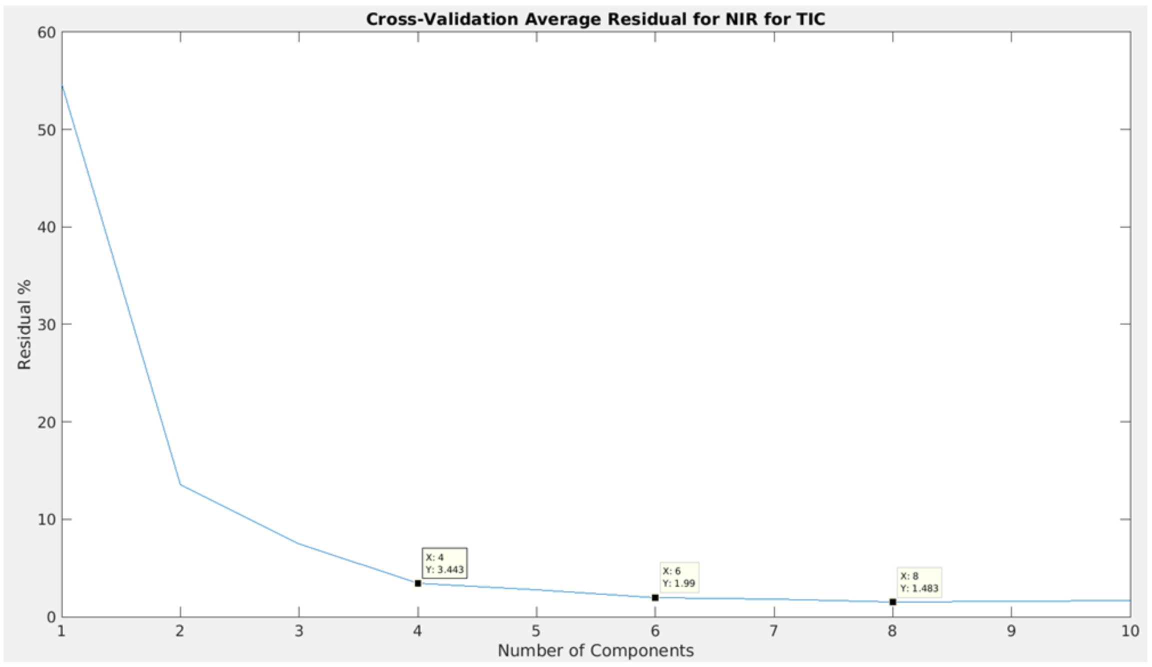

Figure 79: Plot of average residual percentage for a given amount of components for NIR for TIC.

As seen, 8 components gives the lowest predicted residual percentage at 1.483%, while 6 components still gives a reasonable 1.99%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 80.

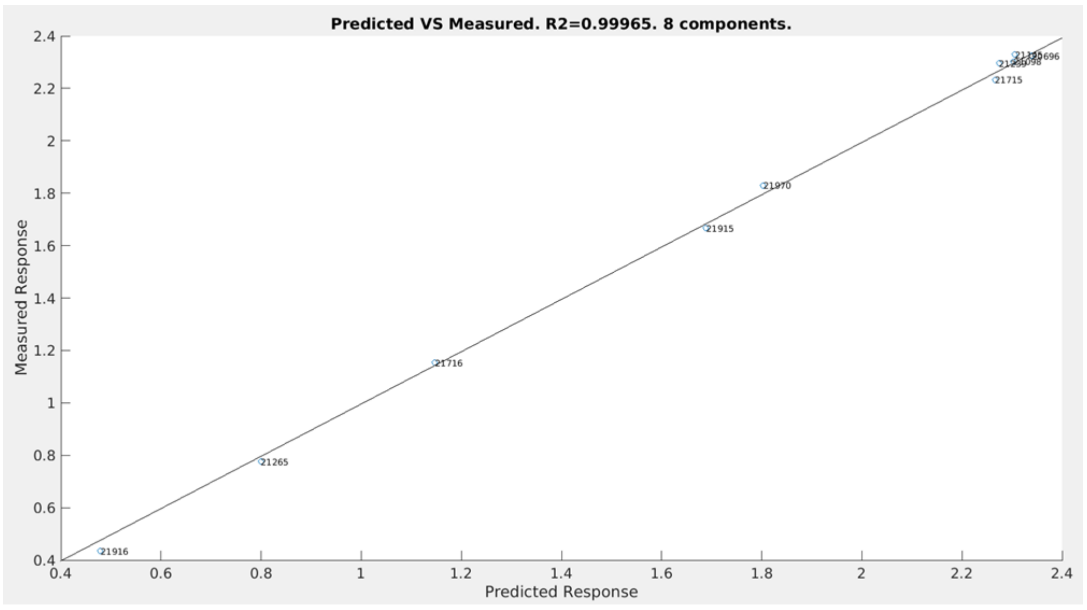

Figure 80: Predicted versus measured response for the NIR model for TIC.

With good marks from the cross-validation, the final model, containing 8 components, was created. The predicted results superimposed onto the measured results can be seen in figure 81.

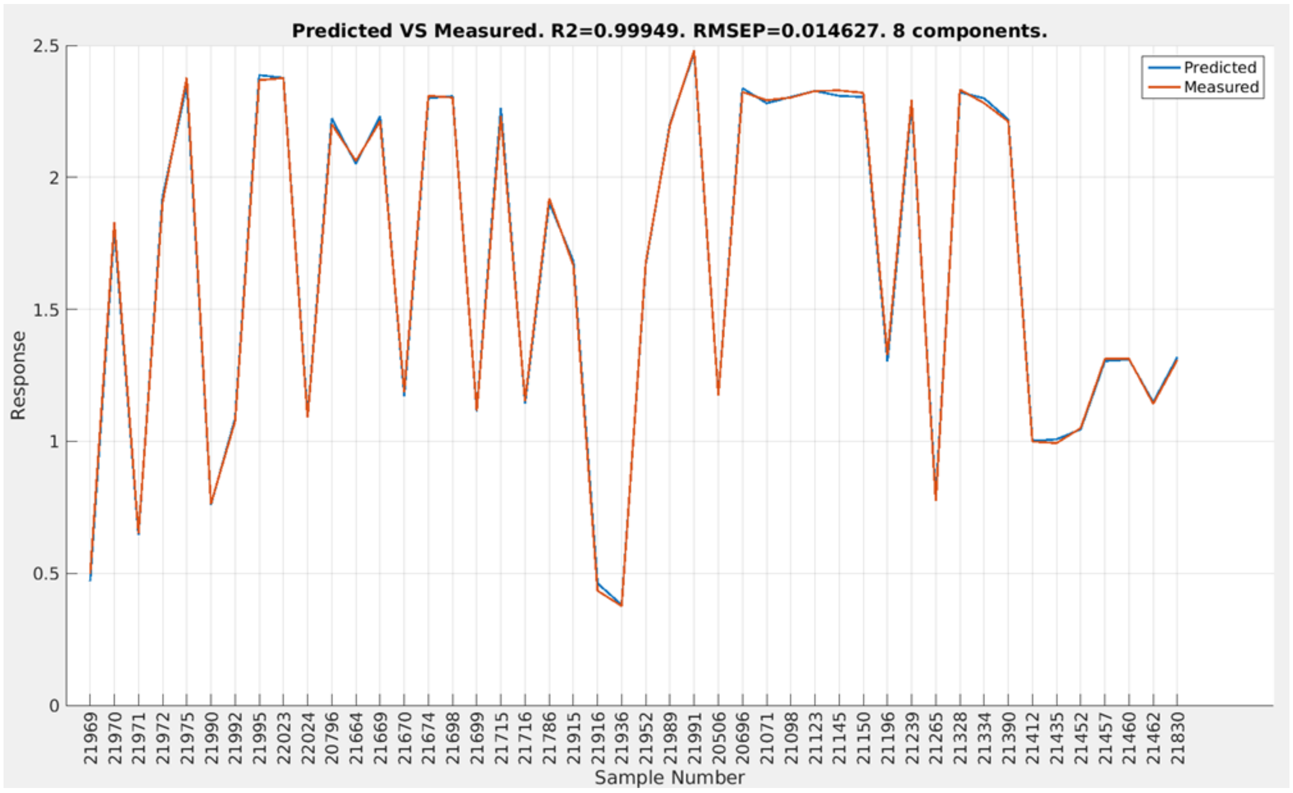

Figure 81: Plot of predicted and measured responses by sample number for NIR for TIC.

Density

The cross-validation results for density were good, but upon a second run of outlier detection, sample 20734 was deemed to be an outlier. Once removed, the RMSEP of the final model, as well as the RMSECV improved by a significant amount. In the end, a final cross-validation was run, showing a minimum RMSECV at 9 components, for a value of 0.6091, as seen in figure 82, which is a residual of 0.06% compared to the average size of the total responses.

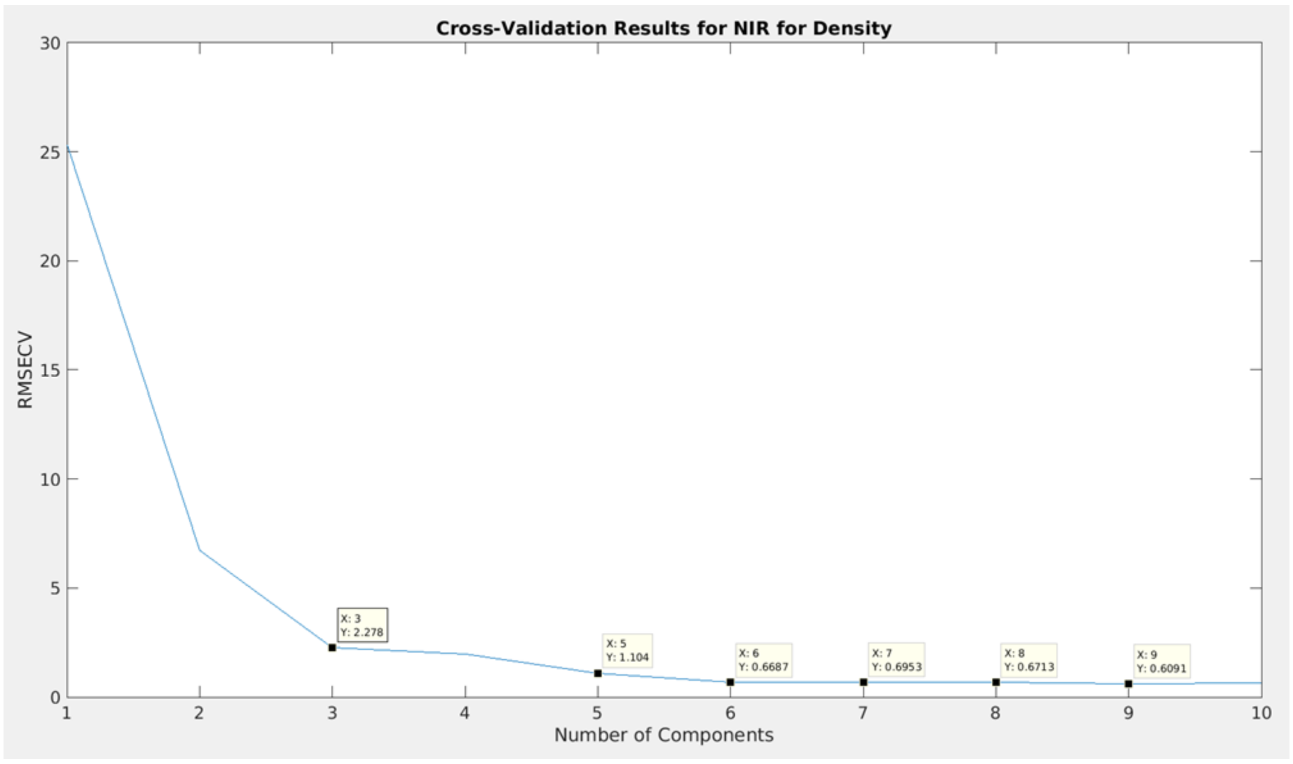

Figure 82: RMSECV versus components used for NIR for density.

Although the suggested number of components is 9, and it is 9 that would be used from this point on, the difference in RMSECV between 6 components and 9 components is relatively minor, and even 3 components is very good, and should be considered to reduce calculation times if needed. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 83.

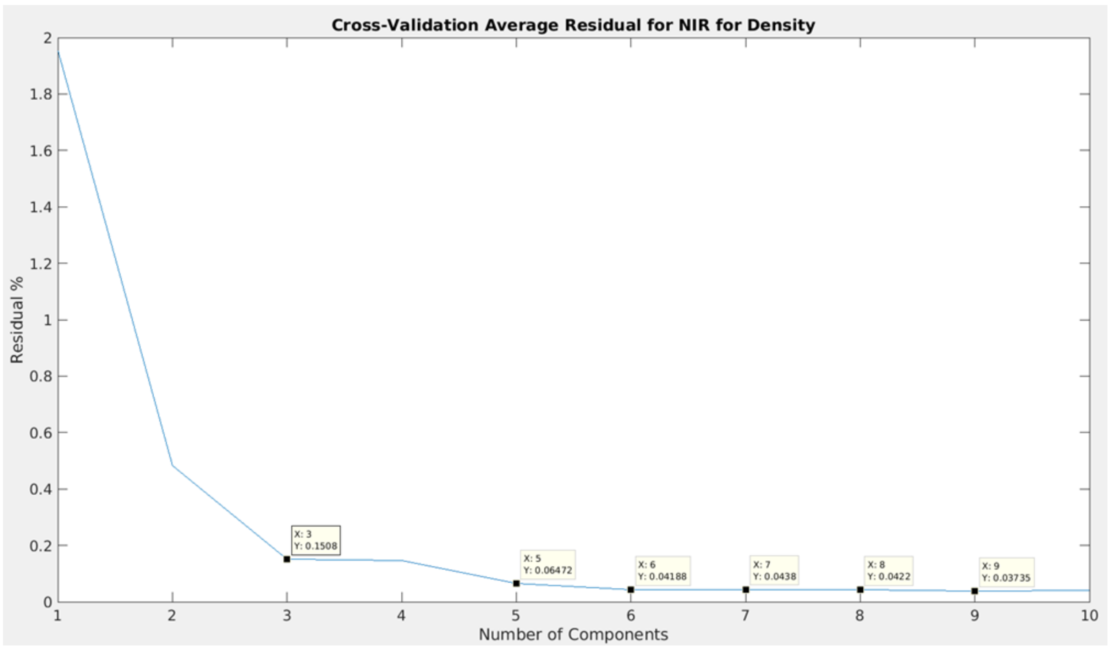

Figure 83: Plot of average residual percentage for a given amount of components for NIR for density.

As seen, 9 components gives the lowest true residual percentage at 0.04%, while 3 components still gives a reasonable 0.15%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 84.

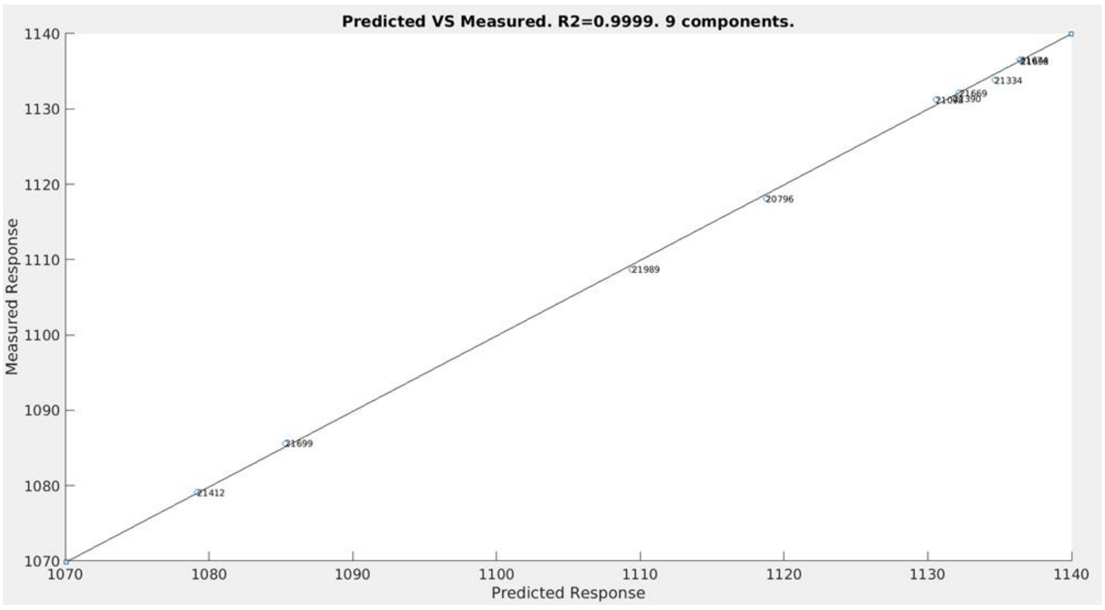

Figure 84: Predicted versus measured response for the NIR model for density.

With good marks from the cross-validation, the final model, containing 9 components, was created. The predicted results superimposed onto the measured results can be seen in figure 85.

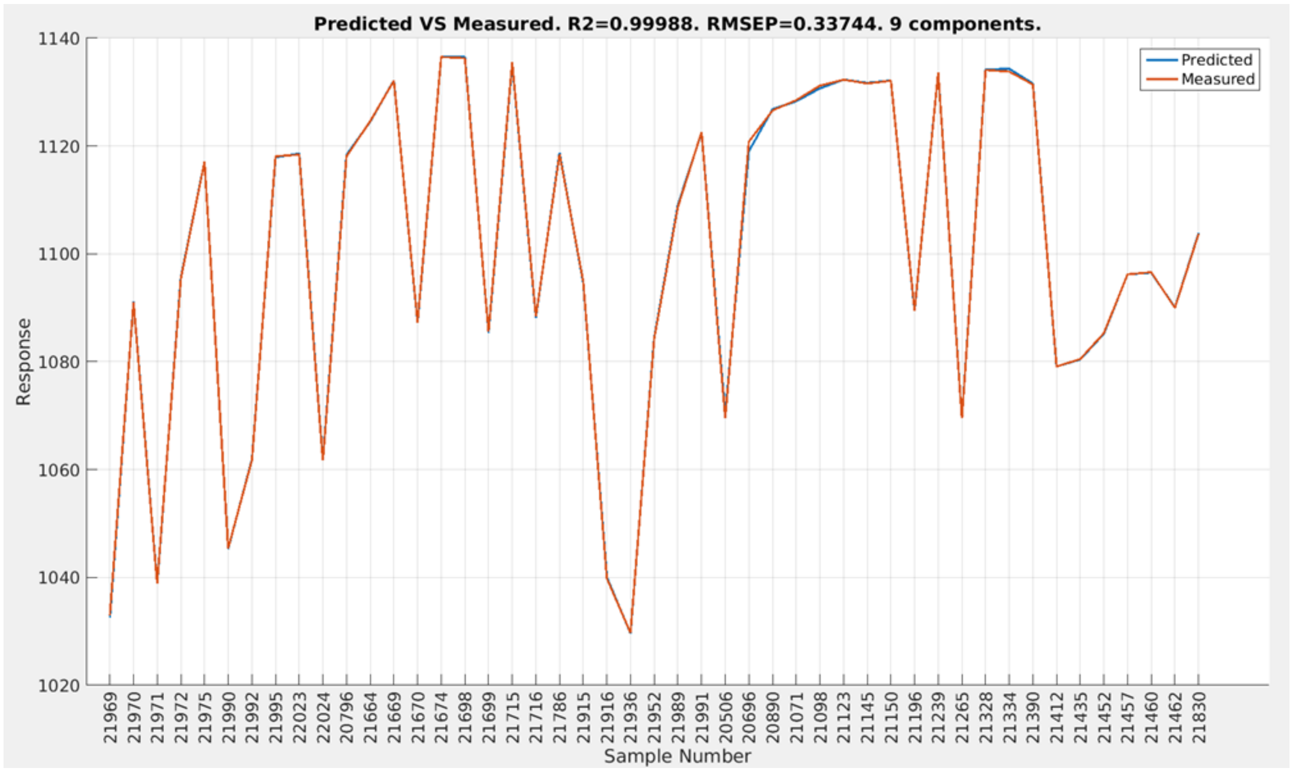

Figure 85: Plot of predicted and measured responses by sample number for NIR for density.

6.4.3 Raman

The individual models for Raman performed poorly in all instances, and it would not be recommended for instruments if only a single instrument type was to be employed. With better noise reduction, or some sort of fluorescence reference spectrum to make corrections with, things might be different, but that is left unknown.

TOT_ALK

The cross-validation results for TOT_ALK were surprisingly good, all things considered. A second round of outlier detection yielded no significant outliers. In the end, a final cross-validation was run, showing a minimum RMSECV at 8 components, for a value of 0.1326, as seen in figure 86, which is a residual of 2.89% compared to the average size of the total responses.

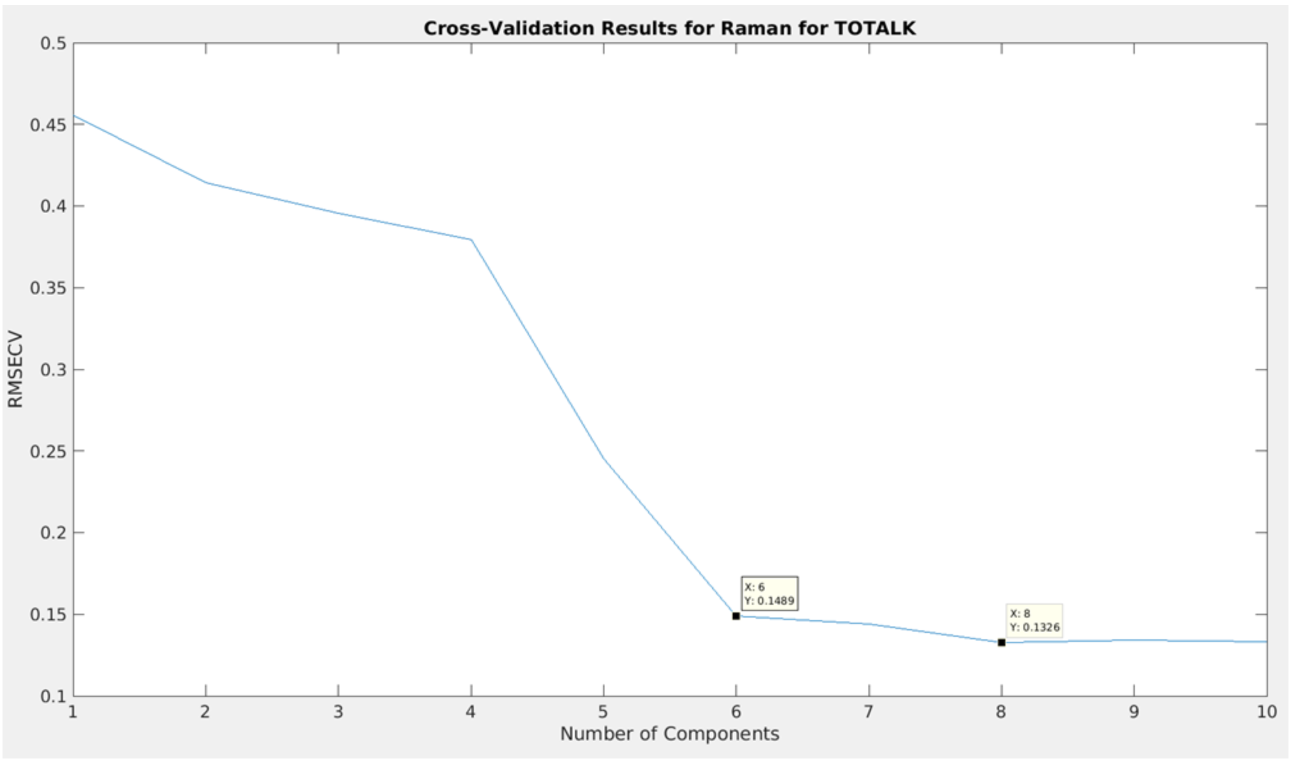

Figure 86: RMSECV versus components used for Raman for TOT_ALK.

Oddly, although the suggested number of components is 8 based on RMSECV, the true residual continued to decrease until 10 components. There appeared to be some rather complex interactions going on that made pegging a stable value difficult, so the suggested number of components by RMSECV was used. To reduce the data volume being processed, 6 components could also be worth considering. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 87.

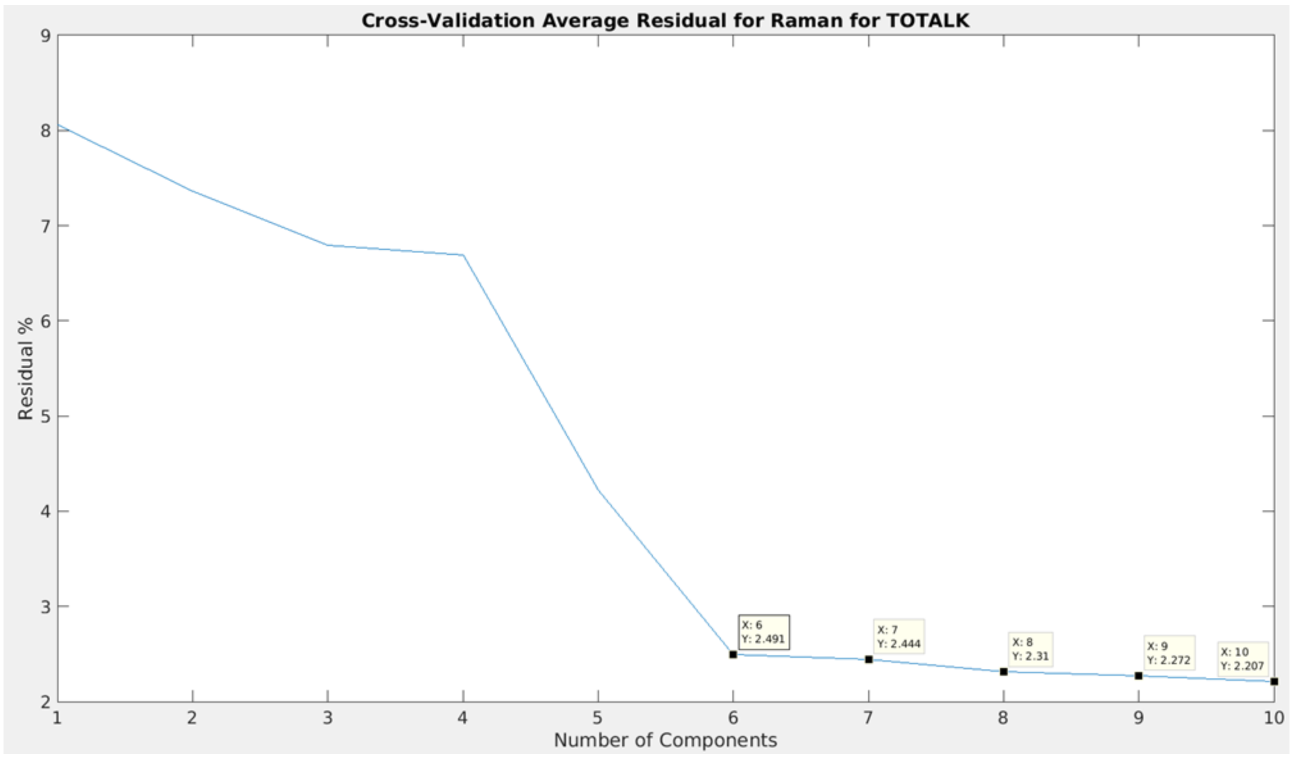

Figure 87: Plot of average residual percentage for a given amount of components for Raman for TOT_ALK.

As seen, 10 components gives the lowest true residual percentage at 2.207%, while the chosen number of components, 8, gives 2.31%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 88.

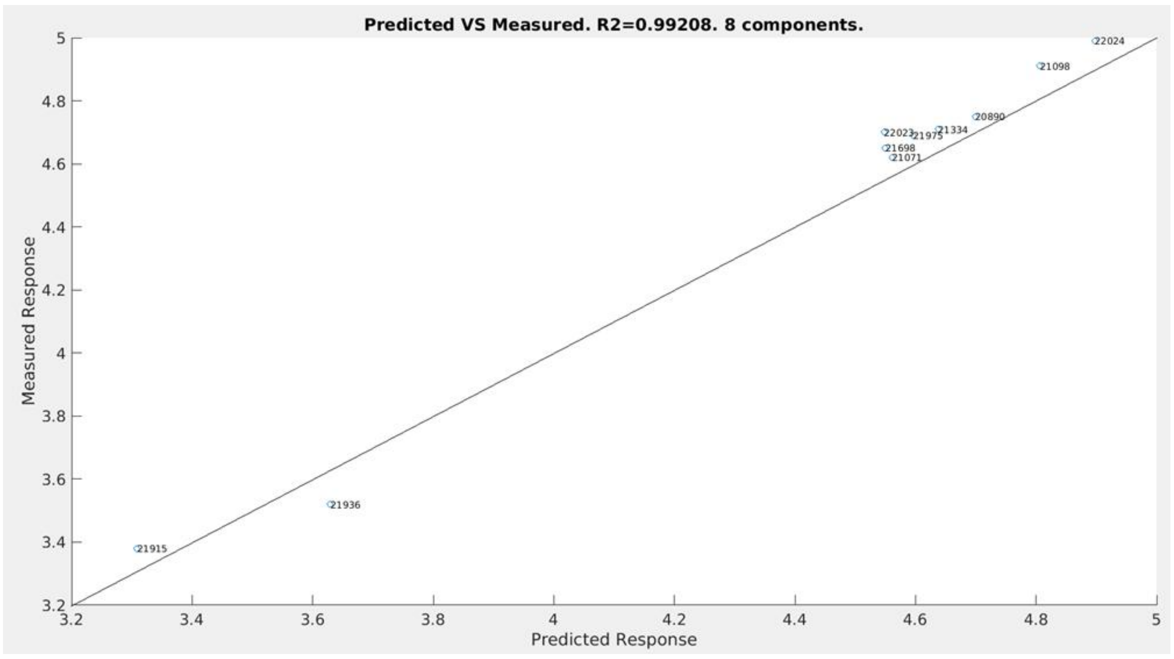

Figure 88: Predicted versus measured response for the Raman model for TOT_ALK.

With good marks from the cross-validation, the final model, containing 8 components, was created. The predicted results superimposed onto the measured results can be seen in figure 89.

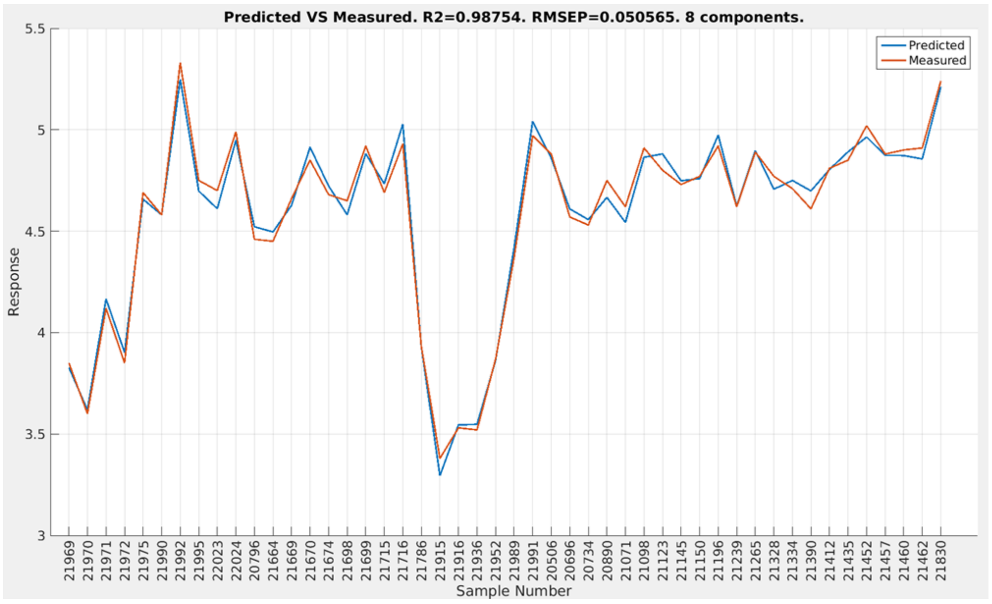

Figure 89: Plot of predicted and measured responses by sample number for Raman for TOT_ALK.

TIC

The cross-validation results for TIC were very poor, showing again that using Raman for TIC measurements seems unwise. A second round of outlier detection yielded no significant outliers due mostly to the fact that everything had such high residuals. In the end, a final cross-validation was run, showing a minimum RMSECV at 8 components, for a value of 0.3101, as seen in figure 90, which is a residual of 18.31% compared to the average size of the total responses.

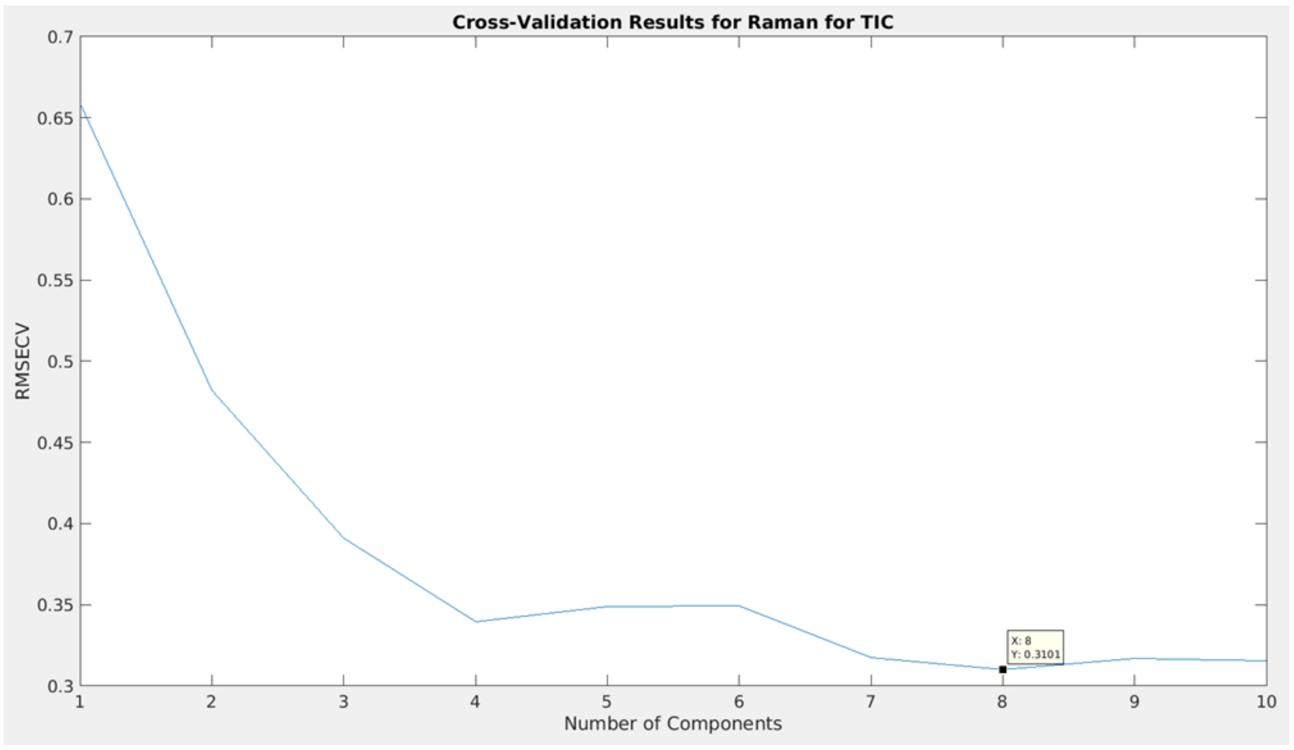

Figure 90: RMSECV versus components used for Raman for TIC.

Although the suggested number of components is 8, and it is 8 that would be used from this point on, the difference in RMSECV between 7 components and 8 components is relatively minor, and even 4 should be considered to reduce calculation times if needed, as the results are already rather poor. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 91.

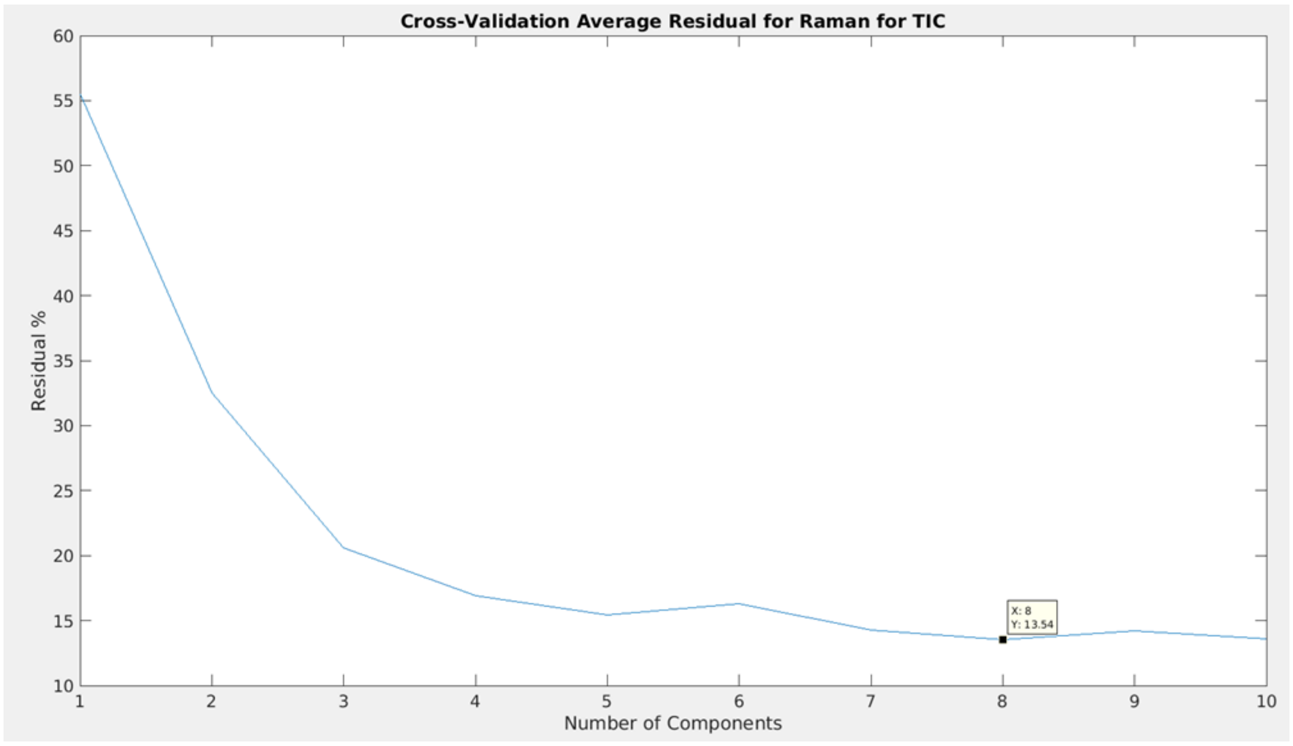

Figure 91: Plot of average residual percentage for a given amount of components for Raman for TIC.

As seen, 8 components gives the lowest true residual percentage at 13.54%, which is very high. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 92. It should be noted that in contrast to previously shown plots of this nature, this sample plot represents the average plot given during cross-validation, and was handpicked, rather than randomly selected. In at least 10% of the cross-validation iterations, plots were made with at least one point wildly off-course. Attempts to pin down the reason were unsuccessful.

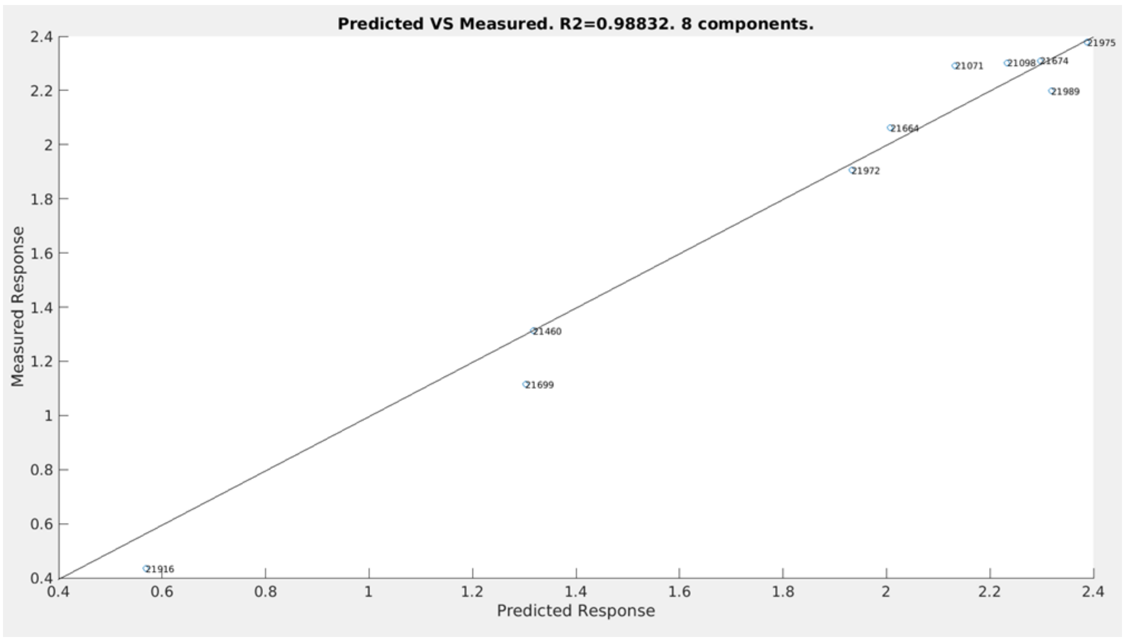

Figure 92: Predicted versus measured response for the Raman model for TIC.

Despite the poor marks from the cross-validation, the final model, containing 8 components, was created. The predicted results superimposed onto the measured results can be seen in figure 93.

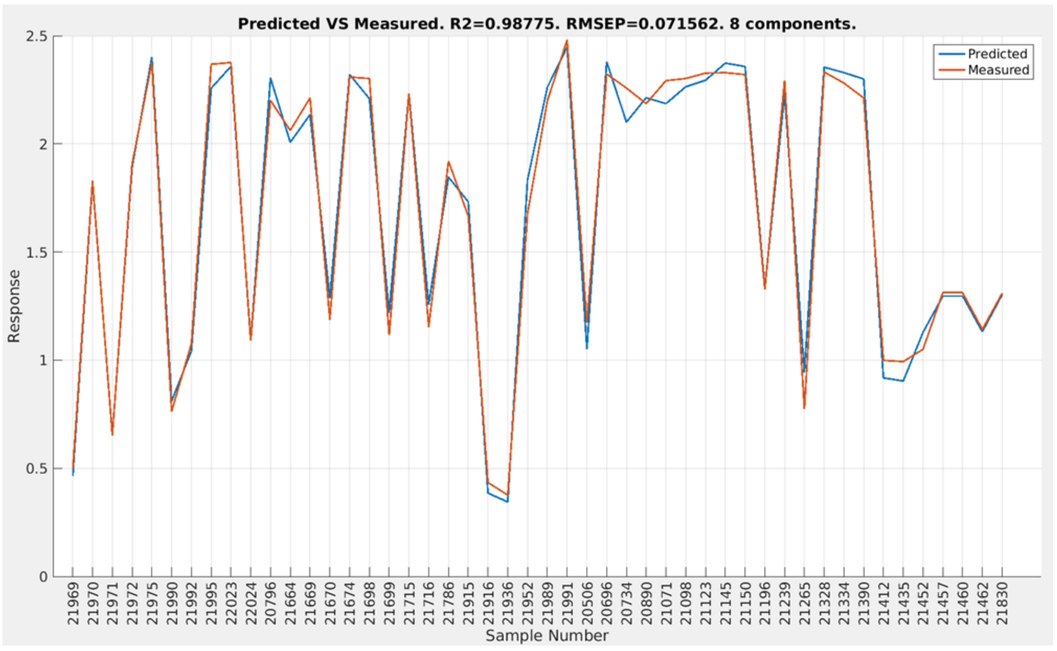

Figure 93: Plot of predicted and measured responses by sample number for Raman for TIC.

Density

The cross-validation results for density were good, all things considered, but upon a second run of outlier detection, samples 21265 and 21239 were deemed to be outliers. Once removed, the RMSEP of the final model, as well as the RMSECV improved by a significant amount. In the end, a final cross-validation was run, showing a minimum RMSECV at 5 components, for a value of 9.539, as seen in figure 94, which is a residual of 0.91% compared to the average size of the total responses.

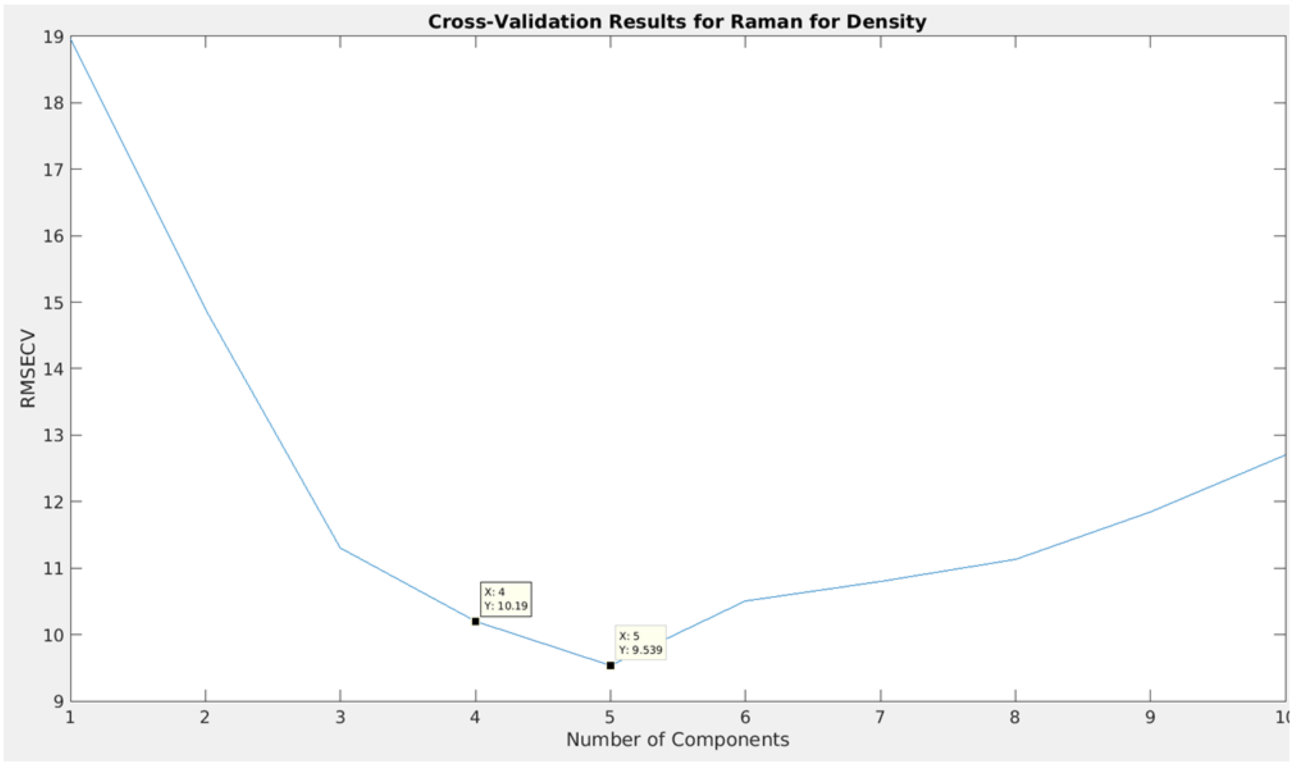

Figure 94: RMSECV versus components used for Raman for density.

Although the suggested number of components is 5, and it is 5 that would be used from this point on, the difference in RMSECV between 4 components and 5 components is relatively minor, and should be considered to reduce calculation times if needed. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 95.

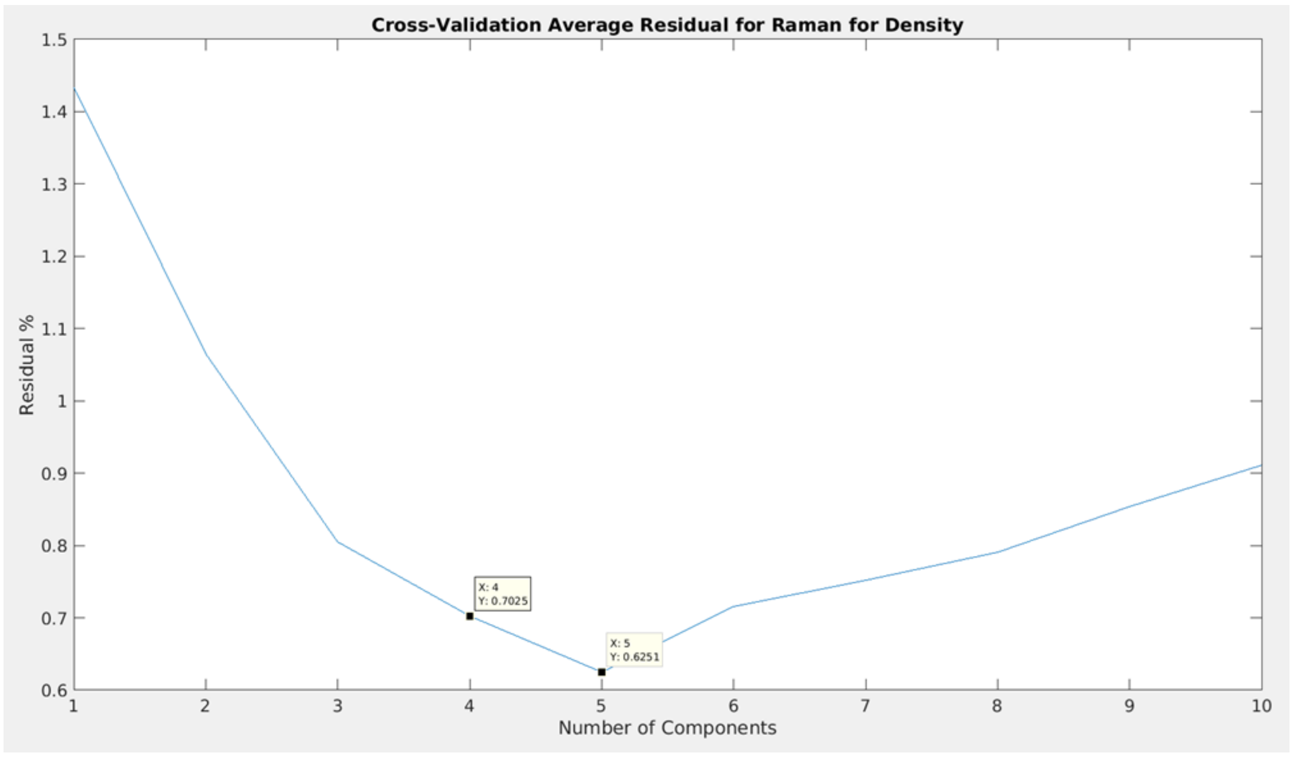

Figure 95: Plot of average residual percentage for a given amount of components for Raman for density.

As seen, 5 components gives the lowest true residual percentage at 0.6251%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 96.

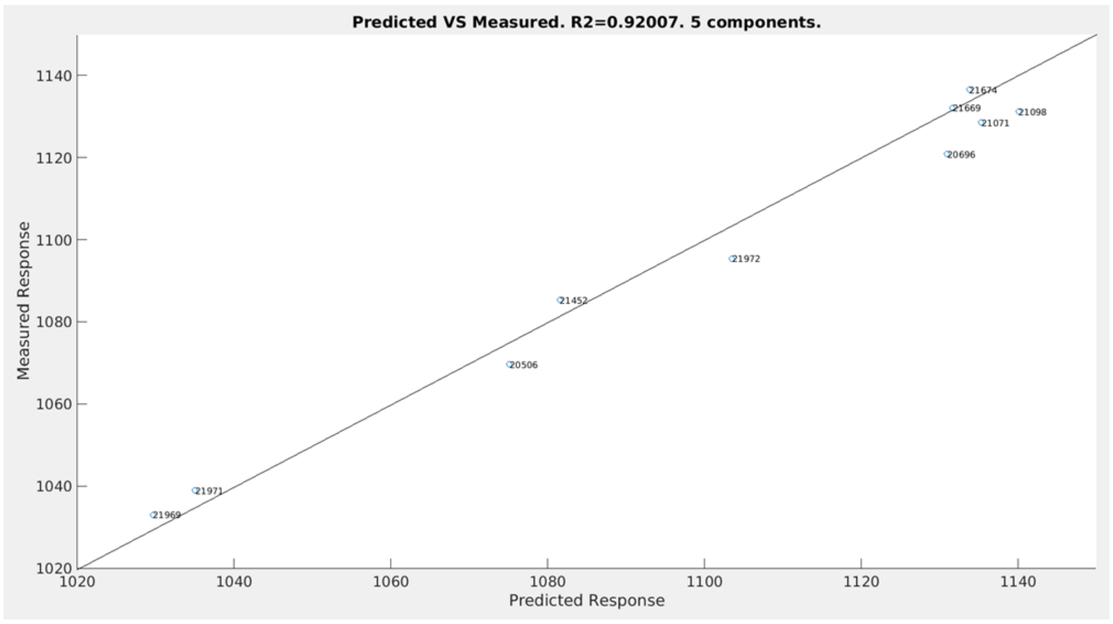

Figure 96: Predicted versus measured response for the Raman model for density.

With good marks from the cross-validation, the final model, containing 5 components, was created. The predicted results superimposed onto the measured results can be seen in figure 97.

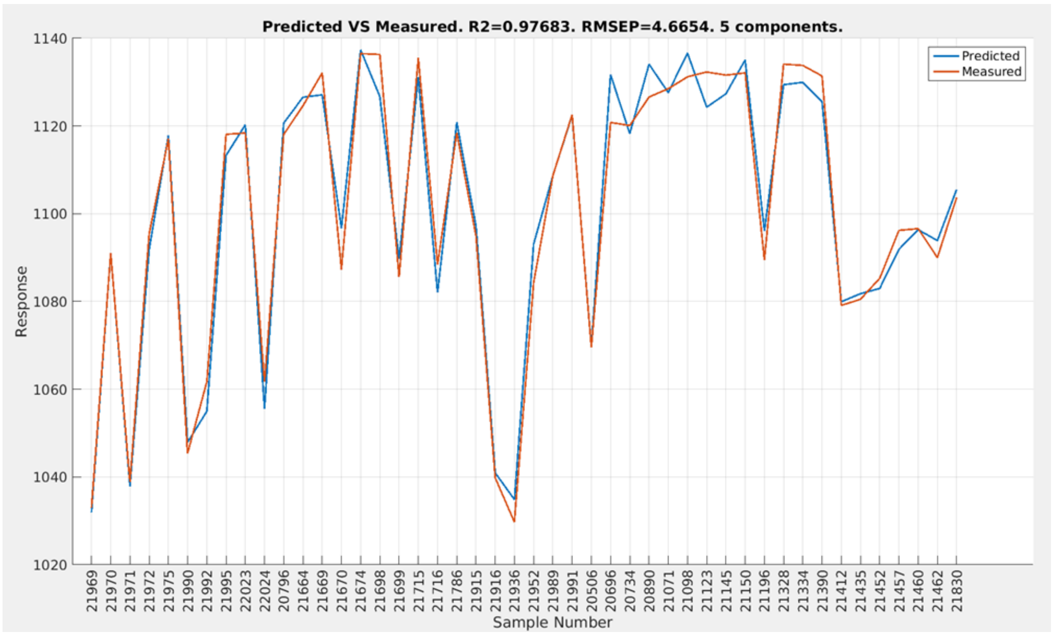

Figure 97: Plot of predicted and measured responses by sample number for Raman for density.

6.5 Low-Level Data Fusion

The three preprocessed spectra groups from each instrument were concatenated in the following order: ATR-FTIR, NIR, Raman. Due to varying wavenumber scales interfering with making a coherent plot, the wavenumbers were simply replaced by variable numbers for easy interpretation. Due to Raman's very poor performance overall, two sets of LLDF sets were made. One containing all three instrument measurements as mentioned above, and one containing only ATR-FTIR and NIR.

6.5.1 Fusion of ATR-FTIR, NIR and Raman.

TOT_ALK

After concatenation, the spectra contained 6952 variables, comprised of 1275 from ATR-FTIR, 560 from NIR and 5117 from Raman, as seen in figure 98. The overall spectrum was dominated in size by the Raman data, which was by far the worst performing dataset.

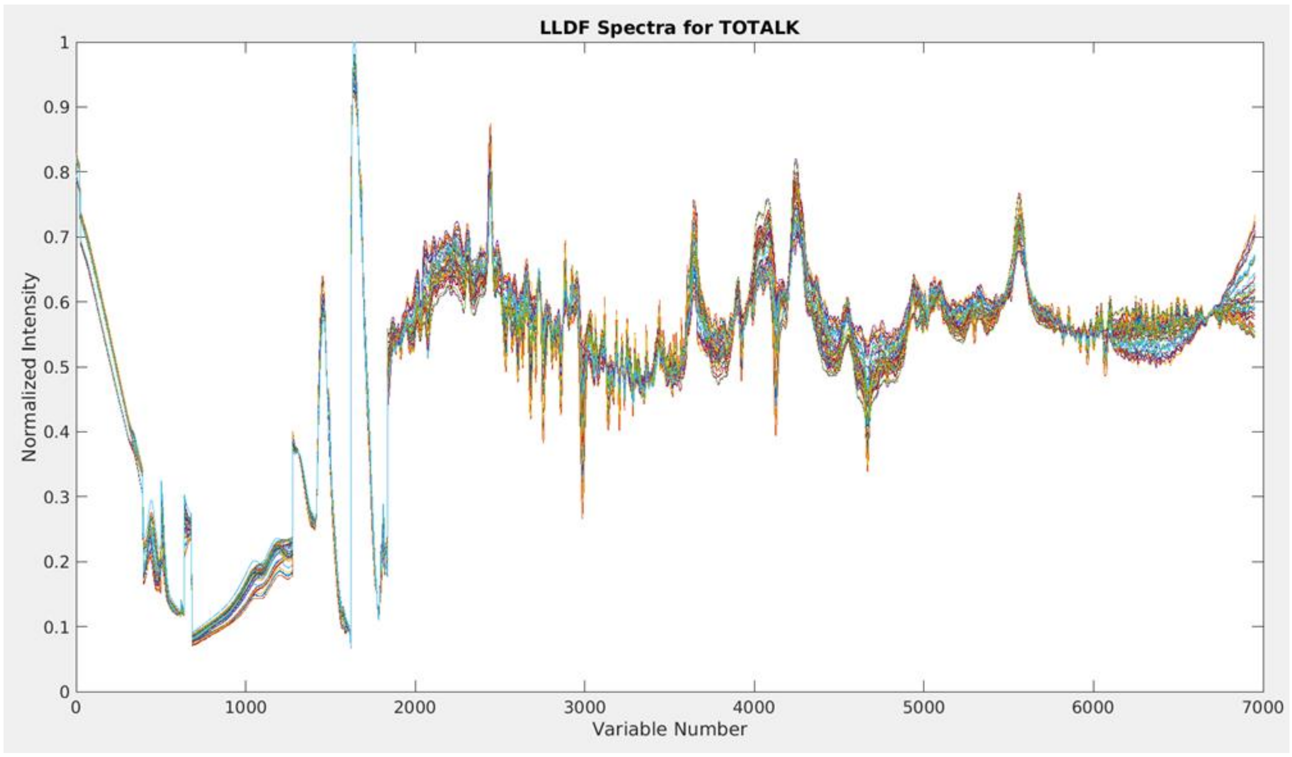

Figure 98: Plot of LLDF's concatenated spectra for all samples for TOT_ALK.

The cross-validation results for TOT_ALK were very promising. A second run of outlier detection was unable to identify any potential outliers. In the end, a final cross-validation was run, showing a minimum RMSECV at 11 components, for a value of 0.06044, as seen in figure 99. Although not bad in general, it is significantly larger than the 0.03343 that the ATR-FTIR individual model produced.

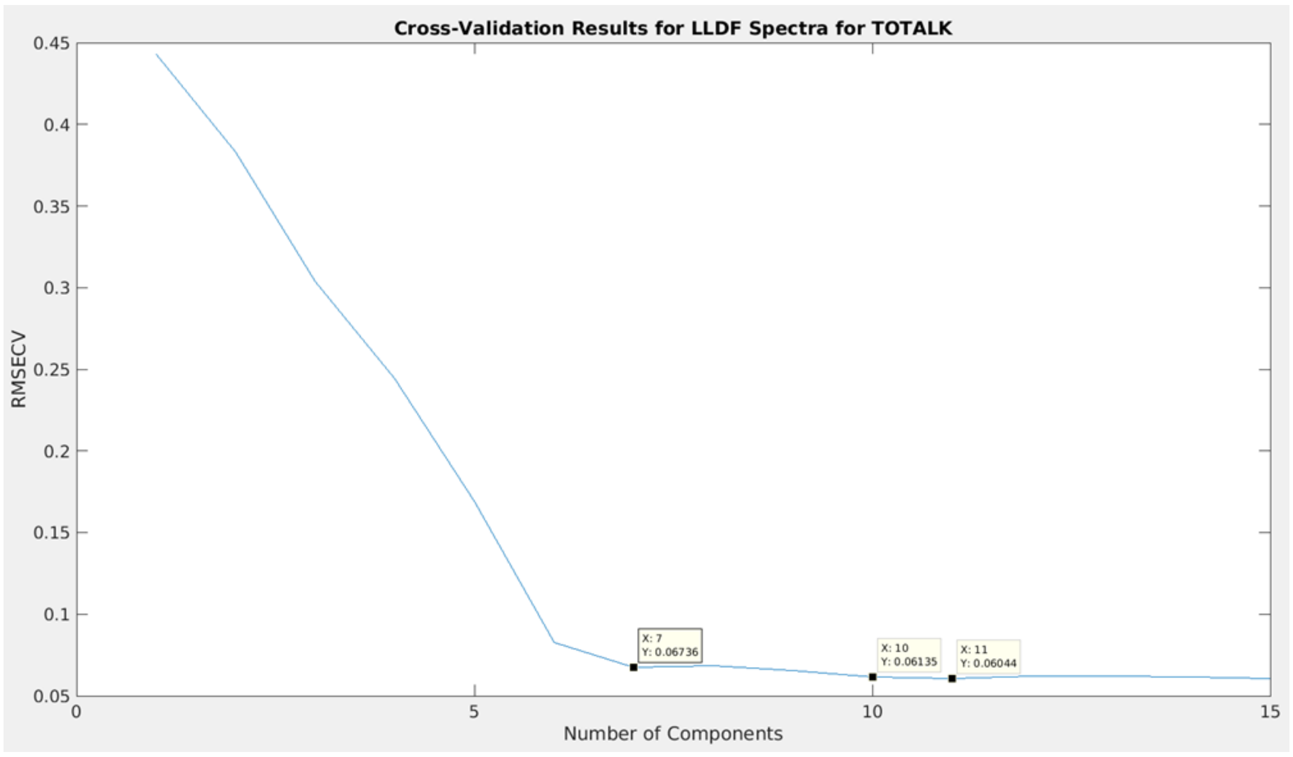

Figure 99: RMSECV versus components used for LLDF for TOT_ALK.

Although the suggested number of components is 11, and it is 11 that would be used from this point on, the difference in RMSECV between 7 components and 11 components is relatively minor and should be considered to reduce calculation times if needed. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 100.

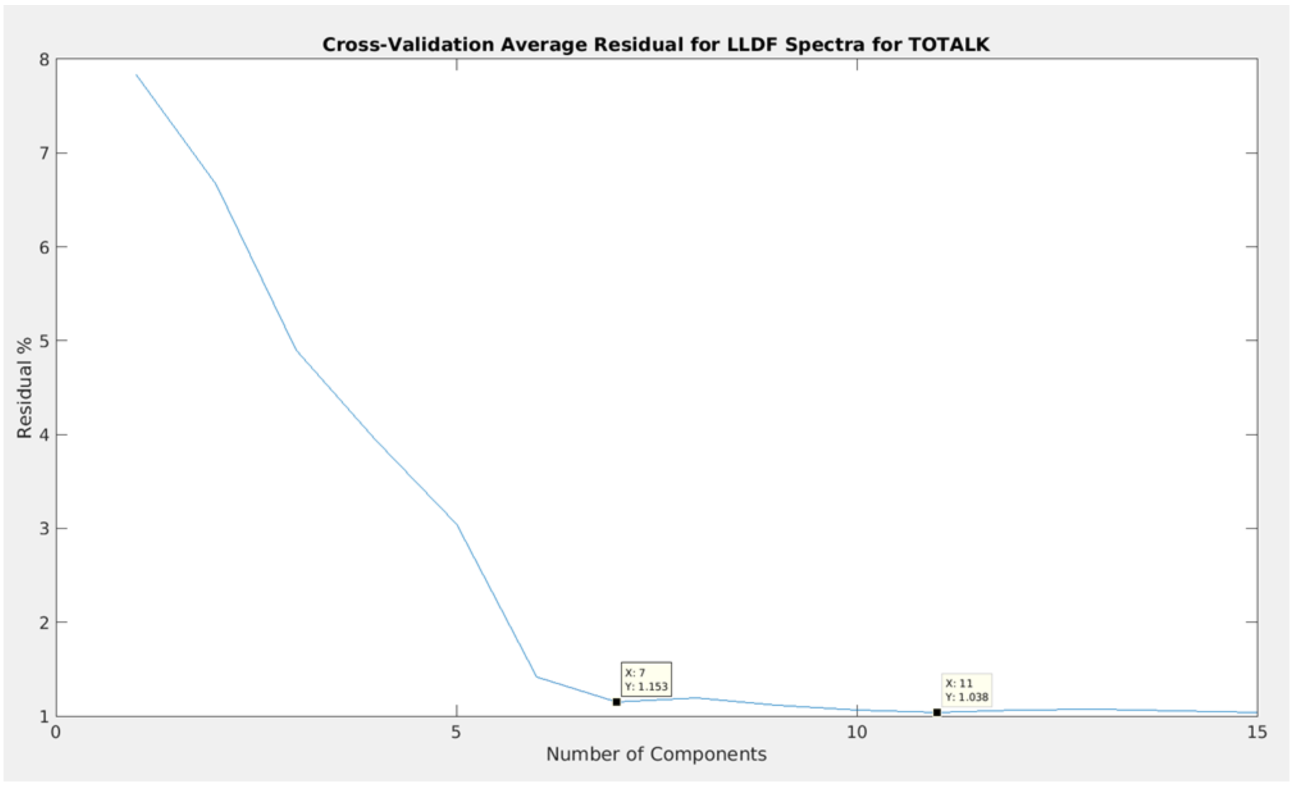

Figure 100: Plot of average residual percentage for a given amount of components for LLDF for TOT_ALK.

As seen, 11 components gives the lowest predicted residual percentage at 1.038%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 101.

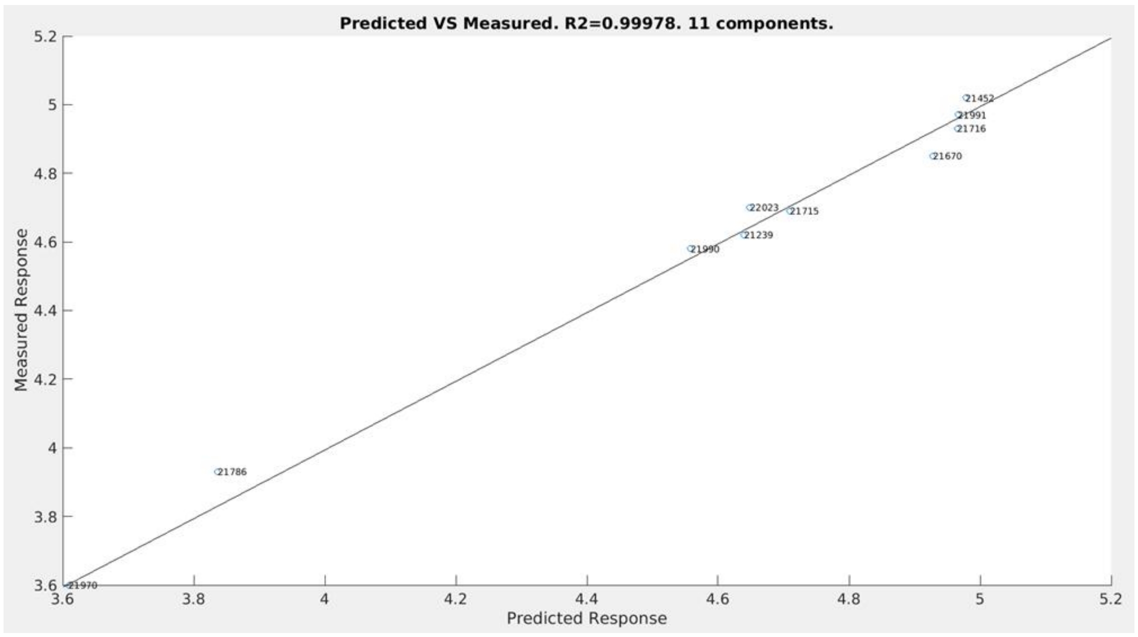

Figure 101: Predicted versus measured response for the LLDF model for TOT_ALK.

With good marks from the cross-validation, the final model, containing 11 components, was created. The predicted results superimposed onto the measured results can be seen in figure 102.

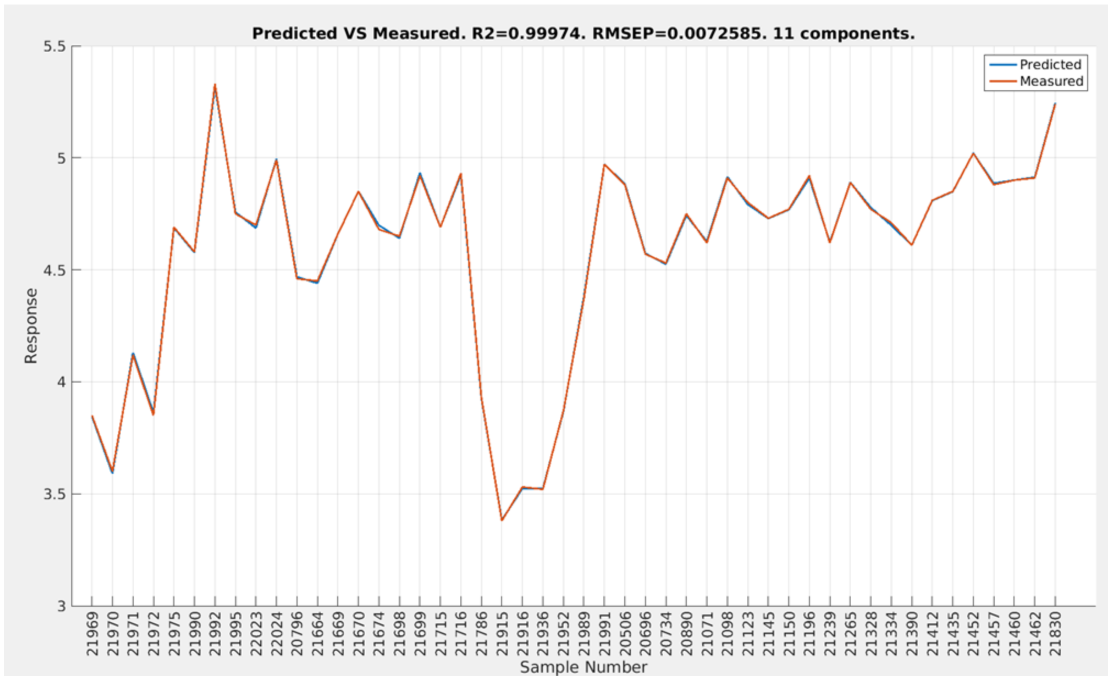

Figure 102: Plot of predicted and measured responses by sample number for LLDF for TOT_ALK.

TIC

After concatenation, the spectra contained 5331 variables, comprised of 1085 from ATR-FTIR, 552 from NIR and 3694 from Raman, as seen in figure 103. The overall spectrum was again dominated in size by the Raman data, which was by far the worst performing dataset, especially for TIC.

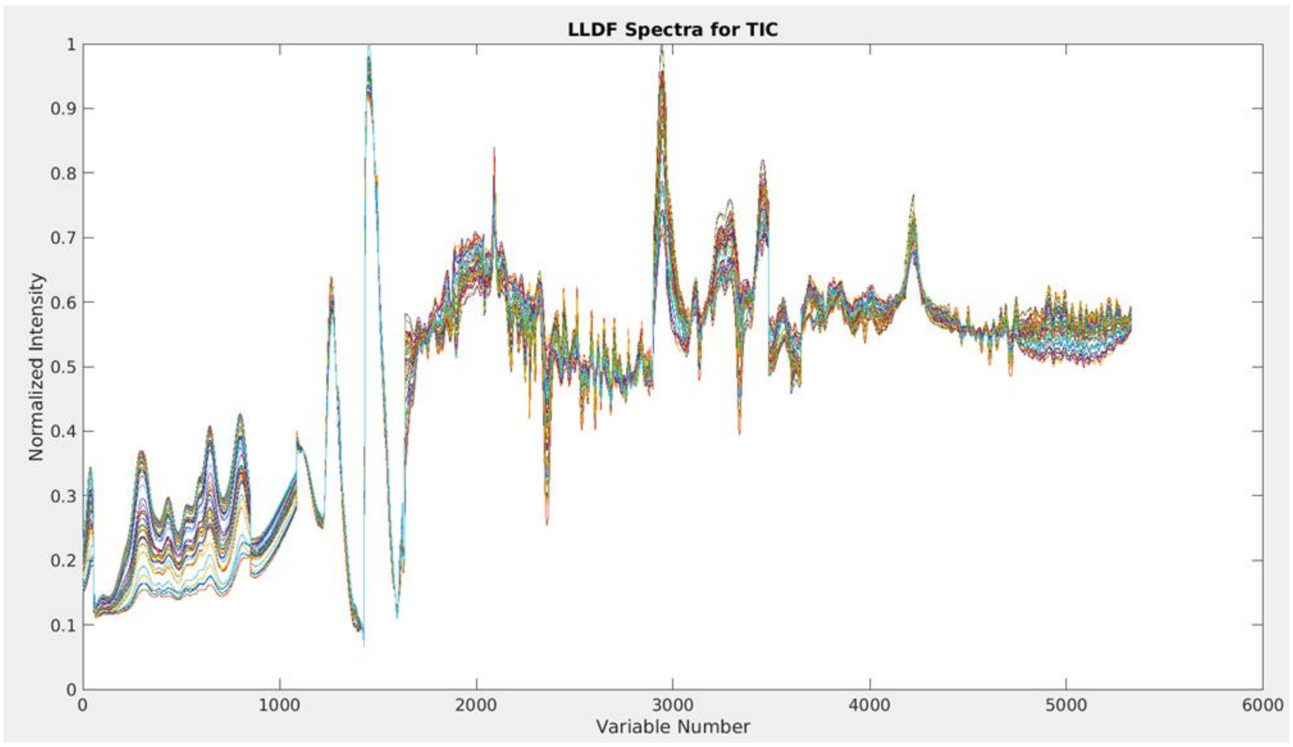

Figure 103: Plot of LLDF's concatenated spectra for all samples for TIC.

The cross-validation results for TIC were very promising. A second run of outlier detection identified samples 20890 and 21123 as outliers. In the end, a final cross-validation was run, showing a minimum RMSECV at 8 components, for a value of 0.02874, as seen in figure 104. This is significantly larger than the 0.01411 that the ATR-FTIR individual model produced.

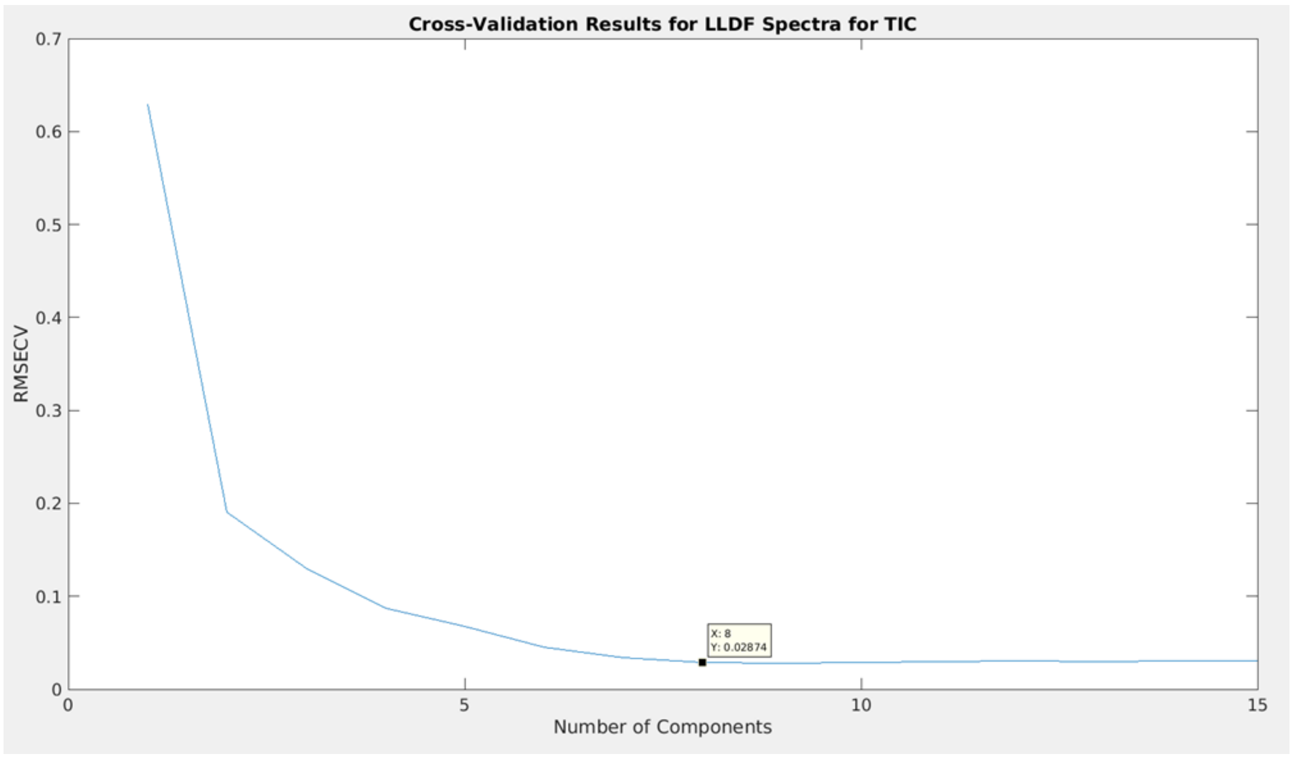

Figure 104: RMSECV versus components used for LLDF for TIC.

Although the suggested number of components is 8, and it is 8 that would be used from this point on, the difference in RMSECV between 5 components and 8 components is relatively minor and should be considered to reduce calculation times if needed. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 105.

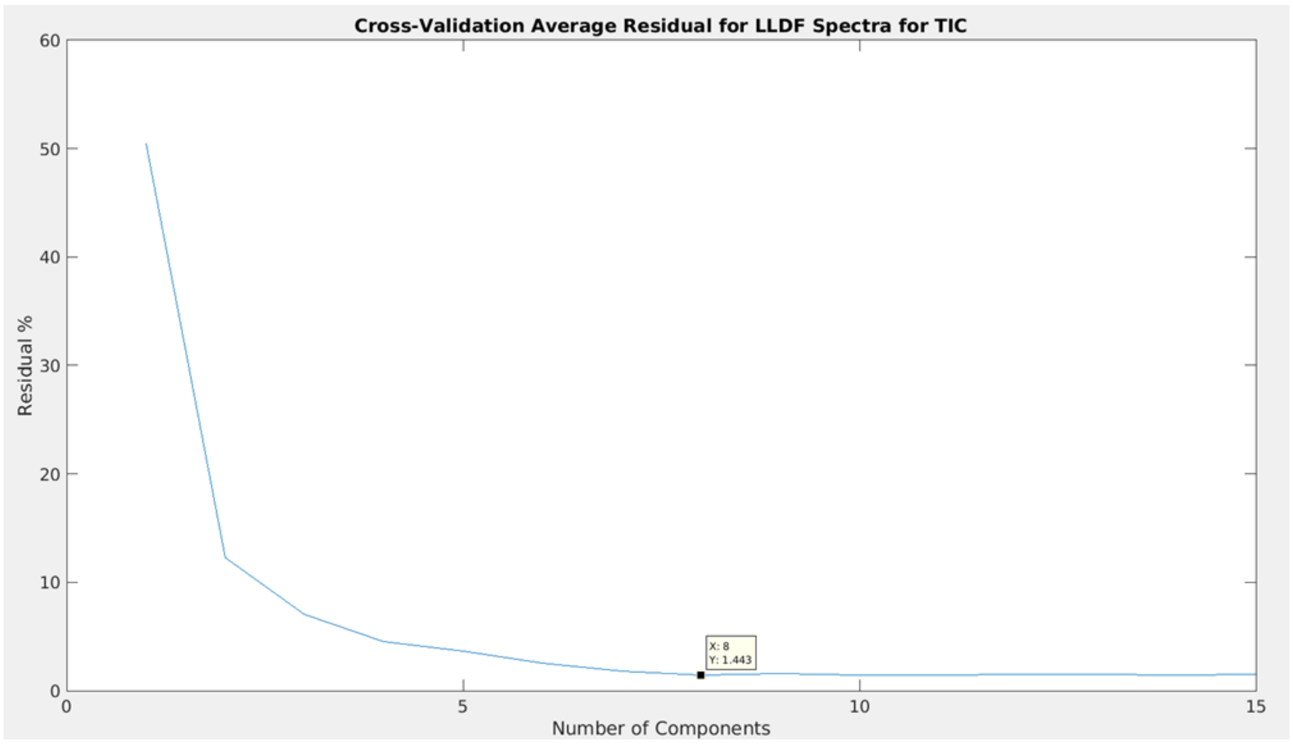

Figure 105: Plot of average residual percentage for a given amount of components for LLDF for TIC.

As seen, 8 components gives the lowest predicted residual percentage at 1.443%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 106.

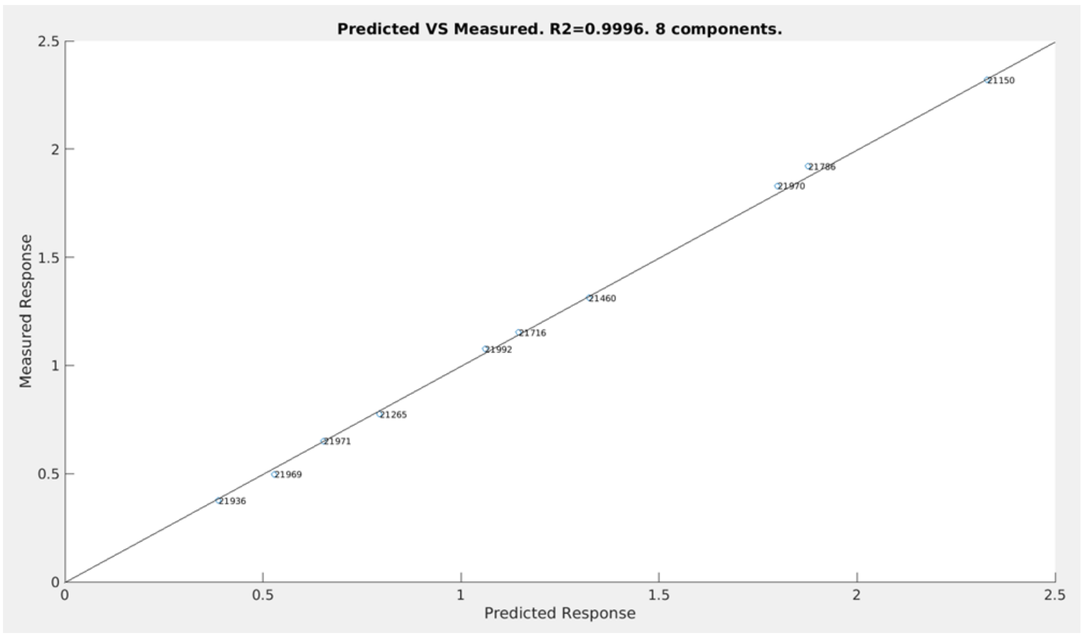

Figure 106: Predicted versus measured response for the LLDF model for TIC.

With good marks from the cross-validation, the final model, containing 8 components, was created. The predicted results superimposed onto the measured results can be seen in figure 107.

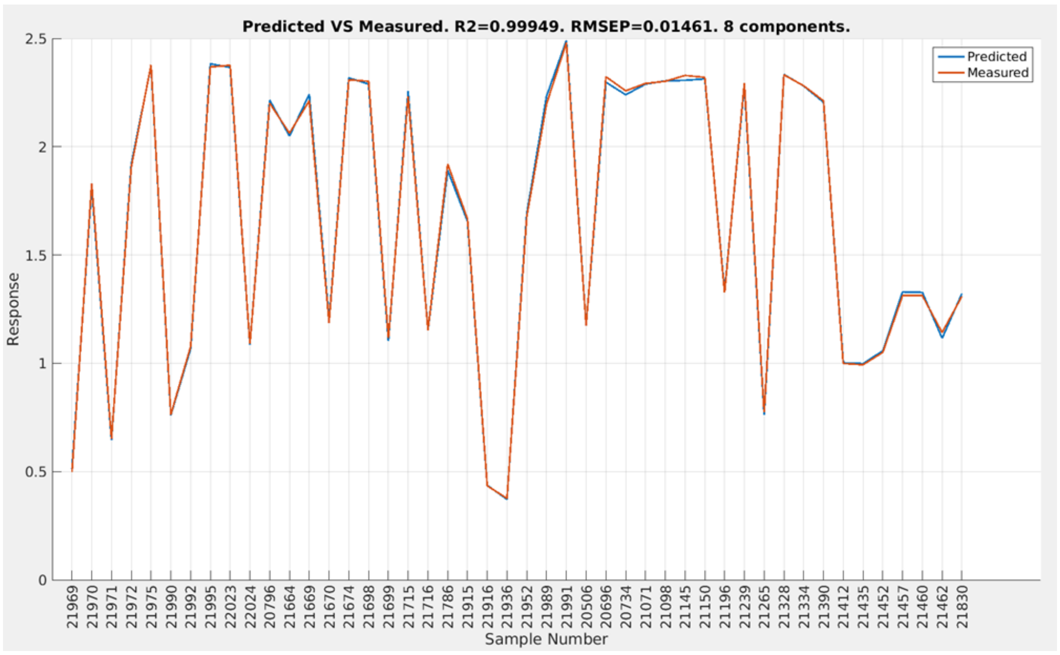

Figure 107: Plot of predicted and measured responses by sample number for LLDF for TIC.

Density

After concatenation, the spectra contained 4377 variables, comprised of 3384 from ATR-FTIR, 503 from NIR and 490 from Raman, as seen in figure 108. The overall spectrum was again dominated in size by the ATR-FTIR data, which was the best performing individual modeling instrument, although not quite as good for density as NIR.

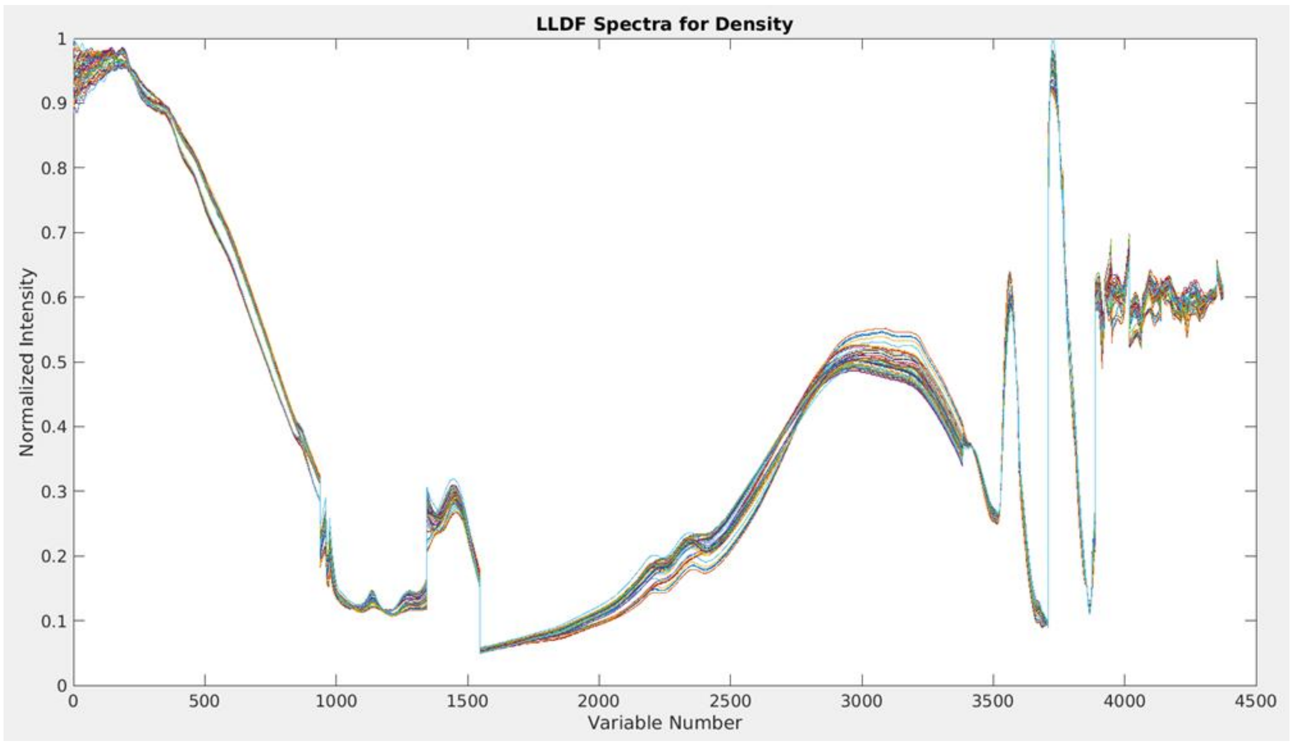

Figure 108: Plot of LLDF's concatenated spectra for all samples for density.

The cross-validation results for density were very promising. A second run of outlier detection identified samples 20890 and 20734 as outliers. In the end, a final cross-validation was run, showing a minimum RMSECV at 7 components, for a value of 0.9925, as seen in figure 109. This is larger than the 0.6091 that the NIR individual model produced, but somewhat similar in size to the 0.8913 that the ATR-FTIR model produced.

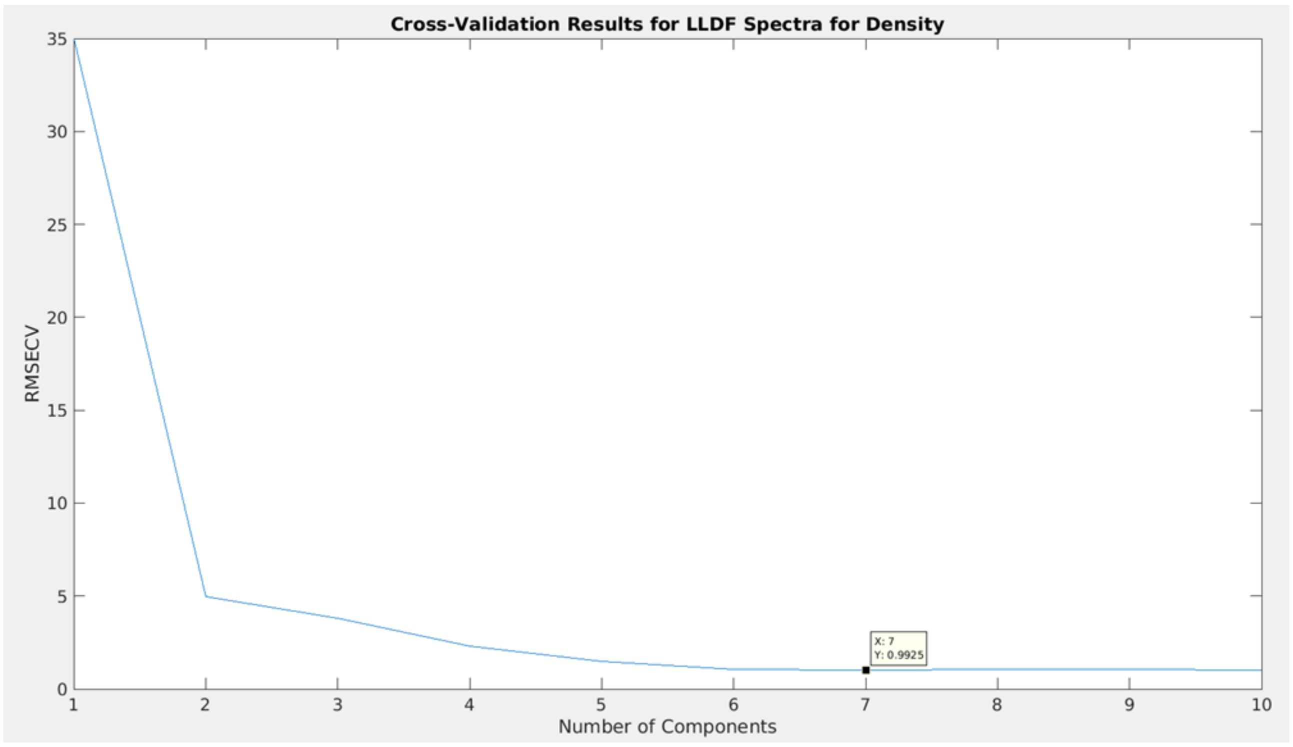

Figure 109: RMSECV versus components used for LLDF for density.

Although the suggested number of components is 7, and it is 7 that would be used from this point on, the difference in RMSECV between 5, 4 or even 3 components and 7 components is relatively minor and should be considered to reduce calculation times if needed. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 110.

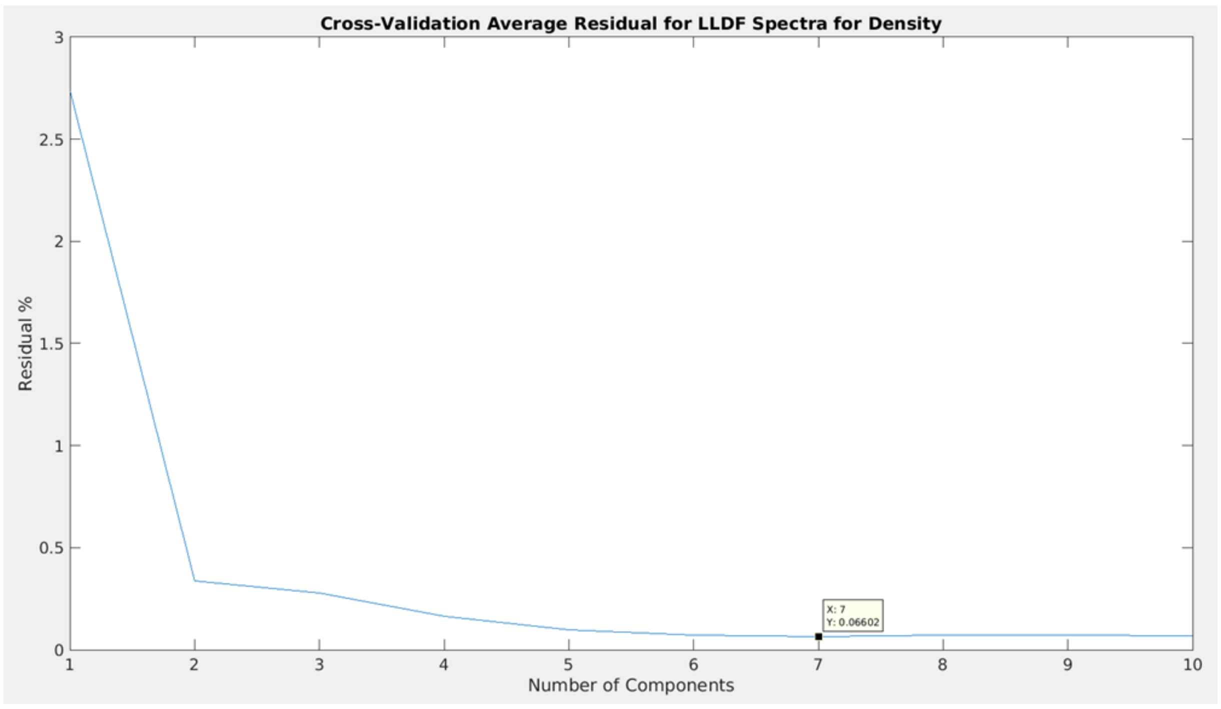

Figure 110: Plot of average residual percentage for a given amount of components for LLDF for density.

As seen, 7 components gives the lowest predicted residual percentage at 0.06602%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 111.

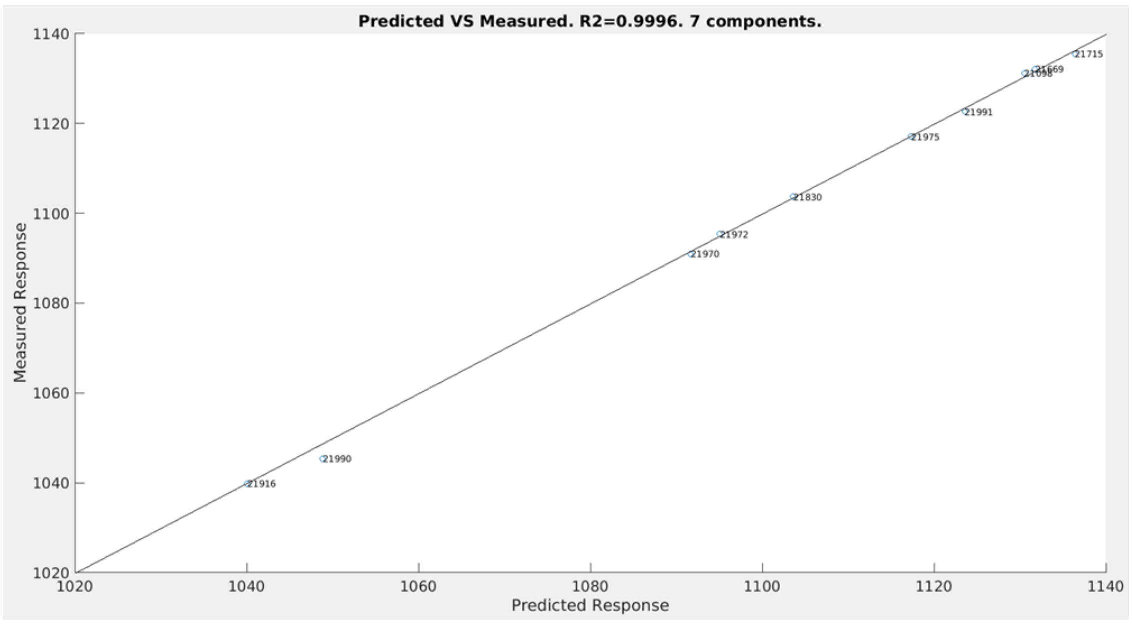

Figure 111: Predicted versus measured response for the LLDF model for density.

With good marks from the cross-validation, the final model, containing 7 components, was created. The predicted results superimposed onto the measured results can be seen in figure 112.

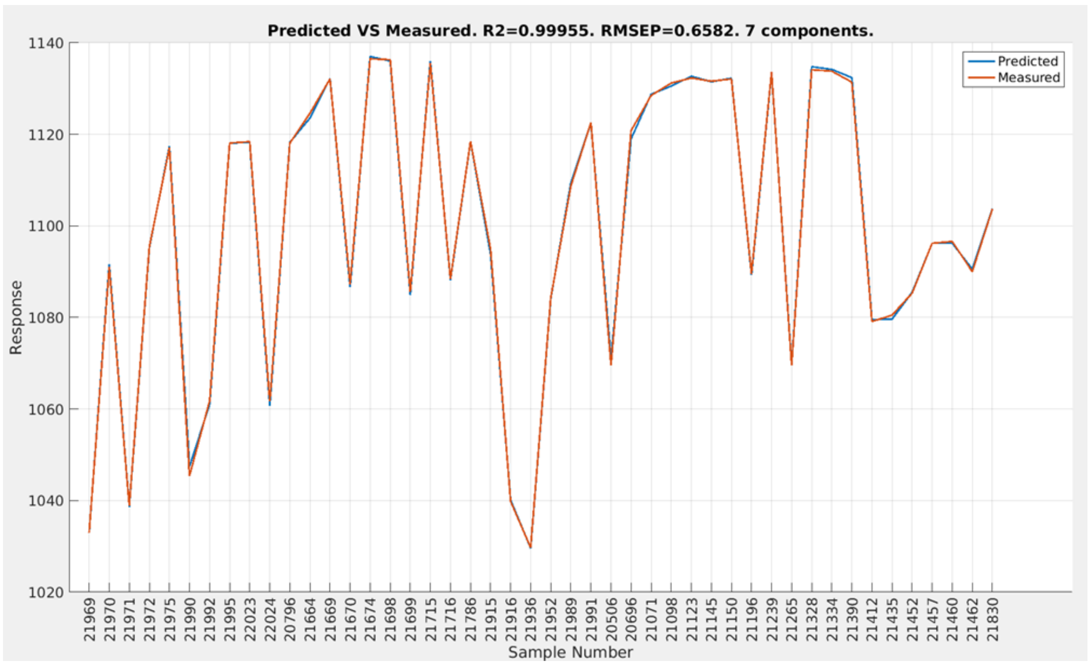

Figure 112: Plot of predicted and measured responses by sample number for LLDF for density.

In every instance, the LLDF model including all three instruments produced worse results than the best individual models.

6.5.2 Fusion of ATR-FTIR and NIR.

TOT_ALK

After concatenation, the spectra contained 1835 variables, comprised of 1275 from ATR-FTIR and 560 from NIR, as seen in figure 113. The overall spectrum was biased towards variables from ATR-FTIR, the best overall performing instrument.

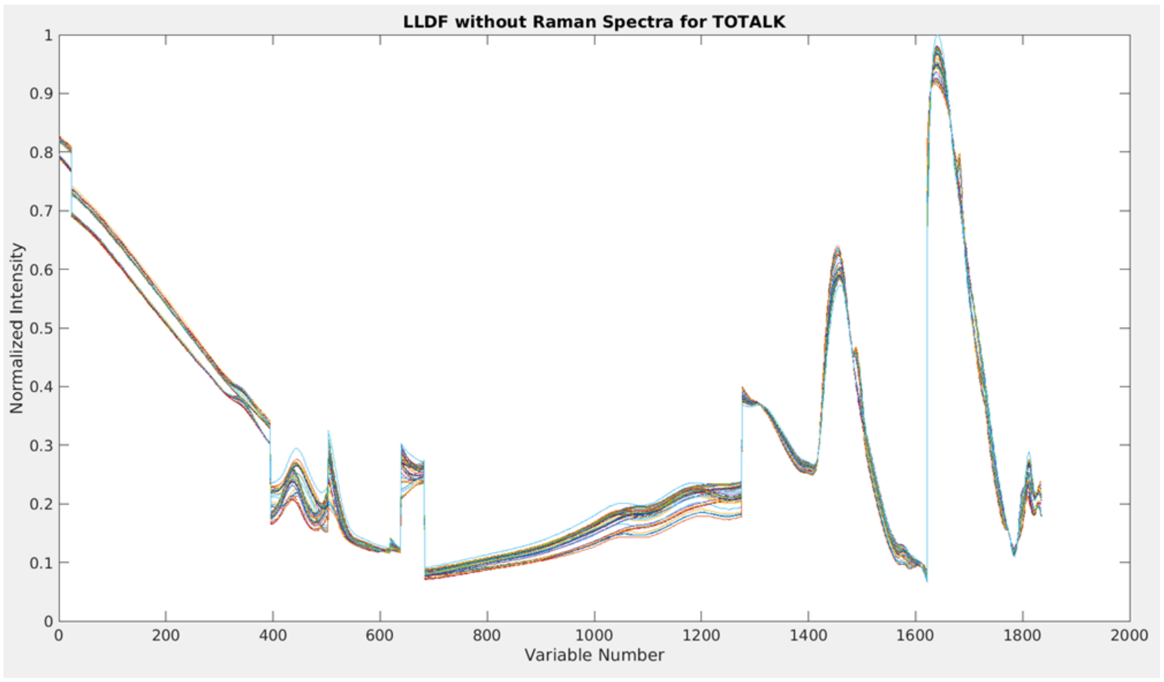

Figure 113: Plot of LLDF's concatenated spectra for all samples without Raman for TOT_ALK.

The cross-validation results for TOT_ALK were very promising. A second run of outlier detection identified 20696, 21098 and 21390 as outliers. In the end, a final cross-validation was run, showing a minimum RMSECV at 12 components, for a value of 0.03094, as seen in figure 114. This is marginally better than the 0.03343 that the ATR-FTIR individual model produced.

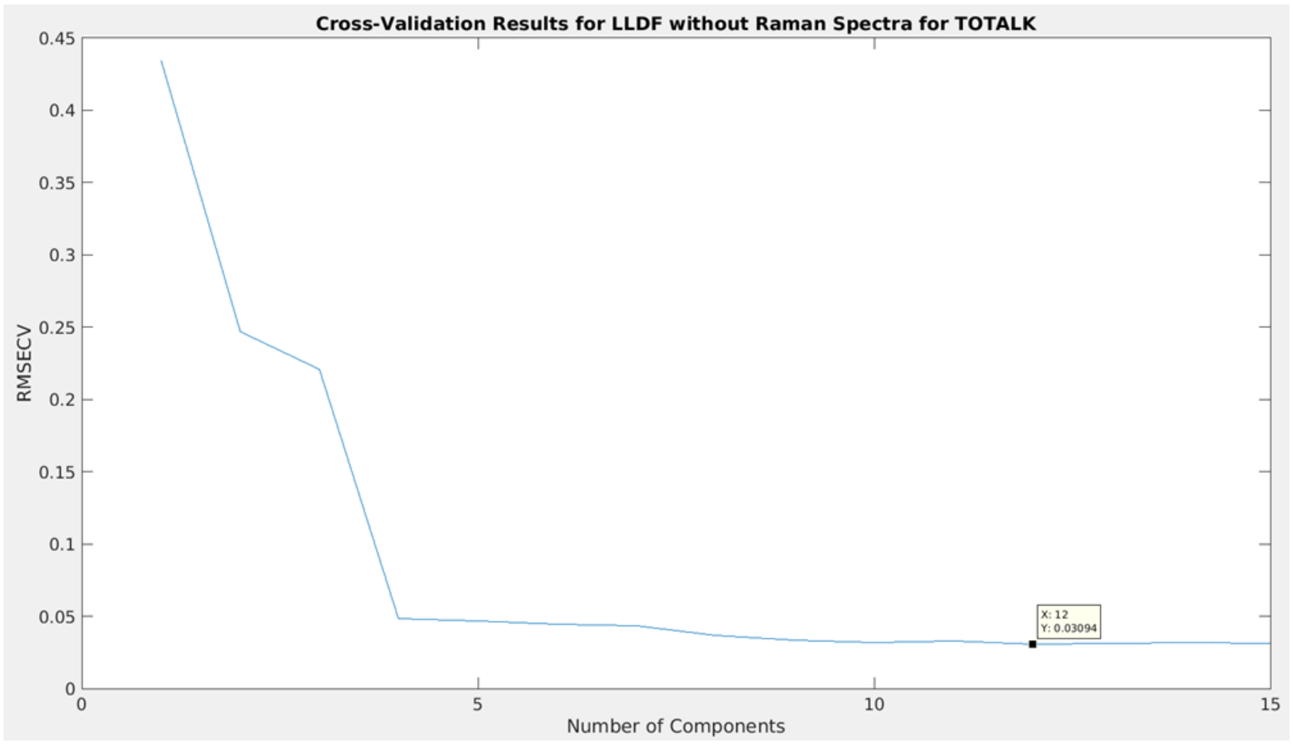

Figure 114: RMSECV versus components used for LLDF without Raman for TOT_ALK.

Although the suggested number of components is 12, and it is 12 that would be used from this point on, the difference in RMSECV between 10 or even 4 components and 12 components is relatively minor and should be considered to reduce calculation times if needed. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 115.

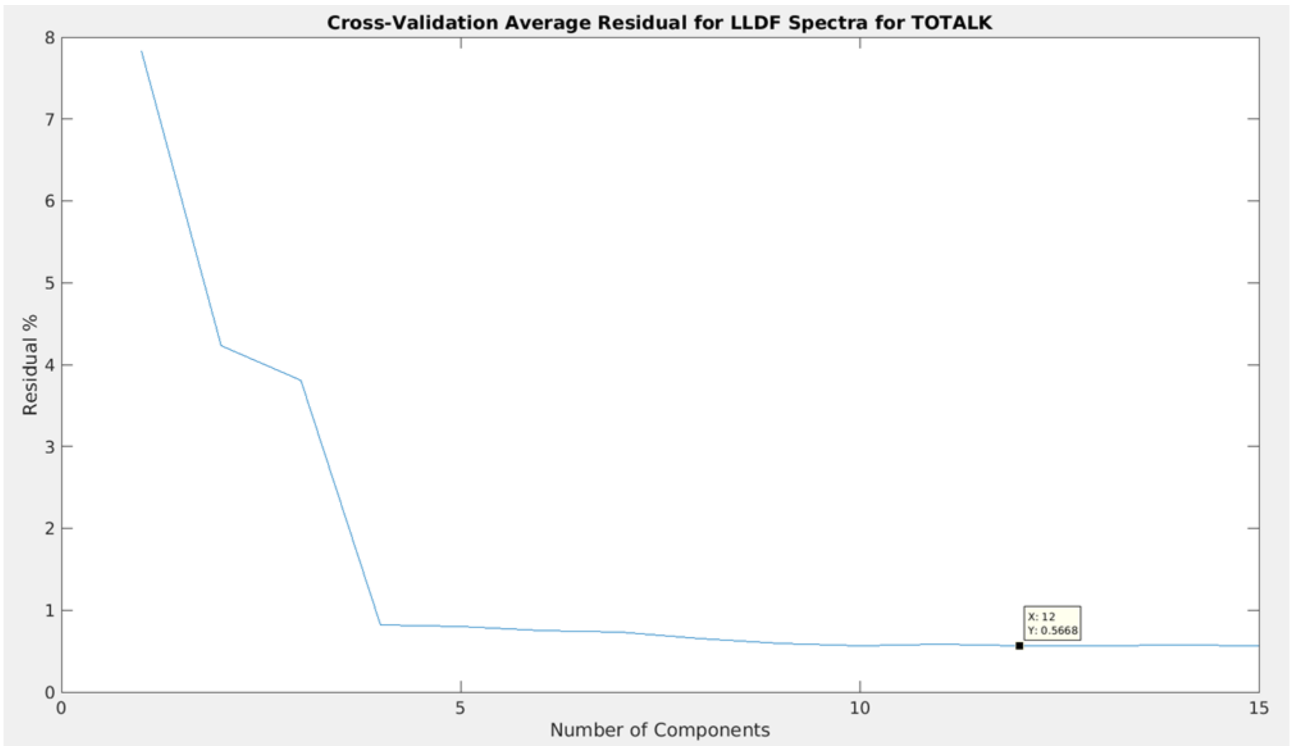

Figure 115: Plot of average residual percentage for a given amount of components for LLDF without Raman for TOT_ALK.

As seen, 12 components gives the lowest predicted residual percentage at 0.5668%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 116.

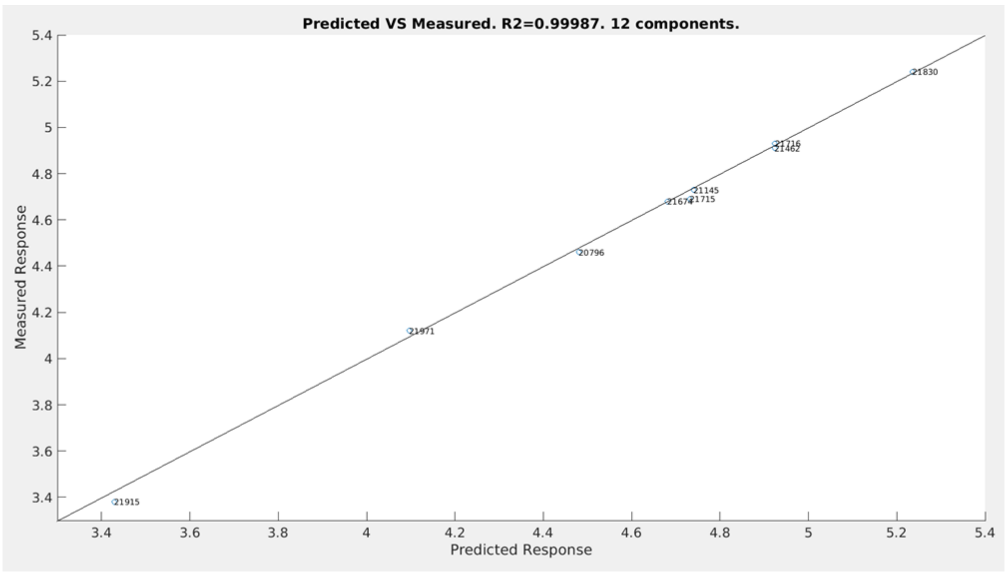

Figure 116: Predicted versus measured response for the LLDF without Raman model for TOT_ALK.

With good marks from the cross-validation, the final model, containing 12 components, was created. The predicted results superimposed onto the measured results can be seen in figure 117.

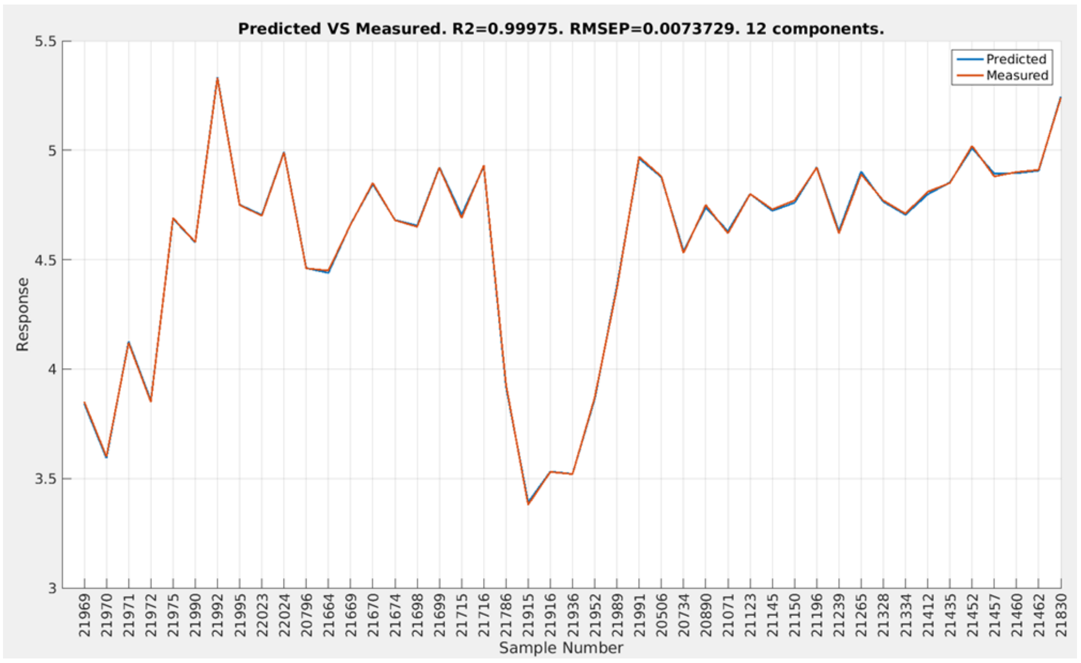

Figure 117: Plot of predicted and measured responses by sample number for LLDF without Raman for TOT_ALK.

TIC

After concatenation, the spectra contained 1637 variables, comprised of 1085 from ATR-FTIR and 552 from NIR, as seen in figure 118. The overall spectrum was again biased towards variables from ATR-FTIR, the best overall performing instrument.

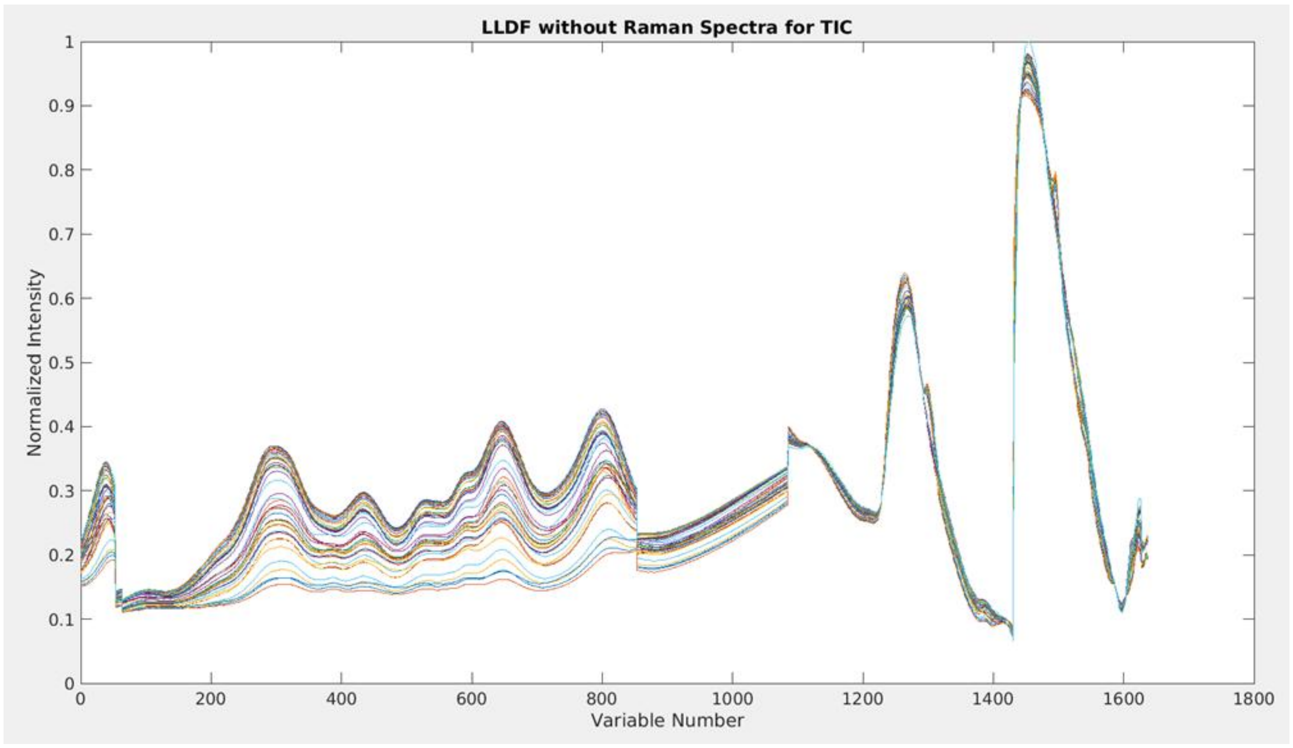

Figure 118: Plot of LLDF's concatenated spectra for all samples without Raman for TIC.

The cross-validation results for TIC were very promising. A second run of outlier detection identified 20796, 20890, 21696 and 21334 as outliers. In the end, a final cross-validation was run, showing a minimum RMSECV at 8 components, for a value of 0.01507, as seen in figure 119. This is marginally worse than the 0.01411 that the ATR-FTIR individual model produced.

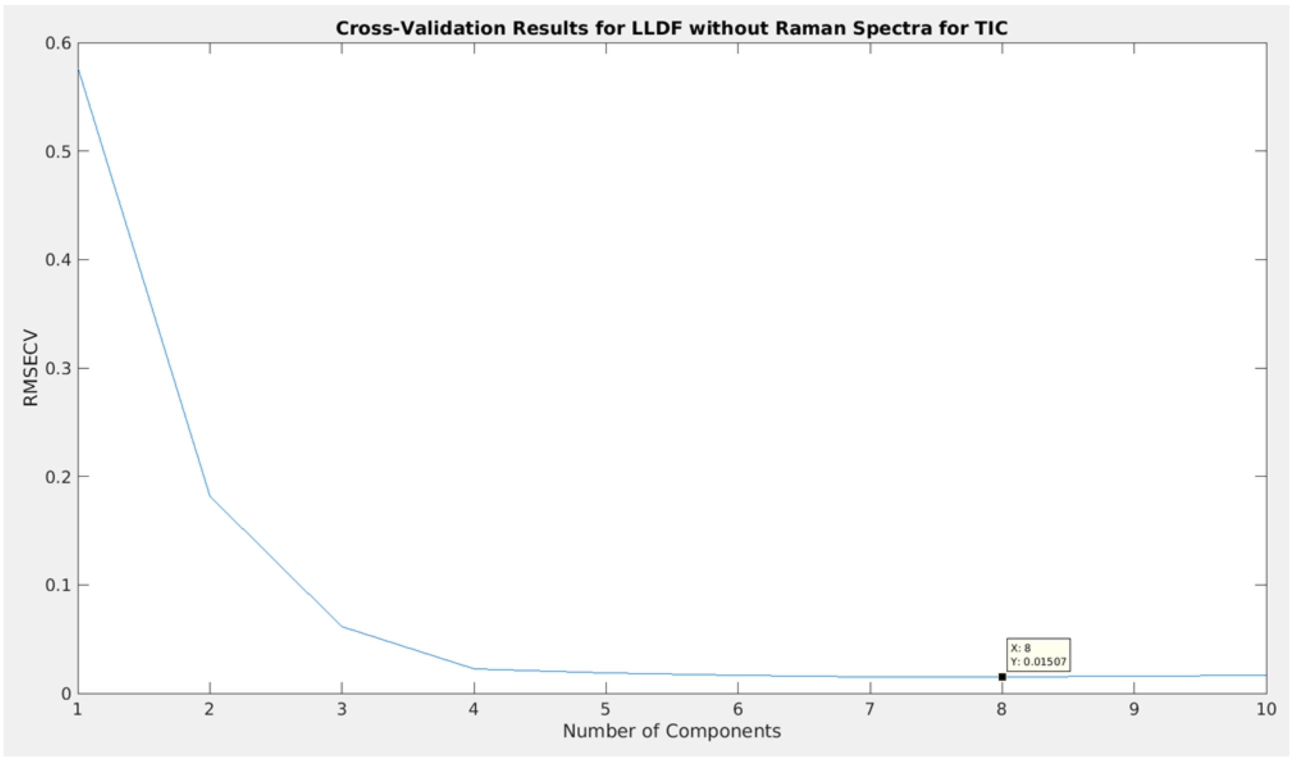

Figure 119: RMSECV versus components used for LLDF without Raman for TIC.

Although the suggested number of components is 8, and it is 8 that would be used from this point on, the difference in RMSECV between 4 components and 8 components is relatively minor and should be considered to reduce calculation times if needed. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 120.

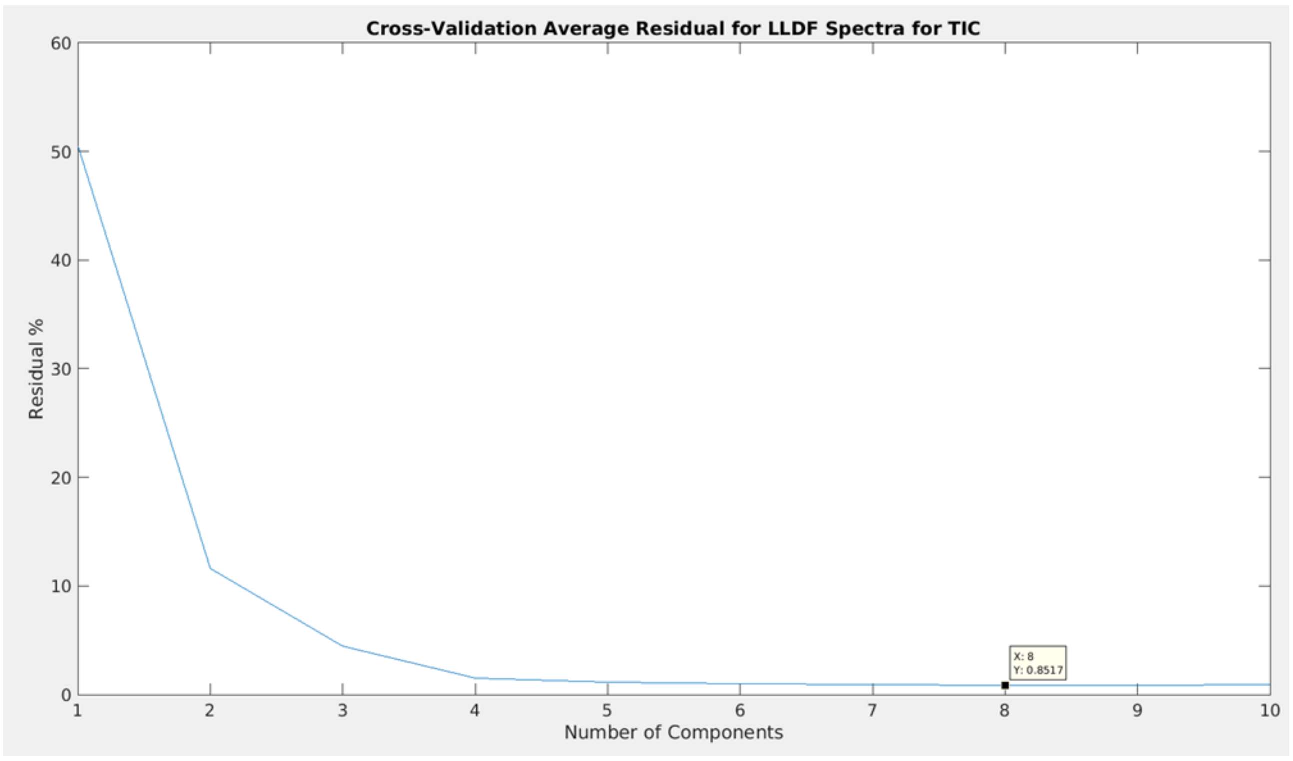

Figure 120: Plot of average residual percentage for a given amount of components for LLDF without Raman for TIC.

As seen, 8 components gives the lowest true residual percentage at 0.8517%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 121.

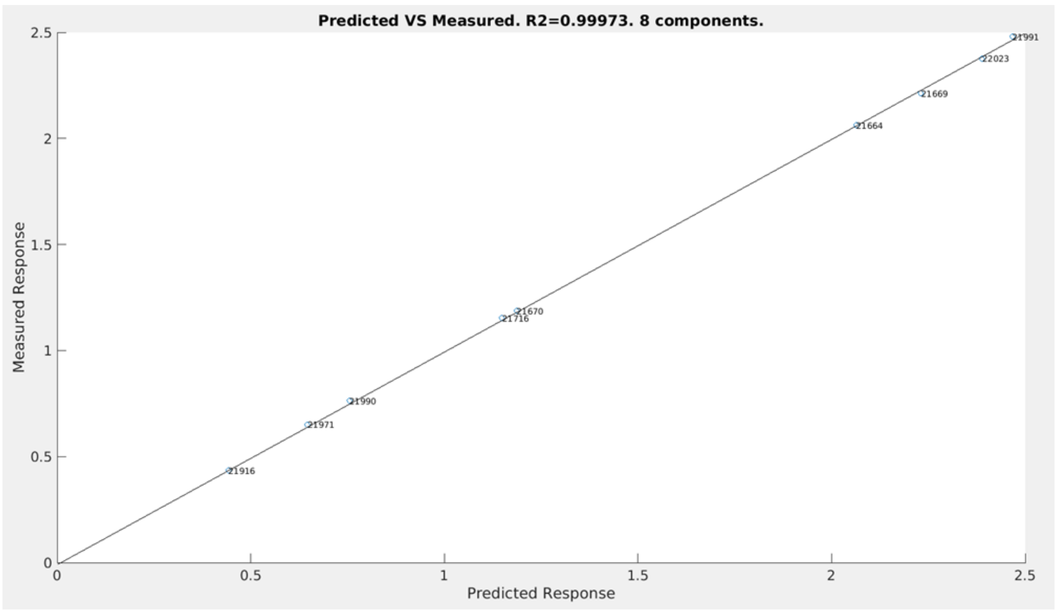

Figure 121: Predicted versus measured response for the LLDF without Raman model for TIC.

With good marks from the cross-validation, the final model, containing 8 components, was created. The predicted results superimposed onto the measured results can be seen in figure 122.

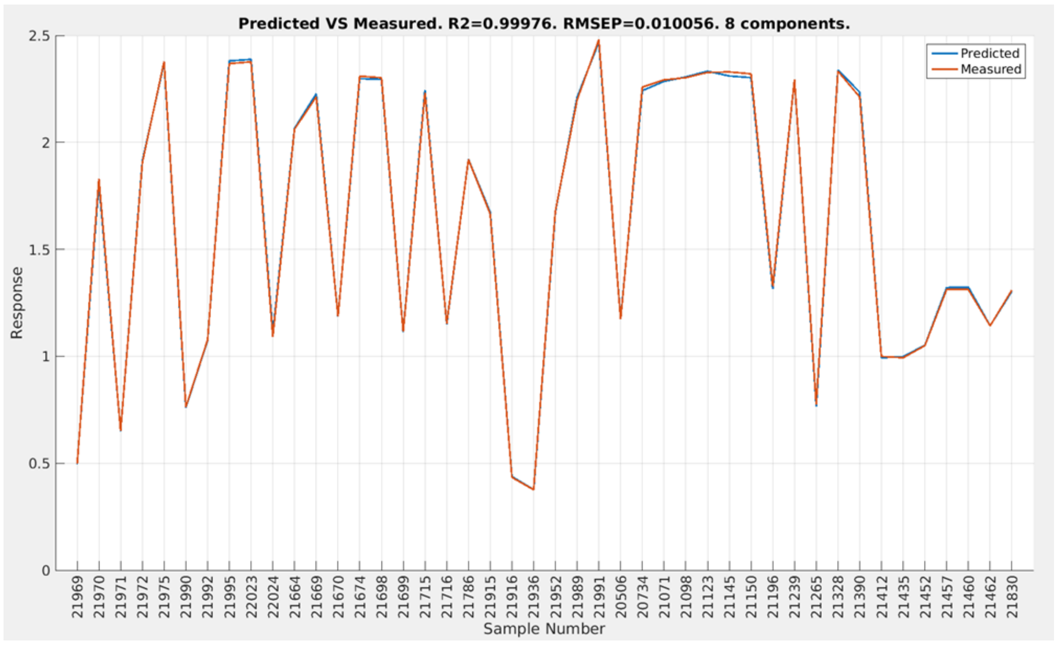

Figure 122: Plot of predicted and measured responses by sample number for LLDF without Raman for TIC.

Density

After concatenation, the spectra contained 3887 variables, comprised of 3384 from ATR-FTIR and 503 from NIR, as seen in figure 123. The overall spectrum was heavily dominated by variables from ATR-FTIR, although it was NIR that provided the best result individually.

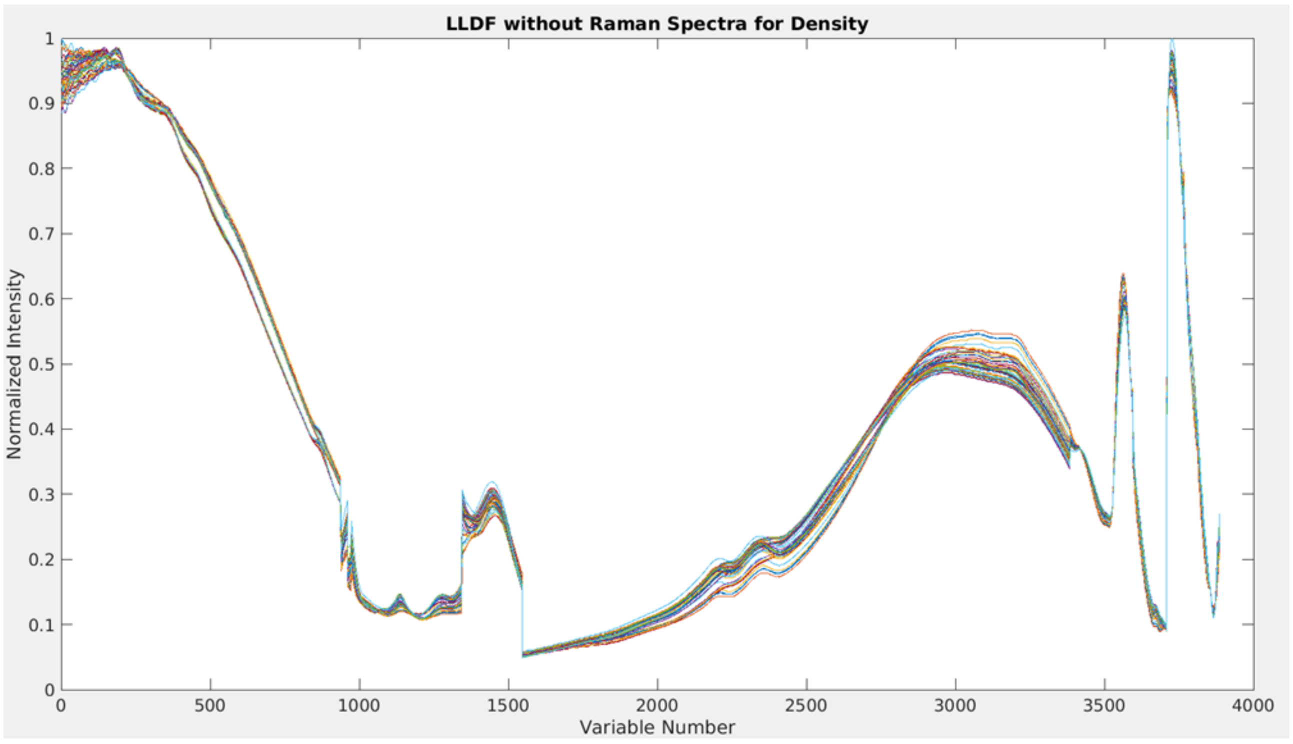

Figure 123: Plot of LLDF's concatenated spectra for all samples without Raman for density.

The cross-validation results for density were very promising. A second run of outlier detection was unable to find any potential outliers. In the end, a final cross-validation was run, showing a minimum RMSECV at 13 components, for a value of 0.6764, as seen in figure 124. This is somewhat similar to the 0.6091 that the NIR individual model produced.

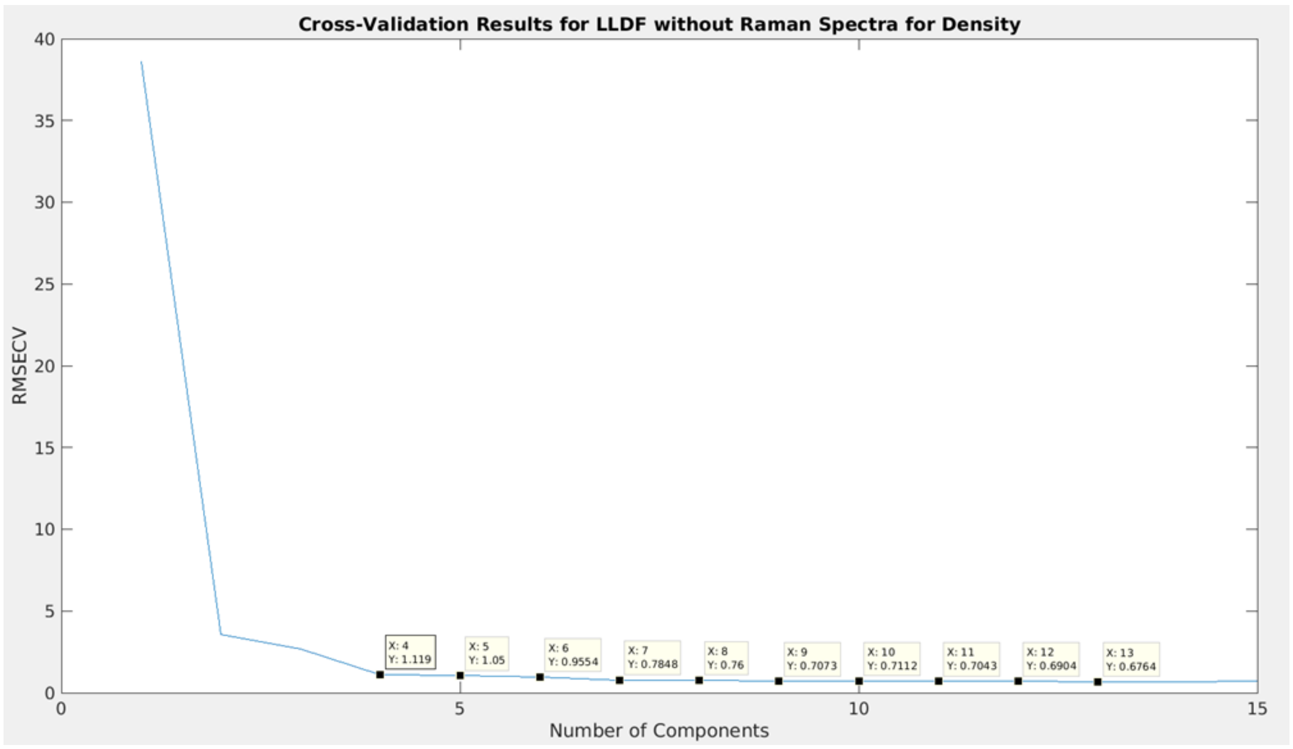

Figure 124: RMSECV versus components used for LLDF without Raman for density.

Although the suggested number of components is 13, and it is 13 that would be used from this point on, the difference in RMSECV between 9 components and 13 components is negligible and should be considered to reduce calculation times if needed. 4 components is still very accurate as well, and would significantly decrease data processing amounts. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 125.

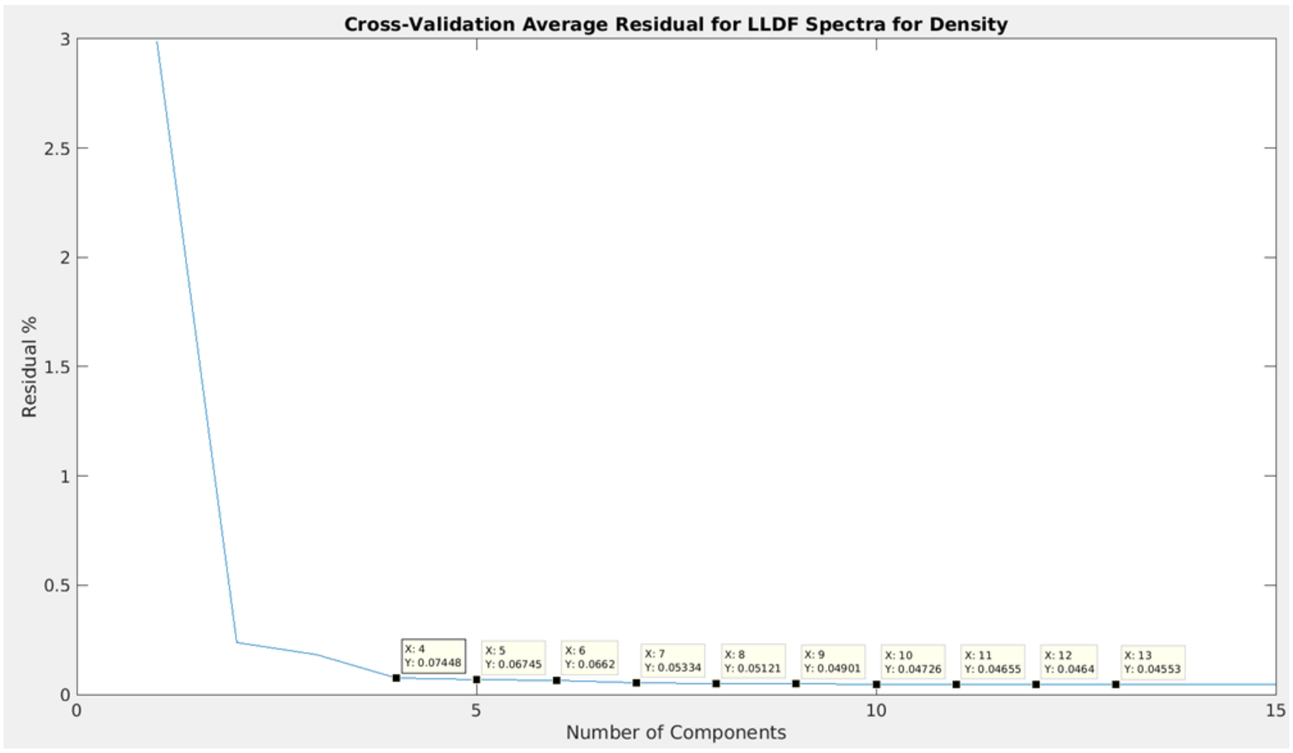

Figure 125: Plot of average residual percentage for a given amount of components for LLDF without Raman for density.

As seen, 13 components gives the lowest true residual percentage at 0.04553%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 126.

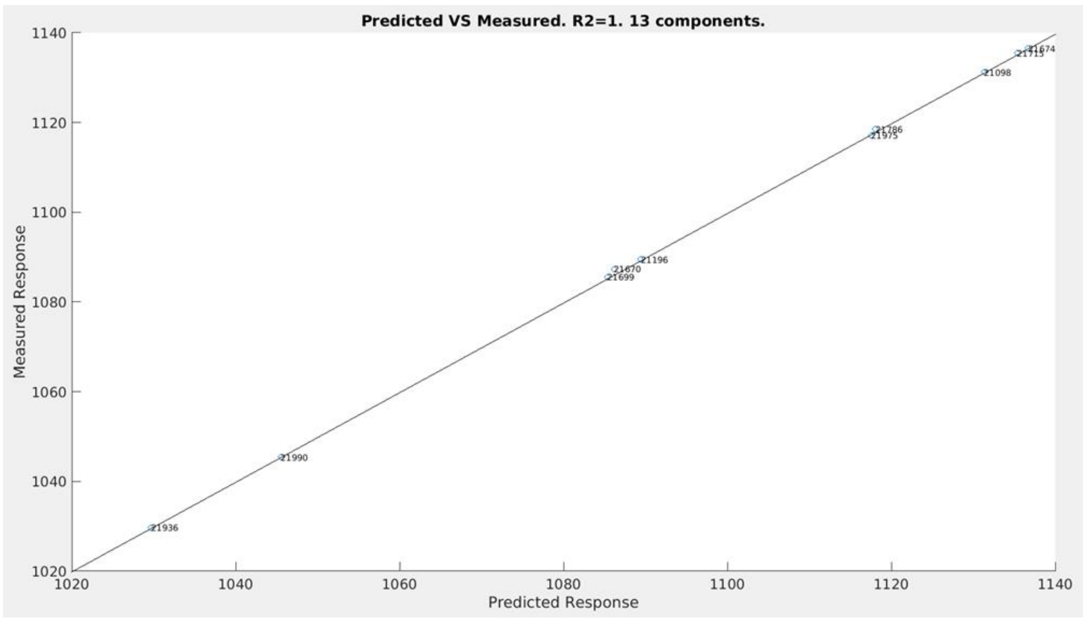

Figure 126: Predicted versus measured response for the LLDF without Raman model for density.

With excellent marks from the cross-validation, the final model, containing 13 components, was created. The predicted results superimposed onto the measured results can be seen in figure 127.

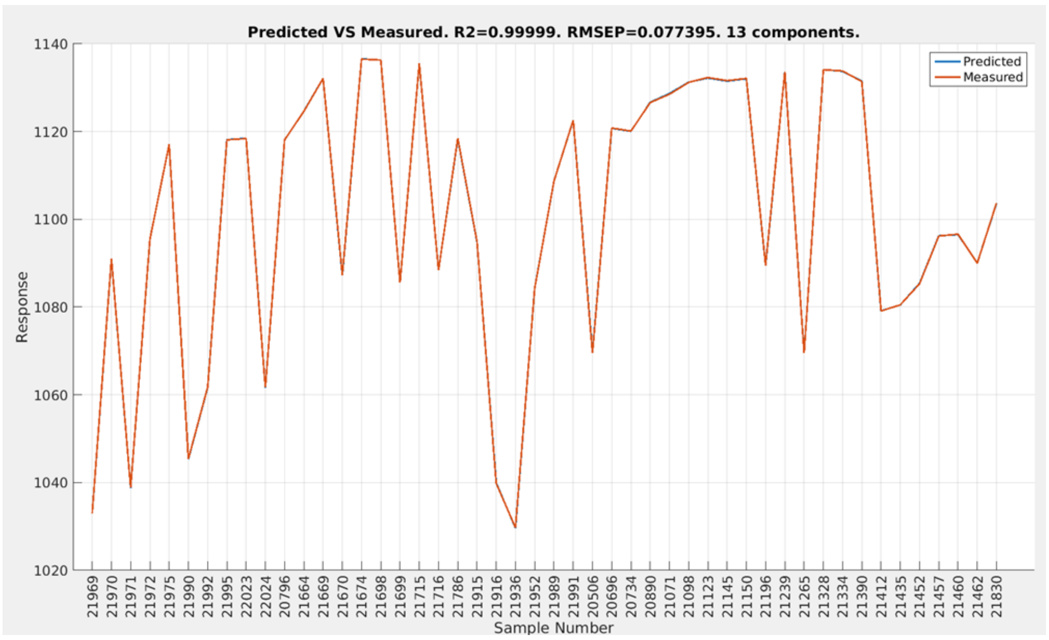

Figure 127: Plot of predicted and measured responses by sample number for LLDF without Raman for density.

In every instance, the LLDF model without Raman produced results approximately as good as, or better than, the individual models alone.

6.5.3 Mid-Level Data Fusion

Due to Raman’s very poor performance overall, two sets of MLDF sets were made. One containing all three instrument measurements as mentioned above, and one containing only ATR-FTIR and NIR. However, they both gave near identical results, within a very small margin of error, so only the full MLDF set will be shown.

TOT_ALK

The cross-validation results for TOT_ALK were very promising. A second run of outlier detection identified samples 21991, 21098 and 21071 as potential outliers, and they were removed. In the end, a final cross-validation was run, showing a minimum RMSECV at 9 components, for a value of 0.03356, as seen in figure 128. This is almost identical to the 0.03343 that the ATR-FTIR individual model produced, and higher than the 0.03094 that the LLDF without Raman produced.

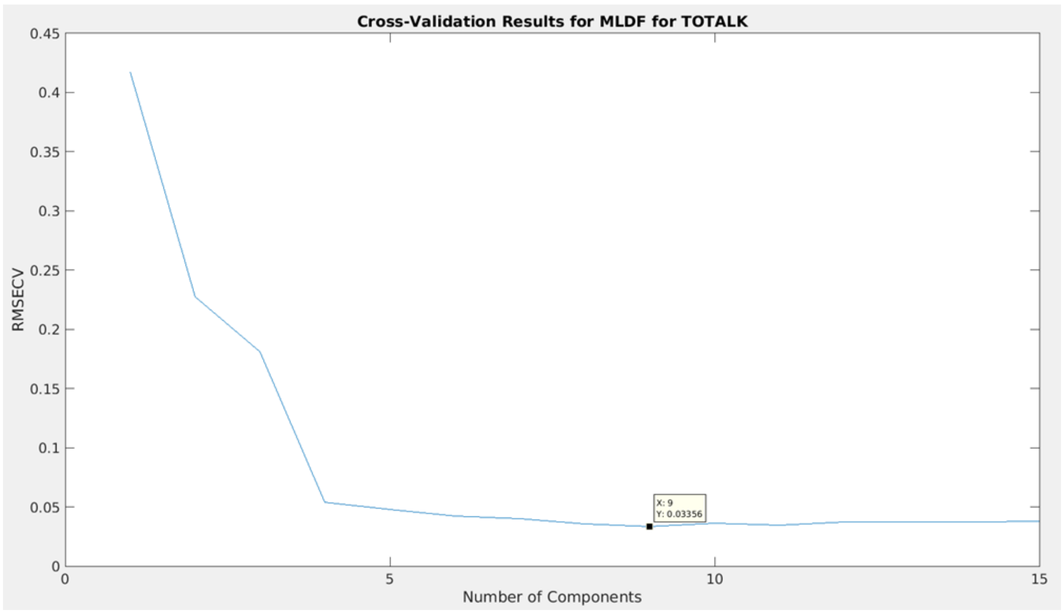

Figure 128: RMSECV versus components used for MLDF for TOT_ALK.

Although the suggested number of components is 9, and it is 9 that would be used from this point on, the difference in RMSECV between 7 components and 9 components is negligible and should be considered to reduce calculation times if needed. 4 components is still very accurate as well, and would significantly decrease data processing amounts. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 129.

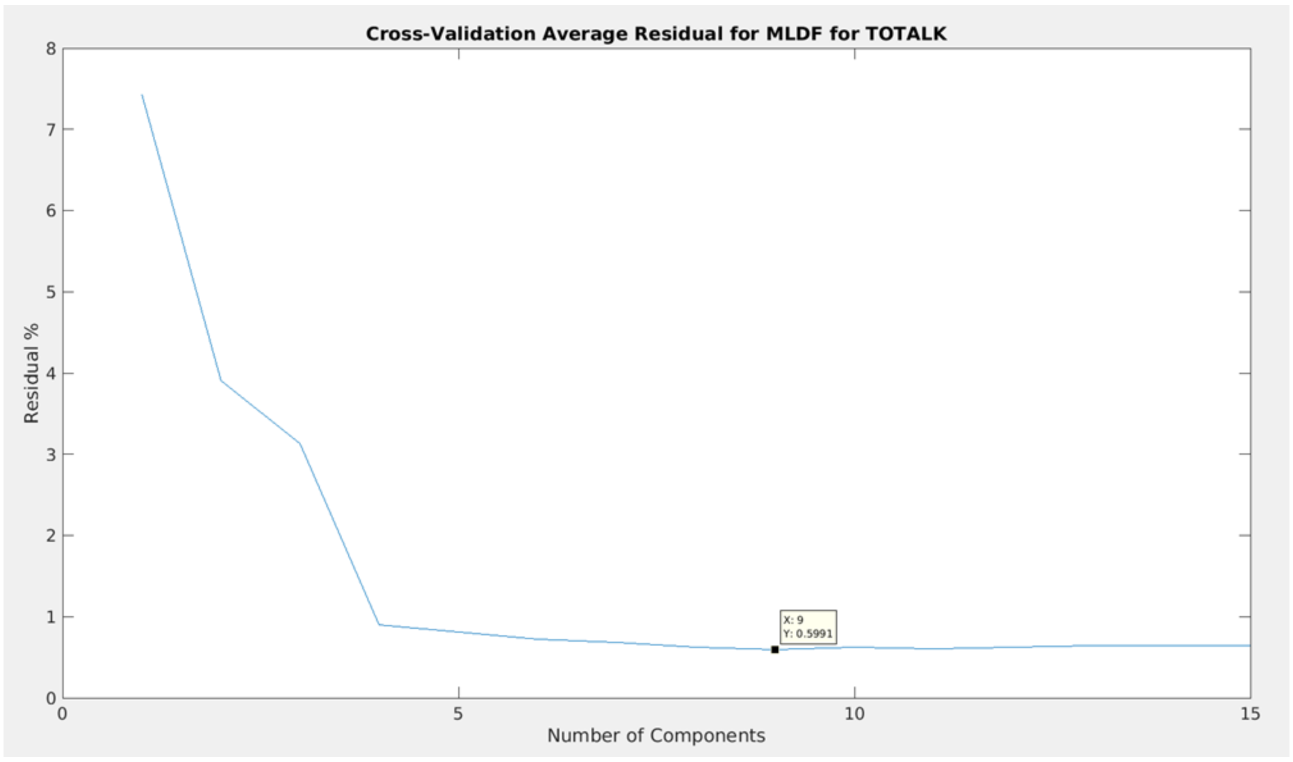

Figure 129: Plot of average residual percentage for a given amount of components for MLDF for TOT_ALK.

As seen, 9 components gives the lowest predicted residual percentage at 0.5991%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 130.

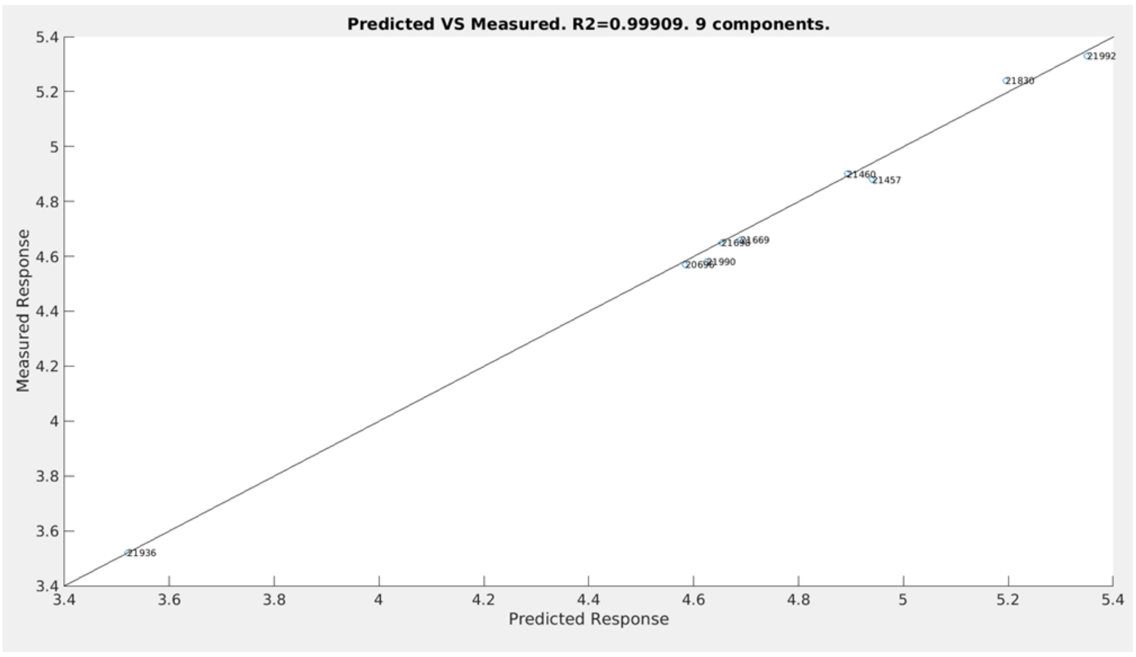

Figure 130: Predicted versus measured response for the MLDF model for TOT_ALK.

With good marks from the cross-validation, the final model, containing 9 components, was created. The predicted results superimposed onto the measured results can be seen in figure 131.

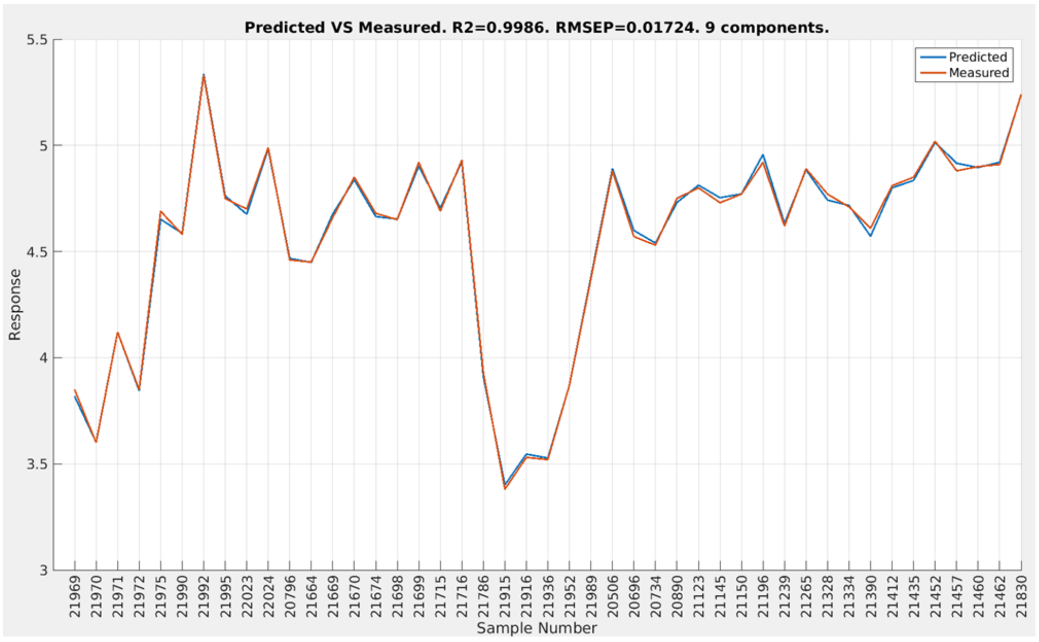

Figure 131: Plot of predicted and measured responses by sample number for MLDF for TOT_ALK.

TIC

The cross-validation results for TIC were very promising. A second run of outlier detection identified samples 20890, 20696 and 21145 as potential outliers, and they were removed. In the end, a final cross-validation was run, showing a minimum RMSECV at 8 components, for a value of 0.01484, as seen in figure 132. This is marginally higher than the 0.01411 that the ATR-FTIR individual model produced, and almost identical to the 0.01507 that the LLDF without Raman produced.

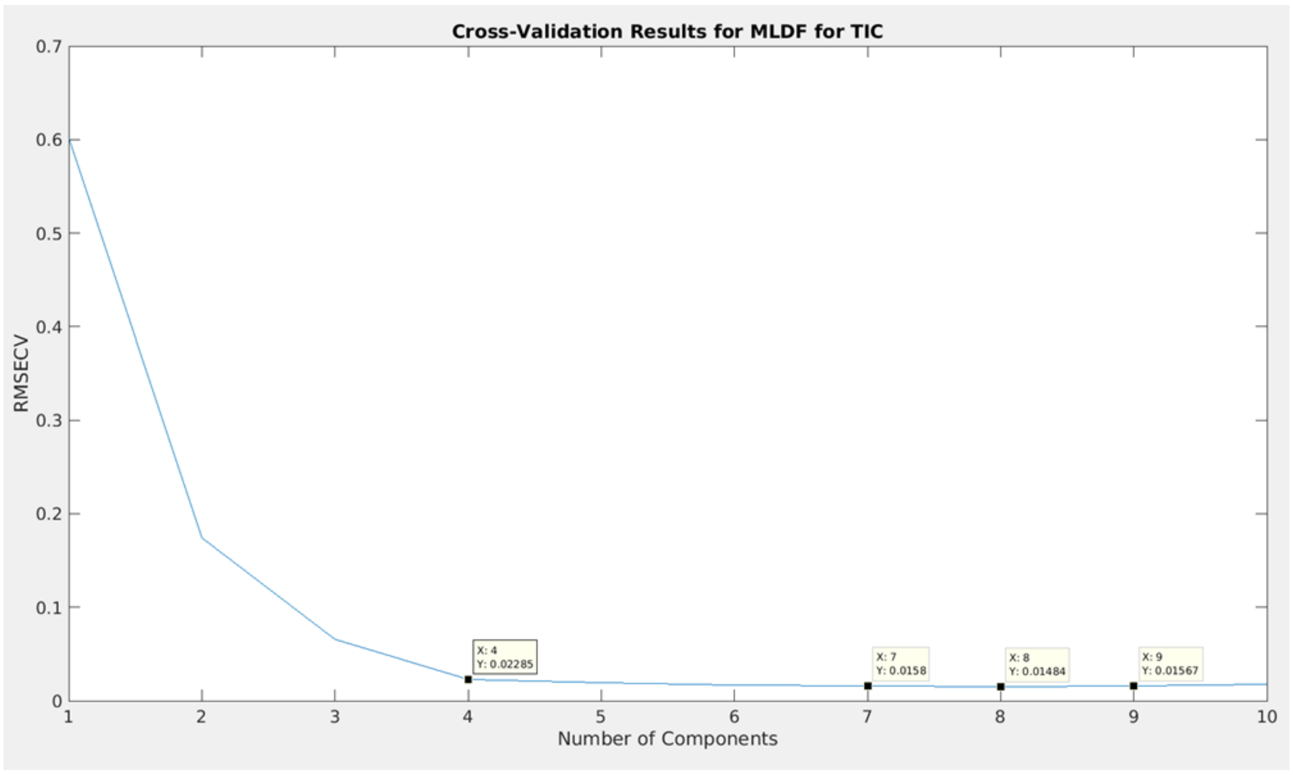

Figure 132: RMSECV versus components used for MLDF for TIC.

Although the suggested number of components is 8, and it is 8 that would be used from this point on, the difference in RMSECV between 5 components and 8 components is negligible and should be considered to reduce calculation times if needed. 4 components is still very accurate as well, and would significantly decrease data processing amounts. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 133.

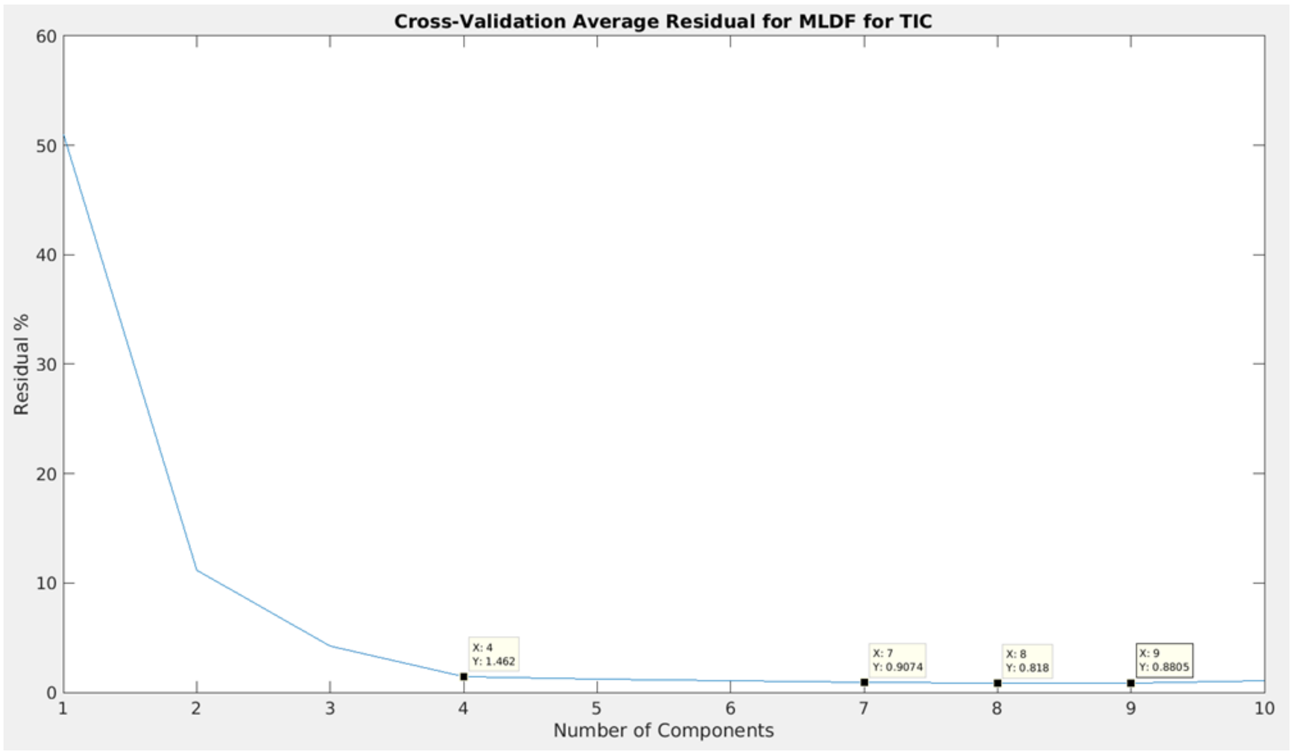

Figure 133: Plot of average residual percentage for a given amount of components for MLDF for TIC.

As seen, 8 components gives the lowest true residual percentage at 0.818%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 134.

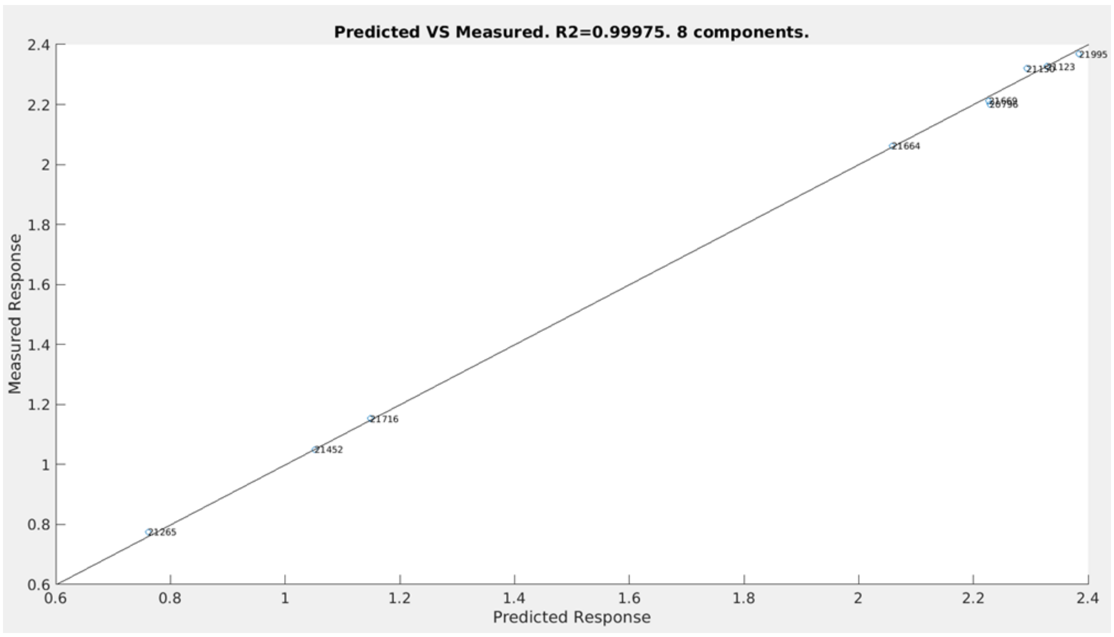

Figure 134: Predicted versus measured response for the MLDF model for TIC.

With good marks from the cross-validation, the final model, containing 8 components, was created. The predicted results superimposed onto the measured results can be seen in figure 135.

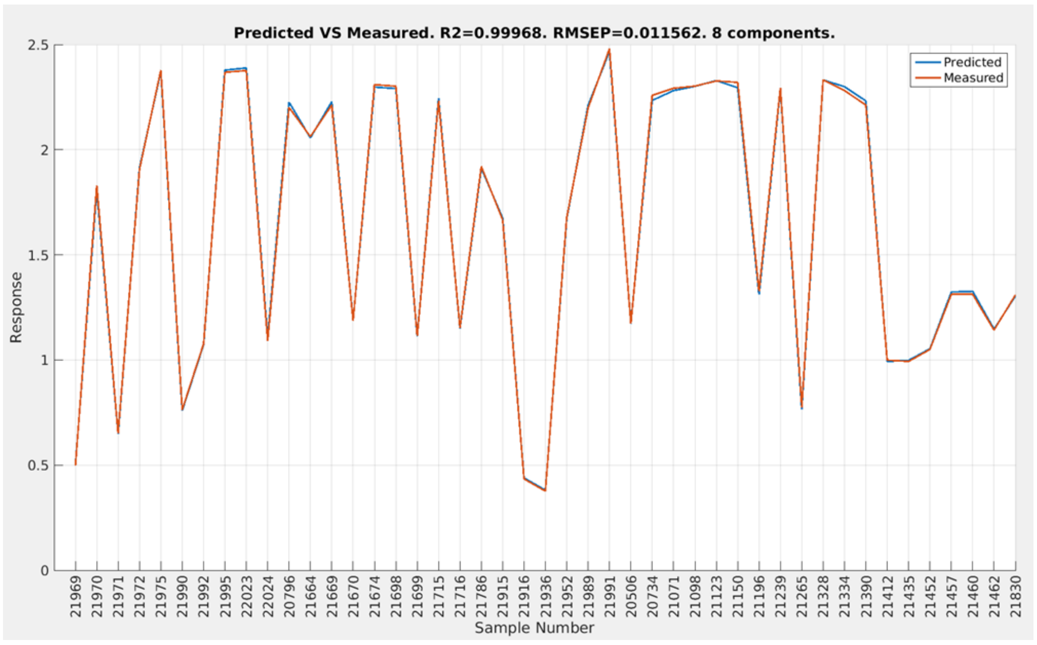

Figure 135: Plot of predicted and measured responses by sample number for MLDF for TIC.

Density

The cross-validation results for density were very promising. A second run of outlier detection identified samples 20696, 20734 and 21071 as potential outliers, and they were removed. In the end, a final cross-validation was run, showing a minimum RMSECV at 15 components, for a value of 0.4116, as seen in figure 136. This is a significant improvement on the 0.6091 that the NIR individual model produced, and an even more significant improvement over the 0.6764 that the LLDF without Raman produced.

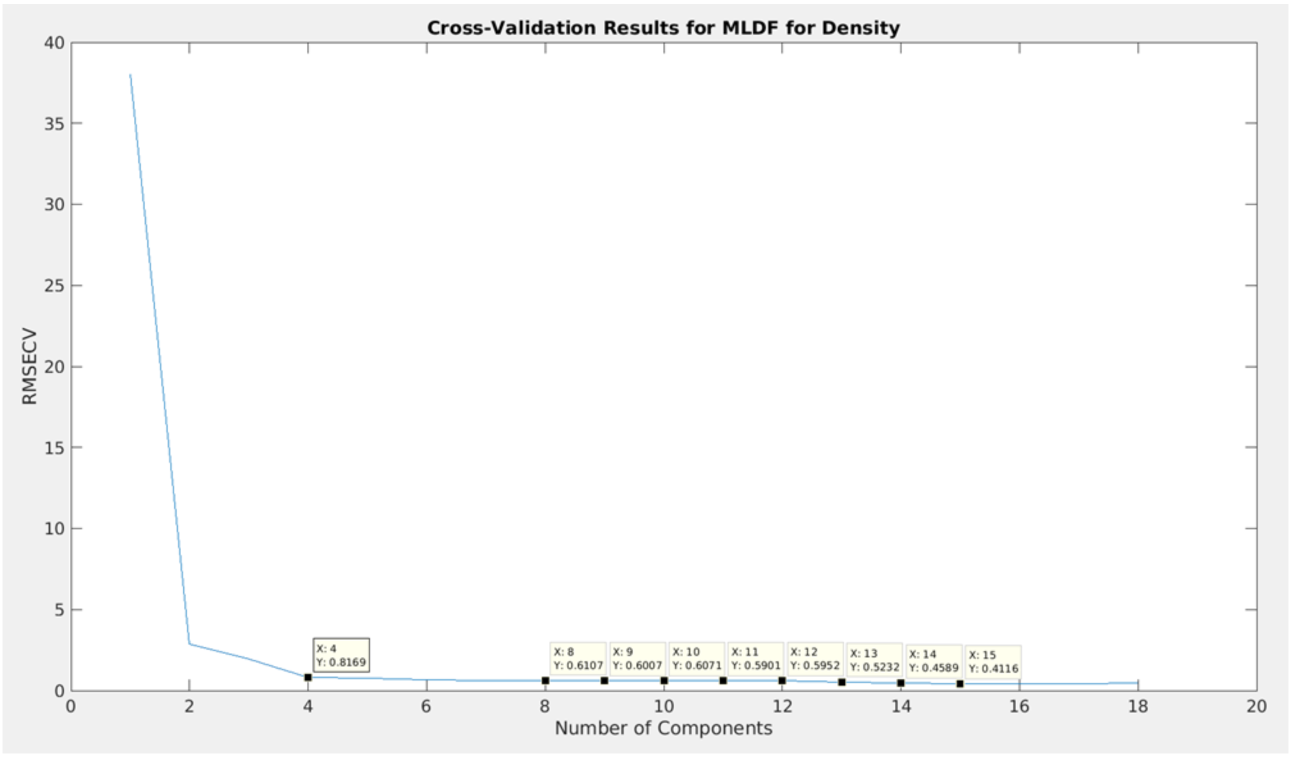

Figure 136: RMSECV versus components used for MLDF for density.

Although the suggested number of components is 15, this seemed excessive, and for little gain in what is already an incredibly small residual. For this reason, a different local minima, at 9 components, would be used going forward. The difference in RMSECV between 4 components and 9 components is negligible considering the low residual and should be considered to reduce calculation times if needed. 2 components is still very accurate as well, and would significantly decrease data processing amounts. In the average predicted residual percentage against number of components plot this is reflected as well, as seen in figure 137.

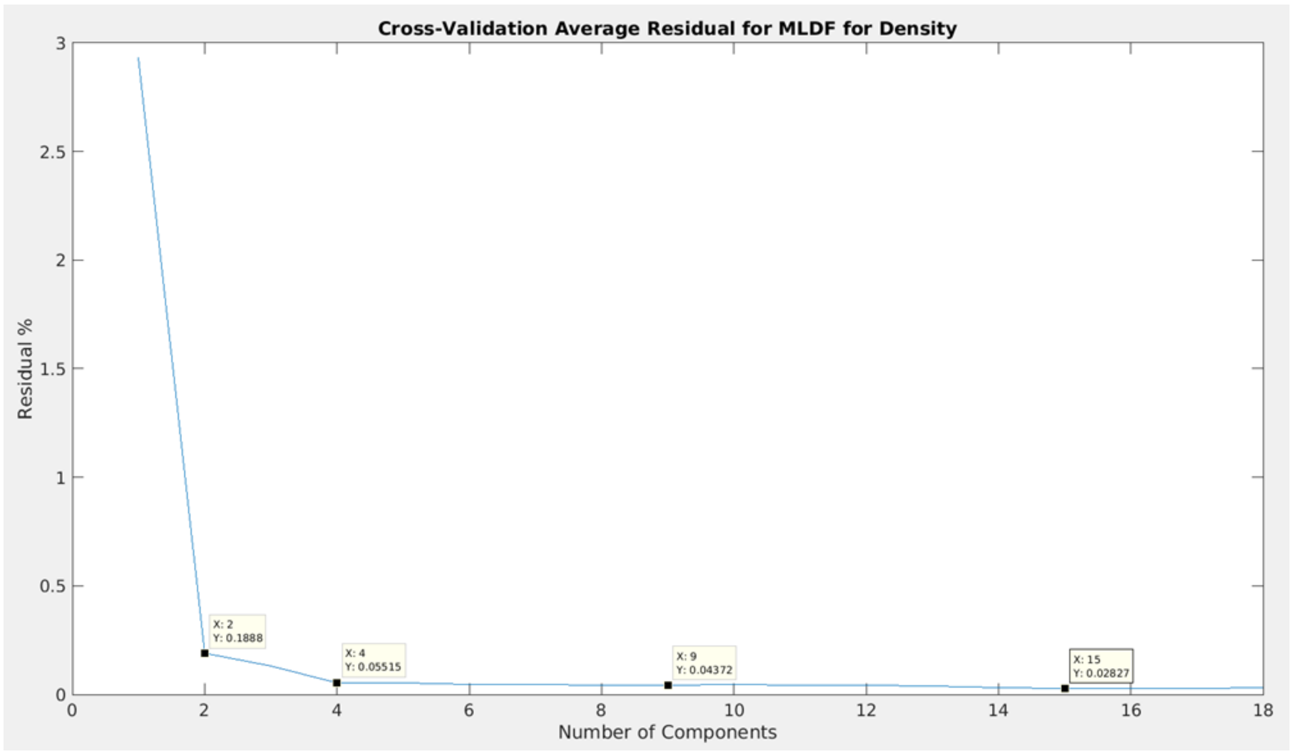

Figure 137: Plot of average residual percentage for a given amount of components for MLDF for density.

As seen, 15 components gives the lowest true residual percentage at 0.02827%. 9 components, which would be used going forward, gave a very respectable 0.04372%. A sample plot of the predicted versus measured values from one of the test sets within the cross-validation can be seen in figure 138.

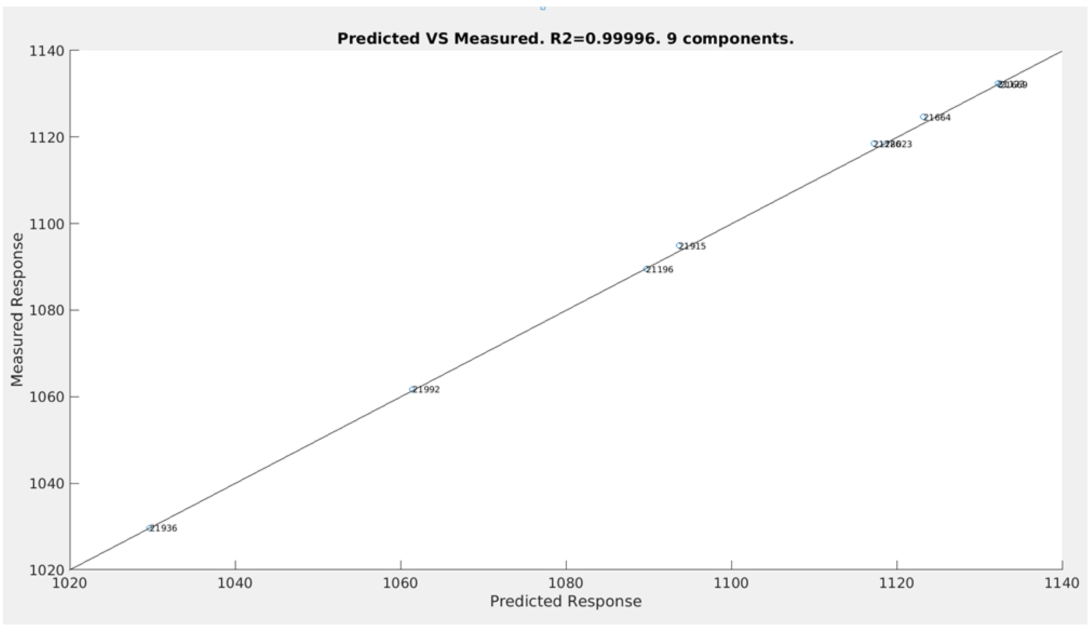

Figure 138: Predicted versus measured response for the MLDF model for density.

With good marks from the cross-validation, the final model, containing 9 components, was created. The predicted results superimposed onto the measured results can be seen in figure 139.

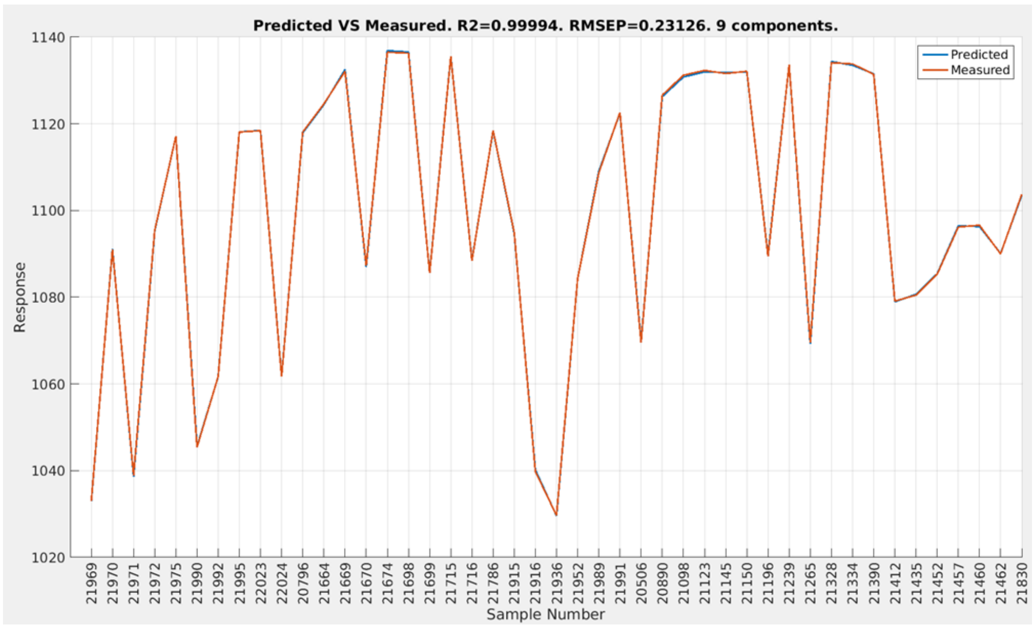

Figure 139: Plot of predicted and measured responses by sample number for MLDF for density.

In every instance, the MLDF model produced results approximately as good as or better than the individual models alone, as well as LLDF without the Raman spectra. This was very impressive given the fact that it was still using the Raman spectra to do so, and shows that MLDF produces results on the same level as the best performing individual model, even when it contains an individual model with very poor prediction accuracy. As the MLDF model without using Raman produced near identical results, it can be assumed that MLDF is very good at suppressing an instrument that is performing poorly.

7 Conclusion

This thesis had the goal of using data fusion techniques to predict properties of various amine solvent samples in an attempt to see what sort of gains in predictive capability could be achieved over usage of the instruments individually. The methods described here have created models for each individual instrument, and then compared those individual results among each other, as well as to the fused data results.

For individual instrument modeling, ATR-FTIR performed better for TOT_ALK and TIC, while NIR did the best for density, while Raman was poor in all areas. The full LLDF models performed worse than ATR-FTIR and NIR, but not by a large margin in most cases. The LLDF models without Raman (LLDF-R) performed at approximately the same level as the best individual models. The MLDF models performed at near identical levels as the best individual instrument, providing the best overall coverage. Tables 1, 2 and 3 show a condensed review of all of these for easy comparison.

Table 1: Table containing the core information from all of the TOT_ALK models.

TOT_ALK						
	ATR-FTIR	**NIR**	**Raman**	**LLDF**	**LLDF-R**	**MLDF**
Components	8	8	8	11	12	9
RMSECV	0.03343	0.05239	0.13260	0.06044	0.03094	0.03356
RMSECV%	0.73%	1.14%	2.89%	1.32%	0.67%	0.73%
Residual %	0.59%	0.85%	2.21%	1.04%	0.57%	0.60%

Table 2: Table containing the core information from all of the TIC models.

TIC						
	ATR-FTIR	**NIR**	**Raman**	**LLDF**	**LLDF-R**	**MLDF**
Components	6	8	8	8	8	8
RMSECV	0.01411	0.02214	0.3101	0.02874	0.01507	0.01484
RMSECV%	0.83%	1.31%	18.31%	1.70%	0.89%	0.88%
Residual %	0.79%	1.48%	13.54%	1.44%	0.85%	0.82%

Table 3: Table containing the core information from all of the density models.

Density						
	ATR-FTIR	**NIR**	**Raman**	**LLDF**	**LLDF-R**	**MLDF**
Components	9	9	5	7	13	9
RMSECV	0.8913	0.6091	9.539	0.9925	0.6764	0.6007*
RMSECV%	0.08%	0.06%	0.91%	0.95%	0.65%	0.57%*
Residual %	0.06%	0.04%	0.63%	0.07%	0.05%	0.04%*

*Using the reduced number of components. Better values possible.

It can be concluded that for this application, Raman spectroscopy is a poor choice, as it consistently produced worse results, both on an individual level, and at a low-level data fusion level, especially for the sake of measuring TIC. For this reason, utilizing Raman spectrometers for process monitoring at CO_2 Technology Center Mongstad's amine plant is not advised.

ATR-FTIR was concluded to be the individual instrument of choice for the above mentioned process, as it produced the best predictive models for two of the three measured responses, and provided accurate predictions across the board.

NIR, while not quite as good at predictive modeling for TOT_ALK and TIC, still provided reasonable predictions, and provided the best predictive capability for density. Although ATR-FTIR is the best choice for an individual instrument based on effectiveness, NIR may still be a viable choice based on cost.

Data fusion techniques were shown to be very effective at predictive modeling, even given data from a poorly predicting instrument. The full LLDF model mitigated the damage done by Raman very well, but was still unable to match the level of the best performing individual model, although it produced reasonable accuracy.

The LLDF model without Raman performed at a similar level as the best individual models, providing some of the best attributes of both machines. Finally the MLDF models, with or without Raman, performed at almost identical levels as the best individual models, showing that it is well equipped to deal with a poorly performing instrument, and provides the perfect middle ground when it comes to performance, allowing one model to make the best possible predictions.

Although it was proposed that separate models be made for lean and rich samples, it has been shown that this is not strictly needed, as very accurate predictions can be made without the need for multiple models. Due to the low number of samples, a detailed analysis was not performed on strictly lean, or strictly rich samples, although it can be speculated that individual models for them may have provided even more accurate predictions.

Finally, the methods for outlier prediction, preprocessing and model generation described in this thesis have shown themselves to be effective at producing quality models that could be implemented for real time process monitoring of an amine based carbon capture plant such as the one located at CO_2 Technology Center Mongstad.

8 References

[1] D. Thimsen, A. Maxson, V. Smith and T. Cents, "Results from MEA testing at the CO2 Technology Centre Mongstad. Part I: Post-Combustion CO2 capture testing methodology," *Energy Procedia,* vol. 63, pp. 5938-5958, 2014.

[2] "Process Control - Applications," Bruker, 2018. [Online]. Available: https://www.bruker.com/applications/quality-control/process-control.html. [Accessed December 2018].

[3] H. Young and R. Freedman, University Physics with Modern Physics 13th edition, San Fransisco, California: Pearson, 2012.

[4] "What Is Spectrometry And What Is It Used For?," ATA Scientific, September 2016. [Online]. Available: https://www.atascientific.com.au/spectrometry/. [Accessed December 2018].

[5] R. Grainger, "University of Oxford," October 2013. [Online]. Available: http://eodg.atm.ox.ac.uk/user/grainger/research/book/protected/Chapter4.pdf. [Accessed December 2018].

[6] D. Rheinstrom, "Atomic Energy Levels," Khan Academy, June 2018. [Online]. Available: https://www.khanacademy.org/science/physics/quantum-physics/atoms-and-electrons/v/atomic-energy-levels. [Accessed December 2018].

[7] "A basic overview of Raman spectroscopy.," Renishaw, [Online]. Available: https://www.renishaw.de/de/a-basic-overview-of-raman-spectroscopy--25805. [Accessed December 2018].

[8] W. Reusch, "Infrared Spectroscopy," Michigan State University, May 2013. [Online]. Available: https://www2.chemistry.msu.edu/faculty/reusch/VirtTxtJml/Spectrpy/InfraRed/infrared.htm. [Accessed December 2018].

[9] G. Reich, "Near-infrared spectroscopy and imaging: Basic principles and pharmaceutical applications.," *Advanced Drug Delivery Reviews,* vol. 57, no. 8, pp. 1109-1143, 2005.

[10] "Experiment 7 - IR Spectroscopy," Columbia University, 2007. [Online]. Available: http://www.columbia.edu/cu/chemistry/ugrad/hssp/EXP_7.html. [Accessed December 2018].

[11] N. Birkner and Q. Wang, "How an FTIR Spectrometer Operates," Chemistry LibreTexts, February 2015. [Online]. Available: https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Spectroscopy/Vibrational_Spectroscopy/Infrared_Spectroscopy/How_an_FTIR_Spectrometer_Operates. [Accessed December 2018].

[12] "Chapter 7. SPIRE Spectroscopy Mode Cookbook," Herschel - European Space Agency, [Online]. Available: http://herschel.esac.esa.int/hcss-doc-15.0/load/spire_drg/html/spire-spectroscopy.html. [Accessed December 2018].

[13] "Attenuated Total Reflection (ATR) - a versitile tool for FT-IR spectroscopy.," Bruker Optics, 2011. [Online]. Available: https://www.bruker.com/fileadmin/user_upload/8-PDF-Docs/OpticalSpectrospcopy/FT-IR/ALPHA/AN/AN79_ATR-Basics_EN.pdf. [Accessed December 2018].

[14] D. Kennepohl, S. Farmer and W. Reusch, "12.8: Infrared Spectra of Some Common Functional Groups," Chemistry LibreTexts, October 2018. [Online]. Available: https://chem.libretexts.org/Bookshelves/Organic_Chemistry/Map%3A_Organic_Chemistry_(McMurry)/Chapter_12%3A_Structure_Determination%3A_Mass_Spectrometry_and_Infrared_Spectroscopy/12.08_Infrared_Spectra_of_Some_Common_Functional_Groups. [Accessed December 2018].

[15] C. Ophardt, "Greenhouse Gases," Virtual Chembook, 2003. [Online]. Available: http://chemistry.elmhurst.edu/vchembook/globalwarmA5.html. [Accessed December 2018].

[16] B. Mojet, S. Ebbesen and L. Lefferts, "Light at the Interface: The Potential of Attenuated Total Reflection Infrared Spectroscopy for Understanding Heterogeneous Catalysis in Water," *Chemical Society Reviews*, vol. 39, no. 12, pp. 4643-4655, 2010.

[17] "Infrared spectrometers: NIR and MIR compared," University of East Anglia, [Online]. Available: https://www.futurelearn.com/courses/food-fraud/0/steps/10515. [Accessed December 2018].

[18] A. Davies, "An introduction to near infrared (NIR) spectroscopy," IM Publications, 2017. [Online]. Available: http://www.impublications.com/content/introduction-near-infrared-nir-spectroscopy. [Accessed December 2018].

[19] M. Prochazka, Surface-Enhanced Raman Spectroscopy, Prague, Czech Republic: Springer, 2015.

[20] S.-M. Thomas, "Infrared and Raman spectroscopy," Science Education Resource Center at Carleton College, January 2018. [Online]. Available: https://serc.carleton.edu/NAGTWorkshops/mineralogy/mineral_physics/raman_ir.html . [Accessed December 2018].

[21] "What is Raman spectroscopy?," InPhotonics, 2012. [Online]. Available: http://www.inphotonics.com/raman.htm. [Accessed December 2018].

[22] Andor, "Raman Spectroscopy," Oxford Instruments, [Online]. Available: https://www.oxinst.com/learning/view/article/raman-spectroscopy. [Accessed December 2018].

[23] "What is Raman Spectroscopy?," Nano Photon, 2016. [Online]. Available: https://www.nanophoton.net/raman/raman-spectroscopy.html. [Accessed December 2018].

[24] O. Akkus and S. Yang, "Fluorescence Background Problem in Raman Spectroscopy: Is 1064 nm Excitationan Improvement of 785 nm?," Case Western Reserve University, Cleveland, Ohio, 2015.

[25] "What is CCS?," Carbon Capture and Storage Association, [Online]. Available: http://www.ccsassociation.org/what-is-ccs/. [Accessed December 2018].

[26] "online, inline, atline and bypass (analysers)," AAVOS International, 2017. [Online]. Available: https://aavos.eu/glossary/online-inline-atline-bypass-analysers/. [Accessed December 2018].

[27] "Data preprocessing," Connecticut State University, [Online]. Available: http://www.cs.ccsu.edu/~markov/ccsu_courses/DataMining-3.html.

[28] "Normalization," International Center for Research - University of Kentucky, [Online]. Available: http://www.analytictech.com/ba762/handouts/normalization.htm. [Accessed December 2018].

[29] S. Saitta, "Standardization vs. normalization," Data Mining Blog, July 2007. [Online]. Available: http://www.dataminingblog.com/standardization-vs-normalization/. [Accessed December 2018].

[30] W. Press, S. Teukolsky, W. Vetterling and B. Flannery, Numerical Recipes in Fortran 77: The Art of Scientific Computing, Cambridge: Cambridge University Press, 1997.

[31] A. Savitsky and M. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures.," *Analystical Chemistry,* vol. 36, no. 8, pp. 1627-1639, 1964.

[32] N. K. Afseth and A. Kohler, "Extended multiplicative signal correction in vibrational spectroscopy, a tutorial," *Chemometrics and Intelligent Laboratory Systems,* vol. 117, pp. 92-99, 2012.

[33] E. Borras, J. Ferre and R. Boque, "Data fusion methodologies for food and beverage authentication and quality assessment – A review," *Analytica Chimica,* vol. 891, pp. 1-14, 2015.

[34] E. Borras, J. Ferre and R. Boque, "Prediction of olive oil sensory descriptors using instrumental data fusion and partial least squares (PLS) regression," *Talanta,* vol. 155, pp. 116-123, 2016.

[35] B. Geurts, J. Engel and B. Rafii, "Improving high-dimensional data fusion by exploiting the multivariate advantage," *Chemometrics and Intelligent Laboratory Systems,* vol. 156, pp. 231-240, 2016.

[36] J. Chen, F. Ye and G. Zhao, "Rapid determination of farinograph parameters of wheat flour using data fusion and a forward interval variable selection algorithm," *Analystical methods,* vol. 9, no. 45, pp. 6341-6348, 2017.

[37] T. Doeswijk, A. Smilde and J. Hageman, "On the increase of predictive performance with high-level data fusion," *Analytica Chimica Acta,* vol. 705, pp. 41-47, 2011.

[38] T. Rajalahti and O. Kvalheim, "Multivariate data analysis in pharmaceutics: A tutorial review," *International Journal of Pharmaceutics,* vol. 417, pp. 280-290, 2011.

[39] R. Tobias, "An Introduction to Partial Least Squares Regression," 2016. [Online]. Available: https://stats.idre.ucla.edu/wp-content/uploads/2016/02/pls.pdf. [Accessed December 2018].

[40] S. O'Halloran, "Lecture 5: Model Checking," [Online]. Available: http://www.columbia.edu/~so33/SusDev/Lecture_5.pdf. [Accessed December 2018].

[41] M. Meloun and J. Militky, "Detection of single influential points in OLS regression model building," *Analytica Chimica Acta,* vol. 439, pp. 169-191, 2001.

[42] K. Dunn, "4.11. Outliers: discrepancy, leverage, and influence of the observations," Learning Chemical Engineering, 2018. [Online]. Available: https://learnche.org/pid/least-squares-modelling/outliers-discrepancy-leverage-and-influence-of-the-observations. [Accessed December 2018].

[43] T. Oberg, "A QSAR for the hydroxyl radical reaction rate constant: validation, domain of application, and prediction," *Atmospheric Environment,* vol. 39, no. 12, pp. 2189-2200, 2005.

[44] W. Jacoby, "Lecture 11: Outliers and Influential data," [Online]. Available: http://polisci.msu.edu/jacoby/icpsr/regress3/lectures/week3/11.Outliers.pdf. [Accessed December 2018].

[45] A. Hebbali, "Studentized residuals vs leverage plot," R Squared Academy, [Online]. Available: https://rsquaredacademy.github.io/olsrr/reference/ols_rsdlev_plot.html. [Accessed December 2018].

[46] R. Bro and A. Smilde, "Principal component analysis," *Analytical Methods,* vol. 6, no. 9, pp. 2812-2831, 2014.

[47] L. Terra, P. Filgueiras and L. Tose, "Petroleomics by electrospray ionization FT-ICR mass spectrometry coupled to partial least squares with variable selection methods: prediction of the total acid number of crude oils," *Analyst,* vol. 139, no. 19, pp. 4908-4916, 2014.

[48] Q. Xu, Y.-Z. Liang and Y.-P. Du, "Monte Carlo cross-validation for selecting a model and estimating the prediction error in multivariate calibration," *Journal of Chemometrics,* vol. 18, no. 2, pp. 112-120, 2004.

[49] R. Remesan and J. Mathew, Hydrological Data Driven Modelling: A Case Study Approach, Springer, 2014.

[50] N. Akarachantachote, S. Chadcham and K. Saithanu, "Cutoff Threshold of Variable Importance in Projection for Variable Selection," *International Fournal of Pure and Applied Mathematics,* vol. 94, no. 3, pp. 307-322, 2014.

[51] T. Rajalahti, R. Arneberg and F. Berven, "Biomarker discovery in mass spectral profiles by means of selectivity ratio plot," *Chemometrics and Intelligent Laboratory Systems,* vol. 95, pp. 35-48, 2009.

[52] "Electromagnetic Spectrum - Introduction," NASA, March 2013. [Online]. Available: https://imagine.gsfc.nasa.gov/science/toolbox/emspectrum1.html. [Accessed December 2018].

9 Appendices

9.1 MATLAB Code for the Savitzky-Golay filter

```
function [G,SAGOoutput] = sago(Xraw,window,order,derivations)
%Savitzky-Golay filter
%Usage:
%
%[G,SAGOoutput] = sago(Xraw,window,order,derivations), where G is the
%smoothed data matrix after any requested derivations, SAGOoutput is the
%smoothed data matrix with wavelength vector, Xraw is the data matrix,
%window is the size of the window, which must be an odd number, order is
%the order of the filter polynomial you wish to use (limited to 2 and 3,
%for 2nd and 3rd order) and derivations is the number of times the output
%matrix should be differentiated, limited to a maximum of 2 times.

%It should be noted that it is assumed that the first row of Xraw is the
%wavelength vector.

%Adam Joshua Armstrong, University of Bergen, 2018

X=[];
X=Xraw(:,2:end)';   %Creation of the data matrix.
waves=Xraw(:,1)';   %Creation of the wavelength vector.
fil_vect=zeros(1,window);   %Creation of the filter vector.
for i=1:size(X,1)
    for j=1:size(X,2)-window+1
        fil_vect(1,:)=X(i,j:j+window-1);    %Filling of window data.
        [b,S,mu]=polyfit(waves(1,j:j+window-1),fil_vect,order); %Regression
of the centered window data.
        x_var=(waves(1,j+(window-1)/2)-mu(1,1));    %Determining the
centered value of center wavelength.
        if order==2
            mid_poly=b(1,3)+b(1,2)*x_var+b(1,1)*x_var^2;    %Determining
value of 2nd order polynomial underived center point.
            if derivations==0
                G(i,j)=mid_poly;
            elseif derivations==1
                mid_poly_1der=b(1,2)+2*b(1,1)*x_var;    %Determining value
of 2nd order polynomial single derived center point.
                G(i,j)=mid_poly_1der;
            elseif derivations==2
                mid_poly_2der=2*b(1,1); %Determining value of 2nd order
polynomial double derived center point.
                G(i,j)=mid_poly_2der;
            else
                disp('Please use a number from 0 to 2, 0 meaning no
derivations, 1 meaning single derivation, and 2 meaning double
derivation.')
                return
            end
        elseif order==3
            mid_poly=b(1,4)+b(1,3)*x_var+b(1,2)*x_var^2+b(1,1)*x_var^3;
%Determining value of 3rd order polynomial underived center point.
            if derivations==0
                G(i,j)=mid_poly;
            elseif derivations==1
                mid_poly_1der=b(1,3)+2*b(1,2)*x_var+3*b(1,1)*x_var^2;
%Determining value of 3rd order polynomial single derived center point.
                G(i,j)=mid_poly_1der;
            elseif derivations==2
                mid_poly_2der=2*b(1,2)+6*b(1,1)*x_var;  %Determining value
of 3rd order polynomial double derived center point.
                G(i,j)=mid_poly_2der;
            else
                disp('Please use a number from 0 to 2, 0 meaning no
derivations, 1 meaning single derivation, and 2 meaning double
derivation.')
                return
            end
        else
            disp('Please select either 2 or 3 for second or third order
polynomials.')
            return
        end
    end
end
SAGOoutput=[];
offset=(window-1)/2;    %Offset size to account for variable reduction.
SAGOoutput=waves(:,offset+1:size(waves,2)-offset); %Creation of reduced
                                                   %wavelength vector.
SAGOoutput(2:size(G,1)+1,:)=G;
end
```

9.2 MATLAB Code for the Raman Spectra Interval Smoother

```
function [ XRamanSMOOTH ] = RamanSmoother(X)
%Smoothing function for incremental data when datapoints are rounded to
%whole nubmers, such as with UiB's Raman spectrometer.
%
%Usage:
%
%[XRamanSMOOTH] = RamanSmoother(X), where X is the predictor matrix, and
%XRamanSMOOTH is the matrix with the incremental wavelengths corrected.
%
%It should be noted that it is assumed that the first object vector is the
%wavelength vector.

%Adam Joshua Armstrong, University of Bergen, 2018

XRamanSMOOTH=[];
count=0;
bottom=0;
    for i=1:length(X)-1
        if X(i,1) == X(i+1,1)       %Determining of the number of repeated
values in the local series.
            if count == 0           %
                bottom=i;           %
            end                     %
            count=count+1;          %
        else
            offset=1/(count+1);     %Determines the size of the increment
to be applied.
            XRamanSMOOTH(bottom,1)=X(bottom,1);
            for j=1:count
                XRamanSMOOTH(bottom+j,1)=X(bottom,1)+offset*j; %Offsets the
wavelegths to evenly spaced increments.
            end
            count=0;
        end
    end
end
```

9.3 MATLAB Code for the (E)MSC Preprocessing

```
function [Xc,waves] = tmcEMSC(Xraw,type,basepoly)
%(Extended) Multiplcative Scatter Correction algorithm.
%Usage:
%
%Xc = tmcEMSC(Xraw,type,basepoly) where Xc is the corrected matrix, Xraw is
%the raw data matrix, type is whether to run an 'EMSC' or 'MSC', and
%basepoly is the order of the detrend polynomial to use.
%(Note, that basepoly is meaningless for MSC)
%
%It should be noted that the only type of EMSC used here is baseline
%correction through a detrend, and not other types of EMSC corrections.
%Also of note is that the first row of the raw data matrix is assumed to
%be the wavelength vector.

%Adam Joshua Armstrong, University of Bergen, 2018

X=Xraw(:,2:end)';    %Creation of the variable matrix.
waves=Xraw(:,1)';    %Creation of the wavelenth vector.
Xr=mean(X);          %Creation of the reference spectrum.
if strcmp(type,'MSC')==1
    for i=1:size(X,1)
        b=polyfit(Xr,X(i,:),1);                %Linear regression of the
reference
                                               %spectrum against the
individual
                                               %sample spectrum.
        Xc(i,:)=(X(i,:)-b(1,2))/b(1,1);        %Correcting of the sample
spectrum
                                               %based on regressing
coefficients.
    end
elseif strcmp(type,'EMSC')==1
    detrendpoly=[];
    for i=1:size(X,1)
        bbase=polyfit(waves,X(i,:),basepoly);%Baseline determination.
        Xc2(i,:)=X(i,:);
        for p=1:basepoly
            detrendpoly=-bbase(1,basepoly+1-p).*waves(1,:).^(p);
            Xc2(i,:)=Xc2(i,:)+detrendpoly;  %Detrending of the spectrum.
        end
        bref=polyfit(Xr,Xc2(i,:),1);
        Xc(i,:)=(Xc2(i,:)-bref(1,2))/bref(1,1);
                                               %MSC performed on the detrended
                                               %spectrum.
    end
else
    disp('Please choose either MSC or EMSC.')
end
plot(waves,Xc)   %Plot of the now corrected spectra.
end
```

9.4 MATLAB Code for the Standalone PLS Function

```
function
[Ypred,Yinit,tmat,pmat,beta,SR,explvartot]=plsfunctSTANDALONE(X,Y,comp,meth
od,labels)
%PLS1 function meant to be used as a standalone function.
%
%Usage:
%
%[Ypred,Yinit,tmat,pmat,beta,SR,explvartot] =
%plsfunctSTANDALONE(X,Y,comp,method,labels), where Ypred is the predicted
%response vector, Yinit is the measured response vector, tmat is the scores
%matrix, pmat is the loadings matrix, beta is the regression coefficient
%vector, SR is the Selectivity Ratio vector, explvartot is the total
%explained variance vector for each component threshold, X is the predictor
%matrix, Y is the response vector, comp is the number of components to use,
%method is a boolean response to whether mean centering is requested, and
%labels is the label vector.
%
%It should be noted that unlike several other functions, wavelength and
%label vectors are not assumed to be included in the first object of X, and
%first variable of Y, respectively.
%
%Adam Joshua Armstrong, University of Bergen, 2018
%
Xinit=X;
Yinit=Y;
Yinflated=Y;
SSexp=[];
SSres=[];
SR=[];
%Mean centering, if requested.
if method == 1
    Ymean=mean(Y);
    Y=Y-Ymean;
    Xmean=mean(X);
    for i=1:size(X,1)
        X(i,:)=X(i,:)-Xmean;
    end
    Xinit=X;
    Yinflated=Y;
end
%Loop that runs the core PLS1 algorithm.
for i=1:comp
    w=X'*Y;
    w=w/norm(w);
    wmat(:,i)=w;
    t=X*w;
    p=X'*t/(t'* t);
    X=X-t*p';
    c=t'*Y/(t'*t);
    cmat(:,i)=c;
    tmat(:,i)=t;
    pmat(:,i)=p;
    weight=wmat*(pmat'*wmat)^(-1);
    beta=weight*cmat';
    Ypred=Xinit*beta;
    explvartot(1,i)=(1-(sum((Yinflated-Ypred).^2))/sum((Yinflated).^2));
    if i==1
        explvarcomp(1,i)=explvartot(1,i);
    else
        explvarcomp(1,i)=explvartot(1,i)-explvartot(1,i-1);
    end
end
%Adjusts Ypred values to remove mean centering effects.
if method == 1
    Ypred=Ypred+Ymean;
end
r1=corrcoef(Yinit,Ypred);
r2=r1(2,1)^2;
RMSEP=sqrt(sum((Yinit-Ypred).^2)/length(Yinit));
figure
set(axes,'XTick',[1:1:length(Y)],'XTickLabel',labels,'XTickLabelRotation',9
0)
hold;
plot(Ypred,'LineWidth',2)
plot(Yinit,'LineWidth',2)
xlabel('Sample Number')
ylabel('Response')
legend('Predicted','Measured')
ax=gca;
ax.FontSize=16;
grid on;
title(['Predicted VS Measured. R2=' num2str(r2) '. RMSEP=' num2str(RMSEP)
'. ' num2str(comp) ' components.']);
hold;
Yres=Yinit-Ypred;
%Creates the SR vector.
if varsel==1
    tTP=Xinit*beta/norm(beta);
    pTP=Xinit'*tTP/(tTP'*tTP);
    XTP=tTP*pTP';
    resTP=Xinit-XTP;
    for i=1:length(XTP)
        SSexp(i,:)=(norm(tTP*pTP(i,:)'))^2;
        SSres(i,:)=(norm(resTP(:,i)))^2;
        SR(i,:)=SSexp(i,:)/SSres(i,:);
    end
else
end
SR=(SR-min(SR))/(max(SR)-min(SR)); %Normalization of the SR vector.
end
```

9.5 MATLAB Code for the Embedded PLS Function

```
function
[Ypred,Yinit,tmat,pmat,beta,SR,explvartot]=plsfunctNEW(X,Y,comp,varsel)
%PLS1 function meant to be used by higher functions, such as MonteCarloCV.
%Not meant to be used as a standalone function. For standalone
%functionality, use plsfunctSTANDALONE.
%
%Usage:
%
%[Ypred,Yinit,tmat,pmat,beta,SR,explvartot]=plsfunctNEW(X,Y,comp,varsel),
%where Ypred is the predicted response vector, Yinit is the measured
%response vector, tmat is the scores matrix, pmat is the loadings matrix,
%beta is the regression coefficient vector, SR is the Selectivity Ratio
%vector, explvartot is the total explained variance vector for each
%component threshold, X is the predictor matrix, Y is the response vector,
%comp is the number of components to use, and varsel is a boolean response
%to whether an SR vector should be made.
%
%Adam Joshua Armstrong, University of Bergen, 2018
Xinit=X;
Yinit=Y;
SSexp=[];
SSres=[];
SR=[];
%Loop that runs the core PLS1 algorithm.
for i=1:comp
    w=X'*Y;
    w=w/norm(w);
    wmat(:,i)=w;
    t=X*w;
    p=X'*t/(t'* t);
    X=X-t*p';
    c=t'*Y/(t'*t);
    cmat(:,i)=c;
    tmat(:,i)=t;
    pmat(:,i)=p;
    weight=wmat*(pmat'*wmat)^(-1);
    beta=weight*cmat';
    Ypred=Xinit*beta;
    explvartot(1,i)=(1-(sum((Yinit-Ypred).^2))/sum((Yinit).^2));
    if i==1
        explvarcomp(1,i)=explvartot(1,i);
    else
        explvarcomp(1,i)=explvartot(1,i)-explvartot(1,i-1);
    end
end
Ypred;
Yres=Yinit-Ypred;
%Creates the SR vector.
if varsel==1
    tTP=Xinit*beta/norm(beta);
    pTP=Xinit'*tTP/(tTP'*tTP);
    XTP=tTP*pTP';
    resTP=Xinit-XTP;
    for i=1:length(XTP)
        SSexp(i,:)=(norm(tTP*pTP(i,:)'))^2;
        SSres(i,:)=(norm(resTP(:,i)))^2;
        SR(i,:)=SSexp(i,:)/SSres(i,:);
    end
```

```
else
end
SR=(SR-min(SR))/(max(SR)-min(SR)); %Normalization of the SR vector.
end
```

9.6 MATLAB Code for the Embedded Monte Carlo Cross-Validation Function

```
function
[RMS,YPredPerm,YPredSet,resVect,PredErrorOuter,RMSECVcomp,YPredSetlabels,r2
] = MonteCarloCV(X,Y,comp,N,ratio,method,response)
%Monte Carlo Cross-Validation function, not necessarily meant to be used as
%a standalone function, but rather utilized bu the CVgrapher function. It
%will still work as a standalone, but will only run once.
%
%Usage:
%
%[RMS,YPredPerm,YPredSet,resVect,PredErrorOuter,RMSECVcomp,YPredSetlabels,
%r2] = MonteCarloCV(X,Y,comp,N,ratio,method,response) where RMS is the root
%mean square, YPredPerm is the predicted response vector, YPredSet is the
%measured response vector corresponding to the same objects used in
%YPredPerm, resVect is the residual vector, PredErrorOuter is the residual
%percentages to be output to the CVgrapher function, RMSECVcomp is the
%RMSECV values for a given comp, YPredSetlabels is the object lable vector
%(which will be the first column of the Y input variable), r2 is the R^2
%value for the regression, X is the predictor matrix, the first object
%being the wavelength vector, Y is the response matrix where each variable
%is a response vector, and the first variable is the object label vector
%(or insert a blank column), comp is the number of components to be
%validated, N is the number of iterations per block, ratio is the ratio of
%of test set split (ie, 0.8 uses 80% of the objects to create the model,
%and 20% to validate it), method is a boolean response for if the predictor
%matrix should be mean centered, and response is the variable from the Y
%matrix which will be the response vector (that is to say that a response
%of '1' would use the first column as the response vector).
%
%Adam Joshua Armstrong, University of Bergen, 2018
%
X=X(2:end,:);
Ylabels=(Y(:,1));
if response ~= 0
    Y=Y(:,response);
end
%Mean centers the datasets, if requested.
if method == 1
    Ymean=mean(Y);
    Y=Y-Ymean;
    Xmean=mean(X);
    for i=1:size(X,1)
        X(i,:)=X(i,:)-Xmean;
    end
end
Mx=size(X,1);
Q=floor(Mx*ratio);
XTrain=[];
YTrain=[];
resVect=[];
PredErrorVect=[];
PRESSvect=[];
%Loop that runs N number of cross-validation iterations.
for k=1:N
    np=randperm(Mx)';
    %Loop that creates and divides training and validation sets.
    for j=1:size(X,1)
        if j <= Q
            XTrain(j,:)=X(np(j,1),:);
            YTrain(j,:)=Y(np(j,1),:);
        else
            XPredSet(j-Q,:)=X(np(j,1),:);
            YPredSet(j-Q,:)=Y(np(j,1),:);
            YPredSetlabels(j-Q,:)=Ylabels(np(j,1),1);
        end
    end
    [Ypred,Yinit,tmat,pmat,beta,SR]=plsfunctNEW(XTrain,YTrain,comp,0);
    YPredPerm=XPredSet*beta;
    %Converts the mean centered results back into the original input
    %magnitude, if mean centering was requested.
    if method == 1
        YPredPerm=YPredPerm+Ymean;
        YPredSet=YPredSet+Ymean;
    end
    Yres=YPredPerm-YPredSet;
    PredErrorInner=abs(Yres)./abs(YPredSet)*100;
    PredErrorVect(k,1)=mean(PredErrorInner);
    resVect(k,1)=mean(Yres);
    PRESSvect(k,1)=sum((Yres.*Yres)/size(YPredPerm,1));
end
RMSECVcomp=sqrt(mean(PRESSvect));
RMS=sqrt(sum(resVect.*resVect)/size(YPredSet,1));
PredErrorOuter=mean(PredErrorVect);
r1=corrcoef(Yinit,Ypred);
r2=r1(2,1)^2;
end
```

9.7 MATLAB Code for the Monte-Carlo Cross-Validation Engine and Graphing Function

```
function [RMSmat,meanRMSECVvect,resMat,PredErrormat,meanPredErrorvect] =
CVgrapher(X,Y,comp,N,M,ratio,method,response)
%Monte Carlo Cross-Validation iterator, used for running multiple
%cross-validation loops, and then plotting results of the compiled data.
%
%Usage:
%
%[RMSmat,meanRMSECVvect,resMat,PredErrormat,meanPredErrorvect] =
%CVgrapher(X,Y,comp,N,M,ratio,method,response) where RMSmat is the matrix
%containing individual RMS values for a given loop and component value,
%meanRMSECVvect is a vector containing the average RMSECV values for a
%given number of components, resMat is a matrix containing the prediction
%residuals for a given number of components, PredErrormat is a matrix
%containing the residual percentage for a given number of components,
%meanPredErrorvect is the average residual percentage for a given number of
%components (in contrast to PredErrormat, which gives it per iteration,
%rather than averaged), X is the predictor matrix, with the first object
%being the wavelength vector, Y is the response matrix where each variable
%is a response vector, and the first variable is the object label vector
%(or insert a blank column), comp is the maximum number of components to be
%validated (note: this will start at 1 component, and work up to the max.
%This can be time consuming at high component values) M is the number of
%blocks to run, and therefore the number of sample plots which will be
%displayed, N is the number of iterations per block, ratio is the ratio of
%of test set split (ie, 0.8 uses 80% of the objects to create the model,
%and 20% to validate it), method is a boolean response for if the predictor
%matrix should be mean centered, and response is the variable from the Y
%matrix which will be the response vector (that is to say that a response
%of '1' would use the first column as the response vector).
%
%It should be noted that the cross-validation will be run N*M times per
%component, which will then generate M sample plots, one from each block.
%This means that N+2 figures will be created, one for each plot, an RMSECV
%plot and a residual percentage plot. So, if you use N=10, M=20 and
%comp=10, 2000 individual cross-validations will take place, and 22 figures
%will be created.
%
%Adam Joshua Armstrong, University of Bergen, 2018

RMSvect=[];
RMSmat=[];
resMat=[];
Remaining=M
YPredPermmat=[];
YPredSetmat=[];
RMSECVvect=[];
%Loop that runs blocks of cross-validaions.
for j=1:M
    %Loop that runs N iterations of cross-validation for each comp.
    for i=1:comp

[RMS,YPredPerm,YPredSet,resVect,PredErrorOuter,RMSECVcomp,YPredSetlabels,r2
]=MonteCarloCV(X,Y,i,N,ratio,method,response);
        RMSvect(i,1)=RMS;
        PredErrorvect(i,1)=PredErrorOuter;
        RMSECVvect(i,1)=RMSECVcomp;
    end
    R2(j,1)=r2;
    RMSECVmat(j,:)=RMSECVvect;
    figure
    scatter(YPredPerm,YPredSet)
    xlabel('Predicted Response')
    ylabel('Measured Response')
    ax=gca;
    ax.FontSize=16;
    title(['Predicted VS Measured. R2=' num2str(r2) '. ' num2str(comp) '
components.']);
    %Loop that labels each point on the predicted vs measured plot.
    for k=1:length(YPredSet)
        text(YPredPerm(k,1),YPredSet(k,1),num2str(YPredSetlabels(k,1)));
    end
    YPredPermmat(j,:)=YPredPerm;
    YPredSetmat(j,:)=YPredSet;
    RMSmat(j,:)=RMSvect;
    resMat(j,:)=resVect;
    PredErrormat(j,:)=PredErrorvect;
    Remaining=Remaining-1
end
meanRMSECVvect=mean(RMSECVmat);
meanPredErrorvect=mean(PredErrormat);
figure
plot(meanRMSECVvect)
xlabel('Number of Components')
ylabel('RMSECV')
ax=gca;
ax.FontSize=16;
figure
plot(meanPredErrorvect)
xlabel('Number of Components')
ylabel('Residual %')
ax=gca;
ax.FontSize=16;
end
```

9.8 MATLAB Code for the Normal Probability Plot Function

```
function [] = normalplotter(scores,labels)
%Normal probability plotting function.
%Usage:
%
%[] = normalplotter(scores,labels), where scores is the score vector, and
%labels is the label vector.
%
%Adam Joshua Armstrong, University of Bergen, 2018

[Y,I]=sort(scores);
prob=[];
%Loop creating the probability intervals.
for i=1:length(scores)
    prob(i,1)=(i-0.5)/length(scores);
end
figure
scatter(Y,prob)
%Loop that labels each point on the plot.
for k=1:length(scores)
    text(Y(k,1),prob(k,1),num2str(labels(I(k,1),1)));
end
end
```

9.9 MATLAB Code for the Embedded Selectivity Ratio Limit Based Monte Carlo Cross-Validation Engine

```
function [RMSmat,meanRMSECVvect,resMat,PredErrormat,meanPredErrorvect] =
optimalSRCVgrapher(X,Y,comp,N,M,ratio)
%Monte Carlo Cross-Validation function used specifically only via the
%optimalSR function, and is not meant to be used as a standalone function.
%
%Usage:
%[RMSmat,meanRMSECVvect,resMat,PredErrormat,meanPredErrorvect] =
%optimalSRCVgrapher(X,Y,comp,N,M,ratio), where RMSmat is the matrix
%containing individual RMS values for a given loop and component value,
%meanRMSECVvect is a vector containing the average RMSECV values for a
%given number of components, resMat is a matrix containing the prediction
%residuals for a given number of components, PredErrormat is a matrix
%containing the residual percentage for a given number of components,
%meanPredErrorvect is the average residual percentage for a given number of
%components (in contrast to PredErrormat, which gives it per iteration,
%rather than averaged), X is the predictor matrix, with the first object
%being the wavelength vector, Y is the response matrix where each variable
%is a response vector, and the first variable is the object label vector
%(or insert a blank column), comp is the maximum number of components to be
%validated (note: this will start at 1 component, and work up to the max.
%This can be time consuming at high component values) M is the number of
%blocks to run, and therefore the number of sample plots which will be
%displayed, N is the number of iterations per block, and ratio is the ratio
%of the test set split (ie, 0.8 uses 80% of the objects to create the
model,
%and 20% to validate it).
%
%Adam Joshua Armstrong, University of Bergen, 2018
RMSvect=[];
RMSmat=[];
resMat=[];
PredError=[];
YPredPermmat=[];
YPredSetmat=[];
RMSECVvect=[];
%Loop that runs blocks of cross-validaions.
for j=1:M
    %Loop that runs N iterations of cross-validation for each comp.
    for i=1:comp

[RMS,YPredPerm,YPredSet,resVect,PredErrorOuter,RMSECVcomp]=MonteCarloCV(X,Y
,i,N,ratio,0,0);
        RMSvect(i,1)=RMS;
        PredErrorvect(i,1)=PredErrorOuter;
        RMSECVvect(i,1)=RMSECVcomp;
    end
    RMSECVmat(j,:)=RMSECVvect;
    YPredPermmat(j,:)=YPredPerm;
    YPredSetmat(j,:)=YPredSet;
    RMSmat(j,:)=RMSvect;
    resMat(j,:)=resVect;
    PredErrormat(j,:)=PredErrorvect;
end
```

```
meanRMSECVvect=mean(RMSECVmat);
meanPredErrorvect=mean(PredErrormat);
end
```

9.10 MATLAB Code for the Selectivity Ratio Based MCCV Controller Engine

```
function [bestSR,SRlimit,bestSRlimit] =
optimalSR(X,Y,comp,N,M,ratio,rangelow,rangehigh,steps,response)
%Function that attempts to find the optimal SR value and number or
%components to use based on RMSECV.
%
%Usage:
%[bestSR,SRlimit,SR,bestSRlimit] =
%optimalSR(X,Y,comp,N,M,ratio,varsel,rangelow,rangehigh,steps,response),
%where bestSR is the matrix containing the best RMSECV value for the given
%SR limit, as well as the corresponding number of components used to
%achieve that RMSECV value, SRlimit is the vector containing the SR limits
%used, bestSRlimit is the best overall performing SR limit, X is the
%predictor matrix, Y is the response matrix, comp is the maximum number of
%components to use in cross-validation, N is the number of iterations per
%cross-validation block, M is the number of blocks, ratio is the ratio of
%of test set split (ie, 0.8 uses 80% of the objects to create the model,
%and 20% to validate it), rangelow is the lower bound for the SR limits to
%be tested, rangehigh is the upper bound for the SR limits to be tested,
%steps is the number of steps in addition to the lower bound that are to
%be tested (ie, steps=5 will check 6 SR values), and response is the
%response vector to be used, corresponding to the Y column it lays in.
%
%It should be noted that it is assumed that the first object in the X
%matrix is a wavelength vector, and this object will be removed.
%
%Adam Joshua Armstrong, University of Bergen, 2018
varsel=1;
Xred=X(2:end,:);
Yred=Y(:,response);
stepsize=(rangehigh-rangelow)/steps;
[Ypred,Yinit,tmat,pmat,beta,SR]=plsfunctNEW(Xred,Yred,comp,varsel);
%Loop that strips X for a given SR limit, and then runs cross-validation.
for i=rangelow:stepsize:rangehigh
    SRlimit(count,1)=i;
    [XSRstripped] = srstrip(X,SR,i);
    [RMSmat,meanRMSECVvect,resMat,PredErrormat,meanPredErrorvect] =
optimalSRCVgrapher(XSRstripped,Yred,comp,N,M,ratio);
    [M2,I]=min(meanRMSECVvect);
    bestSR(count,1)=I;
    bestSR(count,2)=M2;
end
[M3,I2]=min(bestSR);
bestSRlimit=SRlimit(I2(1,2),1);
figure
plot(SRlimit,bestSR(:,2))
xlabel('SR Limit')
ylabel('RMSECV')
ax=gca;
ax.FontSize=16;
end
```

9.11 MATLAB Code for the Outlier Detection Function

```
function [lev,studres] = OutDetect(Yinit,Ypred,tmat,labels)
%Outlier detection plotting function.
%Usage:
%
%[lev,studres] = OutDetect(Yinit,Ypred,tmat,labels), where lev is the
%leverage vector, studres is the Studentized residual vector, Yinit is the
%measured response vector, Ypred is the predicted response vector, tmat is
%the scores matrix, and labels is the label vector.
%
%Adam Joshua Armstrong, University of Bergen, 2018

comp=length(tmat);
r=Yinit-Ypred;
H=tmat*(tmat'*tmat)^(-1)*tmat'+1/length(tmat); %Hat matrix
lev=[];
MSE=mean(r.*r);
studres=[];
%Loop that calculates the leverage and Studentized residuals.
for i=1:length(H)
    lev(i,1)=H(i,i);
    studres(i,1)=r(i,1)/(sqrt(MSE*(1-lev(i,1))));
end
figure
scatter(lev,studres)
xlabel('Leverage')
ylabel('RStudent')
ax=gca;
ax.FontSize=16;
%Loop that labels the points in the plot.
for i=1:48
    text(lev(i,1),studres(i,1),num2str(labels(i,1)));
end
end
```

9.12 MATLAB Code for the Variable Selection Stripping Function, Based on the Selectivity Ratio

```
function [XSRstripped] = srstrip(X,SR,limit)
%SR based matrix variable stripping algorithm
%Usage:
%
%[Xstripped] = srstrip(X,SR,limit), where X is the predictor matrix,
%SR is the SR score vector, and limit is the SR score limit at which
%variables below this value will be removed.

%Adam Joshua Armstrong, University of Bergen, 2018
count=0;
XSRstripped=[];
for i=1:length(SR)
    if SR(i,1)>=limit
        count=count+1;
        XSRstripped(:,count)=X(:,i);%Builds the stripped matrix
sequentially
                                    %only when the value is above or equal
                                    %to the given limit.
    end
end

end
```