

Fachhochschule Kufstein-Tirol

Studiengang Data Science & Intelligent Analytics

Aufgabe zur LAB-Teilbenotung von Software Entwicklung I - LAB

**Verfasser:**

Alexander Hauser, BA

1810837583

Franz Innerbichler, MSc

1810837297

Paul Elmo Klein, BA MA

1810837063

Kufstein, am 10.03.2019

Inhalt

[Abbildungsverzeichnis III](#_Toc3146699)

[1. Projektbeschreibung und Problemstellung 4](#_Toc3146700)

[1.1. Beschreibung Sachverhalt 4](#_Toc3146701)

[1.2. Was ist RAMAN-Spektroskopie 4](#_Toc3146702)

[1.3. Aufbau einer Raman-Applikation 5](#_Toc3146703)

[1.4. Problemstellung 5](#_Toc3146704)

[2. Implementierung 6](#_Toc3146705)

[2.1. Python Bibliotheken 6](#_Toc3146706)

[2.2. Funktionen 7](#_Toc3146707)

[2.2.1. Datenbankerstellung 7](#_Toc3146708)

[2.2.2. Initialisieren und Konvertieren 7](#_Toc3146709)

[2.2.3. Import 9](#_Toc3146710)

[2.3. Berichterstellung 10](#_Toc3146711)

[2.3.1. PDF erstellen 10](#_Toc3146712)

[2.4. Partial Least Squares (=Projection to Latent Structures) 11](#_Toc3146715)

[2.4.1. Python code 11](#_Toc3146716)

[2.5. Grafische Benutzeroberfläche 17](#_Toc3146718)

[2.6. Diskussion der Lösung/Stärken und Schwächen 19](#_Toc3146720)

[2.6.1. Stärken 19](#_Toc3146721)

[2.6.2. Optimierungsmöglichkeiten 20](#_Toc3146722)

[3. Literaturverzeichnis 21](#_Toc3146723)

#### Abbildungsverzeichnis

[Abbildung 1Raman Gerät und Sonde an einem Fermenter mit bakterieller Wirkstoffproduktion 4](#_Toc3147211)

[Abbildung 2: Aufbau Raman-Spektroskopie 5](C:\\FH Ku\\Data Science\\Software_Entwicklung_Projektarbeit_Hauser_Innerbichler_Klein.docx" \l "_Toc3147212)

[Abbildung 3 Spektrenplot in Software R mit Paket „hyperspec“ 6](#_Toc3147213)

[Abbildung 4: Screenshot “def createColumnHeaders(switch):” 7](#_Toc3147214)

[Abbildung 5: Screenshot “def createTable(query, headers):” 7](#_Toc3147215)

[Abbildung 6: Screenshot "def formatFileName(filename):" 7](C:\\FH Ku\\Data Science\\Software_Entwicklung_Projektarbeit_Hauser_Innerbichler_Klein.docx" \l "_Toc3147216)

[Abbildung 7: Screenshot "def getlastID():" 8](C:\\FH Ku\\Data Science\\Software_Entwicklung_Projektarbeit_Hauser_Innerbichler_Klein.docx" \l "_Toc3147217)

[Abbildung 8: Screenshot "def initialData():" 8](C:\\FH Ku\\Data Science\\Software_Entwicklung_Projektarbeit_Hauser_Innerbichler_Klein.docx" \l "_Toc3147218)

[Abbildung 9: Screenshot "def convertSPC(filepath):" 8](C:\\FH Ku\\Data Science\\Software_Entwicklung_Projektarbeit_Hauser_Innerbichler_Klein.docx" \l "_Toc3147219)

[Abbildung 10: Screenshot "def importCSV(filepath):" 9](C:\\FH Ku\\Data Science\\Software_Entwicklung_Projektarbeit_Hauser_Innerbichler_Klein.docx" \l "_Toc3147220)

[Abbildung 11: Screenshot "def importTrainCSVfromFolder(directory):" 9](C:\\FH Ku\\Data Science\\Software_Entwicklung_Projektarbeit_Hauser_Innerbichler_Klein.docx" \l "_Toc3147221)

[Abbildung 12: Screenshot "def importTrainData(headers, filepath):" 10](#_Toc3147222)

[Abbildung 13: Screenshot "def getDataFromCass(query):" 10](#_Toc3147223)

[Abbildung 14: Screenshot Erstellung Daten- und Trainingstabelle 10](C:\\FH Ku\\Data Science\\Software_Entwicklung_Projektarbeit_Hauser_Innerbichler_Klein.docx" \l "_Toc3147224)

[Abbildung 15: Screenshot PDF-Erstellung 10](C:\\FH Ku\\Data Science\\Software_Entwicklung_Projektarbeit_Hauser_Innerbichler_Klein.docx" \l "_Toc3147225)

[Abbildung 16: Python packages für die PLS 11](#_Toc3147226)

[Abbildung 17 Sortierung der RAMAN Spektren 11](#_Toc3147227)

[Abbildung 18 Sortierung der Y-Variablen 11](#_Toc3147228)

[Abbildung 19: Code zum Löschen von nicht notwendigen Wellenlängen 12](#_Toc3147229)

[Abbildung 20: Code zum Breinigen der Daten (noise cancelling) 12](#_Toc3147230)

[Abbildung 21: Code für den Spektralplot nach Smoothing und Bereinigung 12](#_Toc3147231)

[Abbildung 22: Spektralplot alle Wellenlängen nach Index 13](#_Toc3147232)

[Abbildung 23: Ausschnitt der Spektren 1900-2500 (Index) 13](#_Toc3147233)

[Abbildung 24: Train- und Testdatensatz 13](#_Toc3147234)

[Abbildung 25 Funktion „prediction“ Teil1 14](#_Toc3147235)

[Abbildung 26: Funktion „prediction“ Teil2 mit code für den „No of Components“-Plot 14](#_Toc3147236)

[Abbildung 27: Komponentenplot (rotes Kreuz= Komponente mit niedrigster MSEP) 15](#_Toc3147237)

[Abbildung 28: Funktion „prediction“ Teil3: Berechnung einiger Kennzahlen der PLS und Plot der gemessenen und der vorhergesagten Werte 15](#_Toc3147238)

[Abbildung 29: Gemessene Werte (blau) gegen berechnete Werte des „Test“-Datensatzes (hohe Werte können nicht vorhergesagt werden; Messung eines anderen Metaboliten? Gemessene Werte falsch?) 16](#_Toc3147239)

[Abbildung 30: Funktion prediction Teil 4: Code für Plot „Observed vs Predicted 2“ 16](#_Toc3147240)

[Abbildung 31: Plot „Observed vs Predicted 2“ grüne Linie: 0-Fehler-Vorhersage; blaue Linie: aktuelle Regressionslinie 17](#_Toc3147241)

# Projektbeschreibung und Problemstellung

Die vorliegende Arbeit dient als Prüfungsleistung der Lehrveranstaltung „Software Entwicklung I -LAB“ im Data Science and Intelligent Analytics Master an der Fachhochschule Kufstein Tirol. Ziel der Arbeit ist es eine Python-Implementierung für die Erstellung statistischer Berechnungen und mathematische Grafiken zu entwerfen. Zur Verbesserung der Anwenderfreundlichkeit dient eine grafische Benutzeroberfläche.

Die Arbeit beschäftigt sich mit der Installation eines Raman-Spektrometers im Bereich der pharmazeutischen Industrie. Das Augenmerk liegt hierbei auf der Verarbeitung, Speicherung und Auswertung der entstehenden Frequenz-Analysen.



Abbildung 1Raman Gerät und Sonde an einem Fermenter mit bakterieller Wirkstoffproduktion

## Beschreibung Sachverhalt

In einer Fermentationsproduktionseinheit in der pharmazeutischen Industrie sollen mit Hilfe eines Raman-Spektroskops in-situ Echtzeitanalysen von Metaboliten[[1]](#footnote-2) der Escherichia coli Bakterien oder CHO-Zellen[[2]](#footnote-3) gemacht werden gemacht werden. Die hierbei entstehenden Daten sollen analysiert und gespeichert werden. Es handelt sich um einen Batch- Datenstrom.

## Was ist RAMAN-Spektroskopie

**Die Raman-Spektroskopie**, ist eine auf dem Raman-Effekt beruhende spektroskopische Methode, mit der Moleküle in allen Aggregatzuständen untersucht und identifiziert werden können. Dieser Effekt basiert auf der Erscheinung, dass sich im Streuspektrum von mit monochromatischem Licht bestrahlten Molekülen Linien befinden, deren Frequenzen sich von der des eingestrahlten Lichtes unterscheiden. Sie entstehen durch Molekülschwingungen und -rotationen.[[3]](#footnote-4) Diese Art der Spektroskopie ist eine der wichtigsten und gebräuchlichsten Analysemethoden der Chemie und Biologie sowie ihrer Teilbereiche. Namensgeber ist Chandrasekhara Venkata Raman, ein indischer Physiker und späterer Nobelpreisträger, der diese Art der Lichtstreuung und ihre Nutzung erstmals beobachtete.

Ein großer Vorteil dieser Methode ist, dass die zu untersuchenden Proben ohne physischen Kontakt oder Zerstörung untersucht und getestet werden können. Die Größe und Menge der Proben sind ebenfalls sehr variabel.[[4]](#footnote-5) Die Methode macht sich zu Nutze, dass jedes chemische Element eine unverwechselbare Streuung erzeugt.

Die Einsatzfelder in der Praxis sind vielfältig. Mit Hilfe der Raman-Spektroskopie können beispielsweise echte von Industrie-Diamanten unterschieden werden, die Qualität von Treibstoffen bewertet werden oder Gesteinsbohrungen analysiert werden.

## Aufbau einer Raman-Applikation

Ein Spektroskop ist ein optisches Gerät mit dem Licht in sein Spektrum zerlegt und visuell untersucht werden kann. Im vorliegenden Fall wird das Licht mit Hilfe einer Kamera erfasst, man spricht deshalb von einem Spektrometer.[[5]](#footnote-6)

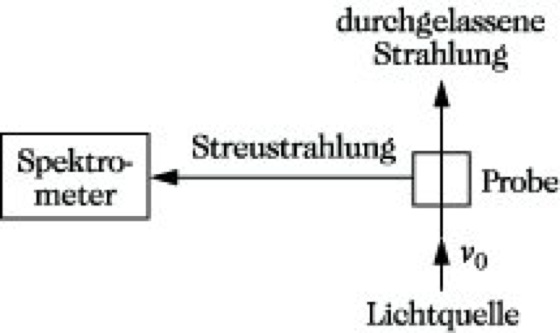
Abbildung 1 zeigt einen schematischen Aufbau der Versuchsanordnung: Bei dem zur Anregung des Raman-Spektrums eingesetzte Lichtquelle handelt es sich um einen Laser. Dieser wird mit einer Linse auf die Probe fokussiert. Die aus der Probe austretende Strahlung wird senkrecht zur Laserstrahlrichtung, bei festen und undurchsichtigen Proben auch in Rückwärtsrichtung, auf den Spalt eines Mehrfachmonochromators abgebildet, um die Raman-Strahlung von der Streustrahlung zu trennen. Mit Hilfe einer Kamera wird die Raman-Strahlung nachgewiesen und mittels eines Computers ausgewertet. Ihr Spektrum besteht aus „Linien“, die einzelnen Schwingungs- oder Rotationsbewegungen der Moleküle zugeordnet werden können. Aus ihrer Intensität kann die Konzentration der Molekülart in der Probe, aus ihrer spektralen Breite und Form ihre Wechselwirkung mit anderen oder gleichen Molekülen in der Probe bestimmt werden.[[6]](#footnote-7)

Abbildung 2: Aufbau Raman-Spektroskopie

## Problemstellung

Die Herausforderungen stellen sich in der Verarbeitung, Analyse und Speicherung der entstehenden Daten. Wie bereits erläutert entsteht während der Spektroskopie ein Datenstrom. Bei mehreren Messungen pro Fermentation und mehreren tausend überprüften Wellenlängen pro Messung akkumulieren sehr schnell eine große Menge an Daten. Dies kann in weiterer Folge zu einem Problem beim Speicherplatz führen, je länger die Apparatur arbeitet.[[7]](#footnote-8)

Je schneller neue Daten eintreffen, desto schneller müssen diese auch verarbeitet und analysiert werden. Sollte die Verarbeitung nicht mit der Eintrittsgeschwindigkeit mithalten können, besteht die Gefahr, dass die Anfragen noch bearbeitet werden während neue Daten schon eintreffen. [[8]](#footnote-9) Die sich stets erweiternde und aktualisierende Datenbasis macht es notwendig, sowohl die Lese- und Schreibgeschwindigkeit, als auch die Berechnungsgeschwindigkeit zu optimieren. Derzeit werden RAMAN-Messungen nur probeweise durchgeführt und mit einer proprietären Software verwertet. Die Auswertung und Bearbeitung erfolgen mit Hilfe der Statistik-Software R-Studio (mit den Packages „hyperspec“ und „pls“, ein Beispielplot in Abbildung 3). Diese Vorgehensweise soll vereinfacht werden, in dem der Nutzer nur noch ein Programm bzw. eine Oberfläche bedienen muss.

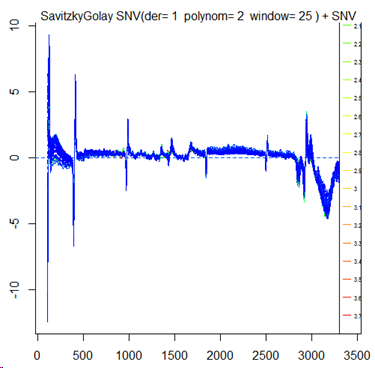


Abbildung 3 Spektrenplot in Software R mit Paket „hyperspec“

# Implementierung

In diesem Abschnitt werden die Implementierung und die Funktionen der Algorithmen in Python Schritt für Schritt erklärt. Zum besseren Verständnis sind Screenshots mit relevantem Code ergänzt. Als Datenbank wird ein Apache Cassandra Cluster verwendet.

## Python Bibliotheken

Um die Funktionalität von Python zu erweitern wird häufig mit Programmbibliotheken gearbeitet. Für eine Vielzahl von Einsatz- und Anwendungsfällen gibt es diese Bibliotheken. Der in dieser Arbeit behandelte Sachverhalt erfordert wissenschaftliche, numerische Berechnungen und mathematische Darstellungen und Visualisierungen wie Matrizen und Vektoren. Die besten Ergebnisse liefert eine Kombination aus den Bibliotheken NumPy, Matplotlib, SciPy, pandas und scikit-learn. Die Ausgabe der Ergebnisse soll in einem PDF-Dokument erfolgen, deshalb wird zusätzlich noch die xhtml2PDF-Bibliothek verwendet. Für die Erzeugung von .csv-Dateien wird auf ein Git-Repository zurückgegriffen. Eine Beschreibung erfolgt im Kapitel 5.1.

## Funktionen

### Datenbankerstellung

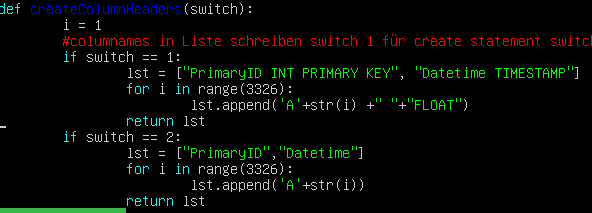
Der „pyDataHelper“ beinhaltet alle Funktionen, die sich mit der Erstellung oder Manipulation der Datenbasis befassen. Die Funktion „**createColumnHeaders(switch)**“ erzeugt die Spaltenüberschriften für die Datentabelle. Ziel ist die effiziente Erzeugung von 3.326 Spaltenüberschriften. Die Variable switch nimmt hierbei Werte von 1 oder 2 an. Option 1 erzeugt die Spalten und Datentypbezeichnungen für das Create-Statement, Option 2 erzeugt lediglich die Spaltennamen für die Abfragen.

Abbildung 4: Screenshot “def createColumnHeaders(switch):”

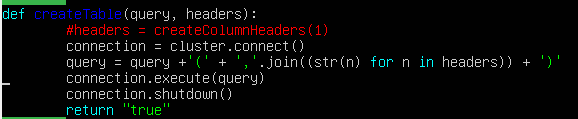
Die Funktion **createTable(query,headers)** dient dazu, die Daten- und Trainingstabelle zu erstellen. Hierzu wird das Create-Statement in Form eines Strings und die Überschrift als Result-String der Funktion createColumnHeaders übergeben.

Abbildung 5: Screenshot “def createTable(query, headers):”

### Initialisieren und Konvertieren

Das Raman-Spektrometer liefert die Daten in einer fixierten Nomenklatur (YYYY-MM-DDHH-MM-SS\_SECTRUM.SPC). Für die Weiterverarbeitung wird das Datum aus dem Dateinamen extrahiert. Dies geschieht in dem die ersten zehn Zeichen des Dateinamens extrahiert und von der Funktion **formatFileName(filename)** zurückgegeben werden.

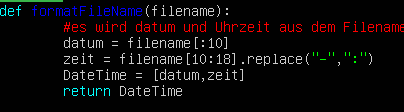


Abbildung 6: Screenshot "def formatFileName(filename):"

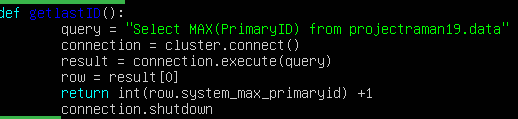
Cassandra unterstützt im Rahmen ihrer Funktionalität keine automatische Erhöhung der Zeilennummer (Auto-Increment). Dahingehend wurde diese Anforderung so gelöst, dass die Funktion **getlastID()** bei jedem Import die maximale ID + 1 zurückgibt.

Abbildung 7: Screenshot "def getlastID():"

Die Funktion **InitialData()** wird nur einmalig aufgerufen um den ersten Eintrag in der Datentabelle zu schreiben. Wäre das nicht der Fall, würde die Funktion getlastID() einen Fehler erzeugen. Diese Zeile wird in der Analyse ausgeklammert.

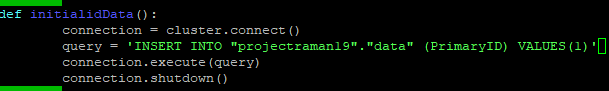
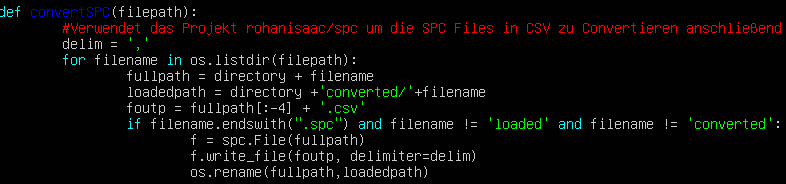


Abbildung 8: Screenshot "def initialData():"

Das Raman-Spektrometer erzeugt die Daten im SPC-Format. Dieses ist für eine weitere Verarbeitung ungeeignet. Es erfolgt deshalb eine Konvertierung in das CSV-Format.[[9]](#footnote-10) Mit der Funktion **convertSPC(filepath)** wird der Importordner nach Datensätzen im SPC-Format durchsucht, alle gefundenen Datensätze in CSV konvertiert und schließlich im Unterordner „converted“ abgelegt.

Abbildung 9: Screenshot "def convertSPC(filepath):"



### Import

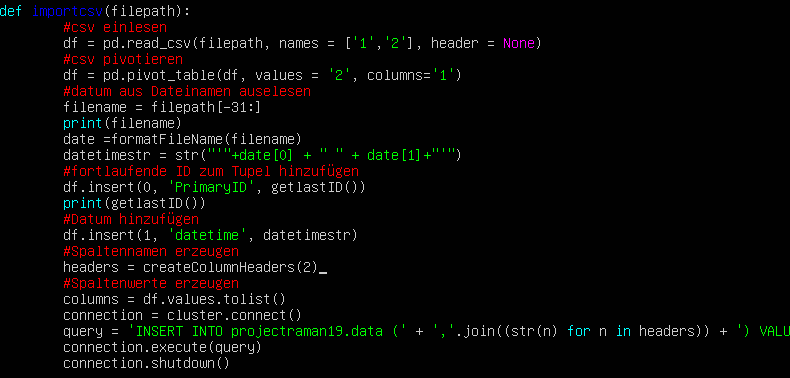
Die Funktion **importCSV(Filepath)** liest das übergebene CSV-File ein. Da die Raman-Daten als Tiefenstruktur überliefert werden, muss diese Struktur erst pivotiert werden. Hierfür wird auf die Bibliothek „pandas“ zurückgegriffen. Anschließend wird die oben bereits beschriebene Funktion formatFileName() aufgerufen um das Datum für das Import-Statement zu schreiben. Um die Datenzeile zu vervollständigen wird ebenfalls die aktuelle PrimaryID über die Funktion getlastID() in den Datensatz eingefügt. Nachdem die Spaltenbeschriftungen über die Funktion createColumnHeaders() erzeugt wurden, werden diese beiden Informationen in die Abfrage übergeben und ausgeführt.

Abbildung 10: Screenshot "def importCSV(filepath):"

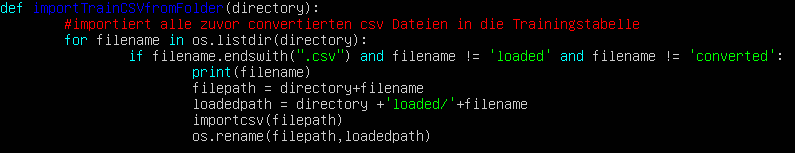
Mit der Funktion **importCSVFromFolder(directory)** werden die vorher konvertierten Dateien iterativ importiert. Hierfür wird über die Funktion importCSV() jedes CSV-File importiert und im Anschluss in den Ordner „loaded“ verschoben. Final wird für die GUI eine Erfolgsmeldung ausgegeben.

Abbildung 11: Screenshot "def importTrainCSVfromFolder(directory):"

Mit Hilfe der Funktion **importTrainData(headers, filepath)** werden die Trainingsdaten werden bei jedem Import neu geladen. In diesem Zusammenhang werden zuerst die alten Y-Daten zuerst gelöscht, um im Anschluss durch das eingelesene CSV zu iterieren und jede Zeile einzulesen.

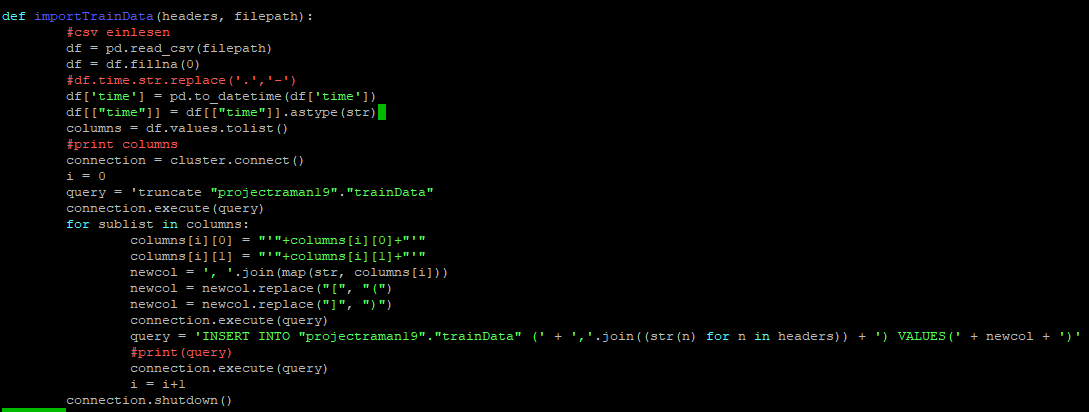


Abbildung 12: Screenshot "def importTrainData(headers, filepath):"

Die Funktion **getDatafromCass(queries)** setzt Abfragen an den Cassandra Cluster ab. Anschließend werden diese als pandas-Datensatz zurückgegeben. Dieser stellt die Basis für die Datenbeschaffung der Regression dar.

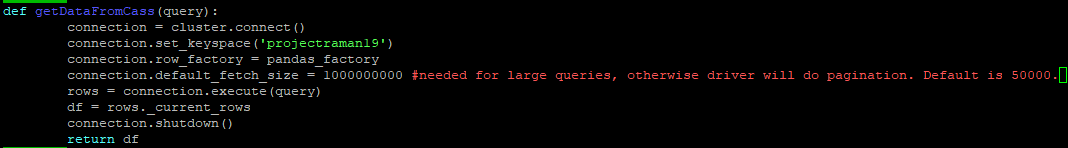


Abbildung 13: Screenshot "def getDataFromCass(query):"

## Berichterstellung

### PDF erstellen

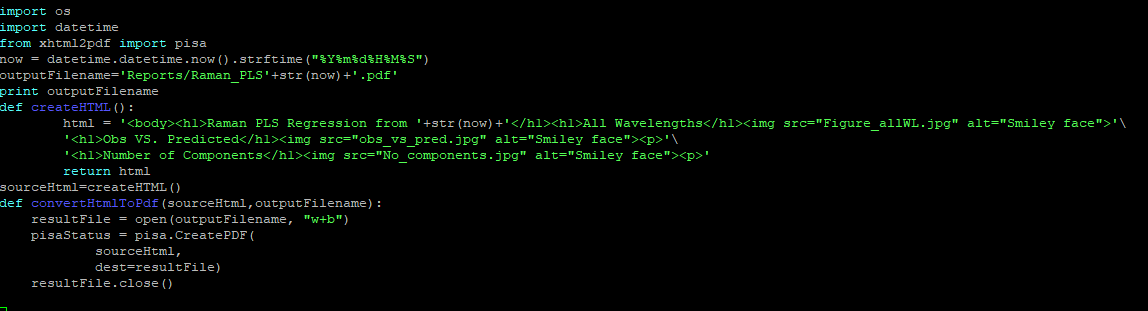
Das **pyCreatePDF.py**-File dient dazu, um auf Basis der durchgeführten Plots einen Report zu erstellen, welcher anschließend mit einem Time-stamp versehen im Verzeichnis „Reports“ abgelegt wird. Hierfür wird die Bibliothek xhtml2pdf verwendet. Diese Funktion wird aktuell nach der Durchführung der Regression aufgerufen und automatisch geöffnet. Ebenfalls kann das Reports Verzeichnis händisch geöffnet werden, um vergangene Visualisierungen abzurufen.

Abbildung 15 Screenshot "pyCreatePDF.py"

## Partial Least Squares (=Projection to Latent Structures)

Die Analyse der 3326 Wellenlängendaten wird mit einem PLS Algorithmus vorgenommen. PLS verwendet im Gegensatz zur PCA Regression nur so wenig Hauptkomponenten wie möglich. Eine sehr gute Abhandlung über PLS kann in folgendem Link nachgelesen werden:

<https://learnche.org/pid/latent-variable-modelling/projection-to-latent-structures>

Es gibt sehr viel Spielarten von PLS weshalb sich das vorliegende Projekt auf die Standard-PLS mit SIMPLS als Algorithmus beschränkt, ohne auf die Unterschiede einzugehen.

Teile des Codes wurden von der Website: <https://www.idtools.com.au/partial-least-squares-regression-python/>

kopiert

### Python code

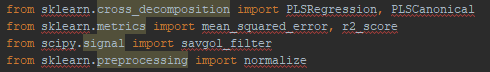
Es werden für das Preprocessing und die PLS folgende Python packages benötigt:

Abbildung 16: Python packages für die PLS

Nach dem Import der Daten vom Cassandra Server der FH Kufstein (Beschreibung in 2.2 und in der Gruppenarbeit derselben Gruppe für Data Engineering, Leiter M. Kohlegger) werden die Daten sortiert und von den beschreibenden Spalten befreit:

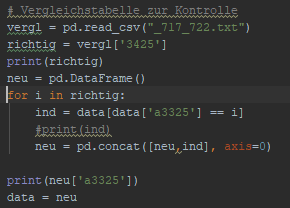


Abbildung 17 Sortierung der RAMAN Spektren

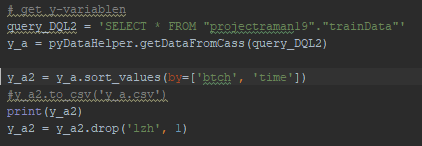
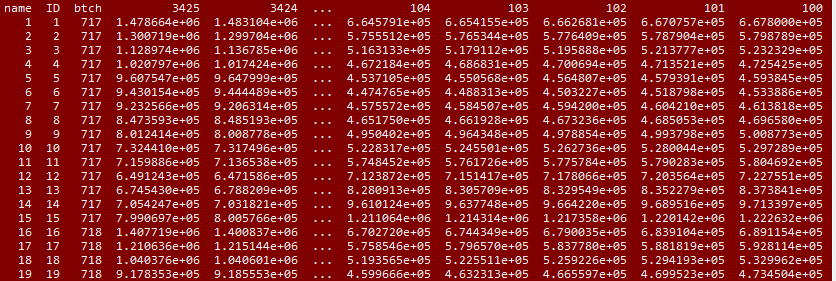


Abbildung 18 Sortierung der Y-Variablen

Tabelle 1: Raman daten: Jede Zeile=1 Messung, Jede Spalte ist eine Wellenlänge



Dann werden die letzten 301 Wellenlängen (von 100-400) gelöscht, weil sie nur Noise enthalten; die Rohspektren werden geplottet und als png ausgegeben. Preprocessing wird mit Savitzky-Golay Algorithmus und Standard Normal Variate gemacht: Fitten eines Polynoms 2. Ordnung über 25 Wellenlängen, Berechnung der ersten Ableitung und Normalisierung der Daten. Zwei Plots zeigen die gesamten bereinigten Wellenlängenbereich und einen Ausschnitt von 1900 – 2500). Zur Berechnung der Prediktionsfähigkeit des Modells werden die Daten in einen Trainings- und einen Testdatensatz geteilt (Abb 15-20).

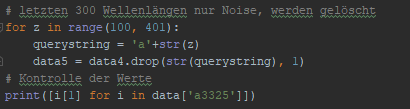


Abbildung 19: Code zum Löschen von nicht notwendigen Wellenlängen

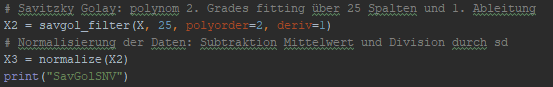


Abbildung 20: Code zum Breinigen der Daten (noise cancelling)

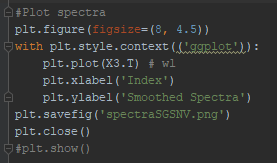


Abbildung 21: Code für den Spektralplot nach Smoothing und Bereinigung

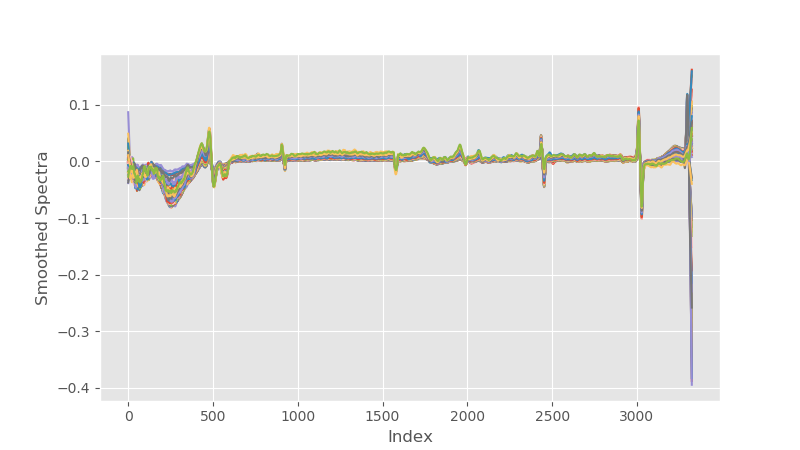


Abbildung 22: Spektralplot alle Wellenlängen nach Index

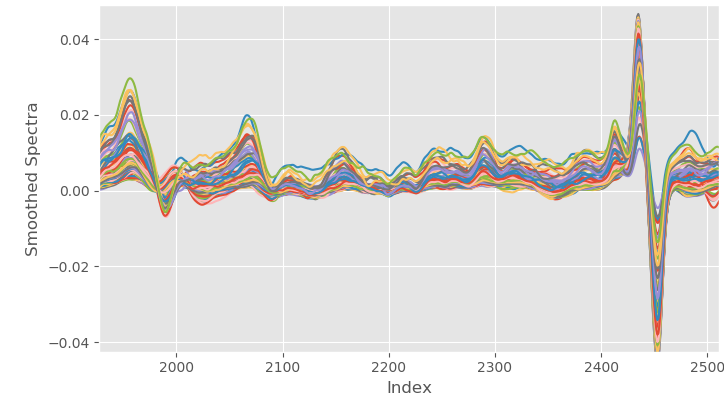


Abbildung 23: Ausschnitt der Spektren 1900-2500 (Index)

Nach der Datenvorbereitung werden die Daten in einen Trainingsdatensatz und einen Testdatensatz aufgeteilt (Abb. )

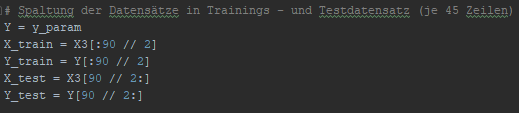


Abbildung 24: Train- und Testdatensatz

Die Funktion „prediction“ führt die PLS durch, und erzeugt Prediktionswerte für den Testdatensatz. Der „mse\_p“ = Mean Squared Error of Prediction wird für eine maximale Anzahl von 40 Hauptkomponenten berechnet. Der Verlauf von mse\_p pro Komponente wird als „NoofComp.png“ geplottet (Optimum bei 4 Komponenten). Schließlich wird die PLS Regression nochmal mit der optimalen Anzahl an Komponenten (mit minimalem MSEP = „msemin“) gerechnet. Der Plot „OvsP.png“ zeigt an, ob die Vorhersage gelungen ist (rote und blaue Kreuze sollten überlappen). Der Plot „OvsP2.png“ enthält auch die wichtigsten Kennzahlen der PLS Regression (Abb. 21-28).

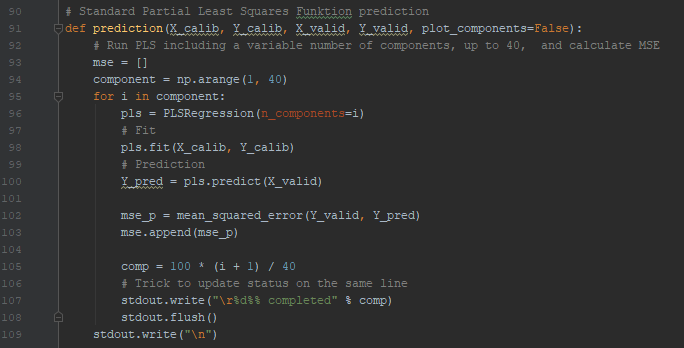


Abbildung 25 Funktion „prediction“ Teil1

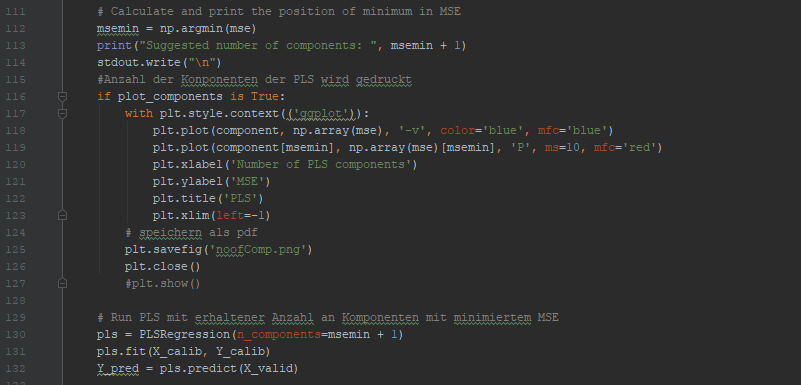


Abbildung 26: Funktion „prediction“ Teil2 mit code für den „No of Components“-Plot

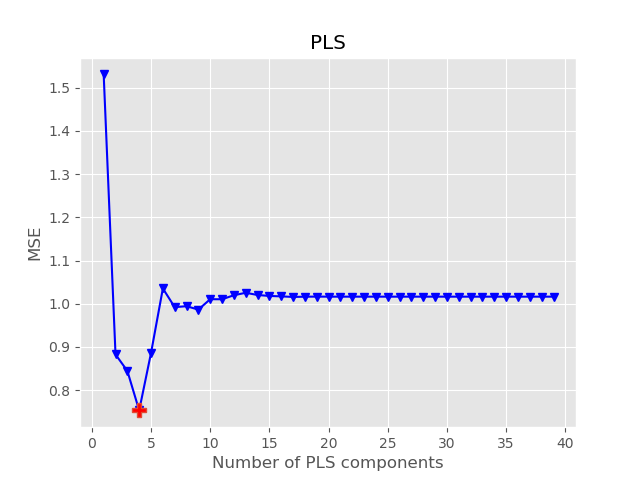


Abbildung 27: Komponentenplot (rotes Kreuz= Komponente mit niedrigster MSEP)

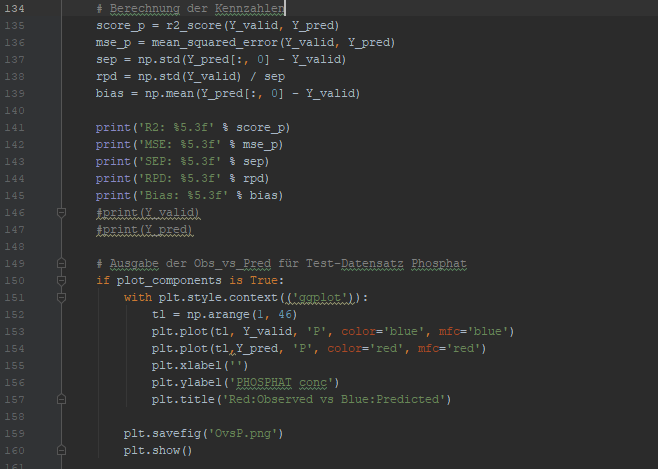


Abbildung 28: Funktion „prediction“ Teil3: Berechnung einiger Kennzahlen der PLS und Plot der gemessenen und der vorhergesagten Werte

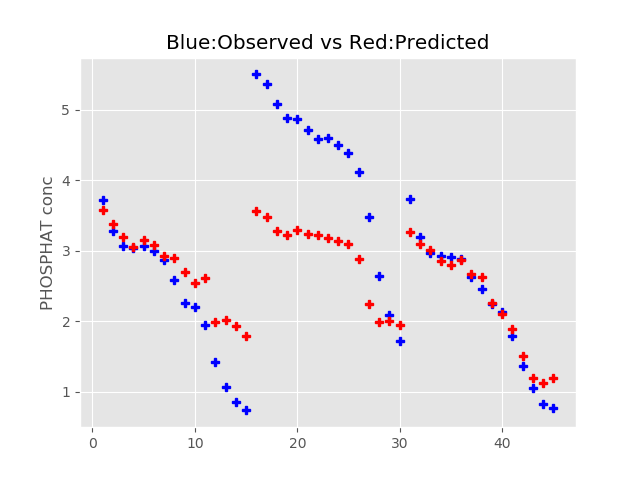


Abbildung 29: Gemessene Werte (blau) gegen berechnete Werte des „Test“-Datensatzes (hohe Werte können nicht vorhergesagt werden; Messung eines anderen Metaboliten? Gemessene Werte falsch?)

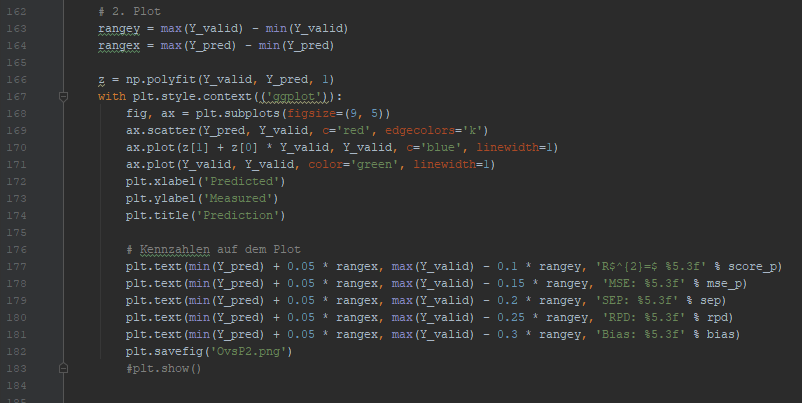


Abbildung 30: Funktion prediction Teil 4: Code für Plot „Observed vs Predicted 2“

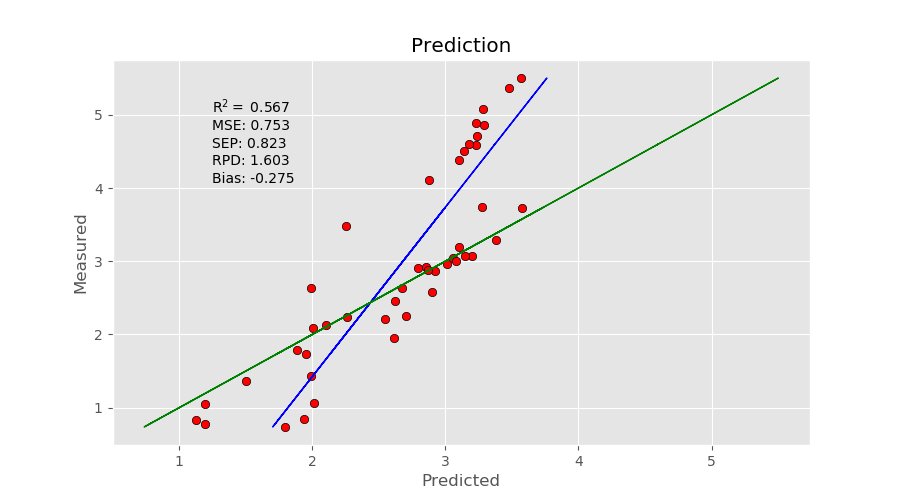
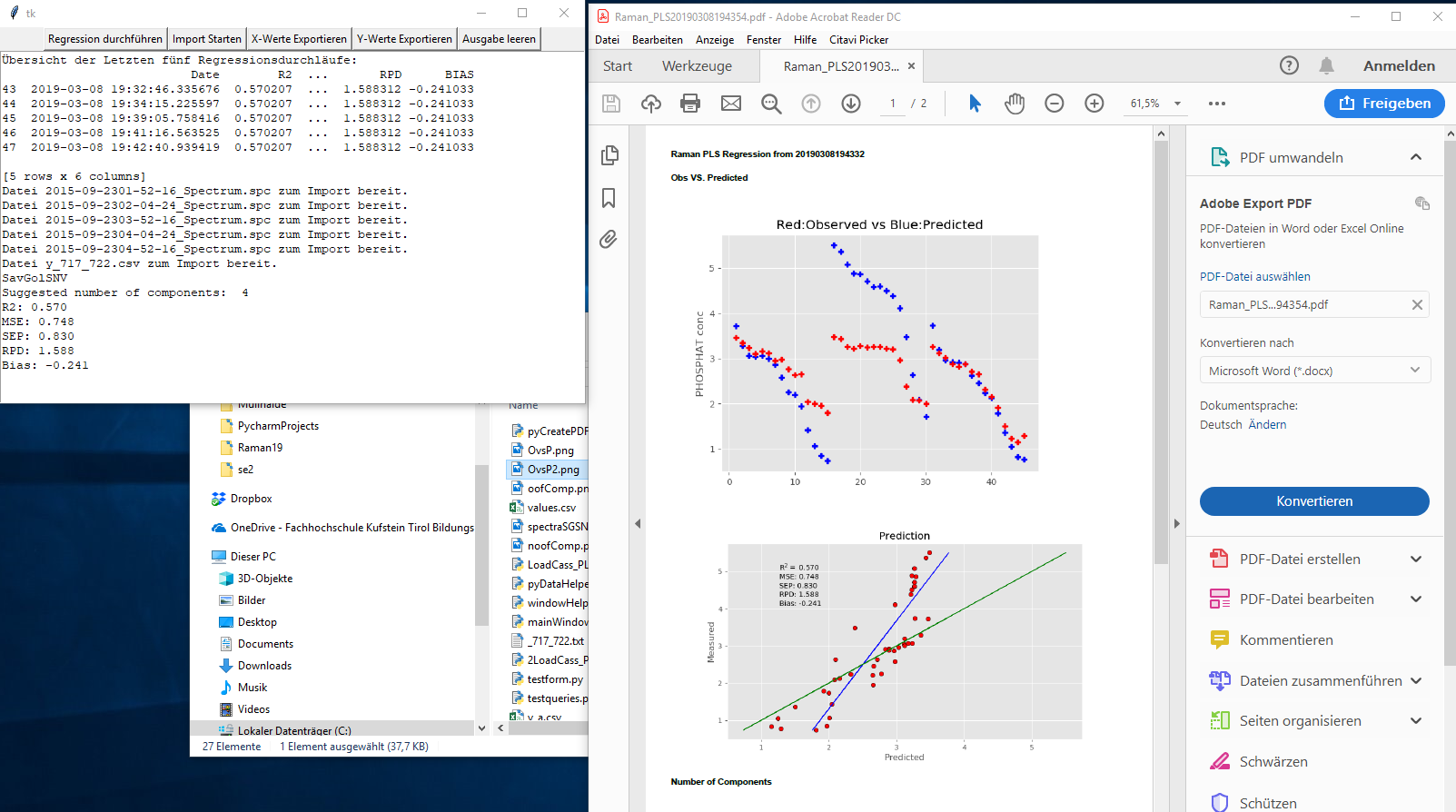


Abbildung 31: Plot „Observed vs Predicted 2“ grüne Linie: 0-Fehler-Vorhersage; blaue Linie: aktuelle Regressionslinie

## Grafische Benutzeroberfläche



Die grafische Benutzeroberfläche wurde mittels TKInter umgesetzt. Der Obere Quadrant stellt den Funktionsbereich dar. Hier können folgende Funktionen durchgeführt werden:

* Regression durchführen
  + Startet die Regression und gibt sowohl die Plots, als auch die Messwerte im unterliegenden Textbereich aus.
* Import starten
  + Startet den Importprozess für die X als auch die Y Werte auf.
* X-Werte exportieren und Y-Werte exportieren
  + Stellt die X und Y Werte im csv Format im Ordner “export” zur Verfügung. Dies ermöglicht eine weitere Auswertung in anderen Tools.
* Ausgabe leeren
  + Löscht den Text im Ausgabebereich

Für diese Funktionalitäten wird auf das Modul “pyWindowHelper.py” zurückgegriffen.

Der Textbereich dient zur Ausgabe von Informationen, wie beispielsweise die Verfügbarkeit von Importdaten oder der Ausgabe der Messwerte der Regression. Initial werden auch die letzten fünf Messwerte der Spektroskopie zurückgegeben.





## Diskussion der Lösung/Stärken und Schwächen

In der vorliegenden Arbeit wurde für den Einsatz eines RAMAN-Spektrogramms im Bereich der pharmazeutischen Forschung ein Software-Prototyp für die Speicherung und Auswertung der entstehenden Daten geschaffen. Gegenüber dem Ausgangszustand wurde eine Verbesserung in zahlreichen Punkten erreicht:

### Stärken

* Die Speicherung der Daten erfolgt in einer auf den Sachverhalt optimierten NoSQL-Datenbank (Cassandra-Cluster).
* Die Usability wurde durch die Erstellung einer grafischen Benutzeroberfläche erheblich vereinfacht und ist nun für einen erweiterten Anwenderkreis verständlich.
* Die Usability und Maintainability wurde durch die Implementierung in Python verbessert. Im vorherigen Zustand wurden mehrere Software-Programme verschiedener Hersteller/Autoren verwendet (u.a. R, R-Studio und Software des Spektrogramm-Herstellers). Bei Softwareübergreifender Zusammenarbeit bestehen immer Risiken hinsichtlich der Kompatibilität und Schnittstellen.
* Die Implementierung erfolgte unter ausschließlicher Verwendung von Open-Source-Software. Dies kann u.a. positive Auswirkungen auf die Total-Cost-of-Ownership haben.
* Die Programme wurden so geschrieben, dass zusätzliche Funktionen je nach Bedarf einfach hinzugefügt werden können. Wie bereits in Kapitel 4.2 erwähnt sollen in einem weiteren Ausbauzustand zu speziellen Zeiten automatisiert Aufgaben ausgeführt werden.

### Optimierungsmöglichkeiten

* Die Verarbeitungsgeschwindigkeit könnte erhöht werden.
* Die RAMAN Spektren sind nicht sortiert; Eine Verbesserung wäre eine ID vor der Einpflegung in die Cassandra an das Spektrum zu hängen. Der Zeitstempel könnte nicht eindeutig sein.
* Python ist etwas längsamer als R. Möglicherweise kann man mit Codeoptimierung die Verarbeitungsgeschwindigkeit erhöhen.

# Literaturverzeichnis

Babcock, B., Babu, S., Datar, M., Motwani, R., & Widom, J. (2002). *Models and Issues in Data Stream Systems.* Stanford University of California, Computer Science, Stanford. Abgerufen am 25. 02 2018 von https://infolab.usc.edu/csci599/Fall2002/paper/DML2\_streams-issues.pdf

Carl Zeiss Spectroscopy GmbH. (2019). *Raman-Spektroskopie*. Abgerufen am 27. 02 2019 von https://www.zeiss.de/spectroscopy/loesungen-und-anwendungen/measuring-principle/raman.html#einfuehrung

Merz, S., & Antwerpes, F. (17. 03 2011). *Metabolit*. Abgerufen am 02. 03 2019 von https://flexikon.doccheck.com/de/Metabolit

Rohan, I. (26. 01 2018). *Module for reading, exploring and converting \*.spc spectroscopic binary data in Python.* Abgerufen am 02. 03 2019 von https://github.com/rohanisaac/spc

Römer, G., & Beutler, B. (27. 09 2012). *CHO-Zelle*. Abgerufen am 02. 03 2019 von https://flexikon.doccheck.com/de/CHO-Zelle

Spektrum Akademischer Verlag. (1998). *Lexikon der Chemie*. Abgerufen am 27. 02 2019 von Raman-Spektroskopie: https://www.spektrum.de/lexikon/chemie/raman-spektroskopie/7804

Spektrum Akademischer Verlag. (1999). *Lexikon der Optik*. Abgerufen am 27. 02 2019 von Raman-Spektroskopie: https://www.spektrum.de/lexikon/optik/raman-spektroskopie/2746

Wikipedia. (22. 03 2018). *Spektroskop*. Abgerufen am 28. 02 2019 von https://de.wikipedia.org/wiki/Spektroskop

1. Metaboliten sind Substanzen, die als Zwischenstufen oder als Abbauprodukte von Stoffwechselvorgängen des Organismus entstehen. Androsteron ist bspw. ein in der Leber gebildeter Metabolit des Sexualhormons Testosteron. – (Merz & Antwerpes, 2011) [↑](#footnote-ref-2)
2. Abgekürzt von englisch Chinese Hamster Ovary und bezeichnet immortalisierte Zellen aus den Eierstöcken der chinesischen Hamsterart Cricetulus griseus. Diese finden vielseitige Anwendung im Bereich der Molekulargenetik, Biotechnologie, sowie der Molekularbiologie. – Vgl. (Römer & Beutler, 2012) [↑](#footnote-ref-3)
3. Vgl. (Spektrum Akademischer Verlag, 1998) [↑](#footnote-ref-4)
4. Vgl. (Carl Zeiss Spectroscopy GmbH, 2019) [↑](#footnote-ref-5)
5. Vgl. (Wikipedia, 2018) [↑](#footnote-ref-6)
6. Vgl. (Spektrum Akademischer Verlag, 1999)(auch Abbildung 1) [↑](#footnote-ref-7)
7. Vgl. (Babcock, Babu, Datar, Motwani, & Widom, 2002, S. 7) [↑](#footnote-ref-8)
8. Vgl. Ebd. [↑](#footnote-ref-9)
9. Für diese Funktion wurde auf eine bereits bestehende Code-Basis zurückgegriffen siehe (Rohan, 2018). [↑](#footnote-ref-10)