**Projektbericht zum Modul Information Retrieval und Visualisierung Sommersemester 2022**

# Visualisierung des Datensatzes Student Behaviour

Paul Tebbe, Matrikelnummer: 216237588

21.12.2022

---

Link zum GitLab-Repository:
`https://github.com/TornadoTebbe/Elm-Visualisation`

Link zur öffentlich zugänglichen Webseite (GitLab Pages):
`https://tornadotebbe.github.io/`

Inhaltsverzeichnis:

# Inhaltsverzeichnis



# 1 Einleitung

Viele verschiedene Faktoren haben einen Einfluss auf die Note der Studenten. Dies geht bereits damit los in welche Familie das Kind hineingeboren wird, welchen akademischen Standard die Eltern haben, welche finanziellen Mittel usw. Weiter spielt die Erziehung und der spätere soziale Umgang eine Rolle. [3] Diese Einflüsse sind schwer zu messen, vor allem schwer genau zu messen.

Zudem sind diese Faktoren kaum von alleine beeinflussbar.
Doch auf welche Faktoren kann der Student selber später Einfluss nehmen. Die Zeit, die täglich in oder eben nicht in die schulische Weiterbildung investiert wird, sollte in der Theorie so einen selbst beeinflussbaren Faktor darstellen. Und ich welcher weise lässt sich auch eine Tendenz in der Fluktuation der Noten erkennen. Bleiben Schüler, die bereits in der Schule schlechtere Noten haben auch im College schlecht? In dieser Arbeit werden sich also folgende Fragen und folgende gestellt Zielstellungen gesetzt:

- Wie korrelieren Noten und Zeitgestaltung und lassen sich daraus Rückschlüsse ziehen, wenn ja welche?
- Wie stehen die Noten im Zusammenhang mit anderen Variablen?
- Wie können die verschiedenen Daten übersichtlich veranschaulicht werden um dem Leser einen schnellen Überblick zu geben

## 1.1 Anwendungshintergrund

Wer in der Schule und im College gute Noten hat und sich somit einen besseren Abschluss erarbeitet hat im späteren Leben deutlich bessere Chancen auf „individuelle Lebenschancen, Selbstverwirklichung, beruflichen Erfolg sowie soziale, politische und kulturelle Teilhabe. “[**Solga**, **Dombrowski 2008**] Die Noten spielen also eine wichtige Rolle, vor allem im Ausblick auf das spätere Leben. Es wäre also wünschenswert, wenn dieser so wichtige Einflussfaktor auf die persönliche Zukunft möglichst in der eigenen Hand liegen würde. Anhand der nun im Anschluss vorgestellten Visualisierungen sollte dies erörtert werden.

## 1.2 Zielgruppen

Die Zielgruppe für die Daten umfasst ein weites Spektrum an Personen. Die Visualisierung sind sowohl für Schüler und Studenten, als auch für Lehrer und Eltern interessant, also jegliche Personengruppe, die Berührungspunkte mit der Notengebung und deren Einflussfaktoren aufweist. Die Informationen, die aus den Visualisierungen gewonnen werden können, können sehr gut dafür genutzt werden zu sehen, wie grade der Einflussfaktor Lernzeit und auch die vorangegangen Noten aus der Schulzeit einen Einfluss auf die Noten im College nehmen. Differenziert werden kann hierbei auch nochmal unter den Geschlechtern. Interessant könnte dieses Projekt aber auch für Forschende im Bereich soziale Ungleichheiten in der Bildung sein. Wenn keine eindeutigen Anzeichen zu sehen sind für eine Korrelation zwischen investierter Lernzeit und besseren Noten, so könnte dies ein Anzeichen dafür sein, dass doch andere Einflussgrößen mehr Auswirkung auf die Noten haben als der reine Fleiß. Dies wäre ein starker Indikator für soziale Ungleichheiten bei der Bildung, wie es bereits in der Einleitung angedeutet wurde.

### 1.3 Überblick und Beiträge

Der in dieser Arbeit verwendete Datensatz differenzieren die Studenten nach verschiedenen Merkmalen. Diese gehen von allgemeinen Unterscheidungen, wie Geschlecht, Alter und Gewicht, über die Aufteilung ihrer Freizeit und die aufgewendete Lernzeit bis hinzu dem körperlichen Wohlbefinden ausgedrückt mittels der Variable ‚stress level '.
Die für diese Visualisierung mit am interessantesten Daten sind die der verschiedenen Noten in der Schule und im college. Die Noten, gerade mit der von den Studenten angegebenen täglich Lernzeit sollte in einem positiven Zusammenhang stehen. Also steigt die Lernzeit am Tag, steigt auch die Note. Das gleiche sollte auch für die Beziehung der Noten untereinander gelten, also welcher Student schon in der 10. und 12.Klasse gute Noten hatte, sollte nun auch im College gut abschneiden. Dies sollte sich außerdem in der späteren Gehaltserwartung widerspiegeln, da meist bessere Abschlüsse zu besseren Gehältern führen. Dies sollte den Studenten bewusst sein, weshalb sie sich dahingehend gut einschätzen sollten. Die körperlichen Voraussetzungen sowie das Department sollten eigentlichen keine größeren Einflüssen auf die Noten haben. Die Zeit verbracht auf sozialen Medien könnte eine negative Korrelation mit den Noten aufweisen, genauso wie der finanzielle Status und das Stress Level bei den Studenten.

## 2 Daten

Wie bereits in den Abschnitten vorher erwähnt, handelt es sich bei den vorliegenden Daten um den Datensatz *Student Behaviour* von *kaggle* . Die darin aufgeführten und gesammelten Daten geben nähere Auskunft über das Verhalten der Studenten und deren Noten. Die Verwendung des Datensatzes zu wissenschaftlichen Zwecken ist abzuraten, da der Verfasser angibt die Daten selber von Universitäten gesammelt zu haben und bis auf diesen Satz keine näheren Informationen zur Datenbeschaffung liefert. Seiner Meinung nach kann der Datensatz dazu genutzt werden das Verhalten der Studenten in Zukunft besser zu deuten.
Im Datensatz auf *kaggle* sind ein paar Rechtschreibfehler bei der Variablenbenennung enthalten, wie z.B. „social medai time“, diese werden daher der Leserlichkeit halber im Folgenden grammatikalisch korrekt geschrieben.
Er enthält Informationen von 235 Studenten in Form einer CSV-Datei. Diese CSV-Datei hat 19 Spalten mit den Informationen der Studenten über *Certification* als Boolean, der ausgibt ob der Student einen certification Kurs besucht hat oder nicht, *Gender* mit Boolean über das Geschlecht, *Department*, welcher Student welchem Derpartment angehört, *Height(CM)* und *Weight(KG)* als Float, die Größe und Gewicht aufzeigen, *10th Mark*, *12th Mark* und *College Mark*, die als Float die Note der Studenten zeigen, *Hobbies*, sowie *stress level* und *financial status* als String, *part-time job* und *Do you like your degree* als Boolean, *salary expectation* als integer (in Elm allerdings als Float deklariert und verwendet, damit es passend visualisiert und gemapt werden kann), *willingness to pursue a career based on their degree* als String in Form einer Prozentangabe (möglich sind hier nur die Angaben: 0,25,50,75 und 100 Prozent) und *tra-*

*velling time*, *social media time* und *daily studying time* als String in Form einer Klassenangabe (also z.B. 30-60 Minuten ist eine Angabe).

Während sich grade die Noten, also 10th Mark, 12th Mark und College Mark sehr gut dazu eignen die Fragestellung zu beantworten und sich zusätzlich dazu noch gut visualisieren lassen, da sie als Float sich relativ einfach veranschaulichen lassen ist dies bei anderen Variablen weniger der Fall. Bei ein paar der Daten ist nichteinmal genau klar, was damit überhaupt ausgesagt werden soll, wie z.B. bei Certification Course. Was für ein Kurs ist das, was zertifiziert man damit, bzw. welche Art von Zertifikation erwirbt man damit. Aufgrund des fehlenden Hintergrundwissen, welches leider im Datensatz nicht näher erläutet wird, kann bei diesen Visualisierungen nicht näher darauf eingegangen werden. Ähnlich verhält es sich bei Department. Auch hier fehlen die nötigen zusätzlichen Informationen, um diese Variable in geeigneter Weise einbauen zu können. Auch die meisten Boolean (part-time job, Do you like your degree, werden bei den Visualisierungen nicht berücksichtigt (bis auf gender), da es sich, bei den hier verwendeten Darstellungen nicht anbietet boolsche Werte zu integrieren. Diese bieten schlichtweg zu wenige Möglichkeiten. Vielseitig genutzt wurde auch die daily studying time, die sich vor allem als *drop-down* zur Sortierung der Noten eignete, sodass schnell ersichtlich wurde mit welcher Lernzeit welche Noten erreicht wurden. Hier wurde also eine der oben gestellten Zielstellungen sinnvoll und gut umgesetzt.

## 2.1 Technische Bereitstellung der Daten

Die Daten wurden als einzelne CSV-Datei in *kaggle* bereitgestellt und auch also solche weiterverarbeitet. Zudem wurden sie im eigenen Github hochgeladen und dort im Ordner Daten gespeichert. Mithilfe dieser Datei wurden der Scatterplot und der Parallelplot erstellt. zur Visualisierung des Baumdiagramms wurde zusätzlich eine json-Datei erstellt *"json.Name"* und auch im Ordner Daten hinterlegt. Zusätzlich dazu finden sich im Github-Repository auch noch Python-Code, mit dessen Hilfe die Daten aufbereitet werden sollen, genaueres in 2.2

## 2.2 Datenvorverarbeitung

Im obigen Abschnitt wurde bereits kurz auf den Datensatz eingegangen und erklärt, dass es sich um eine einzelne CSV-Date handelte. Aufgrunddessen musste keinerlei größere Vorverabeitung an den Daten vorgenommen werden, zumindest für den Parallelplot und den Scatterplot. Die einzige Datenmanipulation, die hier erfolgte ist die Ausreißerbereinigung. Es gab ein paar wenige extreme Ausreißer (bei *salary expectation*), die dazu führen würden, dass die Visualisierung der Daten darunter leidet. Für diese wurde daher der vorher berechnete Mittelwert eingesetzt.

Für das Baumdiagramm wurde zuerst versucht mithilfe von Pyhtoncode die csv-Datei in eine json-Datei umzuwandeln. Leider gelang dies nicht, sodass die csv erst in eine Excel-Datei umgewandelt wurde. Anschließend wurde mithilfe der Excel-Datei manuell eine json-Datei erstellt

um anhand dieser die Baumdarstellung zu visualisieren. In der json wurden dann nur die für das Baumdiagramm relevanten Daten übernommen. Aufgrund anfänglicher Überlegungen die verschieden klassierten *Zeiten*, wie *daily studying time*, *social media time* und *travelling time* in einzelne Klassen in Form von Integers festzuhalten, also *daily studying time* von 0-30 Minuten wäre 1, 30-60 Minuten 2 usw. wurde die csv-Datei in R geladen um die Daten somit zu manipulieren. Gleiches wurde im Verlauf der Areit auch nocheinmal mit *python* probiert. Aufgrund der späteren Nutzung der *daily studying time* als Drop-down Tabelle wurde dieser Gedanke wieder verworfen. Für das Baumdiagramm wurden außerdem der Mittelwert der Noten in den jeweiligen Lernklassen berechnet. Dieser und der vorher für die Ausreißer berechnete Mittelwert sind in der *Student_Behaviour* Csv-Datei abgespeichert.

# 3 Visualisierungen

In den nachfolgenden Abschnitten wird näher auf die oben genannten Zielstellungen eingegangen.

## 3.1 Analyse der Anwendungsaufgaben

Wie bereits in 1 erwähnt, sollen die folgenden Visualisierungen der Zielgruppe dabei helfen sich einen schnellen und übersichtlichen Eindruck über den Zusammenhang der unterschiedlichen Noten und anderen Variablen machen. Also welche Variablen nehmen möglicherweise Einflüsse auf die Note oder stehen in einem Zusammenhang mit diesen. Die Leser sollen sehen können, wie sich die Noten verändern wenn sich gewisse Variablen verändern, also wie ändert sich die Notenverteilung, wenn sich die Lernzeit erhöht oder verringert, welche Individuen der Ausgewählten sind Männer, welche Frauen. Auch auf einen möglichen Zusammenhang der sich im Laufe der Zeit verändernden Noten soll mithilfe der Visualisierungen eingegangen werden. Sind Studenten, die in der 10. Klasse gute Noten hatten auch in der 12. Klasse und im College noch so gut und umgekehrt genauso: War ein Student mit schlechten Noten schon in der 10. und der 12. Klasse schlecht und hängen diese Dinge möglicherweise mit dem Lernaufwand zusammen. All diese Fragen können in anschaulicher und so auch schneller Art und Weise von den Beobachtern auf den unterschiedlichen Visualisierungen überblickt werden, mit Unterstützung von verschiedenen Drop-Down-Möglichkeiten und der Hover-Funktion (hier werden den Lesern in den Plots immer die Auskunft über das Geschlecht und in manchen Fällen noch weitere allgemeine Informationen geliefert) über den Punkten oder den Parallellinien. Das normale mentale Modell, dass bei einem Menschen entstehen sollte, wenn er sich diese Fragen und Anwendungsaufgaben stellt ist, dass es eine positive Korrelation zwischen guten Noten in der Schule und guten Noten im College, als auch zwischen Lernzeit und guten Noten existiert. In den Köpfen der Anwender entsteht also ein stetiger Anstieg der durschnittlichen Notenverteilung, je mehr Lernzeit dazukommt und dies sollte auch in den Abbildungen zu sehen sein. Die Mentalen Modelle sind für die Verständlichkeit der Visualisierungen für Leser allerdings kaum notwendig, da sie relativ simpel gehalten sind

und es auch eine der oben gestellten Anforderungen an das Projekt war, eine gewisse Komplexität nicht zu überschreiten. Die mentalen Modelle unterstützen also mehr die Visualisierungen, als dass sie vorausgesetzt werden um sie zu verstehen. Allerdings ist, wie schon im oberen Bereich der Hausarbeit erwähnt die Gleichung, die Noten beeinflusst ein hochkomplexes Konstrukt, welche von reinen Zahlen kaum abgebildet werden kann, dies muss den Beobachtern bewusst bleiben und verzehrt so möglicherweise das eigene mentale Modell, gerade wenn bedacht wird, dass es sich bei *Student Behaviour* um einen sehr kleinen Datensatz von 236 Studenten und um keine wissenschaftliche Studie, sondern eigens erhobene Daten handelt. Verwirrend wäre dies vor allem , wenn Studenten mit einer geringen Lernzeit sehr gute Noten haben und Studenten mit höher Lernzeit sehr schlechte, die könnte also durchaus dazu führen, dass der Leser mit einer Diskrepanz zwischen seinem mentalen Modell und der abgebildeten Visualisierung konfrontiert wird.

## 3.2 Anforderungen an die Visualisierungen

Da die Leser einschätzen sollen können, ob und wenn ja inwieweit eine Korrelation zwischen den verschiedenen Parametern mit Noten und Zeitaufwand besteht, war es wichtig diese Daten aufeinander abgestimmt, in jeweils einer Visualisierung darstellen zu können.

Der Leser sollte bei allen Visualisierungen die Möglichkeit haben nach dem Parameter Zeit zu differenzieren und sich am Ende die Noten von den unterschiedlichen Studenten aus den unterschiedlichen Klassen/College ausgeben zu lassen. Es sollte also erst eine grobe Differenzierung getroffen werden können, um sich nach dieser ersten Einteilung genauer mit den verschiedenen Noten auseinandersetzen zu können.

Weiterhin war es wichtig die Visualisierungen recht einfach zu halten und schnell erkennbar werden zu lassen, ob die verschiedenen Parameter korrelieren oder nicht. Die Plots bieten die Möglichkeit des Drop-Downs und stellen am Anfang daher eine einfache Methode für den Betrachter dar, sich durch die verschiedenen Optionen zu klicken, ohne ihn direkt am Anfang mit Darstellungen und Informationen zu überhäufen. Durch die Anzahl von drei Drop-Downs (beim Scatterplot) ist es ihm aber wiederum möglich eine große Anzahl an Auswahlmöglichkeiten zu treffen und die Daten sich in vielen verschiedenen Kombinationen anzeigen zu lassen. Bei den Plots bietet sich die zusätzliche Optionen über die Punkte bzw. Linien zu hovern und so noch mehr Informationen über den jeweilig ausgewählten Studenten zu erlangen.

Die Visualisierung sollte in conclusio also anfänglich einfach und übersichtlich gehalten sein und bei mehr Interesse auch weiter Möglichkeiten bieten.

Daher wurde sich für den Anfang für den Scatterplot entschieden, der eine Visualisierung darstellt,die die meisten Menschen kennen und der eine schnelle Übersicht über die Daten liefert,

als zweites für einen Parallelplot, der die ersten Anfänge des Scatterplots erweitert, indem alle Noten auf einmal zu sehen sind. Als letztes das Baumdiagramm, dass alle im bisher im Fokus stehende Daten abbildet.

## 3.3 Präsentation der Visualisierungen

Die eben beschriebenen Präsentationen und Begründungen sollen für jede der drei folgenden Visualisierungen durchgeführt werden.

### 3.3.1 Scatterplot

Wie bereits in 3.2 erwähnt wurde sich als erste Visualisierung für den Scatterplot entschieden, abgebildet in Abbildung 1.

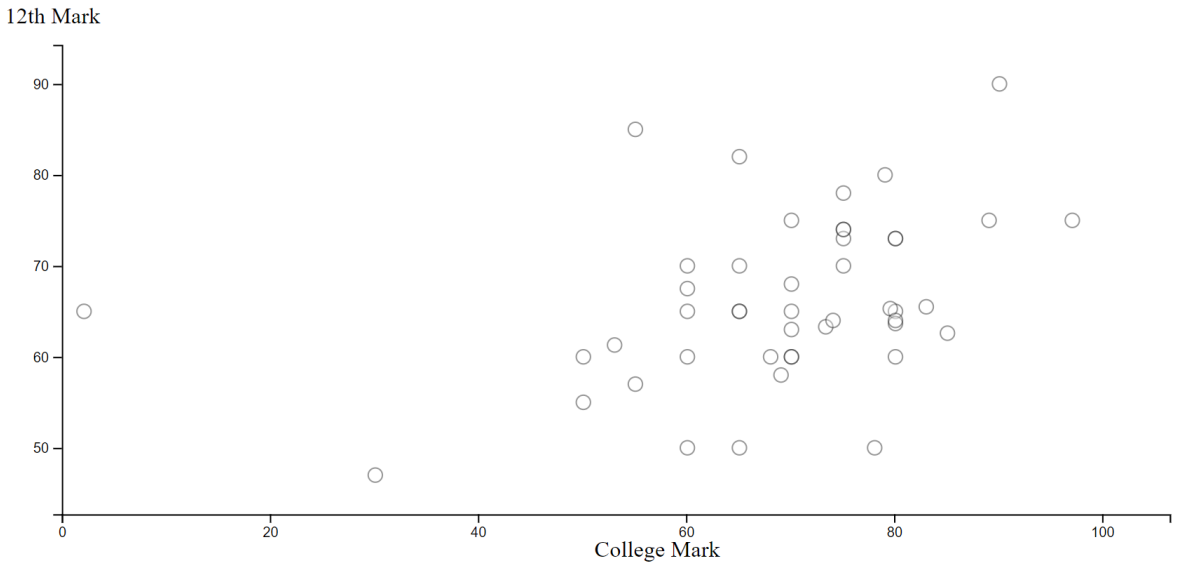



Abbildung 1: Scatterplot

Warum sich für den Scatterplot entschieden wurde, wurde bereits in 3.2 erläutert, zur genaueren Definition des Scatterplots und wie die *Student Behaviour*-Daten darin dargestellt wurden wird in den folgenden Abschnitten näher darauf eingegangen. Der Scatterplot an sich besitzt

zwei Achsen, die X-Achse und die Y-Achse. Deren Größe kann entweder manuell bestimmt werden oder hat vier weitere Möglichkeiten gleiche Daten verschieden darzustellen([4])

- X- und Y-Achse haben einen gleich großen Wertebereich
- Nur die Y-Achse hat einen großen Wertebereich
- Nur die X-Achse hat einen großen Wertebereich
- Der Wertebereich in X und Y wird durch die Daten bestimmt

Die manuelle Einstellung der Achsen wäre sehr großer Aufwand und auch durchaus ungeeignet, weswegen sich hier für eine dynamische Option entschieden wurde.Die beiden Achsen wurden dem Wertebereich angepasst und können so auch flexibel mit den unterschiedlichen reingeparsten Daten interagieren und sich verändern. In Abbildung 1 enthält in diesem Beispiel die X-Achse als Werte die Daten von College Mark und die Y-Achse die Werte der Daten von 12th Mark. Zusätzlich zu den dargestellten Achsen und deren Beschriftung enthält ein Scatterplot Punkte, die die verschiedenen zu untersuchenden Individuen und ihre Werte abbilden sollen (in diesem speziellen Fall die Studenten.
Aus dem Scatterplot kann also anhand der zwei beschrifteten Achsen und den Punkten auf einen Blick erkannt werden, wie die Notenverteilung bei den unterschiedlich ausgewählten Lernzeiten aussieht.

Wie bereits oben erwähnt können die X- und die Y-Achse dynamisch gewählt werden, genauso wie der „Filter“,der die Lernzeit auswählt und unter dieser Prämisse die Daten der Noten auswählt

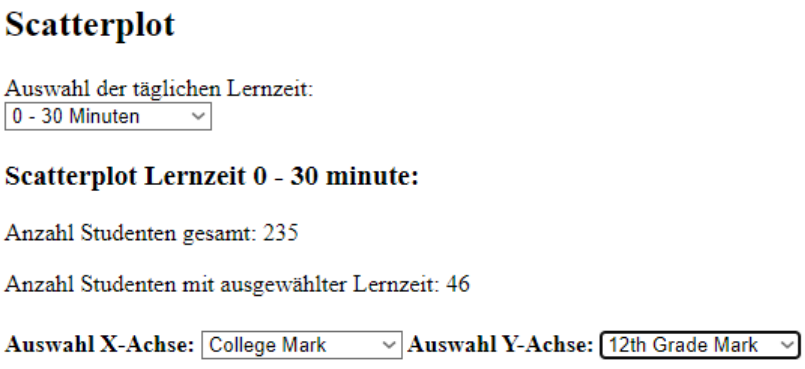



Abbildung 2: Ausgewählte Drop-Down-Funktionen und Head

Zu sehen in Abbildung 2 ist, dass die Lernzeit 0-30 Minuten ausgewählt wurde und zusätzlich dazu die bereits erwähnten Achsen. So ist es dem Leser überlassen sich die verschiedenen Werte

und differenten Korrelationen der Daten ausgeben zu lassen. Dies stellte eine der Anforderungen an die Visualisierung dar und wurde zusätzlich mit der einfachen Handhabung, gegeben durch die Drop-Downs in die Tat umgesetzt. Es lässt sich so sehr leicht auf einen Blick erkenne, in welchem Notenbereich sich Studenten bewegen, die 0-30 Minuten lernen. Zusätzlich dazu wird die Relation zwischen der Note im College und der Note aus der 12.Klasse abgebildet.
Dem Leser werden außerdem zwei zusätzliche Informationen geliefert, einmal wie viele Studenten es insgesamt gibt und außerdem wie viele Studenten von dieser Gesamtanzahl grade ausgewählt sind. Also wie viele Studenten von 236 0-30 Minuten lernen, in dem Fall 46.
So kann sich der Leser ein besseres Bild dazu machen, wie sich die Lernzeit grob verteilt.
Außerdem wird dem Betrachter die Hover-Funktion bereitgestellt, dass bedeutet, dass wenn er über die einzelnen Punkte mit der Maus rüber fährt, leuchtet der Punkt grün auf und gibt ihm das Geschlecht mittig über dem Punkt aus.

Eine weiter ähnliche Visualisierungstechnik ist der QQ-Plot, dieser Teil die Datenwerte in verschiede Quantile ein. Hier werden die Verteilunen verglichen, Hinneburg beschreibt es als „Verschiebungen zwischen zwei Verteilungen werden effizient durch Vergleich der Quantile gefunden“ [4]. Da bei der zugrundeliegenden Arbeit allerdings nicht die Quantile veranschaulicht werden sollen, sondern vielmehr die einzelnen Noten der Studenten interessant sind, macht ein Scatterplot deutlich mehr Sinn.

### 3.3.2 Parallelplot

Die zweite Visualisierung wurde in Form eines Parallelplots dargestellt. Dieser hat, wie bereits in 3.2 gegenüber des Scatterplots den Vorteil, dass hier mehrere Daten auf einmal, sogenannte mehrdimensionale Daten dargestellt werden können.
Es können also von einem Studenten unter der Auswahl einer Lernzeitklasse alle Noten auf einmal darstellen. Hierzu werden die verschiedenen Attribute (Noten und *salary expectation*) vertikal nebeneinander aufgelistet und mithilfe einer Linie verbunden. Diese Linie übernimmt quasi die Aufgabe des Punkts aus dem Scatterplot und veranschaulicht so einzelne Studenten, siehe 3. Hierbei wurde sich wieder für die Noten 10.Klasse, 12.Klasse & College Mark zusätzlich zu *Salary expectation* entschieden. So kann die mögliche Korrelation zwischen den einzelnen Noten und der Lernzeit und der Korrelation der einzelnen Noten untereinander gut veranschaulicht werden.

Der Leser erhält außerdem wieder als Kopf über der Visualisierung die Auswahl zwischen den einzelnen Lernzeiten per Drop-Down zu navigieren und erneut die Information, wie viele Studenten es insgesamt und wieviele Studenten es in der jeweils ausgewählten Zeitklasse gibt, siehe Abbildung 4.
Für den Leser ist so nun, wie in der Zielstellung in 1 formulierten Anforderung auf einen Blick ersichtlich, wie sich die Verteilung der Noten gestaltet. Sollten sich sehr viele Linien im höheren Segment des Parallelplots ansiedeln, so ist klar, dass in der ausgewählten Lernklasse (in diesem

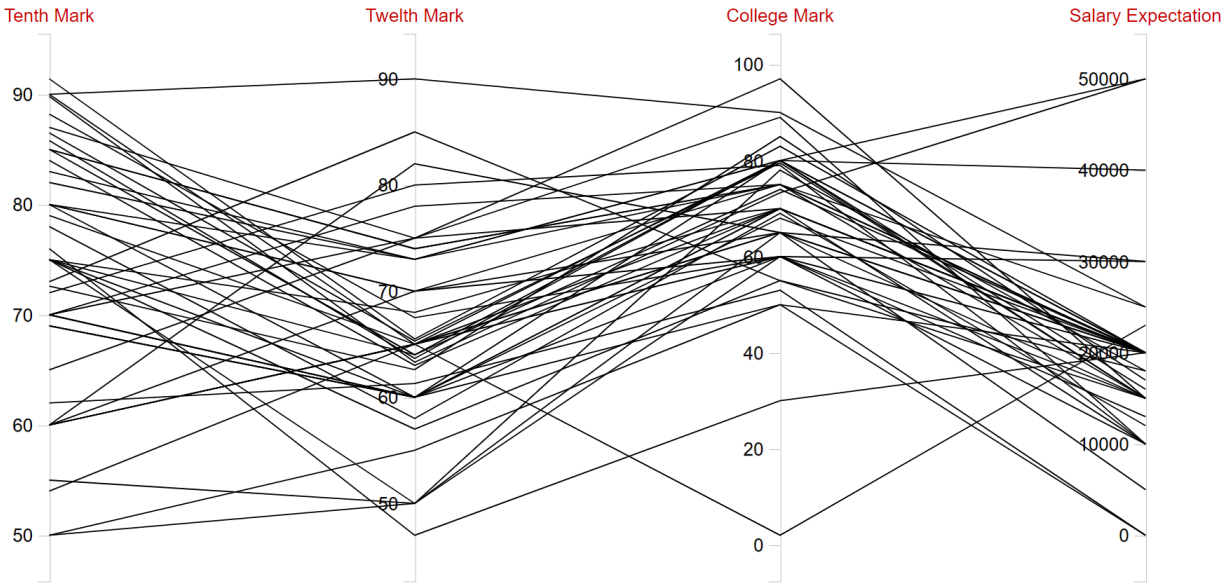



Abbildung 3: Parallelplot

# Multidimensionale Daten für 0 - 30 minute

- Ausgewählte tägliche Lernzeit: 0 - 30 minute
  0 - 30 minutes
- Studenten insgesamt: 235
- Gefilterte Studenten insgesamt: 46

Abbildung 4: Drop-Down Fenster und Head des Parallelplot

Beispiel 0-30 Minuten)viele Studenten gute Noten in der Schule hatten und immer noch gute Noten im College haben, sowie eine hohe Erwartung an ihr späteres Gehalt haben. Genau das gleiche gilt selbstverständlich für den umgekehrten Fall.

Dem Vorteil der mehrdimensionalen Datendarstellung des Parallelplots steht allerdings das Unübersichtliche des Gleichnamigen gegenüber. Wenn auch alle Daten auf einmal abgebildet werden können, so leidet darunter die exakte Differenzierung der unterschiedlichen Individuen. Beim Parallelplot ist es also durchaus schwierig die einzelnen Studenten voneinander zu differenzieren. Denn die Linien schneiden sich an den vertikalen Achsen und so ist nicht genau erkennbar, wo welche Linie welches Studenten weiterführt.
Um hier für den Leser eine gute Lösung bereitzustellen, wurde die *hover* - Funktion eingeführt, mit dessen Hilfe der Leser genauer Informationen enthält und die einzelnen Studenten besser

voneinander unterscheiden kann.
Genau wie die Parallelen Koordinaten gewährleisten die sternförmigen Koordinaten eine verlustfreie und eindeutige Anordnung der Daten. Im Gegensatz dazu werden die sternförmigen Koordinaten aber nicht parallel nebeneinander angeordnet, sondern es findet eine sternförmige Anordnung der Merkmalsachsen statt [2].
Für die Zielsetzung eignet sich der Parallelplot allerdings besser, da es für den Betrachter deutlich leichter ist auf den ersten Blick die Verteilung der Linien einzuordnen. Er kann anhand dieser Verteilung schneller ein Urteil über die Korrelation der Daten fällen, als die bei den sternförmigen Koordinaten der Fall wäre. Hier wäre, wie bereits oben bereits beschrieben, ein gutes mentales Modell und Kenntnisse in Visualisierungsmethoden nötig gewesen, um die Darstellung zu interpretieren; zumindest eine gewisse stärkere Auseinandersetzung mit der Visualisierung an sich.

Eben dieser Fall wollte aber vermieden werden. Es sollte auf den ersten Blick direkt ein ungefähres Urteil über die Daten gefällt werden können und sich dann danach weiter damit auseinandergesetzt werden können. Bei den sternförmigen Koordinaten wäre dies nicht gegeben gewesen.

### 3.3.3 Explizite Baumdarstellung

In der dritten und letzten Visualisierung wurde sich für eine explizite Baumdarstellung entschieden. Diese Veranschaulichungsmöglichkeit ist dargestellt als zusammenhängender Graph und die Beziehung zwischen Knoten mit gerichteten Kanten. Außerdem sind alle Knoten von der Wurzel aus erreichbar [4].

Die Wurzel stellt hier der einzelne Student dar. Dieser hat zwei sogenannte Kinder, also Knoten, die mit gerichteten Kanten von ihm wegführen und sie die Beziehung zueinander visuell darstellen. Diese zwei Kinder sind die zwei Geschlechter, also männlich und weiblich.
Diese Knoten wiederum haben eine weiter Unterteilung in die verschiedenen Lernzeiten, die bereits in den vorherigen Visualisierungen immer als Drop-Down zur Verfügung standen. Die letzten Kinder sind die Noten. Diese wurden diesmal nicht nach Klassen und College unterteilt, sondern nach der Durchschnittsnote aus dem College.
Der Leser kann so, wie dies bereits in den vorherigen Visualisierungen der Fall war, zuerst die Unterteilung der Studenten nach Lernzeit vornehmen. Und danach kann er nach Geschlecht unterteilen und die jeweiligen Leistungen direkt als String ablesen.

Diese Visualisierung hat den Vorteil, dass die Noten des Durchschnitts dargestellt werden können und so dem Leser direkt als String geliefert werden. Zudem ist den meisten Lesern die Baumstruktur durchaus aus vielen normalen Lebensbereichen geläufig, wie z.B. der Ordnerstruktur auf dem PC oder der Turnierstruktur bei einer Weltmeisterschaft [4].

Allerdings bietet die explizite Baumdarstellung auch einige Nachteile, so wird sie sehr schnell unübersichtlich und zu weit verschachtelt, wenn zu viele Knoten dazukommen.
Eine andere Möglichkeit wäre daher die implizite Baumdarstellung gewesen, die die grafische Beziehung zwischen Objekten mittels der drei Möglichkeiten

- Umschließen
- Überlappen
- und Berühren

sinnvoll darstellen kann [4]. Diese haben aber, ähnlich der sternförmigen Koordinaten aus Abschnitt 3.3.2 den Nachteil, dass sie nicht sehr intuitiv für den Leser sind. Der Leser kann bei der ersten Betrachtung, im Gegensatz zur expliziten Baumdarstellung kaum eine Unterteilung der Hierarchie vornehmen. So wird eine der wichtigen Zielstellungen nicht erfüllt. Dieser und der Grund, dass den meisten Menschen die explizite Baumdarstellung weitaus geläufiger ist, wurde sich für letztere Visualisierung entschieden.

## 3.4 Interaktion

Der Leser wird zu Beginn der Visualisierung auf die Startseite geführt. Von hier aus hat er mithilfe verschiedener *href* - Buttons die Möglichkeit auf die unterschiedlichen Visualisierungen zu navigieren. Diese Art der Navigation zieht sich über alle Seiten.

Bei der ersten Visualisierung, dem Scatterplot bieten sich dem Nutzer gleich mehrere Interaktionsmöglichkeiten. Zum einen hat er die Auswahl zwischen drei verschiedenen Drop-Down-Funktionen. In der ersten kann er die Lernzeit auswählen. Hierfür stellen sich ihm die Optionen 5. Die beiden anderen Drop-Downs sind gleich aufgebaut und bieten mehrere Möglichkeiten, auf die in 3.3.1 bereits näher eingegangen wurde.

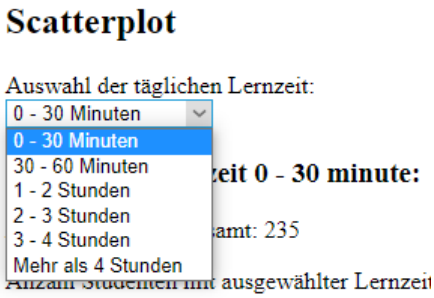



Abbildung 5: Drop-Down Funktion der Lernzeit

Eine weiter Interaktionsmöglichkeit beim Scatterplot stellt die *Hover* - Funktion dar,die dem Nutzer zusätzliche Informationen liefert. Wenn der Nutzer über die einzelnen Punkte *hovert*, so

leuchtet der ausgewählte Punkt grün auf und liefert als „Ausgabe“mittig darüber das Geschlecht und die präferierte Lernzeit, genauer ist die in Abbildung 6 zu sehen.

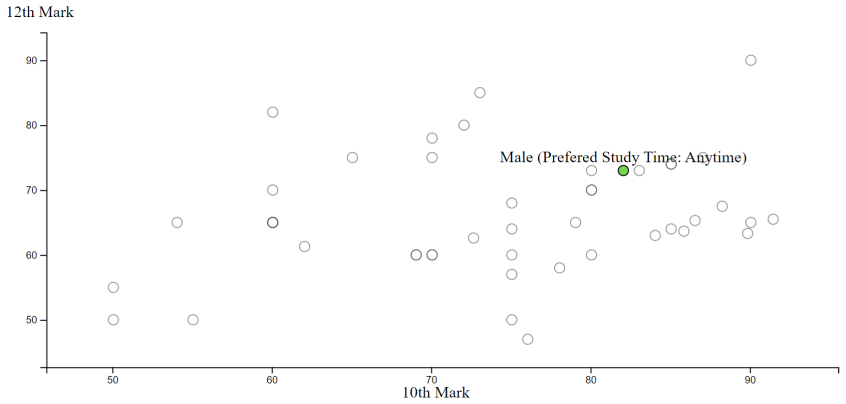



Abbildung 6: Scatterplot mit Hover-Funktion

Wie beim Scatterplot hat der Nutzer beim Parallelplot die Auswahlmöglichkeit der Drop-Down-Funktion für die Lernzeit, siehe Abbildung 5.
Zusätzlich dazu wird hier die Möglichkeit gegeben, die Anordnung der Achsen beliebig anzuordnen. Dies wird diesmal in Form von Button abgebildet um eine größere Breite an Interaktionsmöglichkeiten abzudecken, siehe Abbildung 7. Hier gibt es für jede Achse dieselbe Auswahl an Variablen. Es wäre in der Theorie also auch möglich für jeden Achse denselben Wert festzulegen.

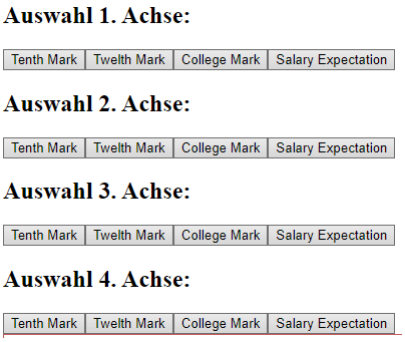



Abbildung 7: Parallelplot mit Button für die Auswahl der Achsen

Bei dem Baumdiagramm ist die Reihenfolge und Hierarchie bereits durch die Beschaffenheit des Diagramms an sich fest. Hier kann allerdings zusätzlich dazu durch die Eingabe der Breite, Höhe und des Radius die Visualisierung des Baums verändert werdend. Der Nutzer hat also keine vorher festgelegte Auswahl, sondern kann fast komplett frei entscheiden, was er in das *input* - Feld schreibt. Die einzige Bedingung, die hier gilt ist, dass es sich um einen Float handeln muss.

# 4 Implementierung

Die nächsten Abschnitte behandeln die Implementierung des Codes. Für die Implementierung wurde sich an den Übungen orientiert und dieser Code dann übernommen und angepasst. Für 4.1 an Übung 1 und Übung 6 und für4.2 an Übung 7.

## 4.1 Scatterplot und Parallelplot

Jede einzelne Datei besteht grundlegend aus einem *main*, welches die Funktionen *init*, *update* *subsriptions* und *view*, siehe Abbildung 8. Der erste Schritt war die Daten zu laden und zu Decoden. Dabei wurde sich wieder am Code der Übung 7 orientiert. Die Daten wurden aus der *Student_Behaviour.csv*-Datei geladen, die im Github abgelegt war. Das Laden der Daten geschah über die eben erwähnte *init*-Funktion, bei der mtihilfe eines *Http.get*-Befehls der Link aus dem eigenen Github geladen wurde. Auf die anderen funktionen wird später genauer und individuell eingegangen.

```
main : Program () Model Msg
main =
    Browser.element
        { init = init
        , update = update
        , subscriptions = subscriptions
        , view = view
        }
```

Abbildung 8: Main

Nachdem die Daten über die *init*-Funktion geladen wurden, müssen sie nun dekodiert werden. Zuerst mussten die verschiedenen Module dafür geladen und importiert werden, in diesem Fall „Csv“und „Csv.Decode“. Mithilfe dieser Module konnte die Funktion zum codieren der Daten geschrieben werden, dieser Code ist zu sehen in Abbildung 9.

```
decodeCsvStudentdata : Csv.Decode.Decoder (Student_Data -> a ) a
decodeCsvStudentdata =
    Csv.Decode.map Student_Data
        (Csv.Decode.field "Certification Course" Ok
            |> Csv.Decode.andMap (Csv.Decode.field "Gender" Ok)
            |> Csv.Decode.andMap (Csv.Decode.field "Department" Ok)
            |> Csv.Decode.andMap (Csv.Decode.field "Height(CM)"(String.toFloat >> Result.fromMaybe "error parsing string"))
            |> Csv.Decode.andMap (Csv.Decode.field "Weight(KG)" (String.toFloat >> Result.fromMaybe "error parsing string"))
            |> Csv.Decode.andMap (Csv.Decode.field "10th Mark" (String.toFloat >> Result.fromMaybe "error parsing string"))
            |> Csv.Decode.andMap (Csv.Decode.field "12th Mark" (String.toFloat >> Result.fromMaybe "error parsing string"))
            |> Csv.Decode.andMap (Csv.Decode.field "college mark" (String.toFloat >> Result.fromMaybe "error parsing string"))
```

Abbildung 9: Ausschnitt der Decode-Funktion

Hier werden die einzelnen Daten als String reingeparst und falls sie auch ein String in der weiteren Verarbeitung sein sollen als „Ok“markiert. Falls sie umgeformt werden sollen, wird

dies mit der Anweisung „String.to“und danach der gewünschte Datentyp, z.B. Float erreicht. In einem Beispiel im Datensatz wäre das beispielsweise bei *10th Mark* der Fall. Damit in elm aber weiter damit als Float gearbeitet werden kann, muss dieser Datentyp nochmal in einem *type* definiert werden. Die Daten wurden also im *type Student_Data* nochmals definiert um mit ihnen später weiterarbeiten zu können. Hier mussten sie als genau der Datentyp definiert werden, als der sie auch vorher decoded wurden. Sonst warf es eine Fehlermeldung aus. Die *type*-Definition sah dann wie in Abbildung 10. Das Decodieren war zu Anfang eine sehr mühselige Arbeit und nahm einiges an Zeit in Anspruch, da hier alle Strings genauso geschrieben werden mussten, wie in der Csv-Datei. Da es auch ein paar Rechtschreibfehler gab (z.B. „social medai“statt „social media“und teilweise Leerzeichen hinter den Strings standen (beispielsweise bei „stress level“war dies anfänglich etwas frustrierend.

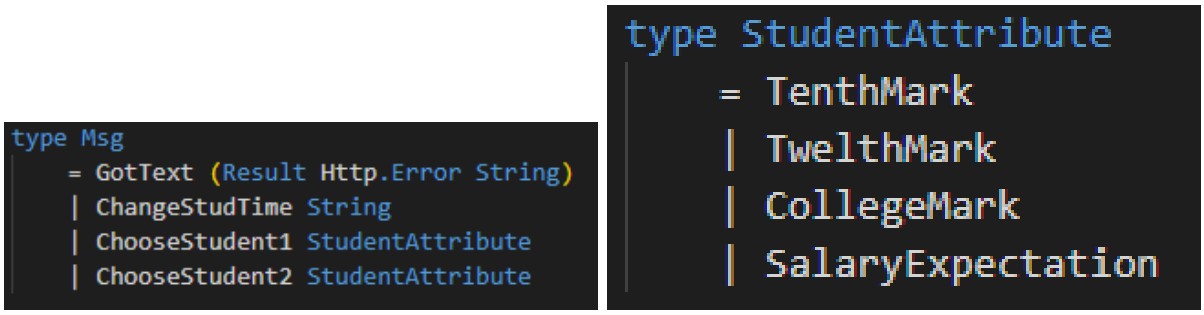
```
type alias Student_Data =
    {   certification : String
    ,   gender : String
    ,   department : String
    ,   height : Float
    ,   weight : Float
    ,   tenthMark  : Float
    ,   twelthMark  : Float
    ,   collegeMark : Float
    ,   hobbies : String
    ,   dailyStudyingTime : String
    ,   preferStudyTime : String
    ,   salaryExpectation : Float
    ,   satisfyDegree : String
    ,   willignessDegree: String
    ,   socialMedia : String
    ,   travellingTime : String
    ,   stressLevel : String
    ,   financialStatus : String
    ,   partTimeJob : String
    }
```

Abbildung 10: Ausschnitt der *type* - Definition Student_Data

Der grundlegende Aufbau des Elm-Programms setzt sich aus vier Dateien zusammen, der *Scatterplotėlm*, *Baumvisualisierungėlm*, *ParallelPlotėlm* und *Mainėlm*. Jede Datei steht für die jeweilige Visualisierung, während die *Mainėlm*-Datei die Startseite der Website darstellt.

Der Aufbau der kompletten Visualisierung stellte ein weiteres größeres Problem dar. Die Verknüpfung der einzelnen *case of* - Fälle ließ sich von mir nicht ohne jegliche Fehlermeldung lösen. Daher musste auf eine andere Art der Technik zurückgegriffen werden. Anstatt alles auf eine Seite zu laden, wurde hier mit einer *Seitennavigationsleiste* gearbeitet So kann zwischen den einzelnen Visualisierungen navigiert werden, siehe Abbildung

Zusätzlich zur *type* - Definition Student_Data wurden bei allen Visualisierungen ein *type Msg*




(a) *type* - Definition *Msg*

(b) *type* - Definition *StudentAttribute*

Abbildung 11: *type* - Definition im Beispiel Scatterplot

definiert. In diesem wurden die verschiedenen Aktionen festgelegt, die dem Leser bei 3.4 zu Verfügung stehen , also die Lernzeit (*ChangeStudTime*), oder die Daten für die Achsen (*ChooseStudent1*, *ChooseStudent2*) zu bestimmen. Zu sehen in Abbildung 11.
Bei dem *type StudentAttribute* werden die Daten festgelegt, die später an den Achsen des Scatterplots abgebildet werden und dynamisch ausgewählt werden könne, in diesem Fall also die Noten aus der 10., 12. Klasse, aus dem College und die Erwartungen an das spätere Gehalt.

Beim Parallelplot verhält sich diese Funktionalität ähnlich, es mussten nur zwei weitere Funktionen hinzugefügt werden und diese wurden dann auch direkt in die *type Msg* eingebettet. Bei beiden Plots wurden zudem Punkte definiert, beim Scatterplot eine *type Point* und beim Parallelplot eine *type MultiDimPoint*.

Weiterhin wurden bei beiden die *Html* Drop-Down Funktion als eingefügt, damit der Leser auf der Website auswählen kann. Die drei Funktionalitäten, Auswahl X-, Y-Achse und Lernzeitasuwahl hatten alle denselben Aufbau. Sie bekamen den Value aus dem Datensatz gegeben und hatten ein „Html.text“Feld, welches dem Nutzer angezeigt wird. Mit „Html.select“konnte der Nutzer nun zwischen den vorher festgelegten Feldern auswählen.

Diese Auswahl wurde in der *update* - Funktion verarbeitet, siehe Abbildung 12. Diese interagiert mit der *type Msg* und arbeitet mit der *case of*. Bei „GotText“werden (in den folgenden Beispielen wird immer davon ausgegangen, dass es zu keiner Fehlermeldung kommt) die Daten geladen und ein *Default* - Wert festgelegt, in diesem Fall liegt also die Lernzeit beim erfolgreichen Laden der Daten bei 0-30 Minuten und die Punkte auf der X-Achse als Noten der 10. Klasse und Y-Achse bei Noten der 12.Klasse.
Falls der Nutzer die Zeit ändert, also die Funktion „ChangeStudTime“aufruft, werden die Daten dahingehend geändert, dass nun, bei gleichbleibenden Achsen (also Noten 10. und 12.Klasse) die Werte sich auf den gewünschten neuen Wert anpassen. Also würde z.B. die Lernzeit auf 1-2 Stunden angehoben werden, wird dies entsprechend angepasst. Gleiches gilt bei Änderung

der Achsen. Hier tritt der Fall „ChooseStudent1“oder „ChooseStudent2“ein, je nachdem welche Achse geändert wird. Sollte es hier bei einem der Schritte zu einem Fehler kommen, wird die „Error-Funktion“durchgeführt.

Bei beiden Plots wurden die Daten zudem noch gefiltert, welche dann in der *view* - Funktion aufgerufen wurden. Diese *view* - Funktion sah auch bei beiden ähnlich aus, sie handelte zuerst die Optionen des *Failure* und *Loading* um bei einem *Success* die Daten und den Plot darzustellen. Hier wurden erst die verschiedenen Werte im *let* - Teil definiert um sie danach im *in* - Teil mit dem Plot an sich darstellen zu können. Im *in* - Teil wurden außerdem die verschiedenen *Html* - Attributen und *hrefs* aufgelistet.

```
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        GotText result ->
            case result of
                Ok fullText ->
                    (Success <| { data = studentListe [fullText], dailyStudyingTime = "0 - 30 minute", x = TenthMark, y = TwelthMark }, Cmd.none)

                Err _ ->
                    (model, Cmd.none)

        ChangeStudTime newTime ->
            case model of
                Success a ->
                    (Success <| { data = a.data, dailyStudyingTime = newTime, x = a.x, y = a.y }, Cmd.none)

                _ ->
                    (model, Cmd.none)

        ChooseStudent1 xNew ->
            case model of
                Success b ->
                    (Success <| { data = b.data, dailyStudyingTime = b.dailyStudyingTime, x = xNew, y = b.y }, Cmd.none)

                _ ->
                    (model, Cmd.none)
```

Abbildung 12: Ausschnitt der update-Funktion aus dem Scatterplot

Beim Scatterplot, welcher als erster visualisiert wurde, wurde die ungefähre Struktur aus den ersten Übungen übernommen. Hier mussten nachdem die Eckpunkte des Scatterplots aufgestellt worden waren, die Punkte und *Maybe Points* erstellt werden. Zusätzlich dazu die Achsenbeschriftung und die oben bereits erwähnten *update* und *view* - Funktionen integriert werden. Danach erfolgte noch weitere Personalisierungen der Visualisierungen um sich die gewollten Werte anzeigen zu lassen.

Ähnlich wurde beim Parallelplot vorgegangen, wobei der *type Msg* hierfür noch um zwei weitere Variablen erweitert wurde, damit alle Möglichkeiten für die Achsen abgedeckt wurden. Bei der Erstellung dieser Visualisierung wurde sich an der Übung sechs orientiert. Dadurch, dass diese auch bereits die Option des Daten-Laden behandest, ging dies deutlich schneller als die Erstellung des Scatterplots. Auch hier wurde wieder, wie beim Scatterplot mit der Hover-Funktion gearbeitet, um die einzelnen zu betrachtenden Datenwerte besser hervorheben zu können.

## 4.2 Explizite Baumdarstellung

Für die Visualisierung der expliziten Baumdarstellung wurde sich an Übung 7 orientiert. Die Daten wurden
Der *type Msg* ähnelt wieder dem der Plots. Abbildung 11, mit den veschiedenen dynamischen Möglichkeiten, die dem Nutzer gegeben werden um mit der Webseite zu interagieren. Die Daten wurden wieder über die *init* - Funktion geladen. Allerdings wurden sie diesmal nicht aus einer Csv-Datei geladen, sondern aus einer *json*. Wie bereits in 2.2 erwähnt, wurde diese hierfür manuell aus der Csv-Datei erstellt, damit mit dieser gearbeitet werden konnte. Im Gegensatz zu den vorherigen Plots wurde bei der Baumvisualisierung eine andere Form der Datendecodierung gewählt, die aus der Übung 7 übernommen wurde, siehe Abbildung 13.

```
treeDecoder : Json.Decode.Decoder (Tree String)
treeDecoder =
    Json.Decode.map2
        (\name children ->
            case children of
                Nothing ->
                    Tree.tree name []

                Just c ->
                    Tree.tree name c
        )
        (Json.Decode.field "name" Json.Decode.string)
        (Json.Decode.maybe <|
            Json.Decode.field "children" <|
                Json.Decode.list <|
                    Json.Decode.lazy
                        (\_ -> treeDecoder)
        )
```

Abbildung 13: Decode Funktion der Baumvisualisierung

Die Interaktionsmöglichkeiten s. mehr dazu in 3.4 sind ähnlich aufgebaut wie bei Scatterplot und Parallelplot. In der *update* - Funktion werden die verschiedenen Messages angegeben, in denen die Optionen abgedeckt werden vgl. . 12, die dem Nutzer vorliegen. Also die Höhe, Weite, etc. des Baumdiagramms zu ändern. Wie in den beiden Plots die Punkte „gezeichnet“wurden, so werden hier mit den Funktionen *Knoten* und *Kanten* die Knoten und Kanten „gezeichnet“. Die Kanten verbinden die einzelnen Knoten miteinander, um so das explizite Baumdiagramm darstellen zu können.

Am schwierigsten gestaltete sich bei dieser Implementierung das Laden der Daten. Genauer gesagt das Umstrukturieren der Csv-Datei in eine *json*. Hierbei wurde anfangs versucht dies mit einer *python* - Anwendung umzuschreiben. Dieser Ansatz funktionierte leider nicht, da die Funktion keine genestete Datei als Ergebnis ausgab und es somit nicht für die Baumdarstellung genutzt werden konnte.

Nach mehrmaligem Rumprobieren wurde die Datei daher manuell erstellt. Dies führte allerdings zum nächsten Problem: Bei der Darstellung der verschiedenen Noten kam es häufig zu Dopplungen bei den *children*, da sich beispielsweise die Noten der besten Frau mit Lernzeit 0-30 Minuten mit der besten Frau der Lernzeit 30-60 Minuten deckten.

# 5 Anwendungsfälle

Nicht einzelne Personen, sondern Muster im großen ganzen sollten erkannt werden, daher stellt es sich etwas schwieriger dar, dieses als einzelnes Bild in der Hausarbeit darzustellen. Vielmehr sollte der Leser dies in den Visualisierungen parallel selber ausprobieren. Anhand der nächsten drei gewählten Anwendungsfälle werden die erstellten Visualisierungen etwas tiefgehender elaboriert.

- Informationsbeschaffung Elternteil des Notenverlaufs bei verändernder Lernzeit
- Verlauf und Zusammenhang der Noten mit sich ändernder Lernzeit
- Korrelation Notendurchschnitt mit Geschlecht und Lernzeit

## 5.1 Anwendung Visualisierung Eins

Ein Anwendungsfall mit dem Scatterplot wäre ein Elternteil, dass sehen will, wie sich die Noten der Studenten über die Jahre hinweg unter dem Einfluss der Lernzeit ansehen, um so eventuell Rückschlüsse auf das eigene Kind und deren mögliche Entwicklung zu ziehen. Dem Elternteil soll also die Gesamtverteilung der Noten nach einer eigen vorgenommenen Filterung zur Verfügung gestellt werden, sodass sie sich einen guten, möglichst unvoreingenommenen Überblick verschaffen kann. Der Scatterplot soll die Daten fast schon *plain* darstellen, sodass dem Benutzer keinerlei Dinge suggeriert werden.

Dies ist, wie in Abbildug 6 zu sehen ist sehr gut gelungen. Der Nutzer bekommt eine sehr einfache Visualisierung vorgestellt, auf denen er einzig die Achsen und die dargestellten Einzeldaten einsehen kann. Beim Fahren mit der Maus über die einzelnen Punkte erhält der Leser allerdings kleine weiter Informationen, wie das Geschlecht und die Zeit, in welcher die Studenten am liebsten lernen. Im Gegensatz zum Parallelplot in 5.2 sind dies wenige und Informationen. So soll weiterhin gewahrt werden, dass der Leser einen allgemeineren Überblick behält.
Mithilfe der in 3.4 erläuterten *Hover* - Funktion lässt sich hier außerdem weitere Informationen über den Einfluss der Lernzeit generieren. So wird nicht nur die Dauer beachtet, sondern zusätzlich noch die Uhrzeit am Tag.

## 5.2 Anwendung Visualisierung Zwei

Im zweiten Beispiel wird eine Visualisierung anhand des Parallelplots angewandt. Hier wird der Notenverlauf in Abhängigkeit der Variable Lernzeit und der am Ende verbundenen Gehaltser-

wartung untersucht. Es sollte also erkennbar sein, dass Studenten die viel Lernen gute Noten und auch eine höhere Gehaltsvorstellung haben, als wenig lernende Studenten, die eher schlechte Noten und dahingehend auch mit schlechterem Gehalt rechnen sollten.

Das dies allerdings nicht immer der Fall ist zeigt sich vor allem deutlich bei der Lernzeit „Mehr als 4 Stunden “. Hier hat eine Studentin also die höchstmöglich auswählbare Lernzeit angegeben und war zwar in der 10. Klasse mit der Note *80* und der 12. Klasse mit *70* zwar noch relativ gut, ging im aber deutlich nach unten. Hier erreichte sie nur die Note *12*. Sie hat also trotz sehr hohem Lernaufwand im College schlechte Noten und gibt trotzdessen den in dieser Rubrik höchsten Wert in der Gehaltserwartung an, mit *30000*.

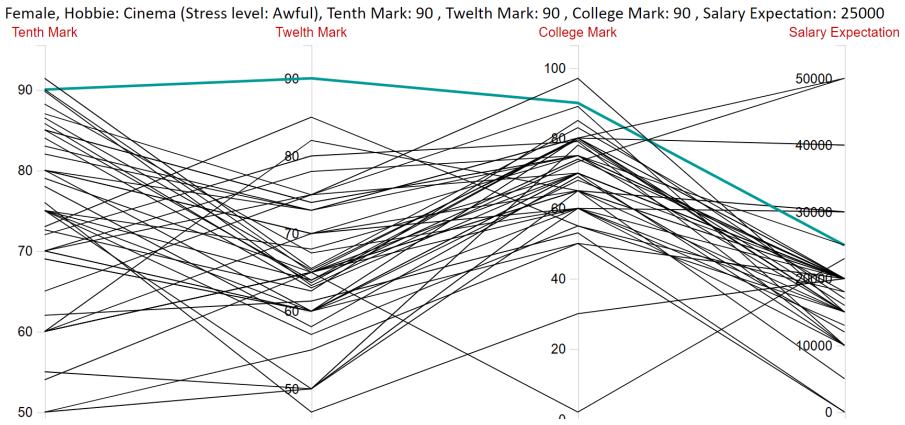
Dies spiegelt also in keinster Weise die im vorhinein getroffenen Erwartungen wieder, dass die vier Variablen bei allen in einem Zusammenhang stehen.
Natürlich lässt sich nicht von einem Individuum auf alle schließen und könnte als Ausreißer dargestellt werden, dennoch spricht es gegen die aufgestellte These.

Viele Gegenbeispiele lassen sich zudem bei der Lernzeit „0-30 Minuten herausfiltern“. Trotz der relativ niedrigen Lernzeit gibt es viele Studenten, die gute bis sehr gute Noten über die Klassen und das College hinweg haben. Zwei dieser Beispiele sind deutlich in Abbildung 14 zu sehen.

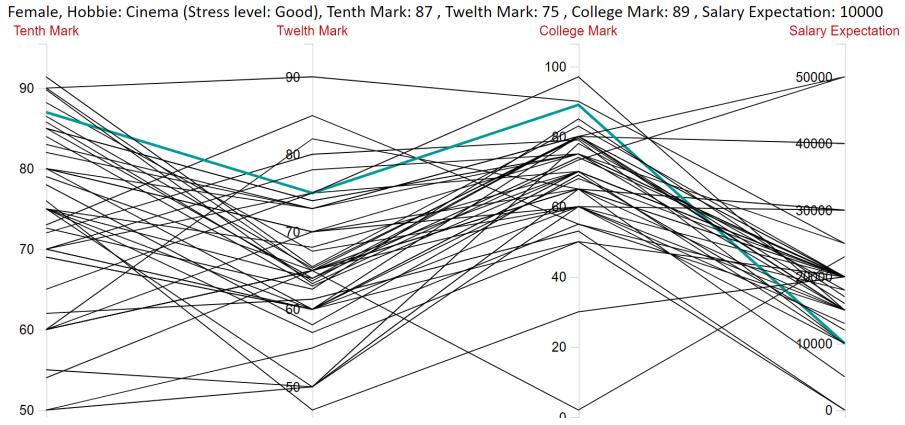
## 5.3 Anwendung Visualisierung Drei

Da bei der Baumvisualisierung erst im Geschlecht und dann in die verschiedenen Lernzeiten unterteilt wird und als letztes „Kind“der Durchschnitt gezeigt wird, soll geschaut werden, inwieweit sich der Durchschnitt bei steigender Lernzeit verändert. Auch die Differenzen zwischen den Geschlechtern soll hier beachtet werden.Dies ist in der gezeigten Abbildung 15 nocheinmal bildlich visualisiert worden.
Bei den Männern ist eine durchaus positive Korrelation zwischen der Lernzeit und dem Notendurchschnitt zu erkennen, vor allem wenn die beiden Extrema verglichen werden. Bei der geringsten Lernzeit liegt der Durchschnitt bei 64, wohingegen er bei der höchsten Lernzeit bei 77 liegt. Es ist also ein starker Anstieg von 13 Notenpunkten im Durchschnitt zu erkennen. Zudem ist eine fast permanente Steigung über alle Lernzeiten hinweg zu erkennen. Nur einmal sinkt der Wert anstatt zu steigen und zwar von 68 auf 67. Danach steigt der Wert jedoch wieder zweimal an.
Bei den Männer lässt sich anhand der vorliegenden Grafik also die Aussage treffen, dass die Lernzeit und der Notendurchschnitt positiv miteinander korrelieren.

Bei den Frauen ist dahingehend leider keine ähnliche Aussage zu treffen. Vor allem der Ausreißer bei der höchsten Lernzeit von über 4 Stunden zieht den Durchschnitt sehr nach unten. Dies ist aber wahrscheinlich auf die geringen Anzahl der Frauen in diesem Segment zurückzuführen.



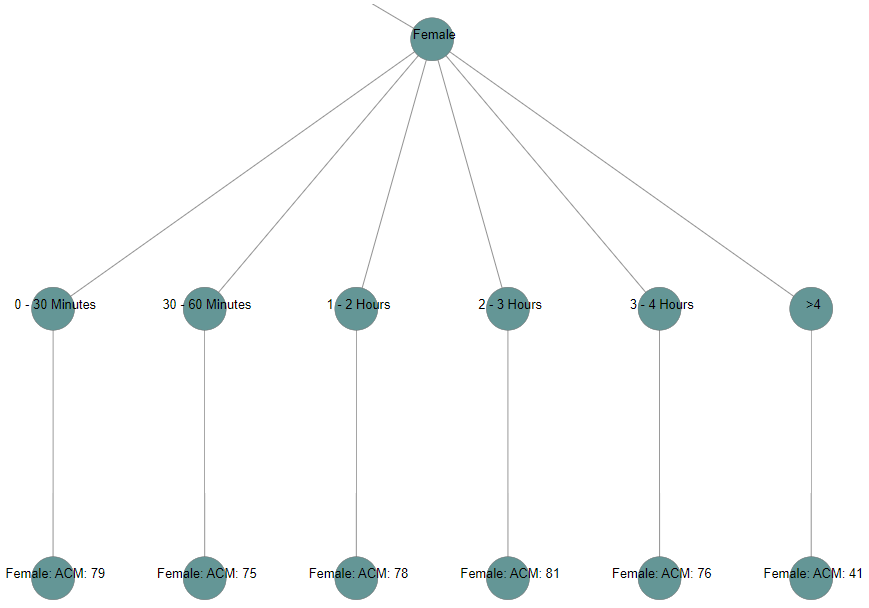

(a)




(b)

Abbildung 14: Zwei Beispiele Parallelplot - Lernzeit: 0 - 30 Minuten


(a) Baumdiagramm Männer




(b) Baumdiagramm Frauen

Abbildung 15: Gegenüberstellung des Notendurchschnitts der Lernzeit bei Männern und Frauen

Zudem ist hier keine stetige Steigung der Werte zu erkennen.

# 6 Verwandte Arbeiten

In der ersten Arbeit von [1] wurde „PerformanceVis“entwickelt, ein visuelles analytisches Werkzeug, mit dessen Hilfe die Leistung und Zulassungsfähikeit der Studenten geprüft werden soll. Hierfür wurde ein Einführungskurs in Chemie der Universitätsweit angeboten wird, mit knapp 1000 Studenten untersucht. Dort wurden in einer Onlinelernplattform Dinge getrackt, wie z.B. Die Häufigkeit der Logins, die Benutzung verschiedener Hilfsmittel und „PerformanceVis“hat sich dabei hauptsächlich auf die Lernspuren der Schüler fokussiert.

Wie in der hier beschriebenen Arbeit und der angefertigten Visualisierungen arbeitet auch [1] mit mehreren Parallelplots, wovon einer in Abbildung dargestellt ist. Hierbei wurden zwei Zielgruppen miteinander verglichen, gekennzeichnet mit den beiden unterschiedlichen Farben. Während die blau gefärbten *polylines* Schüler darstellen, die ein $C$ sowohl im dritten als auch dem letzten Examen darstellen, repräsentieren die orangen gefärbten *polylines* Schüler, die im dritten Examen ein $C$, im finalen Examen aber ein $A$ hatten. Es ist also deutlich erkennbar, dass die Schüler die zu Beginn in den ersten beiden Examen gut waren sich deutlich besser „erholten“von einem $C$, als Schüler, die davor bereits schlechte Noten hatten.

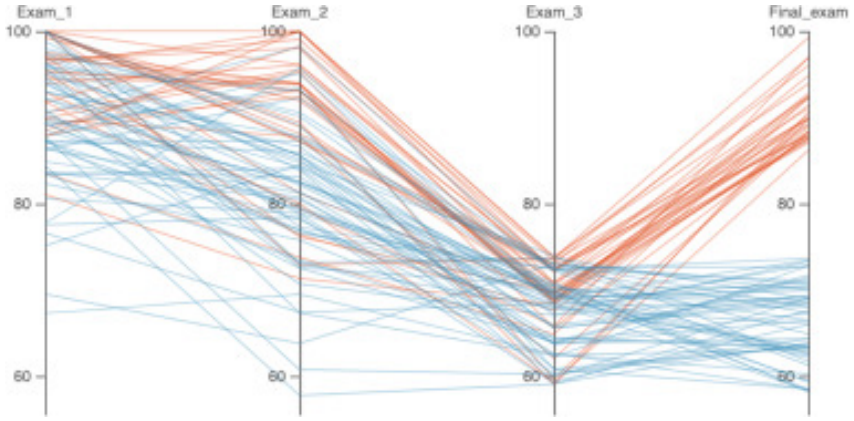



Abbildung 16: Parallelplot mit den unterschiedlichen „Exams“

Für das *tool* „PerformanceVis“und zum besseren Verständnis der „student performance [fluctuation] throughout the course, we should display the grade distributions for each exam.“[1]. Es wurde also Wert darauf gelegt diese Fluktuation mithilfe der Notenverteilung in einem Parallelplot darzustellen, genauso wie es auch in dieser Arbeit der Fall war. Bei dieser Arbeit und vor allem bei dieser Visualisierung ist die Schlussfolgerung, dass die Noten sich untereinander beeinflussen. Bei der eigenen Arbeit konnten wir nicht zu dem gleichen Schluss kommen. Allerdings ist dies wahrscheinlich eher auf die Diskrepanz in der Stichprobengröße zurückzuführen, da in [1] diese fast viermal so groß ist ( 1000 zu  250).

Was sich allerdings ähnelt ist die Art der Visualisierungen. In [1] wurde sich nicht nur auch für den Parallelplot, sonder außerdem auch für ein Baumdiagramm entschieden.

Das Paper von [6] behandelt als eins der wenigen wissenschaftlich auffindbaren Paper im Bereich Informationsvisualisierung den Zusammenhang *Student Behaviour* und diesen Einfluss auf das Abschneiden der Studenten. Dafür werden die nötigen Charakteristiken der studentischen Teilnehmer an den „MOOCs“(Massive Open Online Courses) visuell beschrieben. Von [6] wird zitiert, dass weder Alter noch Geschlecht Einfluss auf den höchsten Abschluss haben. Zu ähnlichen Ergebnissen kam diese Arbeit, als sich keine signifikanten Unterschiede zwischen Männern und Frauen bei dem *Grade average* offenbarte.

In diesem Paper wurde eine etwas andere Art der Visualisierung gewählt 17 und auch ein etwas anderer Ansatz. Xu geht hier nicht so sehr auf den Einflussfaktor Lernzeit ein, sondern vielmehr auf das stetige Mitarbeiten in einem Kurs mit Einfluss auf die Noten. Das stetige Mitarbeiten wird hier im Sinne von „quiz submissions“dargestellt.

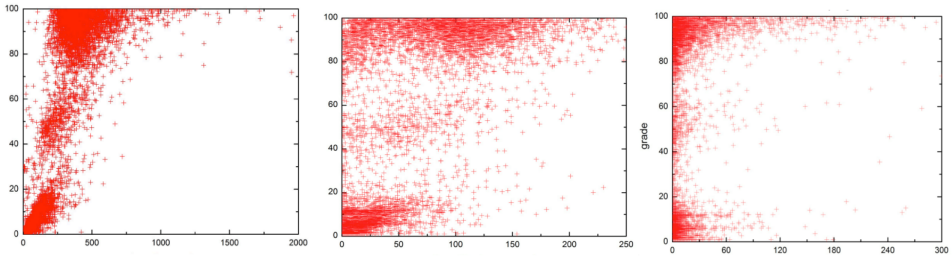



Abbildung 17: Plots von „Grades“against „quiz submissions“

Weiterhin beschreibt [6] „[...] there are multiple ways to be successful in a course, and it is perhaps not necessary to do everything – at least, not for everyone“.
Er stellt hier also die These auf, dass es nicht unbedingt den einen einzelnen Einflussfaktor auf die Note und Erfolgsquote in einem Kurs gibt. Wie auch wir bereits in 1 ähnlich beschrieben haben gibt es deutlich mehr, viele davon lassen sich nicht messen, wie die Kindheit oder die Lernumgebung und der soziale Status der Familie.

# 7 Zusammenfassung und Ausblick

In dieser Arbeit wurde sich viel mit den Zusammenhängen zwischen Noten untereinander und auch der investierten Lernzeit auseinandergesetzt. Es sollte dem Benutzer außerdem ermöglicht werden die Visualisierungen schnell zu überschauen und beim näheren Auseinandersetzen damit aber mehr Möglichkeiten geben. Der Nutzer sollte die verschiedenen Einflussfaktoren sich per-

sönlich darstellen lassen können, ohne den Überblick zu verlieren.

Besonders gut gelungen ist dies beim Parallelplot. Hier eignete sich der Datensatz und die Unterteilung in die Lernzeiten hervorragend für die gestellten Zielsetzung. Der Datensatz war nicht zu überfüllt, dass der Nutzer aufgrund einer Vielzahl von *polylines* den Überblick verliert. vielmehr barg jede einzelne Auswahlmöglichkeit genug Daten um diese gut veranschaulichen zu können. Die Abhängigkeit der Noten untereinander im Zusammenspiel mit der ausgewählten Lernzeit und der Gehaltserwartung konnten hier gut abgebildet werden. Mit den Hover-Elementen konnte außerdem eine einzelne Person hervorgehoben werden und es wurden zusätzliche Informationen geliefert um dem Benutzer ein besseres Allgemeinbild zu liefern.

Ähnlich verhielt sich diese Thematik beim Scatterplot. Dieser wurde allerdings simpler gehalten. Es wurden nicht alle Informationen auf einmal abgebildet, sondern man ließ dem Benutzer immer die Wahl zwischen zwei verschiedenen. Diese Drop-down-Funktionen und die *Hover* - Funktion gabend dem Leser somit einige Freiheiten, mit denen er die Daten genauer inspizieren konnte.

Für einen großen Teil der angesprochene Zielgruppe (Schüler, Lehrer, Eltern), also vor allem Anfänger im Umgang und der Interpretation von Visualisierungen waren diese drei Möglichkeiten also gut gewählt.

Sinnvolle Erweiterungen für den Datensatz wären beispielsweise, die bereits in 1 angesprochenen weiteren Faktoren, die die Noten weitergehend beinflussen, um so mögliche Standardfehler zu verringern. Anbieten würde sich dabei die Bildung oder der finanzielle Status der Eltern, diesen könnte man wieder klassieren und so als Drop-Down Möglichkeit auswählen.

Zudem könnte das Projekt auch noch um zusätzliche Visualisierungen erweitert werden. In 3 wurden in den jeweiligen Unterkategorien der Visualisierungen 3.3.1, 3.3.2 und 3.3.3 jeweils eine Alternative Visualisierungsmöglichkeit zur derzeit vorhandenen vorgestellt. Man könnte diese dem Nutzer auch noch zur Verfügung stellen, um ihm so die Option eines anderen Blickwinkels aber der gleichen Visualisierungstechnik offen zu lassen.
Genauso können aber weitere ergänzende Visualisierungen eingebettet werden, wie beispielsweise ein Mosaikplot.

Abschließend sollen noch kurz die in 1 gestellten Fragen beantwortet werden:

- Wie korrelieren Noten und Zeitgestaltung und lassen sich daraus Rückschlüsse ziehen, wenn ja welche?
    - ⋆ Es lässt sich kaum eine Korrelation erkennen, möglicherweise liegt dies aber auch an Ausreißern

- Wie stehen die Noten im Zusammenhang mit anderen Variablen? ...
  - ⋆ Es lässt sich kein Zusammenhang zwischen den Noten untereinander erkennen
  - ⋆ Auch kein erkennbarer Zusammenhang zwischen Noten und *salary expectation*
- Wie können die verschiedenen Daten übersichtlich veranschaulicht werden um dem Leser einen schnellen Überblick zu geben ...
  - ⋆ Dies geschah mit Hilfe der Visaulisierunge Parallelplot, Scatterplot und Baumdiagramm

## Literatur


[1] Deng. "PerformanceVis: Visual analytics of student performance data from an introductory chemistry course". In: University of Rochester, Rochester, NY 14627, United States, 2019. DOI: https://doi.org/10.1016/j.visinf.2019.10.004. URL: https://www.sciencedirect.com/science/article/pii/S2468502X1930049X?via%3Dihub.

[2] Dr. Ralf Dörner. *Information Retrieval und Visualisierung, Schwerpunkt Information Visualisierung; Vorlesungsskript.*

[3] Earthman. "REVIEW OF RESEARCH ON THE RELATIONSHIP BETWEEN SCHOOL BUILDINGS, STUDENT ACHIEVEMENT, AND STUDENT BEHAVIOR". In: (1996).

[4] Hinneburg. *Information Retrieval und Visualisierung, Schwerpunkt Information Visualisierung; Vorlesungsskript.*

[5] Tamara Munzner. "Process and Pitfalls in Writing Information Visualization Research Papers". In: *Information Visualization: Human-Centered Issues and Perspectives*. Hrsg. von Andreas Kerren u. a. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 134–153. ISBN: 978-3-540-70956-5. DOI: 10.1007/978-3-540-70956-5_6. URL: https://doi.org/10.1007/978-3-540-70956-5_6.

[6] Xu. *Visual Analytics of MOOCs at Maryland*. Atlanta, Georgia, USA, 2014. DOI: 10.1145/2556325.2567878. URL: https://dl.acm.org/doi/10.1145/2556325.2567878.


## Anhang: Git-Historie

* a4ed876 (HEAD -> main, origin/main, origin/HEAD) (Finale Änderung an Main, 2022-12-21) * 0a7c2a0 (Finale Änderung Code, 2022-12-21) * 40a808a (Bericht fast fertig. Bis auf Git-Historie, 2022-12-21) * faf8ad8 (Bericht Zusammenfassung, 2022-12-21) * f8dd5ea (Abschluss anderer Arbeiten, 2022-12-20) * 7e441da (Bericht Anwendung Visualisierung 1-3, 2022-12-20) * 5e01a8b (Update ParallelPlot.html, 2022-12-20) * b2b197a (Ändern Html, 2022-12-20) * f42252d (abändern der Html Datei, 2022-12-20) * 3110aa7 (Abändern der Json, 2022-12-20) * ac86917 (Ändern json, 2022-12-20) * 04a894c (Ändern json, 2022-12-20) * 4002968 (Erstellung neuer Json, 2022-12-20) * 813c5cc (Erstellung Html-Seiten und bugfixing bei Scatterplot, 2022-12-20) * 24c9bbd (Aktualisierung Bericht, 2022-12-20) * 7cd8522 (Einfügen Html-Links, 2022-12-20) * b17aad5 (Bericht Fertig Visualisierungen, 2022-12-20) * 38bc375 (Änderung des Github-Links, 2022-12-19) * 0ab5b34 (Bericht fast Visualisierung zwei fertig, 2022-12-19) * 79e8cec (Bericht Visualisierung eins, 2022-12-19) * 2641485 (Bericht weiterschreiben Vis1, 2022-12-19) * c2bd64a (Versuche Umformulierung Github-Link, 2022-12-19) * 71b7084 (Schreiben Bericht Anforderung an Vis, 2022-12-19) * c82c324 (Änderung ParallelPlot, 2022-12-19) * 1cd92b8 (Lertze Ausreißer, 2022-12-19) * b49ba25 (Letzter Ausreißer, 2022-12-19) * 643af32 (Zweiter und dritter Ausreißer, 2022-12-19) * 60794e0 (Ändern zweiten Ausreißers, 2022-12-19) * 09b3ca2 (Anpassung zweiten

Ausreißers, 2022-12-19) * 9ea7786 (Datenmanipulation in excel, 2022-12-19) * 8e86744 (Anpassung Bericht und Visualisierung, 2022-12-19) * 676ccc0 (Bericht - Analyse Anwendungen, 2022-12-19) * 3b2a645 (Ändern der Scatterplotbeschriftungen, 2022-12-19) * ee240ce (Schreiben des Berichts - Visualisierung, 2022-12-18) * 1b7634b (Hinzufügen Salary Expectation, 2022-12-18) * ae5503f (Schreiben des Berichts - Absatz Daten, 2022-12-18) * e035234 (Schreiben der Zielgruppe, 2022-12-18) * cb75906 (Schreiben des Berichts und Anpassen Scatterplot, 2022-12-18) * 42007d6 (Bericht anfang und Parallelplot hinzufügen Button, 2022-12-15) * 55f7387 (Einfügen des Dropdown Buttons für die Lernzeit, 2022-12-14) * 9dde0e6 (fertiger Paralleplot. Noch ohne Interaktion, 2022-12-14) * b5bf62b (Aufstellen der View-Funktion, 2022-12-14) * ffbcd3f (Abändern der anzeigenden Daten auf Linie, 2022-12-14) * 5985ca6 (Abändern des Parallelplot, 2022-12-14) * 54b977c (Weiterbearbeitung Parallelplot und erstellung Parallelplot, 2022-12-14) * 80f964f (Fast finale Lösung der Baumdarstellung, 2022-12-14) * 7c4ee7d (Änderung Json zu Fehlerbehebung, 2022-12-14) * 6754cf1 (Erste Beginne der ParallelPlots, 2022-12-14) * c8a3ede (Json fertig noch ohne letztes Kind., 2022-12-14) * 6e5727b (changing to final json, 2022-12-14) * b2a1afa (Hinzufügen der Json im selben Ordner, 2022-12-13) * f072361 (Anpassung der Json und Übernahme Code aus Übung, 2022-12-13) * f5b7f82 (einfügen zweiter csv für spätere eventuelle Bearbeitung, 2022-12-12) * 496e4ec (konvertieren der csv in json mithilfe von python, 2022-12-12) * 6ae4fd9 (Löschen alten Scatterplots und Erstellung Baumvis., 2022-12-12) * 03cff2f (Anpassen der Ordnerstruktur im GitHub, 2022-12-12) * e852233 (Bericht Vorlage eingefügt, 2022-12-12) * 37dfae0 (fertiger Scatterplot mit angezeigten Punkten, 2022-12-11) * a8c7d08 (Visualisierung Scatterplot mit drop down, 2022-12-11) * 0d23b10 (updaten der view funktion und einfügen HTMl, 2022-12-08) * 47b29bc (update, 2022-12-08) * 5516ac7 (Scatterplot Anpassung der Scatterfunktionen update und view, 2022-12-08) * 8613cfe (Änderungen der Funktion im Scatterplot, 2022-12-08) * d5326a4 (Scatterplot, 2022-12-07) * 35cf0ba (Erstellen Scatterplot Show, 2022-12-06) * 8dbda6c (Änderungen bei Main der Msg Fkt und Update Fkt, 2022-12-05) * a5bf6fa (Änderung update, 2022-12-05) * 56bc3dd (Anpassung Fehler, 2022-12-05) * 32f3030 (erste Änderungen am Main view, für ScatterImplementierung, 2022-12-03) * 19d4db7 (Anpassen Scatterplot., 2022-12-03) * 3243927 (Erste Versuche Erstellung Scatterplot, 2022-12-01) * 4063f05 (Anzeigen der Daten auf Main und Decoden plus Hochladen, 2022-12-01) * 9559d68 (rumprobieren mit Decoder und Anzeigen lassen auf localhost, 2022-11-29) * c109a87 (Erstellen der Daten und anzeigen lassen auf Localhost, 2022-11-29) * 6c70315 (Hochladen Datenordner, 2022-11-29) * 18e2af1 (Initial commit, 2022-11-29)