Projektbericht zum Modul Information Retrieval und Visualisierung Sommersemester 2022

Visualisierung des Datensatzes Student Behaviour

Paul Tebbe, Matrikelnummer: 216237588

21.12.2022

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 Standart die Eltern haben, welche finanziellen Mittel usw. Weiter spielt die Erziehung und der spätere soziale Umgang eine Rolle. [**Earthman**] Diese Einflüsse sind schwer zu messen, vor allem schwer genau zu messen. Diese Faktoren sind kaum von alleine beeinflussbar, es lässt sich nicht bestimmen, in welche Familie ich geboren werde und welche Voraussetzungen ich also für später mit auf den Weg gegeben bekomme. 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. 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

Tipps zu Latex und Koma-Script für Hausarbeiten sind im LaTeX Reference Sheet for a thesis with KOMA-Script von Marion Lammarsch und Elke Schubert zusammengefasst. Der Bericht fällt in die Kategorie von InfoVis-Paper, die Tamara Munzner Design Study nennt [**Munzner2008**]: In der Einleitung sollen sie zuerst das Zielproblem beschrieben. Daraus sollen sie Fragestellungen motivieren, die mittels Techniken der Informationsvisualisierung beantwortet werden können. In dem Abschnitt direkt unter der Überschrift Einleitung sollen Sie nach einer kurzen Einleitung Fragestellungen und das Zielproblem motivieren und beschreiben. ...

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. “

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. Auch interessant könnte dieses Projekt aber auch für Forschende im Bereich soziale Ungleichheitne 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.

Beschreiben sie die Personengruppe oder Personengruppen, die das von ihnen benannte Anwendungsproblem lösen möchte. Auf welches Vorwissen können sie in dieser Gruppen von Anwenderinnen aufbauen? Welche Informationsbedürfnisse werden durch die Visualisierungen adressiert?

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.ten und 12.Klasse gute Noten hatte, sollte nun auch im College gut abschneiden. 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.

In diesem Abschnitt geben sie einen kurzen Überblick über die Daten und verwendeten Visualisierungen. Dann benennen sie die Beiträge ihres Projekts. Diese Beiträge müssen sie in den hinteren Teilen des Berichts genauer ausführen und belegen.

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 *travelling 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. 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. Aufgrund der späteren Nutzung der *daily studying time* als Drop-down Tabelle wurde dieser Gedanke wieder verworfen.

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:innen 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 einzelnen Visualisierungen wurden daher auf den ersten Blick relativ simple gehalten. 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 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 anfägnlich 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

Präsentieren sie die visuelle Abbildungen und Kodierungen der Daten und Interaktionsmöglichkeiten. Sie müssen begründen, warum und wie gut ihre Designentscheidungen die erstellten Anforderungen erfüllen. Weiterhin müssen sie begründen, warum die gewählte visuelle Kodierung der Daten für das zulösenden Problem passend ist. Typische Argumente würden hier auf Wahrnehmungsprinzipien und Theorie über Informationsvisualisierung verweisen. Die besten Begründungen diskutieren explizit die konkrete Auswahl der Visualisierungen im Kontext von mehreren verschiedenen Alternativen. Machen sie hier nicht den Fehler, einfach nur Visualisierung aus den vorgegebenen Bereichen zu diskutieren, weil das in der Regel nicht sinnvoll ist. Wenn sie sich für einen Scatterplot entschieden haben, ist ein Zeitreihendiagramm in der Regel keine Alternative. Diskutieren sie also nicht einfach Zeitreihendiagramme, weil sie in den Anforderungenen an das Projekt neben Scatterplots stehen, sondern suchen sie nach echten alternativen Visualisierungen, die zum Aufbau eines vergleichbaren mentalen Modells führen. Diskutieren sie die Expressivität und die Effektivität der einzelnen 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.

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

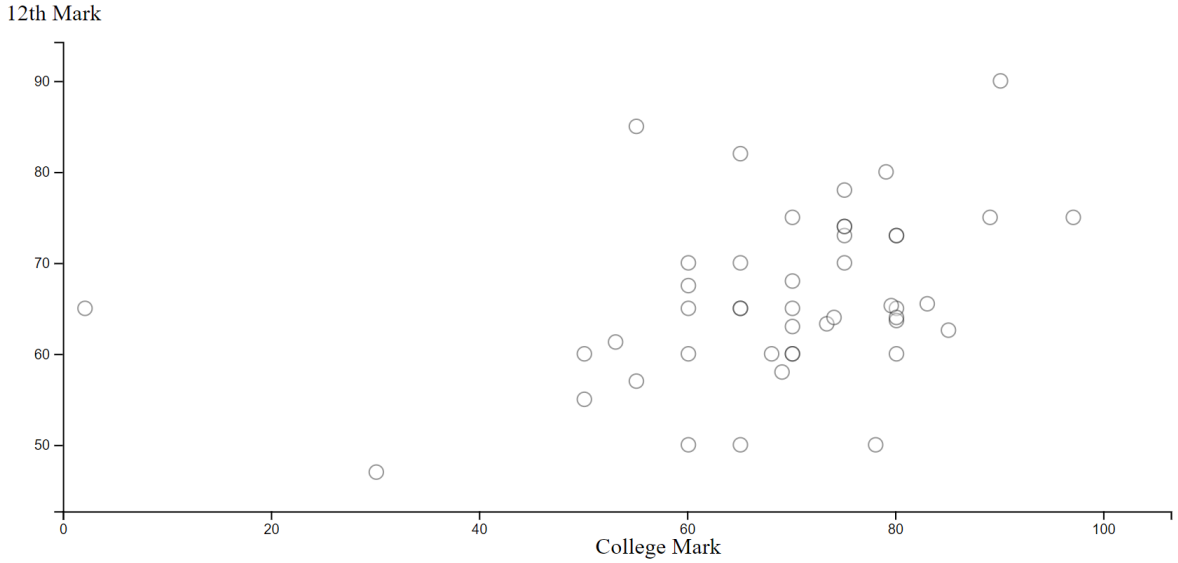


Abbildung 1: Scatterplot

Achsen, die X-Achse und die Y-Achse. Deren Größe kann entweder manuell bestimmt werden oder hat vier weitere Möglichkeiten gleiche Daten verschieden darzustellen([**Hinneburg2022**])

- 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

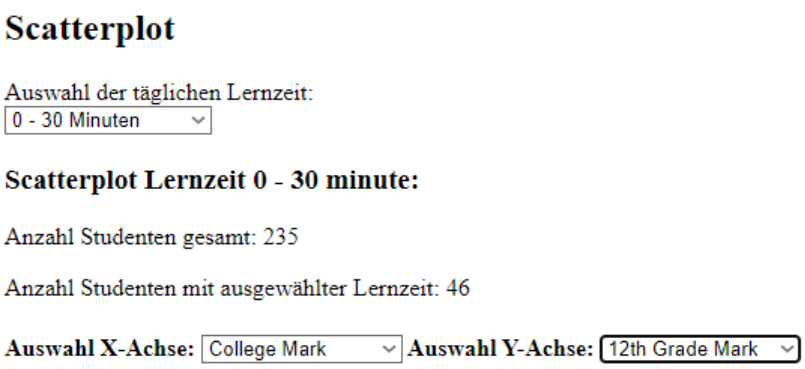


Abbildung 2: Ausgewählte Drop-Down-Funktionen und Head

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 wieviele Studenten es insgesamt gibt und außerdem wieviele Studenten von dieser Gesamtanzahl grade ausgewählt sind. Also wieviele Studenten von 235 0-30 Minuten lernen, in dem Fall 46.
So kann sich der Leser ein besseres Bild dazu machen, wie sich die Lernzeit grobt 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“ [**Hinneburg2022**]. 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) 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.

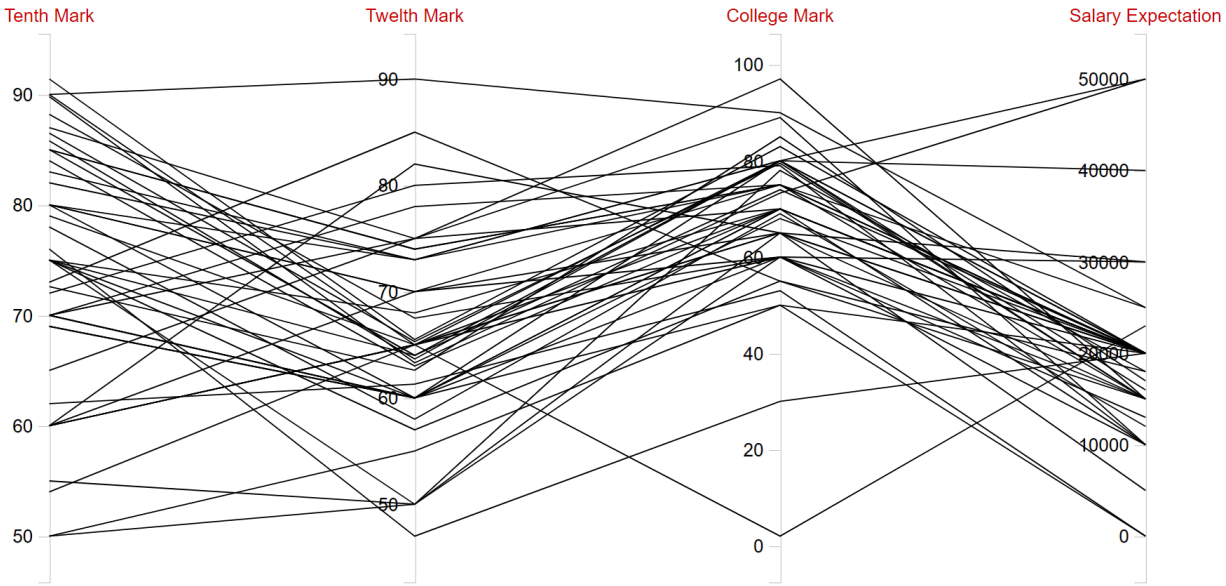


Abbildung 3: Parallelplot

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.

Abbildung 4: Drop-Down Fenster und Head des Parallelplot

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 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 die Ünubersichtlichkeit 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.

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 [**Doerner0304**].
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 [**Hinneburg2022**].
Die auf die Daten angewendete Baumdarstellung ist in Abbildung 5 zu sehen.

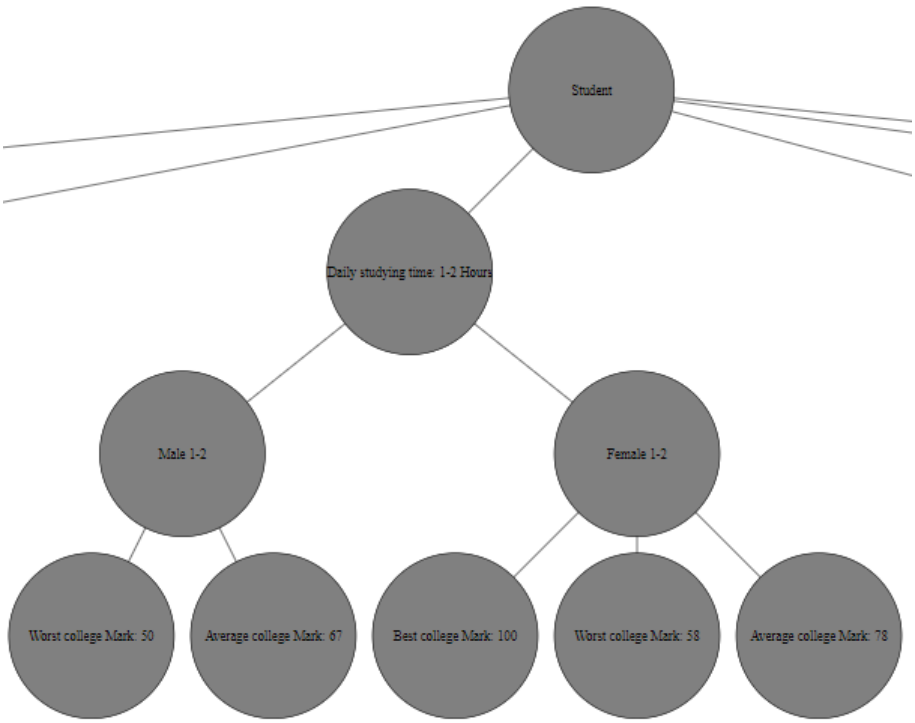


Abbildung 5: Explizite Baumdarstellung

Die Wurzel stellt hier der einzelne Student dar. Dieser hat sechs sogenannte Kinder, also Knoten, die mit gerichteten Kanten von ihm wegführen und sie die Beziehung zueinander visuell darstellen. Diese sechs Kinder sind die unterschiedlichen Lernzeiten, die bereits in den vorherigen Visualisierungen immer als Drop-Down zur Verfügung standen.
Diese Knoten wiederum haben eine weiter Unterteilung in Männlich und in weiblich, umso erneut zwischen den Geschlechtern Differenzen feststellen zu können. Die letzten Kinder sind die Noten. Diese wurden diesmal nicht nach Klassen und College unterteilt, sondern nach der besten, schlechtesten und 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 exakten Noten inklusive des Durchschnitts dargestellt werden können und so dem Leser direkt als String geliefert werden, es wird also keinerlei Hover-Funktion benötigt. 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 [**Hinneburg2022**].
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 [**Hinneburg2022**]. 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

Die präsentierten Visualisierungstechniken müssen interaktiv zu einer Anwendung verknüpft werden. Die Interaktion mit einer Visualisierung soll in den anderen Visualisierungen zu einer Änderung führen. Erklären sie die möglichen Interaktionen mit den einzelnen Visualisierungen und die möglichen Verknüpfungen zwischen ihnen. Begründen Sie warum die konkreten Interaktionen umgesetzt wurden und welche Zwecke für die Anwenderinnen mit ihnen unterstützt werden. Begründen sie ebenfalls warum sie andere Interaktionsmöglichkeiten nicht umgesetzt haben. Wenn sie keine der geforderten Interaktionen umsetzen, erhalten Sie im gesamten Projekt deutlichen Punktabzug.

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 6. 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 6: Main

Nachdem die Laden 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 7.

```
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 7: 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 8. 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.

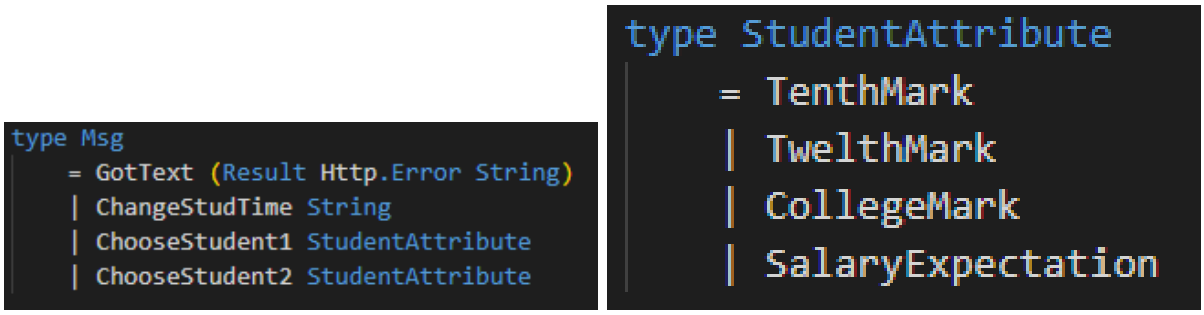
```
type alias Student_Data =
    {   certification : String
    ,   gender : String
    ,   department : String
    ,   height : Float
    ,   weight : Float
    ,   tenthMark  : Float
    ,   twelthMark  : Float
    ,   collegeMark : Float
    ,   hobbies : String
    ,   dailyStudyingTime : String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

Abbildung 8: 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* 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 9.
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.



(a) *type* - Definition *Msg*

(b) *type* - Definition *StudentAttribute*

Abbildung 9: *type* - Definition im Beispiel Scatterplot

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 10. 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 10: 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 s. Abbildung 9, 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 11.

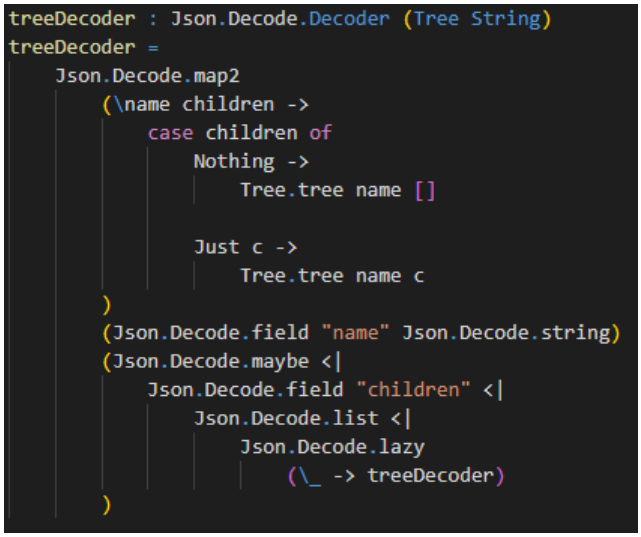


Abbildung 11: Decode Funktion der Baumvisualisierung

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. . 10, 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

Präsentieren sie für jede der drei Visualisierungen einen sinnvollen Anwendungsfall in dem ein bestimmter Fakt, ein Muster oder die Abwesenheit eines Musters visuell festgestellt wird. Begründen sie warum dieser Anwendungsfall wichtig für die Zielgruppe der Anwenderinnen ist. Diskutieren sie weiterhin, ob die oben beschriebene Information auch mit anderen Visualisierungstechniken hätte gefunden werden können. Falls dies möglich wäre, vergleichen sie die den Aufwand und die Schwierigkeiten ihres Ansatzes und der Alternativen.

5.1 Anwendung Visualisierung Eins

5.2 Anwendung Visualisierung Zwei

5.3 Anwendung Visualisierung Drei

6 Verwandte Arbeiten

Führen sie eine kurze Literatursuche in der wissenschaftlichen Literatur zu Informationsvisualisierung und Visual Analytics nach ähnlichen Anwendungen durch. Diskutieren sie mindestens zwei Artikel. Stellen sie Gemeinsamkeiten und Unterschiede dar.

7 Zusammenfassung und Ausblick

Fassen sie die Beiträge ihre Visualisierungsanwendung zusammen. Wo bietet sie für die Personen der Zielgruppe einen echten Mehrwert.

Was wären mögliche sinnvolle Erweiterungen, entweder auf der Ebene der Visualisierungen und/oder auf der Datenebene?

Anhang: Git-Historie