Universität Regensburg

Visualisation and interactive exploration of data changes in data engineering workflows

Bachelorarbeit im Fach Medieninformatik

am Institut für Information und Medien, Sprache und Kultur (I:IMSK)

Vorgelegt von:	Maximilian Schmerle
Adresse:	Engelburgergasse 15, 93047 Regensburg
E-Mail (Universität):	Maximilian.Schmerle@stud.uni-regensburg.de
Matrikelnummer:	2040882
Erstgutachter:	Dr. Johanna Bogon
Zweitgutachter:	Prof. Dr. -Ing. habil. Meike Klettke
Betreuer:	Sebastian Strasser
Laufendes Semester:	8. Semester B.A. Medieninformatik
Abgegeben am:	2023-10-16

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Akronyme

CSS Cascading Style Sheets. 24, 34, 38, 80
CSV Comma-separated values. 23

DAG Directed Acyclic Graph. 14, 32, 33

HTML Hypertext Markup Language. 24, 34, 38, 80

JS Javascript. 24, 38
JSON Java Script Object Notation. 23, 33, 44, 47–49, 54, 57, 66, 79

PiP Pip Installs Packages. 27, 35
PNG Portable Network Graphics. 44

SVG Scalable Vector Graphics. 26, 29, 41, 43, 47, 81

XML Extensible Markup Language. 23

Zusammenfassung

Die Arbeit mit großen Datenmengen stellt einen deutlichen Trend der letzten Jahre dar. Im Hinblick darauf müssen diese Daten von Datenanalysten vorverarbeitet werden, um eine fehlerfreie Nutzung zu garantieren und somit die Qualität aller Folgeprozesse zu verbessern. Daher ist es wünschenswert, Methoden zur Unterstützung der Aufbereitung und Analyse dieser Vorverarbeitungsschritte zu bieten, um somit Fehler zu vermeiden und Klarheit über vollzogene Anpassungen der Daten zu schaffen.

Das Ziel in der vorliegenden Arbeit ist es, mit der Hilfe eines Prototyps, einen Ansatz zur Erfüllung dieser Anforderungen zu bieten. Dieser kann einfach in bestehende Vorverarbeitungsprozesse integriert werden und soll deren Qualität verbessern. Diese Aufbereitung der Vorverarbeitungsschritte wird vor allem durch die Visualisierung ebenjener vorgenommen und ein klarer Fokus liegt auf der Interaktivität dieser generierten Visualisierungen. Ferner liegen die generierten Informationen in einer breiten Palette verschiedener Ausgabeformen vor, um möglichst verschiedene Einsatzgebiete der Anwendung zu erschließen. Ein ebenso weit gefächerter Ansatz soll sich auch in der Anzahl an verarbeitbaren Vorverarbeitungsschritten widerspiegeln, jedoch auch die Möglichkeit eingeräumt werden, modular spezifische Visualisierungen zu spezifischen Schritten anzubieten. Der Fokus liegt des Weiteren auf Python und die damit zusammenhängende Entwicklungsumgebung, sowie Bibliotheken.

Um dieses Ziel zu erreichen, wurden zu Beginn der Stand der Technik evaluiert, um eine Einordnung der Anwendung vorzunehmen. Darauffolgend wurden die Anforderungen bestimmt, die der Prototyp erfüllen sollte um die spätere Implementierung logisch begründen zu können. Daraufhin wird ausführlich die Architektur der Anwendung beschrieben, welche der Erfüllung ebenjener Anforderungen dient und ebenso Schlüsse zu Grundgedanken der geplanten Funktionsweise zulässt. Die abschließende Evaluation der Anwendung, zeigt zu Beginn ein Anwendungsszenario des Prototyps, welches einen genauen Blick auf die Nutzung, sowie die erstellten Ausgaben zulässt. Folgend wird diese Nutzung dann noch mithilfe von Probanden

und Leistungskennzahlen evaluiert und erlaubt somit eine Einschätzung des Prototyps. Diese Ergebnisse können als Anknüpfungspunkte weiterer Forschung genutzt werden und zeigen, Schwächen sowie Stärken der aktuellen Umsetzung auf.

Abstract

Over the last few years, working with large amounts of data has been a major trend. To guarantee flawless use and improve the quality of the following processes, data has to be pre-processed by data analysts. It is therefore vital to provide methods to support the preparation and analysis of these pre-processing steps to avoid errors and provide clarity about the adjustments made to the data.
The goal of this essay is to provide an approach to fulfilling these requirements with the help of a prototype that can be integrated into existing pre-processing cycles with the aim of improving their quality. The preparation of the pre-processing steps is mainly done by visualizing them, with a clear focus on the interactivity of these generated visualizations. Moreover, the generated information comes in a variety of output shapes in order to access as many different areas of application as possible. An equally large variety of approaches should be reflected in the number of pre-processing steps, but with the possibility of additionally offering modular visualizations for specific steps. Furthermore, the focus lies on Python and the related developmental environment, as well as libraries. Firstly, to achieve this goal, the state of the technology was evaluated to classify the application. Subsequently, the requirements were determined, which are to be upheld by the prototype to be able to justify later implementations logically. Then, the architectural property of the application is described, which serves to fulfill the requirements and allows conclusions to be drawn from the basic ideas of the planned functionality. The concluding evaluation shows a usage scenario of the prototype, which allows a detailed view of the usability, as well as the generated outputs. Lastly, the usage is evaluated with the help of test subjects and key performance indicators, which allows the assessment of the prototype. The results can be used as starting points for further research and show the possibilities and limits of recent implementations.

1 Einleitung

Das Aufkommen und die andauernde Entwicklung von Softwaretechnologien der letzten Jahre wurde äußerst positiv durch die Verfügbarkeit großer Datenmengen, sogenannter *Big Data*, beeinflusst. Wird repräsentativ für diese Entwicklung der *Gartner Hype Cycle* für das Jahr 2022 (Abbildung 1) betrachtet, so fällt auf, dass eine Vielzahl der dort präsentierten Technologien auf die Sammlung und Weiterverarbeitung von Daten angewiesen sind. Um diese Unmengen an Daten verwertbar und somit zugänglich zu gestalten, muss eine Vorverarbeitung ebenjener stattfinden. Dieser Aspekt des Datenmanagements, die Datenvorverarbeitung, beschreibt ein Set an Techniken und Prozeduren, welche benutzt werden, um Rohdaten in ein nutzbares Format für Analyse und Modellbildung zu bringen. Sie ist enorm wichtig für die Arbeit mit Daten, da sie meist am Anfang der Handhabe ebenjener steht. Deswegen besitzt die Datenvorverarbeitung einen direkten Einfluss auf die Qualität von allem, was danach kommt , seien es nun Analysen, Modellbildungen oder die Speicherung der Daten (Milani et al., 2021, S. 2). Somit kann festgestellt werden, dass *Big Data* und die damit verbundene Prozedur der Datenvorverarbeitung, heutzutage eine zentrale Rolle in vielen Domänen einnehmen, wie auch aus den Arbeiten von Kitchin (2014), Davenport & Patil (2012) und M. Chen et al. (2014) erkennbar ist.

Um jetzt eine optimale Qualität dieser Daten bieten zu können, gilt es, sie und vorrangig die darauf angewandten Vorverarbeitungen zu untersuchen und gegebenenfalls anzupassen. Für die Analyse dieser quantitativen Informationen bietet sich wie in der Arbeit von Tufte (2001) erläutert, eine visuelle Darstellung an. Durch diese Art der Darstellung können die den Daten inneliegenden Informationen wesentlich schneller verarbeitet werden, als in Textform Fahad & Yahya (2018). Aktuell geschieht diese Visualisierung jedoch hauptsächlich nur in dem letzten Schritt der

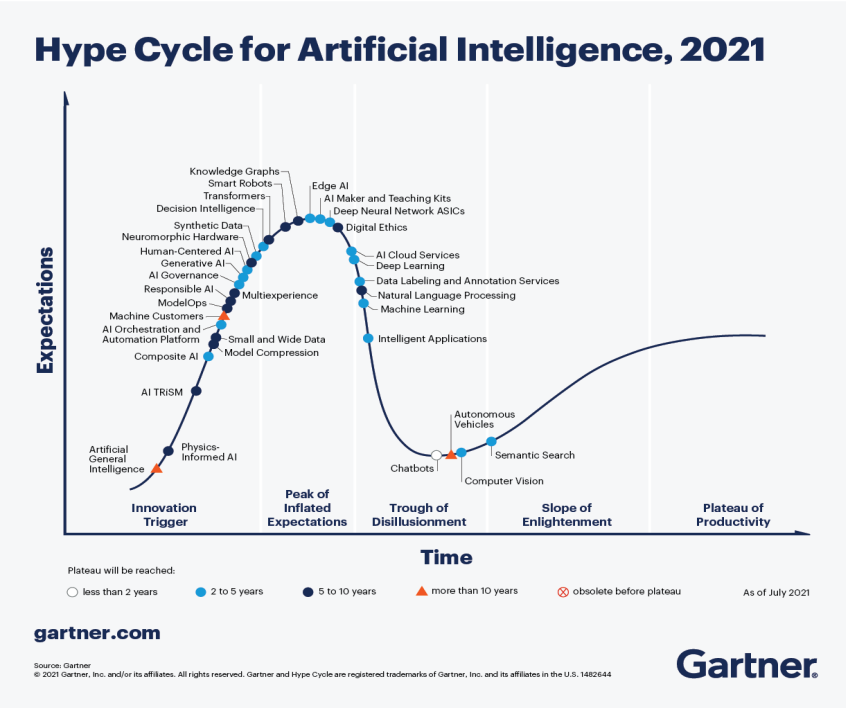


Abbildung 1: Gartner Hype Cycle

Vorverarbeitung, also zur Darstellung der schlussendlich nutzbaren Daten (Milani et al., 2021, S. 2). Dadurch kann zwar überprüft werden, dass die letztendlichen Daten in der angestrebten Qualität vorliegen, jedoch kann bei Fehlern schwerer auf inhärente Probleme der spezifischen Schritte der Vorverarbeitung geschlossen werden. Außerdem kann es gerade für unerfahrenere Datenanalysten schwer sein, die Auswirkungen der einzelnen Vorverarbeitungen anhand des finalen Ergebnisdatensatzes zu ermitteln. Eine Möglichkeit zur detaillierten Visualisierung der einzelnen Vorverarbeitungsschritte gilt also als wünschenswert und eine erste Umsetzung wird mithilfe des in dieser Arbeit vorgestellten Prototypen (PP-Vis) angegangen.
Somit kann als Forschungsfrage, die Machbarkeitsanalyse einer Anwendung, welche den Prozess der Datenvorverarbeitung verständlicher gestaltet, gesehen werden. Diese setzt sich aus der Spezifikation von Anforderungen an dieses Produkt zusammen, sowie der Umsetzung erster Grundkonzepte und Evaluation ebenjenes. Es wird ein genereller Nutzungsansatz verfolgt, sich also nicht auf spezifische Vorverarbeitungsprozesse beschränkt und somit können für verschiedenste Szenarien automatisiert visuelle Informationen geliefert werden. Ebenfalls einen generel-

len Ansatz verfolgt der Prototyp auch in den Ausgabeformen der Visualisierungen. Hier kann zwischen einer Bild-, Web- und Datenform der Visualisierungen gewählt werden. Um die generierten Visualisierungen noch nutzerfreundlicher zu gestalten, legt die hier vorgestellte Anwendung außerdem einen Fokus auf eine interaktive Handhabe der konstruierten Inhalte Yi et al. (2007). Die Integration dieses Prototyps in bestehende Datenvorverarbeitungsabläufe ist so einfach und intuitiv gestaltet, dass dieser nur eine ergänzende Funktion einnimmt und keine Anpassungen der Vorverarbeitung an sich bewirkt.

Die grobe Struktur der Arbeit verfolgt zuerst eine Einordnung gegenüber verwandten Arbeiten, gefolgt von der Definition der spezifischen Anforderungen an den Prototypen. Danach werden die verwendeten Technologien und das Vorgehen bei der Entwicklung erläutert, bevor danach genauer auf die Gestaltung und Funktionsweise der Anwendung eingegangen wird. Abschließend wird ein Nutzungsszenario des Prototyps präsentiert und ebenjener einer Nutzer-, sowie Leistungsbewertung unterzogen.

2 Stand der Technik

Beginnend mit dem aktuellen Stand der Technik, sollten neben Grundlagenarbeiten vorwiegend bestehenden Systeme zur Visualisierung von Datenvorverarbeitung ausgewertet werden, um so eine Einordnung und Abgrenzung des Prototyps zu schaffen. Grundsätzlich lässt sich jedoch vorab feststellen, dass die Visualisierung von Datenvorverarbeitungsprozessen einem sehr starken Wachstum unterliegt, was auch an dem steigenden Interesse für maschinelles Lernen, sowie der *Data Science* an sich liegt. So zeichnet sich in der Übersichtsstudie von Yuan et al. (2021) zu Visualisierungen für maschinelles Lernen im Bereich der Datenqualitäts- und Featureverbesserung ein deutlich steigender Trend ab.

Als Grundlagenarbeit für die Visualisierung der Datenvorverarbeitung kann die Arbeit von Milani et al. (2021) dienen. In dieser wird der durch Keim et al. (2010) und Sacha et al. (2014) definierte visuelle Analyseprozess um die Datenvorverarbeitung ergänzt und dieser ihre Relevanz eingeräumt. Wie anhand der Abbildung 2 zu erkennen ist, nimmt die Datenvorverarbeitung in diesem Modell eine wichtige Stellung in der visuellen Datenanalyse ein. Des Weiteren werden in der Arbeit für diese visuelle Analyse der Datenvorverarbeitung Richtlinien festgelegt, welche in Kapitel 6.3 dieser Arbeit als Metriken der Leistungsbewertung benutzt werden. Auch findet sich in dieser Arbeit eine erste Übersicht der Forschung zur konkreten Umsetzung der Visualisierung von Datenvorverarbeitungsprozessen.

So werden die Studien von Bernard et al. (2019) und Gschwandtner et al. (2014) erwähnt, welche sich mit der visuellen Analyse der Vorverarbeitung von Zeitreihendaten beschäftigen. Die Studie von Bernard et al. (2019) stellt hierbei eine Anwendung vor, welche es erlaubt, Vorverarbeitungsroutinen aus einem Set gegebener Prozesse zusammenzustellen und diese dann visuell zu analysieren. Durch den Fokus auf einen spezifischen Datentyp und ein Set an vordefinierten Vorverarbei-

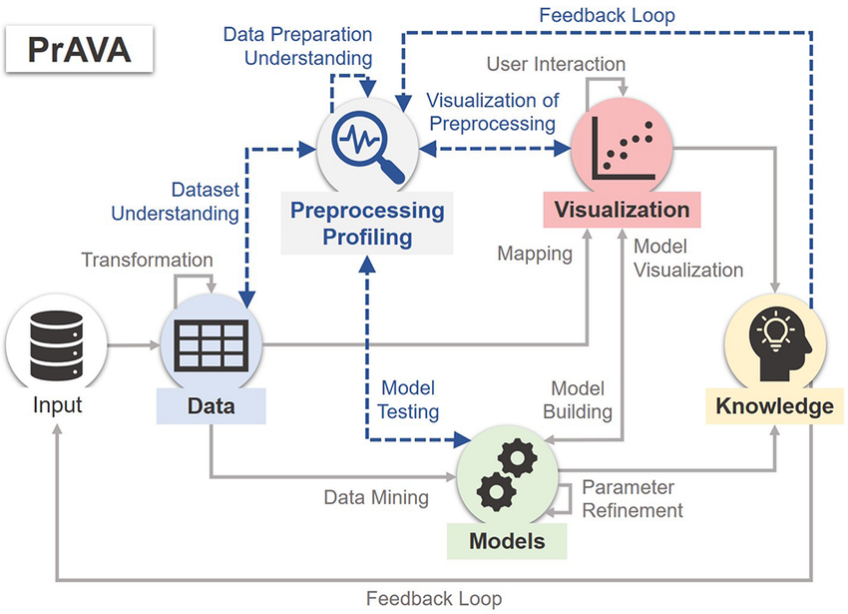


Abbildung 2: Preprocessing Profiling Approach for Visual Analytics

tungsprozessen können in dieser Arbeit sehr angepasste Visualisierungen erzeugt werden. Da der hier vorgestellte Prototyp einen allgemeineren Ansatz verfolgt und unabhängig von dem Typ der Daten, sowie der auf diese angewendeten Vorverarbeitungen agieren soll, ist eine oberflächlichere Art der Darstellung hier das Ziel. Ebenso eine Unabhängigkeit von Daten- und Vorverarbeitungsart wird durch die in der Arbeit von Grafberger et al. (2021) präsentierte Anwendung erreicht. Hier wird mit *ML Inspect* eine Python-Bibliothek vorgestellt, mit deren Hilfe Datenvorverarbeitungsabläufe ohne zusätzlichen Code analysiert und in einem ersten Schritt in Form eines Directed Acyclic Graphs (DAG) der Übersicht dienend dargestellt werden. Danach wird der Code untersucht, um Probleme in der Vorverarbeitung zu finden, welche zu Fehlern in der Verteilung der Daten führen könnten. Kann einer dieser feststellbaren Verteilungsfehler erkannt werden, wird dieser dem Nutzer mitgeteilt und kann auch in Form von Histogrammen visualisiert werden. In der im Folgenden erläuterten Anwendung sollen jedoch keine Informationen zu spezifischen Fehlern in den Daten visualisiert werden. Ganz im Gegenteil sollen möglichst alle visualisierbaren Aspekte der Daten erzeugt und die Identifikation, sowie Interpretation der Probleme dem Nutzer überlassen werden. Somit dient der Prototyp primär als Darstellungswerkzeug und nicht als Assistent bei der Überprüfung

von Datenproblemen.

Eine große Menge an weiteren Arbeiten verfolgt analog ebenjenes Prinzip der visuellen Aufbereitung von Datenqualitätsproblemen. Beispiele hierfür sind die Arbeiten von Kandel et al. (2012) und C. Chen et al. (2020). Obwohl sich diese in ihrer Zielsetzung zwar von PP-Vis unterscheiden, können ihre Visualisierungen jedoch trotzdem, als gute Inspiration für die in diesem Prototyp benötigten Diagramme dienen. Ein Beispiel hierfür wäre die Darstellung von Ausreißern in der Arbeit von C. Chen et al. (2020).

Im Hinblick auf verwandte Arbeiten fällt der Blick auch immer wieder auf Forschungen zur Datenherkunft, der sogenannten *Data Provenance*. Diese beschreibt die Struktur und Entwicklung von Daten, womit Arbeiten zu dieser Thematik hervorragende Informationen für die hier zu entwickelnde Anwendung bieten können. Eine Beispielstudie wäre die Arbeit von P. Chen et al. (2012), welche verschiedene Methoden vorstellt, um die Datenherkunft von *Big Data* zu visualisieren und somit analysierbar zu machen. Weitere Studien wie die Arbeit von Krause et al. (2015) stellen den Umgang mit der Datenvorverarbeitung, in ihrem visuellen Analyseprozess vor. Jedoch liegt hier dieser nicht im Fokus, sondern wie in diesem Fall die Visualisierung des *Clusterings* der Daten.

3 Anforderungsanalyse

3.1 Nutzergruppen

Um die verschiedenen Anforderungen an den Prototypen zu spezifizieren, müssen im Vorfeld Nutzergruppen und deren Bedürfnisse an die Anwendung erörtert werden. Als übergeordnete Zielgruppe können ganz klar Datenanalysten gesehen werden. Jedoch lässt sich diese weiter unterteilen und somit können *Datenanalyse-Einsteiger*, also Personen, welche neu in der Materie sind oder gerade erst mit der Lehre zu der Thematik begonnen haben, als erste mögliche Nutzergruppe ermittelt werden. Diese Nutzer eignen sich Wissen über die Datenanalyse und entsprechend über die Vorverarbeitung gerade erst an, weshalb ein Werkzeug zur visuellen Aufschlüsselung dieser Prozesse einen erheblichen Beitrag zur Förderung des Verständnisses leisten könnte.
Im Rahmen der Lehre könnten auch Dozenten der Datenanalyse als Nutzergruppe identifiziert werden, denn ihr vermitteltes Wissen zur Datenanalyse würde durch den Prototyp angereichert werden. So könnte ohne großes eigenes Zutun, die Erklärung von Vorverarbeitungsschritten durch eine automatisierte Visualisierung gestützt werden.
Abseits von der Lehre können auch im Bereich der Tätigkeit als Datenanalyst zwei Nutzergruppen identifiziert werden. In der Forschung, wie auch in der Industrie könnte eine automatisierte Visualisierung den Vorverarbeitungsprozess von Daten ungemein unterstützen. Es könnte zum einen an einer aufwendigen eigenen Erstellung von Visualisierungen gespart werden, sowie den Nutzern ein vielfältiger Einblick in ihre Vorverarbeitung gestattet werden. Dies kann gerade dahingehend passend sein, dass viele Datenanalysten begrenzte Zeit besitzen und es ihnen so schwerfallen kann, eigene Analysen zu betreiben, wie die Arbeit von Vartak & Mad-

den (2018) nahelegt.

3.2 Anforderungen

Nach der vorangegangenen Definition der Nutzergruppen des Prototyps lassen sich nun benötigte Grundfunktionalitäten desselben bestimmen. Diese sind wenig spezifisch und sollen der groben Abbildung der vorab definierten Ziele für die Entwicklung von PP-Vis dienen, sowie ein besseres Verständnis gegenüber dem entwickelten Funktionsumfang vermitteln. Als Inspiration hierfür wurde das in den Arbeiten von Bors et al. (2019) und Bernard et al. (2019) verwendete Aufgabenmodell genutzt. Hierdurch werden dem Prototyp grundlegende Aufgaben gegenübergestellt, die dieser erfüllen muss und gleichzeitig deren Bedeutung sowie Umsetzung diskutiert. Wird analog für die hier vorgestellte Anwendung verfahren, so können die folgenden vier Anforderungen erhalten werden, welchen sich anhand ihrer Reihenfolge eine gewisse Bedeutsamkeit ableiten lässt. Beginnend mit der wichtigsten Anforderung an den Prototyp und den darauffolgenden mit absteigender Priorität.

Visualisierung

Bei der Visualisierung handelt es sich um die wichtigste Aufgabe von PP-Vis. Sie dient als zentraler Mehrwert der Anwendung, um die Vorverarbeitung von Daten besser verständlich zu machen. Somit zieht sich die Umsetzung dieser Aufgabe durch den kompletten Prototypen, welcher eine Vielzahl verschiedener Visualisierungsmöglichkeiten automatisch, basierend auf den Eingabedaten, bieten soll. Der Fokus hierbei soll klar auf der Visualisierung der durch die Vorverarbeitung bedingten Änderungen dieser Daten liegen. Jedoch können für diese Darstellung von Veränderungen unterschiedliche Methoden an geeigneter Stelle genutzt werden. So kann etwa eine Veränderung durch die Darstellung einer Differenz zweier Werte abgebildet werden, jedoch auch durch die Abbildung beider Werte nebeneinander. Dieses Prinzip lässt sich auch auf den Kontext des Prototyps anwenden, welcher zum einen durch seine Grundstruktur, wie die parallele Darstellung von Informationen mehrerer Datensätze, Aussagen zu Veränderungen zulässt. Zum anderen kann diese Veränderung jedoch auch durch die Darstellung der Wertigkeit von Un-

terschieden in dafür bestimmten Diagrammen abgebildet werden. Diese existieren in vielen verschiedenen Ausführungen und sollten ein wichtiges Visualisierungsmittel der Anwendung sein. Da deren schlussendliche Erstellung oft sehr generisch ist und durch Bibliotheken vereinfacht werden kann, ergibt sich als wichtigste Teilaufgabe des Prototyps die Aufarbeitung und dynamische Überführung der Eingabedaten in die Diagrammform, um Analysen dieser durch den Nutzer zu ermöglichen.

Des Weiteren ist die Auswahl oder Kombination einer oder mehrerer Diagrammarten, wie Balken- oder Kurvendiagramme, ein weiterer wichtiger Teil dieser Aufgabe. Der Prototyp muss die Fähigkeit besitzen aus dieser Sammlung von Diagrammen, geeignete für gegebene Datentypen und der Art ihres Auftretens zu bestimmen. So kann die Werteverteilung eines numerischen Attributs wohl am besten durch eine Art von Kurvendiagramm abgebildet werden. Besitzt es jedoch nur wenige individuelle Werte, eignet sich vielleicht ein Balken- oder Kreisdiagramm besser. Um diese Entscheidungen zu ermöglichen, muss eine Verarbeitung der Daten stattfinden. Wichtig ist es außerdem, dass die Anwendung ein möglichst breites Spektrum unterschiedlicher Arten von Visualisierungen der gleichen Daten und deren Veränderungen bietet. Hierbei zu beachten ist, dass es sich dabei nicht um sich wiederholende gleiche Darstellungen handeln soll, sondern dem Nutzer unterschiedliche Blickwinkel auf die Datenvorverarbeitungen ermöglicht werden sollen. So kann eine tabellarische Darstellung von Attributen Informationen zu deren Metriken wie Datentyp und Durchschnittswerten liefern, während die Darstellung der Attribute in Diagrammform Aussagen über deren Verteilung zulässt.

Wie in Kapitel 2 erläutert, soll die grundlegende Verfahrensweise des Visualisierungsprozesses der Anwendung unabhängig von der jeweiligen Art des Datenvorverarbeitungsschrittes sein. Es sollte jedoch trotzdem die Möglichkeit bestehen, spezifische Arten von Veränderungen der Datensätze zu erkennen und dadurch ausgelöst spezifische Darstellungen zu generieren, welche ergänzend zu den anderen angezeigt werden. Abschließend sollte der Prototyp auch noch die elementare Entscheidung treffen, ob an einer Stelle eine Visualisierung überhaupt benötigt wird,

beziehungsweise eine Bereicherung für die Nutzeranalyse der Daten darstellt.

Interaktivität

Als direkte Ergänzung und Unterstützung der Visualisierung kann die Aufgabe von PP-Vis gesehen werden, eine gewisse Interaktivität zu bieten, deren Bedeutung in der Studie von Lam et al. (2011) hervorgehoben wird. Diese interaktive Nutzungsmöglichkeit sollte primär bei den im Prototyp verwendeten Diagrammen zum Erkunden und Analysieren von Datenvorverarbeitungsprozessen bestehen. Beispiele hierfür wären das Navigieren, Vergrößern und Filtern von Daten in den verwendeten Diagrammen, um unterschiedliche Blickwinkel auf die generierten Informationen zu erlangen. Jedoch sollten sie auch in allen möglichen der Verwendung förderlichen Bereichen der Anwendung vertreten sein, um möglichst in ganzem Maße den durch Yi et al. (2007) beschriebenen positiven Effekt von Interaktivität auf die Visualisierung von Informationen zu erzielen.
Wichtig ist es des Weiteren, das Bedien- und Steuerungselemente der Anwendung möglichst einfach gestaltet sind, um den Nutzer zum einen nicht zu überfordern und eine intuitive Benutzung zu gewährleisten. Der Prototyp sollte außerdem ein Gleichgewicht zwischen Nutzbarkeit und Funktionalitäten schaffen (Shneiderman et al. (2016)), sowie nur Funktionalität mit einem ersichtlichen Vorteil anbieten. Somit bietet die Vergrößerung bestimmter Teile eines Kurvendiagramms einen klaren Vorteil bei der Analyse, die Änderung der Farbe dieses Diagramms jedoch eher weniger. Über die Diagramme hinaus sollte auch eine gewisse Interaktivität gewährleistet werden, um an geeigneten Stellen die Nutzung des Prototyps zu verbessern. Wie die interaktive Auswahl von Vorverarbeitungsschritten, um deren spezifische Informationen anzuzeigen.

Generalität

Damit der Prototyp letztlich erfolgreich genutzt werden kann, sollte er zudem eine gewisse Generalität für Eingabe- und Ausgabedaten, sowie Vorverarbeitungsprozesse bieten. Die Eingabe betreffend, sollten möglichst viele Datentypen visualisierbar sein und infolgedessen auch deren Entwicklung durch die Vorverarbeitung hindurch. Der Prototyp sollte also jeden Datensatz, unabhängig von den verschiede-

nen vorkommenden Datentypen der Attribute, als Eingabe akzeptieren können und daraus eine Ausgabe generieren können. Nicht unterstützte Datentypen oder Vorverarbeitungsschritte sollten gegebenenfalls ignoriert werden, jedoch nie zu Fehlern im Programm führen.
Im Hinblick auf die Ausgabe sollte die Anwendung auch verschiedene Nutzungsszenarien abdecken werden. Diese sollten sich zum einen hinsichtlich ihrer Komplexität, wie auch ihrem Format unterscheiden und so möglichst jeder Nutzergruppe Vorteile bieten. Vor allem im Hinblick auf die Fülle möglicher Vorverarbeitungsschritte würde der Versuch, alle mit spezifischen Darstellungen abzudecken, den Rahmen der Arbeit sprengen. Jedoch sollten die von der generellen Vorverarbeitung unabhängigen Informationen immer generiert werden können. Somit könnte die Nutzung der Anwendung von dem jeweiligen Forschungs- oder Arbeitsfeld unabhängig für Datenanalysten möglich sein. Zusammenhängend hiermit sollte die Verfahrensweise der Anwendung auch unabhängig von dem benutzten Vorverarbeitungsprogramm oder der verwendeten Bibliothek sein. Lediglich die zu übergebenden Ergebnisse der Vorverarbeitungen sollten in einer bestimmten Form vorausgesetzt werden.

Zugänglichkeit

Betreffend des Nutzungskontextes sollten nicht zu große Einschränkungen vorhanden sein und der Prototyp in der gewählten Entwicklungsumgebung in allen dort möglichen Szenarien verwendet werden können. Damit verbunden ist ein einfacher Zugriff auf den Prototypen von der gewählten Umgebung aus, sowie eine gewisse Vielfältigkeit in der Art und Weise, wie auf seine Funktionen zugegriffen wird. Einmal importiert, sollten sowohl die Befehle zum Ansprechen der Anwendung, sowie die Navigation und Bedienung leicht verständlich nutzbar sein. Im Zusammenhang mit diesen Befehlen sollte ebenfalls eine gewisse Personalisierbarkeit über die Eingabe hinweg geboten werden. Ein Beispiel hierfür wären selbst definierbare Speicherpfade. Auch im Zusammenhang mit dieser Anforderung steht eine gemäßigte Laufzeit der Anwendung. Sie sollte in keinem Fall eine Verzögerung des eigentlichen Vorverarbeitungsgeschehens verschulden, um so eine optimale Nutzung zu

garantieren. Abschließend sollten durch die Nutzung des Prototyps keine großen Änderungen an der Vorverarbeitung an sich nötig sein und auch die Verwendung an sich sollte möglichst einfach und intuitiv sein.

4 Technische Rahmenbedingungen

Anhand der Anforderungen des Prototyps lässt sich nun eine Wahl bezüglich verwendeter Entwicklungswerkzeuge, sowie Bibliotheken treffen. So wird als Programmiersprache Python verwendet, da es sich hierbei um eine der meistgenutzten Entwicklungssprachen für die Arbeit mit *Big Data*, wie auch im Bereich der Datenanalyse und Datenvorverarbeitung handelt (Siddiqui et al. (2017)). Somit kann der Prototyp einfach von den Nutzergruppen in ihre Programme und Arbeitsabläufe integriert werden. Ferner fungiert Python nicht nur als Programmiersprache, sondern auch als Laufzeitumgebung für die Ausführung der Skripte des Prototyps. Somit wird ein aktueller Python-Interpreter für die Ausführung des Pythoncodes von PP-Vis vorausgesetzt. Mit dem Hinblick auf die identifizierten Nutzergruppen sollte diese Installation jedoch schon vorhanden sein.

4.1 Pandas

Durch die Verwendung von Python als Programmiersprache liegt es nahe für die Zwischenspeicherung und Verarbeitung der Datensätze, die Datenverarbeitungsbibliothek *Pandas* zu nutzen. Diese stellt Python exklusiv einen großen Umfang an Methoden zur Handhabe und Analyse von Datensätzen bereit (McKinney et al. (2010)). Somit dient sie in dieser Arbeit als zentrales Mittel, um die Datensätze zu strukturieren und zu analysieren.
Ein wichtiger Teil dieser Bibliothek sind die sogenannten *Pandas Dataframes*, bei welchen es sich um zweidimensionale Datenstrukturen mit der Fähigkeit, heterogene Daten zu speichern, handelt. Im Prototyp werden durch Operationen auf diesen *Dataframes* und der Nutzung bestehender Verknüpfungen zwischen *Pandas* und Visualisierungsbibliotheken, zentrale Funktionalitäten desselben ermöglicht. Um eine Eingrenzung der Komplexität der Eingabedaten der Anwendung zu schaffen, kann

diese ausschließlich Datensätze in Form von *Pandas Dataframes* verarbeiten. Dies führt zum einen zu einer Vereinfachung der Eingabeschicht des Prototyps, da keine komplizierten Dateiformatidentifizierungen und -transformationen benötigt werden. Außerdem kann somit auch die Eingabe von Eingabedatensätzen in PP-Vis optimiert werden. Es muss zwar zuerst eine Umwandlung in einen *Dataframe* stattfinden, jedoch deckt *Pandas* viele mögliche Datenrahmenerstellung für verschiedenste Datensatzformate ab und bietet dem Nutzer somit gestützt durch eine Vielzahl an Dokumentationen die Möglichkeit, einfach seine Daten für den Prototyp vorzubereiten.

4.2 Tidy Datasets

Um neben den *Pandas Dataframes* einen weiteren Rahmen für die zu analysierenden Daten zu schaffen, wird in dieser Arbeit die Annahme getroffen, dass ausschließlich *Tidy Datasets* als Input für den Prototypen vorliegen (Wickham & Wickham (2016)). Diese Datensätze sind nach dem *Tidy-Data-Prinzip* strukturiert und unterliegen somit gewissen Formatierungsregeln, was ihnen eine grundlegende Konsistenz gewährt. Auf einen als *Tidy Data* betitelten Datensatz müssen somit folgende Regeln zutreffen:

- jede Variable bildet eine Spalte
- jede Beobachtung bildet eine Reihe
- jeder Beobachtungstyp bildet eine Tabelle

Dabei sind diese Datensätze formatunabhängig und können in sämtlichen gängigen Formen vorliegen. Beispiele hierfür wären Comma-separated values (CSV), Extensible Markup Language (XML) oder Java Script Object Notation (JSON). Lediglich die Formatierungsregeln müssen eingehalten werden, was eine gute Vorhersagbarkeit der Datenstruktur bietet, welche durch den Prototyp verarbeitet werden soll. Außerdem können komplizierte Input-Grenzfälle vermieden werden.

4.3 Webtechnologien

Für einen großen Teil der Ausgabe des Prototyps werden die Webtechnologien Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) und Javascript (JS) verwendet. Mit ihrer Hilfe werden, wie weiter unten ausführlicher beschrieben, dem Nutzer die durch den Prototyp erstellten Inhalte in Form von Webdokumenten ausgegeben. Klare Vorteile dieser Technologien sind zum einen die universelle Zugänglichkeit, das bedeutet, dass sie meist ohne eine zusätzliche Installation von Programmen im nativen Webbrowser des jeweiligen Geräts geöffnet und angezeigt werden können. Darüber hinaus gibt es zu diesen Webtechnologien eine große Menge an Anleitungen und Dokumentationen, was die Erstellung dieser Webdokumente ungemein vereinfacht.
Des Weiteren wird durch JS eine einfache Möglichkeit geboten, die Anwendung interaktiv zu gestalten, was vorwiegend für die zweite Anforderung an den Prototypen von Vorteil ist. Auch gibt es durch die langjährige Entwicklung viele benutzerfreundliche Optionen, was die Implementierung der Webelemente betrifft. Beispiele hierfür wären die Wiederverwendung von Gestaltungsklassen, um doppelten Code zu verhindern und die automatische Zuteilung von Attributen für bestimmte Elemente wie Texte und Überschriften, was deren semantische Korrektheit deutlich verbessert. Eine weitere wichtige Eigenschaft, welche primär durch CSS erzielt wird, ist die Adaptivität bezüglich verschiedener Bildschirmgrößen, womit die Anwendung flexibler genutzt werden kann.
Als letzten großen nennenswerten Vorteil der Webtechnologien kann die einfache Skalierbarkeit von solchen Webdokumenten gesehen werden, indem einfach weitere Elemente ergänzt und bestehende Gestaltungsklassen wiederverwendet werden. Jedoch gilt es zu beachten, dass es sich bei den Webtechnologien um ein komplett neues Entwicklungsframework handelt, welches deutliche Unterschiede zu Python besitzt und somit für Erweiterungen einen gewissen Einarbeitungs- und Entwicklungszeitraum benötigen könnte.

4.4 Visualisierungsbibliothek

Essenziell ist die Wahl der Visualisierungsbibliothek, welche eine zentrale Rolle in der Anwendung einnimmt, da sie die wichtige Anforderung der Visualisierung an den Prototypen zu einem großen Teil mit erfüllt. Mit der Entscheidung der Python-basierten Implementierung sollte die zu wählende Visualisierungsbibliothek für Python konzipiert sein, um so einfach mit dem restlichen Teil des Prototyps verwendet werden zu können. Hierfür existiert eine Vielzahl an Bibliotheken, welche sich in ihrer Komplexität und Vielfalt an Funktionen unterscheiden. Die Hauptfunktionalität ist jedoch bei allen das Abbilden von Daten durch Diagramme.
Einen guten Überblick über die bekanntesten bietet das Buch von Embarak et al. (2018) und die wissenschaftlichen Arbeiten von Stančin & Jović (2019), sowie Dhruv et al. (2021). Ermitteln lassen sich Matplotlib, Seaborn und GGPlot als bekannteste Vertreter für die Erstellung nicht interaktiver Diagramme, sowie Altair, Bokeh, Pygal und Ploty für die Erstellung interaktiver Diagramme. Es gilt zu beachten, dass hierbei von einer unmodifizierten Interaktivität gesprochen wird, denn Beispielsweise könnten nicht interaktive Bibliotheken mithilfe einer grafischen Nutzeroberfläche, wie in der Arbeit von Podrzaj (2019) angeschnitten, um eine gewisse Interaktivität erweitert werden. Für das hier vorhandene Nutzungsszenario, wird die Funktionalität, interaktive Diagramme zu schaffen, im Hinblick auf die Anforderung nach Interaktivität des Prototyps, in jedem Fall benötigt. Hierfür ist es Prinzipiell irrelevant, ob diese schon vorhanden ist oder künstlich ergänzt wird. Jedoch boten nach praktischen Tests Bibliotheken mit schon bestehender Interaktivität eine bessere und reibungslose Steuerung, sowie kaum spürbare Verzögerungen. Außerdem verfügten die Bibliotheken mit integrierter Interaktivität oft über zusätzliche Steuerungselemente wie Interface-Komponenten zur Vergrößerung oder freien Navigation in der Diagrammansicht. Da diese zusätzlichen Funktionen sehr wünschenswert, jedoch aufwendig selbst anhand fremd erstellter Diagramme implementierbar sind, sollten diese Visualisierungsbibliothek bevorzugt werden.
Bei der Betrachtung der Bibliotheken mit integrierter Interaktivität bietet Altair die einfachste Handhabe, denn viele Funktionen, wie das Bauen von Legenden, sind au-

tomatisiert und der deklarative Charakter der Graphenerstellung erlaubt Komplexität mit minimalem Code (VanderPlas et al. (2018)). Jedoch bietet Altair aufgrund seiner einfachen Handhabe auch weniger Anpassungsmöglichkeiten und genauso wie die interaktive Visualisierungsbibliothek Pygal, ein relativ kleines Nutzerökosystem. Dies führt bei beiden Bibliotheken zu weniger Dokumentationen, einer kleineren Community und somit einer geringeren Unterstützung bei der Entwicklung mit diesen Bibliotheken. Des Weiteren besitzt Pygal auch die Möglichkeit, interaktive Darstellungen zu generieren, fokussiert sich jedoch vor allem auf das erstellen von Scalable Vector Graphics (SVG), welche statisch in HTML-Dokumente eingepflegt werden und somit über keine weiteren Steuerungselemente verfügen. Über SVG's hinaus sollten jedoch trotzdem, da für einen großen Teil der Ausgabe des Prototyps Webtechnologien gewählt wurden, die Visualisierungsbibliotheken ihre Diagramme in dieser Codeform ausgeben können, damit sie nahtlos in das Grundgerüst des Prototyps eingebunden werden können. Einen Fokus auf die Erstellung von Diagrammen für die Präsentation in Webbrowsern bieten vorrangig Bokeh (Jolly (2018)) und Plotly, wobei die Entwickler von Plotly mit Plotly-Dash zusätzlich noch ein Framework zur Erstellung interaktiver Webanwendungen bieten(Dabbas (2021)). In diesem Fall beständen dann jedoch begrenzte Möglichkeiten zur Erweiterung oder der Integration externer Funktionalitäten, da es sich hier um eine geschlossene Anwendungsumgebung handelt.

Als weiteren wichtigen Entscheidungspunkt, die Kompatibilität betreffend, kann bei dem Vergleich dieser Bibliotheken die oben erläuterte Pandas Datenverarbeitungsbibliothek gesehen werden. Als Grundträger der Daten in der Anwendung ist eine einfache Nutzung der Pandas Daten durch die Visualisierungsbibliothek von großem Vorteil, um komplizierte Datentransformationen zu verhindern. So kann Laufzeit gespart und mögliche Transformationsfehler vermieden werden. Diesen Punkt erfüllen vorwiegend Plotly, Bokeh und Altair, welche eine nahtlose Verwendung von Dataframes zur Diagrammerstellung ermöglichen. Anhand der Abbildung 1 lässt sich erkennen, dass Altair, Bokeh und Plotly alle benötigten Anforderungen erfüllen, jedoch die letzten zwei ein größeres Ökosystem besitzen. Die

	Matplotlib	Seaborn	GGPlot	Altair	Bokeh	Pygal	Plotly
Python	✓	✓	✓	✓	✓	✓	✓
Interaktiv	✗	✗	✗	✓	✓	✓	✓
Steuerungsel.	✗	✗	✗	✓	✓	✗	✓
Html-Ausgabe	✗	✗	✗	✓	✓	✓	✓
Pandas Eingabe	✓	✓	▬	✓	✓	▬	✓
Github-Stars	18.100	11.200	3.700	8.500	18.000	2.600	14.200
Geschl. Issues	8.604	2.280	316	1.689	6.668	245	1.351

Tabelle 1: Vergleich von Visualisierungsbibliotheken

Entscheidung zwischen Bokeh und Plotly findet nunmehr nur noch aufgrund von persönlichen Präferenzen wie der allgemeinen Ästhetik der Bibliotheken statt, weswegen sich bei PP-Vis schlussendlich für Plotly entschieden wurde.

4.5 Bibliotheksintegration

Damit der Nutzer die Funktionalitäten des Prototyps nutzen kann, muss dieser in dessen Programm eingebunden werden. Die einfachste Methode wäre es hier, den Prototyp als Download für den Nutzer anzubieten. Dessen Ort müsste dieser dann zur Nutzung im Code referenzieren, um Zugriff auf die Funktionen zu erhalten. Jedoch kann dies bei manchen Nutzungsmethoden zu Problemen und Umständen kommen. So muss zum Beispiel bei der Nutzung in online *Jupyther-Notebooks* wie *Google Collab* erst ein Upload der Prototypdateien stattfinden. Ähnlich verhält es sich, wenn der Prototyp um eine installierbare Setup-Datei ergänzt wird, die die Installation in den bestehenden Python-Interpreter ermöglicht.
Eine einfachere Möglichkeit, ohne komplexe Installationsprozesse, bietet der in alle Pythonumgebungen ab Version 3.4 integrierte Paketmanager Pip Installs Packages (PiP). Mit ihm wird eine einfache Installation und Aktualisierung von Python-Paketen ermöglicht (*Installing Packages - Python Packaging User Guide 2023* (2023)). Er bietet außerdem ein frei zugängliches Python-Paketarchiv, über welches einfach auf jegliche dort bereitgestellte Pakete zugegriffen werden kann und dies auch von Online-Pythonumgebungen aus. Somit bietet sich diese Methode perfekt an, um

als Bibliotheksintegrationswerkzeug zu dienen und den Nutzern Zugang zu dem Prototyp zu ermöglichen.

5 Systembeschreibung

Da es sich bei der Entwicklung des Prototyps um die zentrale Aufgabe dieser Arbeit handelt, wird hier nun auf seinen programmatischen Aufbau, die dafür getroffenen Designentscheidungen und die genaue Struktur und Interaktion der verschiedenen Komponenten ebenjenes eingegangen. Seine Entwicklung fand in *Visual Studio Code* statt und als Interpreter wurde die Python Version 3.11.4 verwendet. Alle verwendeten SVG's wurden mit dem Programm *Adobe Illustrater* erstellt und durch ihre SVG-Zeichenkette in die Anwendung eingebunden. Getestet wurden die Funktionalitäten mithilfe von lokalen *Jupyter Notebooks*.

5.1 Designentscheidungen

Um ein konkretes Vorgehen, für die zu implementierenden Visualisierungen festzulegen, gilt es im ersten Schritt Entscheidungen zur Wahl von Darstellungsmöglichkeiten zu treffen. Im engeren Sinne betrifft dies hauptsächlich die Art der verwendeten Diagramme und weiterer visueller Aspekte, die einen unmittelbaren Einfluss auf das Benutzererlebnis besitzen. Die verschiedenen Diagrammarten, wie anhand der Abbildung 3 zu erkennen, eignen sich optimal zur verständlichen und interpretierbaren Darstellung von großen Datenmengen auf kleinem Raum und sind damit ideal für die Verwendung im Prototypen geeignet.

5.1.1 Diagramme

Da Plotly alle gängigen Diagrammarten anbietet, gilt es, diese hier nun für den gegebenen Einsatzzweck zu evaluieren und optimal auf die gegebenen Datentypen abzustimmen. Der Einsatzzweck setzt sich aus zwei Hauptaufgaben zusammen, zum einen die Visualisierung des Ist-Zustands der Datensätze an sich, zum anderen die Visualisierung der Veränderungen zwischen den Datensätzen. Hierbei, wie

oben angedeutet, ist zu beachten, dass an gewissen Stellen des Prototyps diese Visualisierung von Veränderung auch dadurch erreicht wird, dass zwei Diagramme verschiedener Datensätze nebeneinander dargestellt werden.
Da vorausgehend zwar Format (*Pandas Dataframes*) und Struktur (*Tidy Dataset*) der zu visualisierenden Datensätze festgelegt wurde, jedoch der Typ der eingehenden Datenpunkte offen bleibt, sollte mit allen fundamentalen Python Datentypen gerechnet und für diese entsprechende Diagrammarten bereitgestellt werden. Ausgeschlossen werden können Datentypen, welche die erste Regel der *Tidy Datasets* verletzten und mehrere Variablen für einen Datenpunkt zulassen würden. Des Weiteren wurden Datentypen, welche eine zu komplexe Darstellungsweise erfordern würden auch ausgeschlossen und werden bei der Verarbeitung durch den Prototyp ignoriert (Tabelle 2).

Im Vorfeld der Entscheidungsfindung können nun die Datentypen genauer be-

Datentyp	**Verfahrensweise**
Integer, Float, String, Boolean	Verarbeitbar
List, Tuple, Set, Dictionary, Range	Verletzung der ersten Tidy Dataset Regel
Complex, Bytes	zu komplexe Darstellung benötigt

Tabelle 2: Datentyp Verfahrensweise

trachtet werden, um so die Komplexität der Aufgabe zu verringern. So können etwa Integer und Floats beide als Repräsentationen numerischer Werte gesehen werden, welche sich lediglich in ihrer Präzision unterscheiden, weswegen sie als eine zu visualisierende Kategorie zusammengefasst werden können. Wenn jedoch numerische Werte in einer sehr diskreten Form vorliegen, legen andere Arbeiten nahe, diese mit den Methoden für kategoriale Attribute darzustellen (Szoka (1982)). Mit ebenjenen Methoden zur Visualisierung wird auch ein Teil der String Attribute dargestellt, genauer der Teil, der eine kategoriale Eigenschaft besitzt. Der andere Teil, welcher diese nicht besitzt, muss in einer angepassten Weise dargestellt werden, da hier sonst jeder einzelne Wert einer neuen Kategorie entsprechen würde. Zum Schluss können auch die Boolean-Attribute zu der Visualisierungskategorie

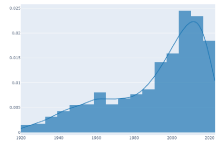
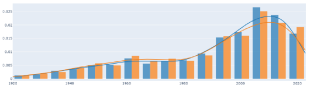
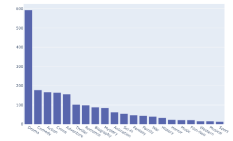
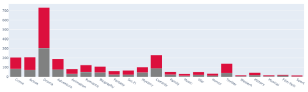
	Zugeordnete Attribute	Generelle Diagramme	Veränderungs Diagramme
Numerische Kategorie	• kontinuierliche Float und String Attribute	 Histogramm + Kurvendiagramm	 Histogramm + Kurvendiagramm (Vergleichsansicht)
Kategorielle Kategorie	• diskrete Float und String Attribute • kategorielle String • Attribute boolean Attribute	 Balkendiagramm	 Balkendiagramm (Gestapelt)
String Kategorie	• nicht-kategorielle String Attribute		 Balkendiagramm (Gestapelt & 1-Balken)

Abbildung 3: Diagramm Wahl

der kategorialen Werte gezählt werden, da sie in jedem Fall zwei Kategorien, *True und False*, beinhalten.

Für die Darstellung der numerischen stetigen Attribute schlägt die Literatur, wie zum Beispiel die Arbeit von Hardin et al. (2012) Histogramme vor. Jedoch wird in anderen Fällen auch von Kurvendiagrammen als passende Darstellungsform gesprochen. Ein Beispiel hierfür ist die Arbeit von Fahad & Yahya (2018). Da Plotly durch sein *Figure Factory Module* ermöglicht, eine Kombination dieser beiden Diagramme zu produzieren, bietet sich dieses Vorgehen für diesen Fall optimal an und wird beispielhaft in der Abbildung 2 dargestellt. Sollen nun Veränderungen dieser numerischen Werte abgebildet werden, wird oft auf zwei Linien in einem Kurvendiagramm verwiesen (Szoka (1982)). Um jedoch eine gewisse Persistenz der Diagramme zu gewährleisten, werden die oben erwähnten Histogramme in einer parallelen Ansicht ergänzend beibehalten. Als beste Visualisierungsmethode kategorialer Werte werden, wie in der Arbeit von Szoka (1982), meistens Balken oder Kuchendiagramme empfohlen. Jedoch weist einige Literatur daraufhin, dass ein Vergleich von Daten durch Balkendiagramme deutlich leichter ist und somit die-

se hierfür bevorzugt werden sollten (Hardin et al. (2012)). Da auch hier wieder der Veränderung eine Darstellungsweise geschuldet wird, kann sich, wie anhand der Arbeit von Szoka (1982) angedeutet, für eine gestapelte Darstellung der Balkendiagramme entschieden werden. Die Kategorie der String-Werte ist die schwerste, um sie in Diagrammen zu visualisieren, da meist nur die Anzahl der Werte des Attributs dargestellt werden kann und infolgedessen deren Mengenveränderungen. Somit findet eine Darstellung ohne vorliegende Veränderung in diesem Fall nicht statt, liegt eine vor, wird hier analog zu den kategorialen Attributen verfahren, um wie oben erwähnt eine Konsistenz zu gewährleisten. Ein Überblick der vorausgehenden Einordnung und Diagrammwahl lässt sich der in Abbildung 2 dargestellten Tabelle entnehmen.

5.1.2 Vorverarbeitungsworkflow

Hauptziel der Anwendung ist die Aufschlüsselung eines Vorverarbeitungsprozesses, weshalb ebendieser auch verständlich und als Übersicht dienend durch den Prototyp abgebildet werden sollte. Beim Vergleich der Herangehensweise verwandter Arbeiten kann erkannt werden, das dieser oft als Graph abgebildet wird (Grafberger et al. (2021)). Die Arbeit von Simmhan et al. (2005) verweist sogar explizit darauf, dass sich Datenabstammung optimal mithilfe eines Graphen abbilden lässt und die Arbeit von Bors et al. (2019) spricht von einer Darstellungsmethodik, wo Datensätze als Knoten und Transformationen derselben als Kanten abgebildet werden. Somit liegt es Nahe, diese Methodik auch hier anzuwenden und den Vorverarbeitungsworkflow ganz im Sinne der Graphtheorie, als gerichteten Graph mit Knoten und Kanten abzubilden.

Die dafür optimale verwendete Art von Graph laut der oben genannten Literatur ist der Directed Acyclic Graph (DAG). In dieser Darstellungsform wird jeder Vorverarbeitungsschritt als Knoten dargestellt und die Kanten, bei welchen es sich in diesem Fall um Pfeile handelt, verdeutlichen die Reihenfolge der Ausführung. Da der Prototyp durch seinen später noch genauer erläuterten Input und Art der Darstellung, jedoch auch einen nicht zu vernachlässigenden Fokus auf die entstandenen

Abbildung 4: Vorverarbeitungsworkflow

Datensätze zwischen den verschiedenen Vorverarbeitungsschritten legt, sollten diese auch ihren Platz in der Darstellung finden. So wurde sich hier für ein Ablaufdiagramm, mit Orientierung an Aspekten des DAG's entschieden (Abbildung 4). Die Rechtecke stellen die Datensätze dar und die Pfeile, die zwischen den Datensätzen stattgefundenen Veränderungen. Es handelt sich um eine lineare Darstellung und das abwechselnde Auftreten von Datensatz und Veränderung wird vorausgesetzt. Als Farbe für den Datensatz wurde blau gewählt und für die Veränderung gelb. Diese Farbgebung wird konstant auch an anderen Stellen des Prototyps verwendet, wenn spezifische Informationen zu den Datensätzen oder deren Veränderung abgebildet werden.

5.2 Architektur und Funktionsweise

Die oben dargestellte Abbildung 5 zeigt bildlich die grobe Struktur des hier vorgestellten Prototypen. Klar zu erkennen ist, dass eine visuelle Unterteilung in drei größere Komponenten möglich ist, welche ihren Platz in dem schlussendlich installierbaren Pip-Packet finden. Die drei großen Komponenten Interaktion, Content Generierung und Ausgabe stehen miteinander in Verbindung und durch ihre Verknüpfung lässt sich schon hier ein Ablauf der Arbeitsweise der Anwendung ermitteln.

Grob zusammengefasst nimmt die Interaktionsschicht alle Nutzereingaben an, stößt mit diesen Prozesse in der Kontengenerierung an, welche dann ihre erzeugten Daten durch die Ausgabeschicht dem Nutzer zurückgibt. Da es sich bei Interaktion, Contentgenerierung und dem Pip-Packet um die drei größten Akteure des Prototypen handelt, werden diese in den weiteren Kapiteln genauer behandelt. Die Ausgabe Komponente, welche in ihrer Funktion vor allem die Anforderung nach einer Generalität der Ausgabe von PP-Vis erfüllt, ermöglicht die Ausgabe von generierten Inhalten als Website, als Bild und als JSON-Datei. Sie beinhaltet eine Sammlung von

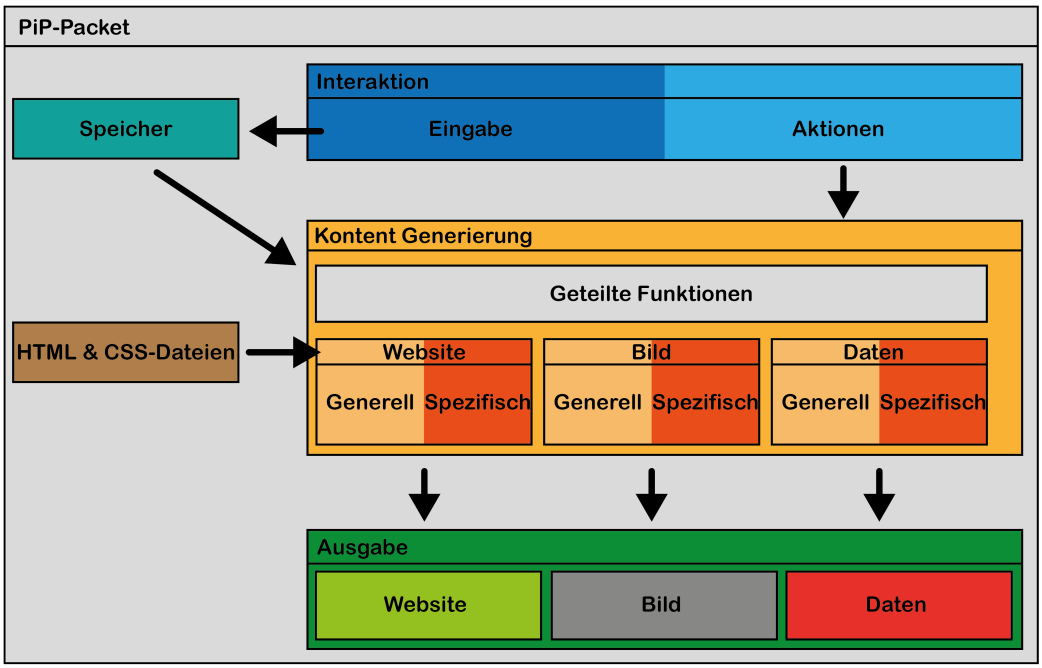


Abbildung 5: Prototyp Komponenten

Funktionen, welche die Überführung der durch die Contentgenerierung erzeugten Inhalte in Web-, Bild- und Datenformate ermöglichen. Außerdem lassen sich für Speicherort und Name der Dateien durch die Interaktionsschicht eigene Werte festlegen.

Ebenfalls anhand der Abbildung zu erkennen, ist der Bereich der geteilten Funktionen. Wird ein Prozess der Anwendung, in mehreren Contentgenerierungsarten benötigt, so teilen sich die Komponenten die zugehörige Funktion und lagern diese in jenen Bereich aus. Das und eine generelle Unterteilung von Komponenten in mehrere spezialisierte Python-Dokumente, bedeutet das sich eine Komponente auch über mehrere Dateien erstrecken kann und somit nicht als geschlossene Einheit fungiert. Bei geteilten Funktionen kann es sich zum Beispiel um, an mehreren Stellen verwendete Daten- oder Diagrammerstellungsmethoden handeln. Abschließend lässt sich noch der die Website Generierung stützende Bereich der HTML und CSS-Dateien erkennen. Durch ihn werden vordefinierte Variablen, bestehend aus statischen Teilen der HTML-Elemente, sowie einem Großteil des CSS-Codes bereitgestellt, um die Pythondateien übersichtlicher zu halten.

5.2.1 Pip-Packet

Wie anhand der Abbildung 5 zu erkennen, dient der PiP-Paket-Teil als übergeordnete Komponente der Anwendung, mit dessen Hilfe Funktionalität nach außen angeboten wird und auch eine Installation des Prototyps erfolgen kann. Er gibt außerdem einen Speicherort und, wie anhand Abbildung 6 zu erkennen, eine Struktur für die Dateien der Anwendung vor. An diesem befinden sich auch für PiP-Pakete gängig eine Lizenz, eine *Readme* mit Informationen zur Nutzung und eine *.toml-Datei* mit Metadaten des Pakets befinden. Wichtig hier zu erwähnen ist, dass für diesen ersten Prototyp noch keine Veröffentlichung im oben beschriebenen Python Paketarchiv geplant ist, weswegen die Dokumentationsdateien des PiP-Pakets noch von geringerer Bedeutung sind. Sobald der Prototyp jedoch veröffentlicht wurde, lässt er sich in jeder Python Umgebung durch folgenden Befehl installieren **pip install pp_vis**.

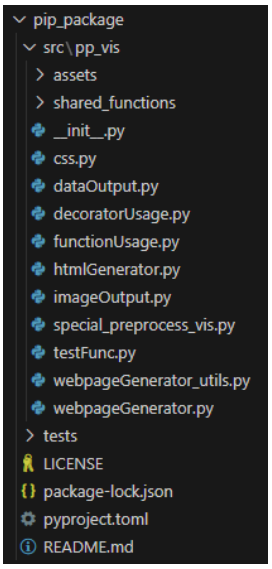


Abbildung 6: Pip Ordner

Ein großer Vorteil für die Nutzung dieses Prototyps als PiP-Packet ist, dass durch die Festlegung von Bibliotheks-Abhängigkeiten der Anwendung in der *.toml-Datei*, diese automatisch mit installiert werden, wenn sie sich bislang nicht in der Pythonumgebung auffinden lassen. Somit müsste ein Nutzer nicht im Vorfeld oben genannte und im Prototyp verwendete Bibliotheken wie Pandas und Plotly separat installieren. Auch auf dieses Feature wurde aufgrund der frühen Version der Anwendung verzichtet. Einmal installiert, lassen sich die öffentlichen Methoden von PP-Vis nun in der Python Umgebung verwenden. Diese Anwendungsschnittstellen werden durch die *init.py* des Pakets definiert und bereitgestellt und können durch einen Import der Methoden gefolgt von einem Aufruf ebenjener ausgeführt werden, wie anhand des Codebeispiels 1 zu erkennen ist.

5.2.2 Interaktion

Diese Komponente der Anwendung beinhaltet alle Funktionen, die durch das Pip-Paket als Anwendungsschnittstellen definiert werden, und stößt durch den Aufruf dieser, tiefer liegende Methoden des Prototyps an und dient somit als Vermittler zwischen Nutzer und Funktionalitäten des Programms. Unterteilen lässt sich der Teil des Prototyps in die Bereiche Aktionen und Eingabe.
Der Eingabebereich stellt Methoden bereit, durch welche der Prototyp zum einen die Datensätze an sich, wie auch Informationen zu deren Namen vom Nutzer erhält. Der Datensatz-Parameter der Funktionen ist erforderlich, der des Namens optional und erhält durch Fehlen einen Default-Wert. Es werden zwei verschiedene Arten von Eingabemethoden durch die Interaktionskomponente definiert, zum einen Standard-Funktionen und zum anderen Dekorator-Funktionen. Die Standard-Funktionen werden in der Komponente definiert und können dann nach dem Importieren durch den Nutzer über einen Funktionsaufruf mit entsprechenden Parametern genutzt werden 1.

```
1   import pp_vis
2
3   # method usage
4   pp_vis.add_data(first_df, "Augmented data", "
        Augmentation")
5
6   # decorator usage
7   @pp_vis.add_decor
8   def preprocess_removenan(df):
9       data_without_nan = data_with_multiple_nan.dropna()
10      return data_without_nan
11  data_without_nan = preprocess_removenan(
        data_with_multiple_nan)
12
13  # helper functions
14  pp_vis.list_data()
15  pp_vis.delete_data()
```

Codebeispiel 1: Methode und Dekorator

Auch Dekorator-Funktionen werden standardmäßig definiert und können nach ihrem Import verwendet werden. Im Gegensatz zu den Standard-Funktionen werden

sie aber genutzt, um schon vorhandene Funktionen zu dekorieren und ihr Verhalten somit zu modifizieren. Wie auf der Abbildung 1 zu erkennen, ruft der Dekorator die dekorierte Funktion auf und ermöglicht Aktionen davor und danach, was im Kontext des Prototyps der Speicherung des Datensatzes vor und nach Bearbeitung durch die dekorierte Vorverarbeitungsfunktion entspricht. Somit muss bei dieser Verwendung darauf geachtet werden, dass nur Vorverarbeitungsfunktionen dekoriert werden, welche genau einen Datensatz als Eingabeparameter erhalten und den veränderten Datensatz als Ausgabe zurückgeben. Ist das nicht der Fall, muss auf die Standard-Funktion zurückgegriffen werden, welcher jedoch auch nicht mehrere Datensätze übergeben werden können. Die Dekorator-Funktion existiert in zwei Ausführungen und kann so einmal mit Default-Werten für Datensatz- und Vorverarbeitungsnamen verwendet werden. Ferner existiert sie in einer Version, in welcher diese Namen durch Nutzereingabe abgefragt werden. Ergänzend ist hier außerdem zu erwähnen, dass eine abwechselnde Nutzung von Methoden und Dekorator Verwendung ohne Probleme möglich ist.

Die Reihenfolge der Übergabe der Datensätze an den Prototypen spielt hier eine wichtige Rolle, denn sie wird genauso als Reihenfolge der Vorverarbeitung interpretiert und eine Sortierung im Nachhinein ist nicht mehr möglich. Die Eingabefunktionen können für alle drei Ausgabeformen verwendet werden und stoßen von sich selbst noch keine Ausgabeerstellung an. Bei dem Aktionsbereich handelt es sich um Methoden, die das restliche Spektrum an Prototyp Funktionalität abdecken. Spezifischer lassen sich Funktionen zur Erstellung der Ausgabeoptionen identifizieren, welche als oben dargestellte Standard-Funktion vorliegen und meist keine Übergabeparameter benötigen. Es muss jedoch im Vorfeld mindestens eine Eingabefunktion aufgerufen worden sein. Des Weiteren existieren noch Hilfsfunktionen, welche die Löschung und die der Übersicht dienenden Darstellung der eingegebenen Datensätze ermöglichen (Codebeispiel 1).

5.2.3 Generelle Contentgenerierung - Website

Zur generellen Contentgenerierung, können alle Visualisierungen und Informationen gezählt werden, welche nicht vorverarbeitungsspezifisch sind und somit für jeden beliebigen Datensatz und seine Veränderung erstellbar sind. Beispiele hierfür wären Mittelwerte der numerischen Attribute oder Diagramme, die Klassenverteilungen kategorialer Attribute abbilden. Durch die Erstellungsfunktionen der Interaktionskomponente ausgelöst, werden hier im Herzstück des Prototyps der Ausgabeart entsprechende Visualisierungen und Informationen über die eingegebenen Datensätze und deren Veränderungen dynamisch generiert. Das Hauptziel der in diesem Kapitel genauer erläuterten Komponente ist die Darstellung dieser Daten in einer interaktiven Website.

Da es sich bei der Webseitenausgabe um die zentrale und komplexeste Art der Ausgabe der Anwendung handelt, wird sie gesondert in diesem Kapitel betrachtet. Das Grundprinzip dieser Komponente ist die Generierung von HTML-Objekten, welche mit CSS angereichert werden, um ein funktionales und ansprechendes Erscheinungsbild zu schaffen. Abschließen wird noch JS-Code ergänzt, welcher die Implementierung einer grundlegenden Steuerung dieser Elemente ermöglicht. Programmatisch umgesetzt wird dies durch dynamisches Parsen von HTML, CSS und JS Code in einen Python String, welcher dann zum Schluss der Erstellung als HTML-Content in eine HTML-Datei geschrieben wird. Diese Funktionalitäten werden von Python standardmäßig angeboten und benötigt keine weiteren Bibliotheken. Ebenso können HTML an sich, wie auch CSS und JS in derselben Datei, in ihren entsprechenden Tags stehen (Codebeispiel 2).

```
1 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de"
     lang="de">
2 <head>
3 <style></style>
4 </head>
5 <body>
6 <script></script>
7 </body>
8 </html>
```

Codebeispiel 2: HTML Struktur

Das bedeutet, dass nur eine HTML-Datei erstellt werden muss, welche dann einfach mit Hilfe der schon in Python vorhandenen Webbrowser-Bibliothek geöffnet werden kann.

Angesprochen zur Abbildung der Website wird der standardmäßig installierte Webbrowser des genutzten Geräts, in welchem sich diese dann wie jede gängige Website verhält und benutzen lässt.

```
1       PREPROCESS_SVG = """<?xml version="1.0" encoding="UTF
            -8"?><svg id="Ebene_1" xmlns="http://www.w3.org
            /2000/svg" viewBox="0 0 20...
2
3       def generate_preprocess_button(name, ind):
4           return f"""<button class="svg-container" onclick="
                preprocessingButtonClicked({ind})"><svg width=
                "220" height="90">{PREPROCESS_SVG}</svg><div
                class="centered-text text-preprocess">
                Preprocessing: {name}</div></button>"""
```

Codebeispiel 3: HTML-String Parsen

In dem Codeausschnitt 3 dargestellt, ist die oben beschriebene Methode durch Python, einen dynamischen HTML-String zu parsen. Dies geschieht in der abgebildeten Python-Methode durch zum einen die Bereitstellung der statischen Elemente der jeweiligen HTML-Komponente als String. In welche dann die übergebenen Parameter Name und Index geparst werden, gefolgt von dem schlussendlichen Zurückgeben des finalen Strings. Somit besitzt diese Methode die Fähigkeit, Button-Elemente den Eingabeparametern entsprechend zu erzeugen.

Ein weiterer Teil der erstellten Website sind Diagramme, denn wie oben erläutert, erlaubt die Plotly Visualisierungsbibliothek die Umwandlung von Graphen in HTML-Objekte mit schon vorhandenen Javascript Steuerungselementen. Diese Objekte können dann einfach im analogen Vorgehen zum restlichen Erstellen der Website in den HTML-String geparst werden und so dynamisch erstellt und verwendet werden. Die dafür benötigten Graphen, wurden durch die Eingabe der durch die Eingabefunktionen erhaltenen Datensätze in Methoden der Plotly-Visualisierungsbibliothek geschaffen. Die Herangehensweise hierbei unterscheidet sich jeweils, basierend auf Art des Graphen, zusätzlich benötigten Funktionen und anhand des gewähltem

Plotly-Moduls zur Erstellung des Graphen. In der Abbildung 4 ist die Erstellung eines Barplots mit dem Plotly *Graph-Objects-Module* zu erkennen. Daraufhin wird ein HTML-Element erstellt, welches dann in einen HTML-String geparst wird.

```
1       categorical_figure = go.Figure(px.parallel_categories(
            df))
2       categorical_figure.update_layout(
3           title=None, showlegend=False, margin=dict(l=15, r
                =15, t=15, b=15), width=800
4       )
5       chart_html_categorical_parallel = pyo.plot(
6           categorical_figure, include_plotlyjs=False,
                output_type="div"
7       )
8       chart_holder = f"""<div class="chart-holder">{
            chart_html_categorical_parallel}</div>
```

Codebeispiel 4: Plotly Html Erstellung

Durch das Überführen in den HTML-String finden sie auf der Website ihren Platz und dienen somit dem Nutzer zum einen als Einblick in seine übergebenen Datensätze, deren Attribute, sowie Veränderungen. Dem gleichen Muster folgend konstruiert der Prototyp in dieser Komponente nun verschiedene Bereiche der Webseitenausgabe und eben jene sollen nun im folgenden genauer betrachtet werden.

Tabellarische Merkmalsinformationen

Als ein erstes Analysetool werden durch die Webseitenkomponente, wie aus der Abbildung 7 ersichtlich, tabellarische Informationen aus den Datensätzen generiert. Hierbei handelt es sich zum einen um generelle Informationen zu den Attributen des Datensatzes, wie die Anzahl an Nullstellen, den Datentyp oder auch formatabhängige Werte wie Durchschnitt und Mode. Außerdem werden Beispieldaten zufällig ausgewählt und in einer weiteren Tabelle abgebildet. Abschließend lässt sich auch noch eine Überschrift des Datensatzes erkennen, welche um Informationen zu Name, Anzahl an Beobachtungen und Menge der Attribute erweitert wird.

Spezifische Merkmalsvisualisierung

Die Abbildung 8 des Generell-Plots-Bereichs des Prototyps zeigt eine weitere Nutzung der Anwendung als Analysewerkzeug der Attribute eines Datensatzes. Im

Dataset Starting data with 2025 data points and 6 variables

Variable information

Attribute	DType	NaN-Values	Mean	Median	Mode
title	object	0			Drishyam
release_year	int64	0	1991.97	2000.0	
runtime	object	0			130 min
genre	object	0			Drama
rating	float64	0	7.969	7.9	
gross(M)	float64	0	62.858	10.9	

Sample data

	title	release_year	runtime	genre	rating	gross(M)
1869	You Can't Take It with You	1938	126 min	Comedy	7.8	4.66
1900	English Vinglish	2012	134 min	Comedy	7.8	1.67
1249	Dallas Buyers Club	2013	117 min	Biography	7.9	27.30
1958	Harry Potter and the Deathly Hallows: Part 1	2010	146 min	Adventure	7.7	295.98
2505	21 Grams	2003	124 min	Crime	7.6	16.29

Abbildung 7: Tabellarische Merkmalsinformationen

oberen Teil lassen sich für jedes kategoriale und numerische Attribut entsprechende Verteilungsdiagramme anzeigen. Darunter werden diese zwei Arten von Attributen zusammengefasst in sogenannten *Parallel-Diagrammen* abgebildet. Die inhärente Interaktivität dieser Diagramme, sowie die zusätzlichen Auswahlmöglichkeiten durch die Dropdown-Menüs erfüllen die ersten zwei Anforderungen an den Prototyp und bieten dem Nutzer die Möglichkeit, seine eingegebenen Datensätze interaktiv und visuell aufbereitet zu analysieren. Wichtig im Hinblick auf die Parallel-Diagramme ist, dass diese als einzige Diagramme dieser generierten Website in einem SVG-Format dargestellt werden, um deren übersichtliche Darstellung der Verteilung aller Attribute zwar zu nutzen, dadurch aber keine schlechte Performance der Seite zu erzielen. Eine wichtige Funktion, um auch diese generellen Graphen, sowie die vorher erläuterten generellen Informationen als Werkzeug zur Abbildung von Veränderung zu nutzen, ist die simultane Darstellungsmöglichkeit dieser Bereiche für aufeinander folgende Datensätze. Wie anhand der Abbildung 9 zu erkennen, ermöglicht es diese Darstellung, dass Unterschiede und Gemeinsamkeiten leicht ermittelt werden können.

Vergleichende Merkmalsvisualisierung

Auch an Stellen, wo ein direkter Vergleich von zwei Datensätzen stattfindet, werden die Diagramme auf der Website verwendet. In der Abbildung 10, werden die Unterschiede von Attributen zweier Datensätze grafisch aufbereitet und auch eine Attributauswahl in Form eines *Dropdown-Menüs* ist hier wieder möglich.

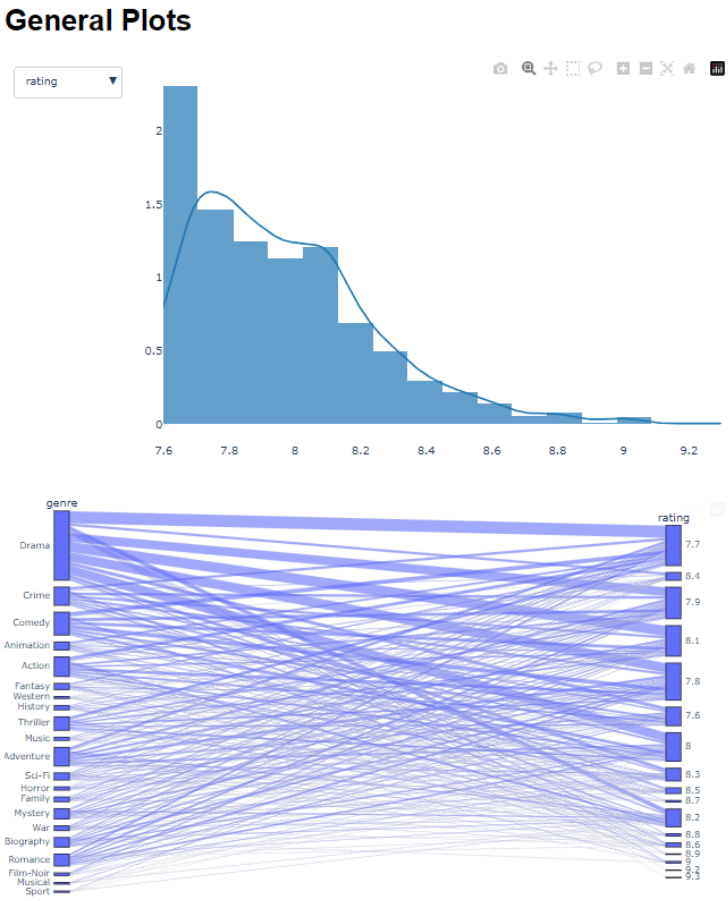
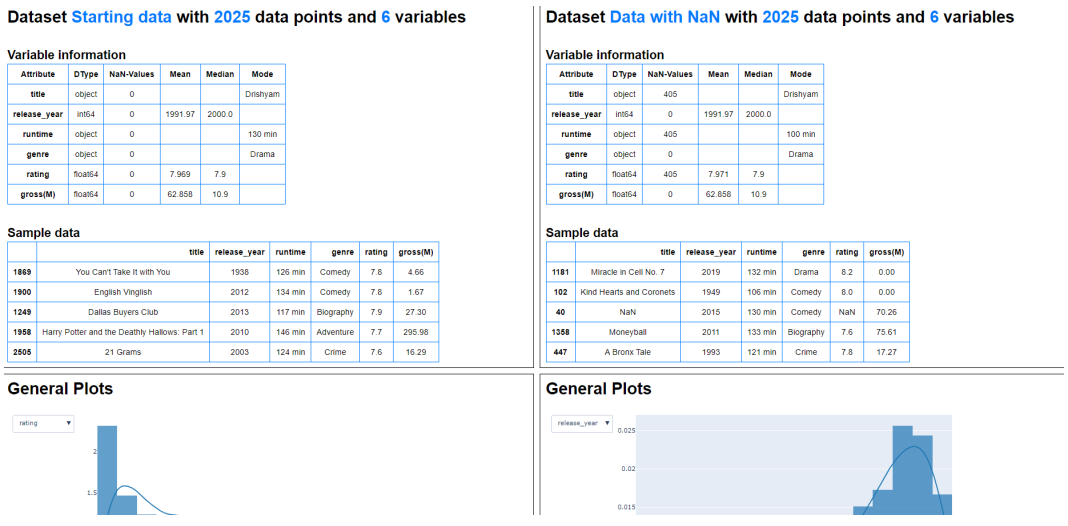


Abbildung 8: Spezifische Merkmalsvisualisierung



Dataset Starting data with 2025 data points and 6 variables

Variable information

Attribute	DType	NaN-Values	Mean	Median	Mode
title	object	0			Drishyam
release_year	int64	0	1991.97	2000.0	
runtime	object	0			130 min
genre	object	0			Drama
rating	float64	0	7.969	7.9	
gross(M)	float64	0	62.858	10.9	

Sample data

	title	release_year	runtime	genre	rating	gross(M)
1869	You Can't Take It with You	1938	126 min	Comedy	7.8	4.66
1900	English Vinglish	2012	134 min	Comedy	7.8	1.67
1249	Dallas Buyers Club	2013	117 min	Biography	7.9	27.30
1958	Harry Potter and the Deathly Hallows: Part 1	2010	146 min	Adventure	7.7	295.98
2505	21 Grams	2003	124 min	Crime	7.6	16.29

Dataset Data with NaN with 2025 data points and 6 variables

Variable information

Attribute	DType	NaN-Values	Mean	Median	Mode
title	object	405			Drishyam
release_year	int64	0	1991.97	2000.0	
runtime	object	405			100 min
genre	object	0			Drama
rating	float64	405	7.971	7.9	
gross(M)	float64	0	62.858	10.9	

Sample data

	title	release_year	runtime	genre	rating	gross(M)
1181	Miracle in Cell No. 7	2019	132 min	Drama	8.2	0.00
102	Kind Hearts and Coronets	1949	106 min	Comedy	8.0	0.00
40	NaN	2015	130 min	Comedy	NaN	70.26
1358	Moneyball	2011	133 min	Biography	7.6	75.61
447	A Bronx Tale	1993	121 min	Crime	7.8	17.27

General Plots

General Plots

Abbildung 9: Vergleich zweier Datensätze

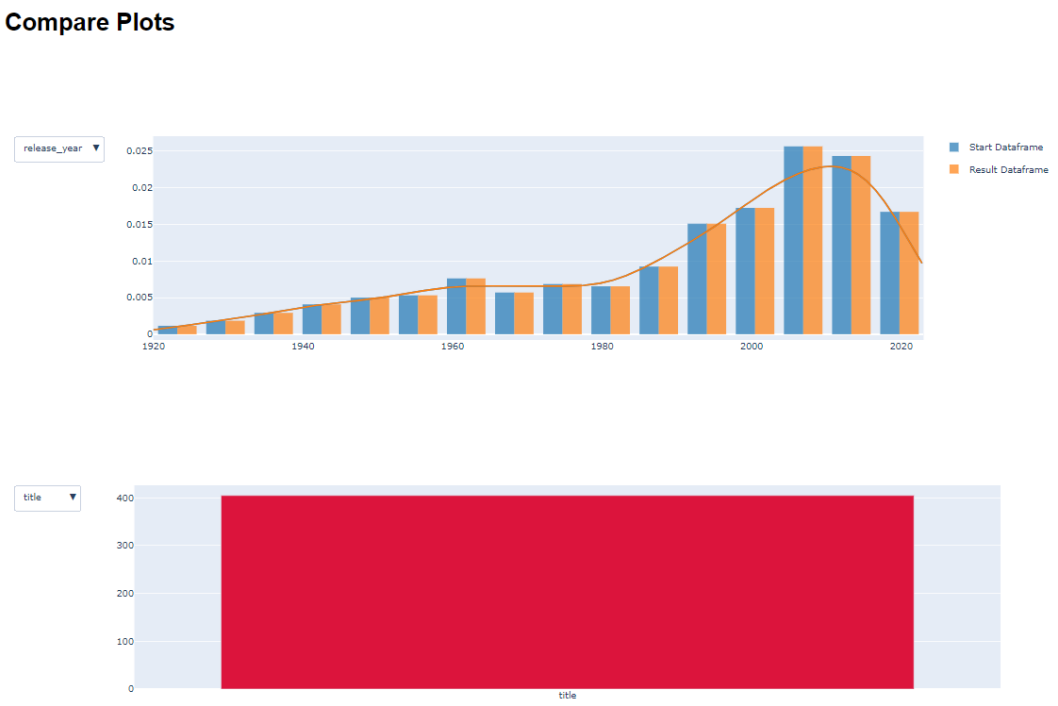


Abbildung 10: Vergleichende Merkmalsvisualisierung

Vorverarbeitungsworkflow

Abschließend findet auch die Darstellung des Vorverarbeitungsprozesses hier einen Platz (Abbildung 4). Sie dient zur Übersicht und für die Steuerung der Website und setzt sich aus SVG's zusammen, in welche dynamisch Text ergänzt werden kann, wofür der in der Interaktionskomponente definierte Titel der Datensätze verwendet wird. Da SVG's spezifisch für Webseiten entwickelt wurden, ist das Darstellen als String unproblematisch und infolgedessen auch das Parsen in den HTML-String. Mithilfe dieser Komponente wird in der Anwendung zwischen den Darstellungen für die verschiedenen Datensätze navigiert und durch Klicken auf das SVG mit dem entsprechenden Datensatznamen die Seite mit den vorausgehend erläuterten generellen Informationen und Graphen für diesen dargestellt. Wird auf das SVG eines Vorverarbeitungsschrittes geklickt, so wird zuerst die in Abbildung 9 abgebildete Simultandarstellung ausgegeben, gefolgt von den Vergleichsgraphen 10. Außerdem werden auch spezifische Informationen eines Vorverarbeitungsschrittes hier dargestellt, dieser Prozess wird jedoch in Kapitel 5.2.5 expliziter erläutert. Die durch die Website verwendeten Graphen und Tabellen können an vielen Stellen nicht einfach

aus den Eingabedaten erstellt werden, sondern benötigen weitere Aktionen und Anpassungen, um geeignete Informationen aus ihnen zu generieren. Dies geschieht meist mit durch Pandas bereitgestellten Hilfsfunktionen, welche auf den die Datensätze haltenden *Dataframes* ausgeführt werden können, jedoch auch durch klassische Python-Operationen wie die Iterierung über *Arrays* oder der Elementzugriff auf ebenjene.

5.2.4 Generelle Contentgenerierung - Export

Bildausgabe	**Groß**	**Mittel**	**Klein**
Anzahl der Datensätze	Alle	Start+Ende	keine
Vorverarbeitungsübersicht	true	true	true
Attributinformationen	true	true	false
Beispieldaten	true	true	false
Parallel-Graphen	true	true	false
Zusätzliche spezifische Vorverarbeitungsgraphen	true	false	false

Tabelle 3: Aufschlüsselung Bildausgabearten

Neben der Erstellung der Website als Ausgabe können durch den Prototyp noch zwei weitere Arten erzeugt werden. Analog durch eine Erstellungsfunktion ausgelöst, können zum einen Bilder im Portable Network Graphics (PNG) Format generiert werden, die Informationen zum Vorverarbeitungsprozess kompakt abbilden. Des Weiteren kann auch die Erstellung von informativen Daten in Form von JSON-Objekten angestoßen werden, welche in anderer Form nutzbare Schlüsse zu dem Vorverarbeitungsprozess liefern. Wichtig hier noch einmal zu erwähnen ist, dass sich alle drei Ausgabearten dieselben Eingabefunktionen teilen, womit auch alle drei nacheinander ausgeführt werden könnten.
Wird die Bildergenerierung nun genauer betrachtet, so lassen sich drei Modi erkennen. Zum einen kann ein ausführliches, sehr großes Bild erstellt werden, was Informationen zu jedem Vorverarbeitungsschritt liefert, sowie eine generelle Übersicht des Workflows und spezifische Visualisierungen der Vorverarbeitungsschritte. Die zweite Möglichkeit, ein Bild zu erstellen, gibt nur noch Informationen zum

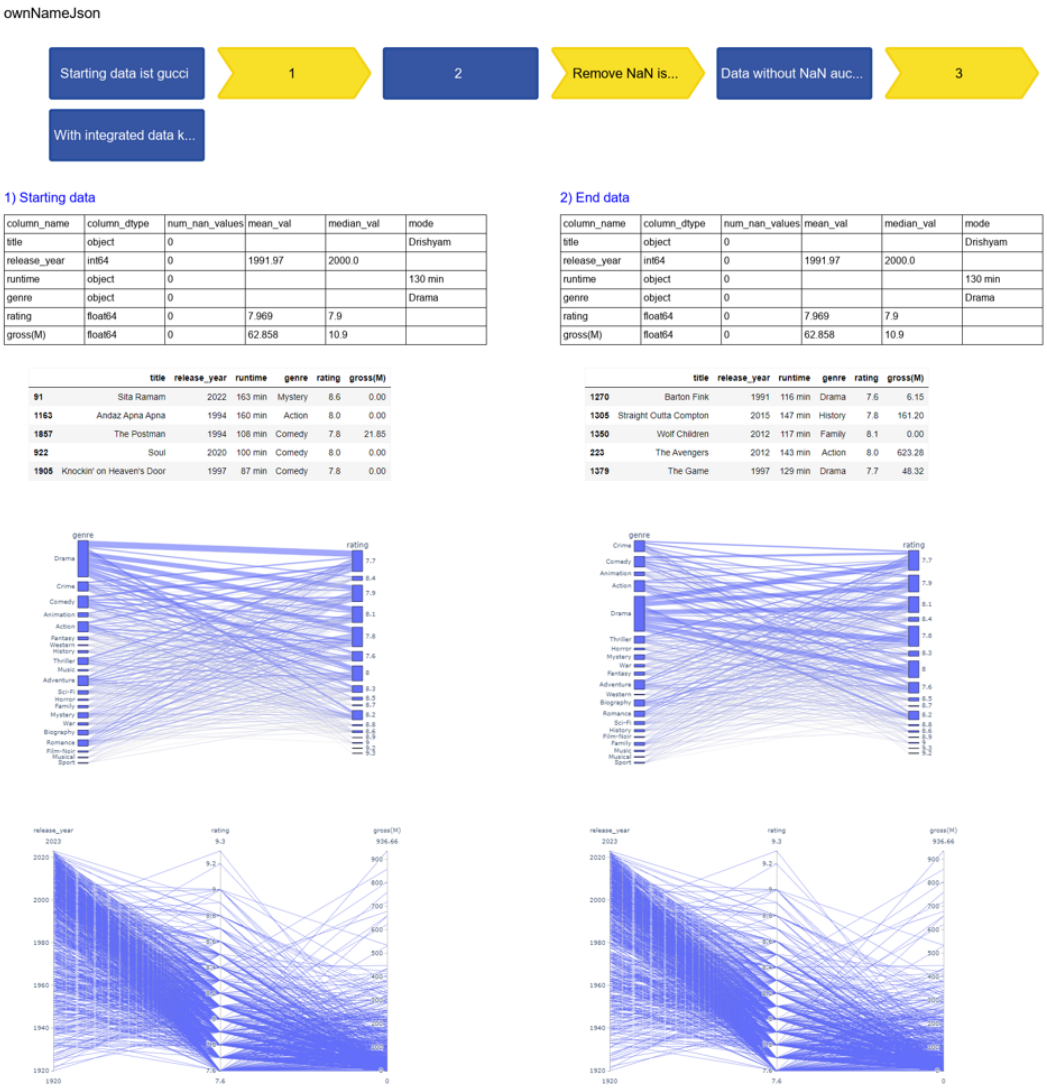
Start- und Enddatensatz aus, sowie die Workflow-Übersicht und der letzte Modus nur noch die Übersicht. Eine Zusammenfassung der drei Möglichkeiten kann der Tabelle 3 entnommen werden. Für die Erstellung der Bilder wird die Python *Pillow-Bibliothek* verwendet, welche einen virtuellen Canvas schafft und das Zeichnen von Objekten auf ihm, sowie die Ausgabe des Bildes ermöglicht.

```
1      image = Image.new("RGBA", (img_width, 1800), color="
          white")
2      draw = ImageDraw.Draw(image)
3
4      font_title = ImageFont.truetype("arial.ttf", 24)
5      draw.text((20, 20), title, font=font_title, fill="
          black")
6
7      start_categorical_figures = px.parallel_categories(
          savedData[0][1])
8      start_categorical_fig_bytes =
          start_categorical_figures.to_image(format="png")
9      start_categorical_image = Image.open(io.BytesIO(
          start_categorical_fig_bytes))
10     image.paste(start_categorical_image, (20,
          current_y_position))
```

Codebeispiel 5: Bild Erzeugung

In der Abbildung 5 zu erkennen, ist zum einen die Erstellung des Canvas mit zu definierenden Parametern. Danach wird ein Textobjekt und seine zugehörige Font erstellt und an der gegebenen Stelle eingefügt. Abgeschlossen wird der Prozess mit dem Hinzufügen einer durch Plotly erstellten Grafik, welche hier nicht wie bei der Webseitenausgabe in ein HTML-Element umgewandelt wird, sondern in ein PNG-Format konvertiert. Danach wird das Bild mithilfe des *io-Modules* der Python-Standardbibliothek aus den vorher erzeugten Byte-Daten geöffnet und in den Canvas gelegt.

Im Hinblick auf eine Beispielausgabe der Bildgenerierung 11, lässt sich diese zuerst als mittleres Bild einordnen. Wären die Informationen des unteren Teils für jeden Datensatz vorhanden, so würde es sich um die größtmögliche Bildausgabe handeln, wären gar keine Datensatzinformationen vorhanden, um den kleinsten Typ der Bildausgabe. An oberster Stelle zu erkennen, ist der durch den Nutzer



column_name	column_dtype	num_nan_values	mean_val	median_val	mode
title	object	0			Drishyam
release_year	int64	0	1991.97	2000.0	
runtime	object	0			130 min
genre	object	0			Drama
rating	float64	0	7.969	7.9	
gross(M)	float64	0	62.858	10.9	

	title	release_year	runtime	genre	rating	gross(M)
91	Sita Ramam	2022	163 min	Mystery	8.6	0.00
1163	Andaz Apna Apna	1994	160 min	Action	8.0	0.00
1857	The Postman	1994	108 min	Comedy	7.8	21.85
922	Soul	2020	100 min	Comedy	8.0	0.00
1905	Knockin' on Heaven's Door	1997	87 min	Comedy	7.8	0.00

column_name	column_dtype	num_nan_values	mean_val	median_val	mode
title	object	0			Drishyam
release_year	int64	0	1991.97	2000.0	
runtime	object	0			130 min
genre	object	0			Drama
rating	float64	0	7.969	7.9	
gross(M)	float64	0	62.858	10.9	

	title	release_year	runtime	genre	rating	gross(M)
1270	Barton Fink	1991	116 min	Drama	7.6	6.15
1305	Straight Outta Compton	2015	147 min	History	7.8	161.20
1350	Wolf Children	2012	117 min	Family	8.1	0.00
223	The Avengers	2012	143 min	Action	8.0	623.28
1379	The Game	1997	129 min	Drama	7.7	48.32

Abbildung 11: Erzeugtes mittleres Bild

bestimmbare Titel, gefolgt von der Vorverarbeitungsprozessübersicht. Diese wird dynamisch erzeugt und kann sich so über mehrere Zeilen erstrecken, wenn genug Datensätze eingegeben wurden. Außerdem werden zu lange Titel automatisch gekürzt, bevor sie aus ihrem SVG herausragen würden. Hierdurch erfüllt der Prototyp an dieser Stelle die Anforderung nach Zugänglichkeit, was bedeutet, dass er beliebige Eingabemengen und Namen ermöglicht, was dem Nutzer umständliche Anpassungen dieser erspart. Dies spiegelt sich auch in der dynamischen Größe des Bildes an sich, der Breite einzelner Reihen der Datensatzinformationen und deren durch die Menge an Attributen ermittelten Höhe wieder.
Im Falle der großen Bildausgabe werden, falls bestehend, zusätzliche spezifische Vorverarbeitungsdiagramme an das Ende der restlichen Darstellungen angehängt. Dies geschieht für Diagramme der Nullstellen und Ausreißer, jedoch nicht für die Erkennung von Datenmanipulationen. Alle drei spezifischen Generierungen werden in Kapitel 5.2.5 genauer erläutert und eine beispielhafte Darstellung ist in der Evaluation zu finden, wo diese auch analysiert wird. Im Bereich der Datensatzinformationen, kann analog zur Webseitengenerierung die Methodik der Visualisierung von Veränderung durch Gegenüberstellung aufgefunden werden. Auch die Inhalte der Datensatzinformationen wurden hier wiederverwendet und lassen sich in eine Tabelle zu Attribut Metriken, einer Tabelle für Beispieldaten und den Parallelgrafiken für numerische sowie kategoriale Werte segmentieren. Trotz der exakt gleichen Diagramme können die hier Gezeigten natürlich nicht interaktiv dargestellt werden und müssen so untereinander platziert werden. Dies ist primär bei dem in Abbildung 24 dargestellten Bereich der spezifischen Diagramme der Fall (Abbildung 24).
Bei der Contentgenerierung in Datenform werden alle für die Webseite und das Bild benötigten Daten als JSON-Datei ausgegeben. Mithilfe dieser präparierten Ausgabe lassen sich dann einfach eigene Grafiken oder Übersichten generieren. In der Abbildung 12 zu erkennen ist ein Ausschnitt aus einer erzeugten Datei, welche die Attribut-Übersicht in einer JSON-Objektform enthält. Diese Unterteilung der Datei in JSON-Objekte ist ein wiederkehrendes Muster, denn zum einen können diese einfach aus *Python-Dictionaries* konstruiert werden. Zum anderen fördern sie eine

Erstellung der JSON-Datei in einer lesbaren und gut strukturierten Form, was die Zugänglichkeit erleichtert. Abschließend gilt es noch zu erwähnen, dass auch die Bild- und Datengenerierung Informationen zu den spezifischen Vorverarbeitungsschritten erzeugt, was jedoch im nachfolgenden Kapitel genauer beschrieben wird.

5.2.5 Spezifisch Content Generierung

In den spezifischen Teilen der drei Content-Generierungs-Komponenten wird auf geteilte Funktionen zugegriffen, die anhand zweier aufeinanderfolgender Datensätze spezifische auf den Daten ausgeführte Vorverarbeitungsschritte erkennen können. Ein wichtiger Aspekt, der hier zu beachten ist, besteht darin, dass diese Identifikationen zwar mit dem Ziel implementiert werden, die spezifischen Vorverarbeitungen zu erkennen. Es jedoch aufgrund der vielen Überschneidungen der Auswirkungen verschiedener Vorverarbeitungen irreführend sein könnte, die schlussendliche Visualisierung als Erkennung eines spezifischen Schrittes zu betiteln. Ein Beispiel hierfür wären Änderungen der Nullstellen zwischen zwei Datensätzen, was bei einer Abnahme beispielsweise auf den Vorverarbeitungsschritt der Nullstellen-Entfernung schließen ließe. Dieser Effekt auf die Nullstellen könnte jedoch auch durch andere Arten der Vorverarbeitung erzielt werden. Deswegen wird in diesem Fall nicht von einer spezifischen Nullstellen-Entfernung gesprochen, sondern von dem beobachteten Effekt der Änderung der Nullstellen.


```
"Data: Data with NaN": {
    "Attributes": [
        {
            "column_name": "title",
            "column_dtype": "object",
            "num_nan_values": 405,
            "mean_val": "",
            "median_val": "",
            "mode": "Beauty and the Beast"
        },
        {
            "column_name": "release_year",
            "column_dtype": "int64",
            "num_nan_values": 0,
            "mean_val": 1991.97,
            "median_val": 2000.0,
            "mode": ""
        },
        {
            "column_name": "runtime",
            "column_dtype": "object",
            "num_nan_values": 405,
            "mean_val": "",
```


Abbildung 12: JSON Beispiel

Wurden die schlussendlich spezifischen Inhalte erzeugt, so werden sie an den generellen Content angehängt und dem Nutzer ergänzend präsentiert. Diese Erkennung und Generierung folgt einem modularen Prinzip, da eine Identifikation aller möglicher spezifischer Vorverarbeitungen den Rahmen dieser Arbeit sprengen würde.

Somit werden im Folgenden ausschließlich die schon möglichen Vorverarbeitungsprozesse vorgestellt. Eine beliebige Erweiterung um weitere Prozesse ist jedoch ohne Probleme möglich.

Datenmanipulation

Die erste erkennbare Vorverarbeitung bezieht sich auf die Prozesse des Zusammenfügens oder Auftrennens eines Datensatzes. Dies kann durch den Prototyp identifiziert werden, indem die Datenpunkte zweier aufeinanderfolgender Datensätze miteinander verglichen werden. Sowohl Überschneidungen als auch Unterschiede werden so bestimmt und im Falle existierender Unterschiede in eine visuelle Darstellung überführt. Diese findet wie anhand der Abbildung 14 zu erkennen, in grafischer Form auf der Website ihren Platz, wie auch in Datenform in der JSON-Datei (Abbildung 13). Die Abbildung der Website kopiert dabei einfach die generellen Informationen und Graphen der allgemeinen Content-Generierung und ermöglicht somit die Analyse der Veränderung, durch den Vergleich der nebeneinander dargestellten Informationen. Hierbei stehen die Informationen zu den übereinstimmenden Datenpunkten immer ganz links als *Unaltered-Dataset*, die zu den hinzugefügten Datenpunkte rechts davon als *Appendix-Dataset* und die der entfernten Datenpunkte als *Appendum-Datenset* ganz rechts.

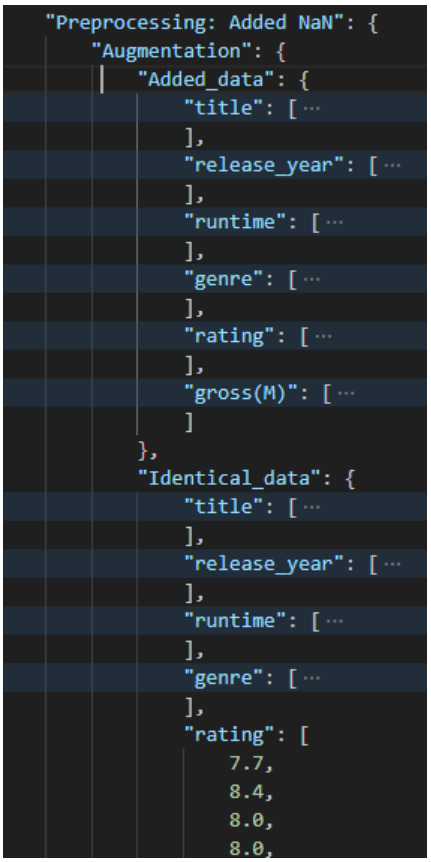


Abbildung 13: Datenausgabe Datenmanipulation

In der JSON-Datei wird für die neu generierten Visualisierungsdaten ein zusätzliches JSON-Objekt erstellt, welches die Darstellungsdaten aller generierter Informationen enthält. Auf eine ergänzende Darstellung in den drei Bildausgaben wurde für

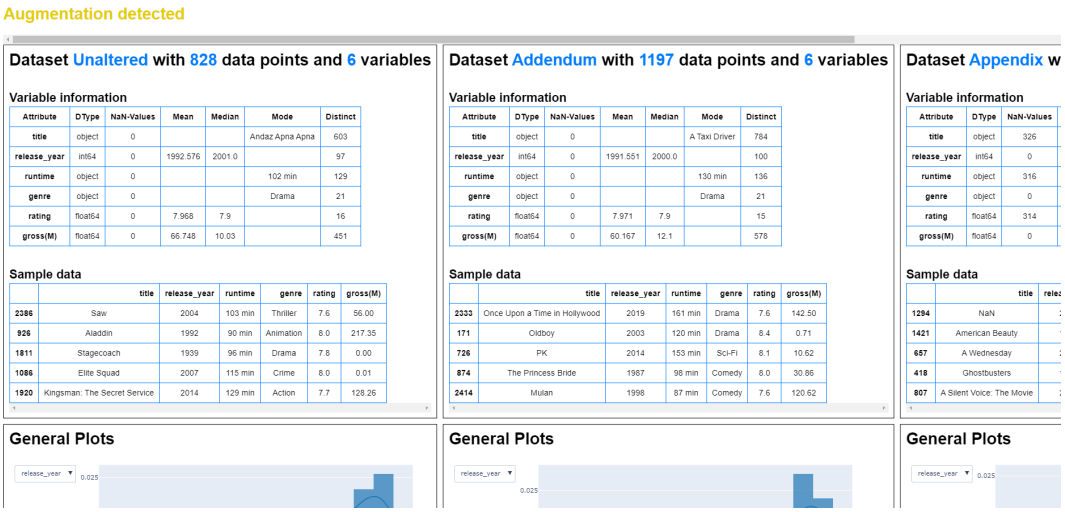


Abbildung 14: Datenmanipulation Darstellung

diesen Vorverarbeitungsprozess verzichtet, da diese sonst an Übersichtlichkeit hätten einbüßen müssen. Eine nennenswerte Einschränkung dieser und aller folgender spezifischen Visualisierung ist der Grenzfall, dass in einem Vorverarbeitungsschritt exakt die gleichen Datenreihen entfernt und hinzugefügt wurden. Dies kann nicht durch die bestehenden Erkennungsalgorithmen des Prototyps erkannt werden und somit werden diese Datenreihen als übereinstimmend identifiziert.

Nullstellen

Auch für Vorverarbeitungsprozesse, durch welche Änderungen an den Nullwerten der Datensätze stattfinden, existiert eine spezifische Content-Generierung. Diese wird angestoßen, wenn eine Änderung der Nullwert enthaltenden Reihen der Datensätze erkannt wird. Also entweder Beobachtungen mit Nullwerten ergänzt oder entfernt werden, sowie wenn schon bestehende Datensatzreihen Nullwerte imputiert oder ergänzt bekommen. Wie schon bei der vorausgehenden spezifischen Contentgenerierung kann auch hier nicht zwischen identischen, gleichzeitig entfernten und ergänzten Beobachtungen der Datensätze unterschieden werden. Außerdem existiert der Sonderfall, dass eine Datenreihe mit einer oder mehreren Nullstellen entfernt wird und ein weitere durch das Einfügen von Nullwerten eine identische Form annimmt. Auch dieser sowie der entsprechend gegensätzliche Vorverarbeitungsgrenzfall wäre nicht identifizierbar.

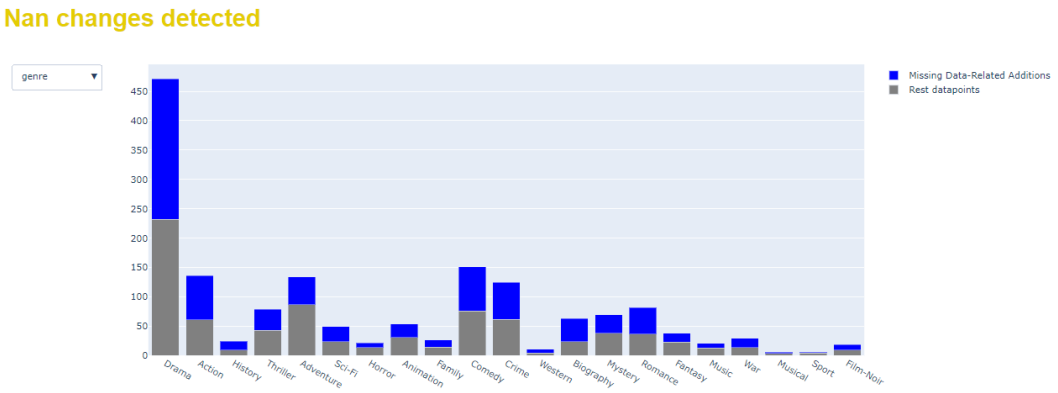


Abbildung 15: Nullstellenänderungen Darstellung

Die generierte Darstellung dieses Vorverarbeitungsprozesses, welche sich für die Webseitenausgabe in der Abbildung 15 vorfinden lässt, stützt sich auf die strukturelle Korrelation zwischen den Nullwerten und den im selben Eintrag liegenden Datenpunkten. Dies resultiert daraus, dass eine Darstellung von Nullwerten auf der logarithmischen Skala eines Diagramms nicht möglich ist und somit für eine visuelle Aufbereitung der Veränderung die korrelierenden Datenpunkte benötigt werden. Diese werden in vier mögliche Kategorien unterteilt und in für jedes Attribut separaten Grafiken dargestellt.

- Missing Data-Related Additions: Die Datenreihen dieser Werte enthalten Nullwerte und wurden entweder im letzten Vorverarbeitungsschritt ergänzt oder die Nullwerte hinzugefügt
- Missing Data-Related Exclusions: Diese Werte befanden sich in Einträgen, welche entweder entfernt oder deren Nullwerte imputiert wurden
- Missing Data-Related Unaltered: Bei diesen Werten handelt es sich um Datenpunkte, welche sowohl im vorausgegangenen wie im aktuellen Datensatz Nullwerte in ihrer Datensatzreihe besaßen
- Rest datapoints: Diese Datenpunkte besitzen keine strukturelle Korrelation zu Nullwerten und dienen nur der Einordnung der restlichen Arten. Im Falle der Abwesenheit der ersten zwei Kategorien werden diese Punkte und die Missing *Data-Related Unaltered* Datenpunkte nicht dargestellt

Nur die größtmögliche Bildausgabe wird durch die Visualisierungen dieses Vorver-

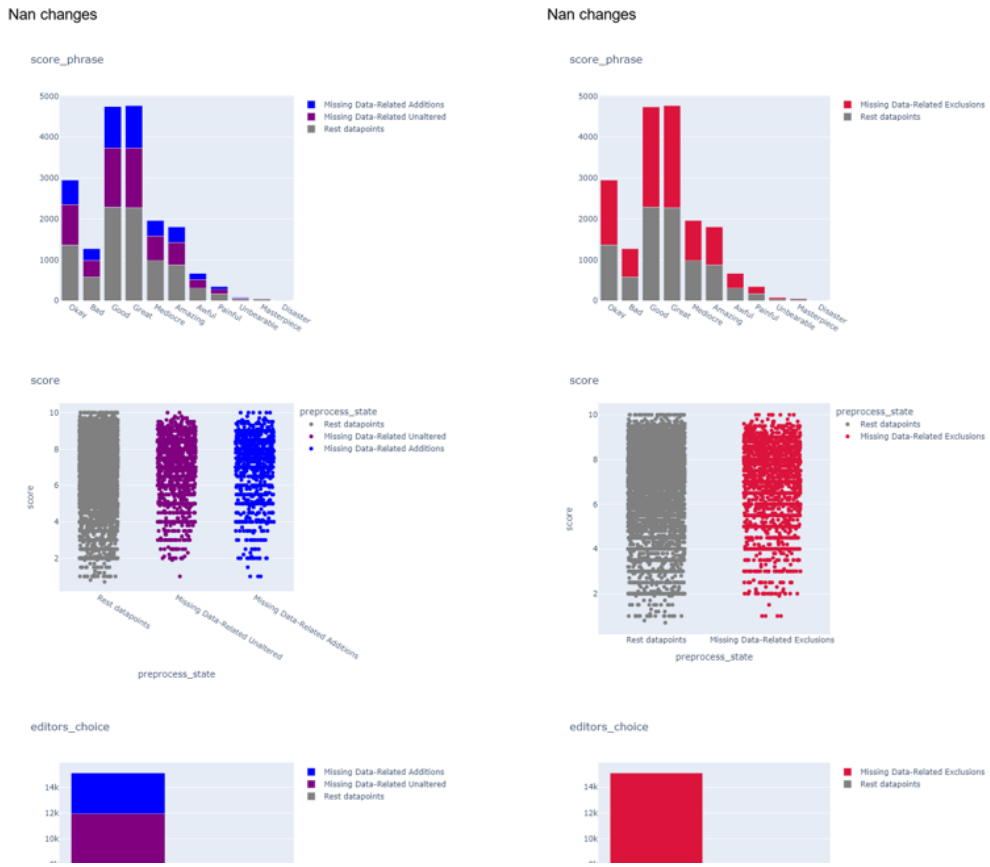


Abbildung 16: Nullstellenänderungen Bildausgabe

arbeitungsprozesses erweitert und ein Ausschnitt davon ist in der Abbildung 16 zu sehen. Der größte Unterschied ist hier, dass die jeweiligen Diagramme, nicht wie in der Webseitenausgabe, durch ein Burgermenü navigiert werden können, sondern untereinander dargestellt werden. Sonst sind die Darstellung und die Kategorien identisch. Hinsichtlich der Datenausgabe wird analog zur ersten vorverarbeitungs-spezifischen Contenterstellung ein zusätzliches JSON-Objekt erzeugt, welches die entsprechenden Visualisierungsdaten beinhaltet.

Ausreißer entfernen

Werden Änderungen der ausreißenden Werte bestimmter Attribute zwischen den verschiedenen Datensätzen erkannt, so wird wie auch bei den vorangegangenen Prozeduren eine spezifische Visualisierung erstellt. Hierfür gilt es vorausgehend zu erläutern, wie die Anwendung einen Datenpunkt als Ausreißer festlegt. Dies geschieht für numerische Attribute durch die Berechnung des Z-Scores für jeden

Abbildung 17: Ausreißer Webseitenausgabe

Datenpunkt, gefolgt von einer Filterung anhand dieser Werte und einem gesetzten Schwellwert. Für nicht numerische Attribute stellt der Prototyp in seiner aktuellen Version, noch keine Alternativen zur Ausreißer-Erkennung bereit und ignoriert diese somit bei der Identifikation dieses spezifischen Vorverarbeitungsprozesses. Mit Hilfe des Z-Scores lässt sich die Menge an Standardabweichungen eines Datenpunktes von dem Mittelwert seines Informationspunktes quantifizieren (Rousseeuw & Hubert (2011)), womit dieser die benötigte Bewertung von Werten für diese spezifische Vorverarbeitung ermöglicht. Der gesetzte Schwellwert setzt eine maximale Größe dieses Z-Scores fest und identifiziert somit alle diese Grenze übertretenden Datenpunkte als Ausreißer. Im Rahmen des Prototyps wurde der Schwellwert, basierend auf einer Empfehlung aus der Arbeit von Razzaq & Zaibidi (2023) auf den Wert 3 festgesetzt. Des Weiteren lässt sich durch die Betrachtung der Formel des Z-Scores erkennen, dass dieser sowohl positiv, wie auch negative Werte einnehmen kann, indem sich der Datenpunkt entweder über oder unter dem Durchschnitt des Informationspunktes befindet. Somit werden für die Nutzung nur eines Schwellwerts, die absoluten Z-Scores der jeweiligen Datenpunkte benötigt.

Nach der Analyse und Filterung der Attribute, werden die Ergebnisse, wie auch schon bei den vorausgehenden Verfahren in Visualisierungen überführt. Diese sind in Abbildung 17 dargestellt und zu erkennen ist eine Gegenüberstellung der restlichen Datenpunkte, sowie der hinzugefügten, entfernten und gleichgebliebenen ausreißenden Werten des jeweiligen Attributs. Der präsentierte Vergleich basiert auf dem vor der betrachteten Vorverarbeitung liegendem Datensatz sowie dem Er-

gebnisdatensatz ebenjener. Wie auch schon bei den anderen spezifischen Content Generierungsmethoden wird gleichsam hier die erste Aufgabe von PP-Vis nach einer Visualisierung, und zwar genauer der Visualisierung von Veränderung erfüllt. Analog zu den Nullstellen findet diese spezifische Visualisierung auch nur in der größten Bildausgabe der Anwendung ihren Platz und die generierten Daten wiederholt in einem zusätzlichen JSON-Objekt.

6 Evaluation

Nach der technischen und strukturellen Aufbereitung von PP-Vis gilt es im folgenden ebenjenen zunächst in einem Testszenario zu analysieren und danach eine Bewertung des Prototyps vorzunehmen. Dies soll sowohl durch Testpersonen als auch Leistungskennzahlen stattfinden.

6.1 Prototyp-Test

6.1.1 Daten und Vorverarbeitung

Bei dem für den Test verwendeten Datensatz handelt es sich um den auf Kaggle verfügbaren *IGN Games* Datensatz, dessen Komposition der Tabelle 4 zu entnehmen ist. Dieser enthält alle verarbeitbaren Datentypen, ist frei verfügbar und vollständig. Somit bietet er sich optimal für das Testen der Anwendung an, benötigt dennoch gewisse Anpassungen, um auch die spezifischen Vorverarbeitung-Contentgenerierungen auslösen zu können. Im Rahmen dieser Anpassungen wurden in dem Datensatz Nullstellen, an zufällig ausgewählten Stellen, für die Attribute *Title* und *Score* eingesetzt. Des Weiteren wurde der Datensatz in drei Teile zerlegt, um diese in einem Datenfusionsschritt wieder zusammenzufügen. Zum Schluss wurde noch, das *Editors Choice* Attribut zu einem Boolean-Datentypen umgewandelt.

Bei den in dieser beispielhaften Prozedur ausgeführten Datenvorverarbeitungsschritten handelte es sich um folgende, welche nach der Reihenfolge ihrer Nennung auf den Datensatz angewendet wurden. Beginnend mit der Datenfusion des zerlegten Datensatzes, folgt daraufhin die Entfernung der Nullstellen und danach die Beseitigung stark ausreißender Werte. Abgeschlossen wird die Vorverarbeitung mit dem *Oversampling* der unterrepräsentierten Klassen des Genre-Attributs. Die Instal-

ID	Integer
Score Phrase	String
Title	String
URL	String
Platform	String
Score	Float
Genre	String
Editors Choice	String
Release Year	Integer
Release Month	Integer

Tabelle 4: Attribute IGN Games

lation des Prototyps und die entsprechende Pythonumgebung wurden für diesen Test vorausgesetzt.

6.1.2 Eingabe

Das Zuführen der Daten in den Prototypen erfolgt in einem *Jupyther Notebook*, in welchem der Datensatz eingelesen und zwischen jedem Vorverarbeitungsschritt ein entsprechender Zwischenstand in Form eines *Pandas Dataframes* an diesen übergeben wird. Wie aus dem Codeausschnitt 6 zu erkennen, wird für die ersten zwei Vorverarbeitungsschritte die Eingabe durch Funktionen genutzt. Für die letzten zwei die Eingabe über das Dekorieren von Funktionen, welche die Vorverarbeitung vornehmen. Als optionale Parameter können den Funktionen die Namen der Datensätze und der Vorverarbeitungsschritte übergeben werden. Im Falle der Dekorationseingabe gibt es, wie oben erläutert, die Möglichkeit, durch den Input-Dekorator über Nutzereingaben die entsprechenden Parameter festzulegen. Die in diesem Fall erzeugte Abfrage lässt sich der Abbildung 18 entnehmen. In diesem Beispiel werden jeweils die erste Funktions- und Dekorator-Eingabe mit Parametern und die zweite ohne Parameter vorgenommen. Am Ende der Vorverarbeitung wird, wie aus dem Code 6 ersichtlich, zuerst eine Übersicht der gespeicherten Datensätze und ihrer Namen erzeugt. Danach wird die Erstellung einer Website durch den Prototyp angesto-

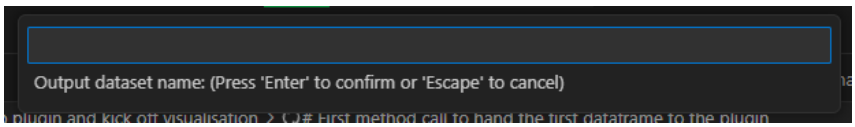


Abbildung 18: Nutzerabfrage

ßen, gefolgt von der Generierung eines Bildes mit Informationen zu allen Schritten und eines weiteren mit ausschließlich der Abbildung des Prozesses. Bei der letzten erzeugten Ausgabe handelt es sich um die JSON-Datei, welcher wie auch den Bildausgaben, Parameter zu Titel und Speicherpfad übergeben werden. Abschließend wird der Zwischenspeicher des Prototyps geleert, woraufhin dann nochmals der finale Datensatz der Vorverarbeitung übergeben und eine abschließende Website erzeugt wird.

```
    # First method call to hand the first dataframe to the
        plugin
    pp_vis.add_data(first_df, "Starting data")

    # Dataframe concatenation
    concatenated_data = pd.concat([first_df, second_df,
       third_df], ignore_index=True)

    # Second method call to hand in the second dataframe
    pp_vis.add_data(concatenated_data, "Concatenated data"
       , "Data Fusion")

    # Remove rows with nan values
    removed_nan_data = concatenated_data.dropna()

    # Third method call to hand in the dataframe with the
       removed nan rows
    pp_vis.add_data(removed_nan_data)

    # Definition of the outlier removal and normalisation
       methods and decorate them with the plugin
    @pp_vis.add_decor
    def remove_numeric_outliers(df, z_score):
        result_df = df
        for col in df.columns:
            if df[col].dtype == "int64" or df[col].dtype
               == "float64":
                z_scores = np.abs((result_df[col] -
                   result_df[col].mean()) / result_df[col
                   ].std())
```

```
23                  result_df = result_df[z_scores <= z_score]
24          return result_df
25
26      @pp_vis.add_fast_decor
27      def normalize_attributes(df, attr_name):
28          y = df[attr_name]
29          x = df.drop(columns=[attr_name])
30          x_numeric_columns = x.select_dtypes(include=['
               number', 'float']).columns
31          x_numeric = x[x_numeric_columns]
32          x_non_numeric = x.drop(columns=x_numeric_columns)
33          smote = SMOTE(sampling_strategy='auto',
               random_state=42)
34          x_numeric_resampled, y_resampled = smote.
               fit_resample(x_numeric, y)
35          x_resampled = pd.concat([x_non_numeric, pd.
               DataFrame(x_numeric_resampled, columns=
               x_numeric_columns)], axis=1)
36          resampled_data = pd.concat([x_resampled, pd.Series
               (y_resampled, name=attr_name)], axis=1)
37          return resampled_data
38
39      # Execution of the functions
40      removed_outliers_data = remove_numeric_outliers(
           removed_nan_data, 3)
41      normalize_attribute_data = normalize_attributes(
           removed_outliers_data, "release_month")
42
43      pp_vis.list_data()
44      pp_vis.create_webpage()
45      pp_vis.create_image(1, "Vorverarbeitungsprozess
           Detailliert", "./output")
46      pp_vis.create_image(2, "Vorverarbeitungsprozess
           Übersicht", "./output")
47      pp_vis.create_data("MeineDaten", "/output")
48
49      pp_vis.clear_data()
50      pp_vis.add_data(normalize_attribute_data)
51      pp_vis.create_webpage()
```

Codebeispiel 6: Vorverarbeitung und PP-Vis Nutzung

6.1.3 Webseitenausgabe

Beginnend mit der Analyse der ersten Webseitenausgabe, welche auch im Anhang 7 zu finden ist kann festgestellt werden, dass die Vorverarbeitungsübersicht erfolgreich konstruiert wurde. So wurden, wie anhand der Abbildung 19 ersichtlich, dy-

Dataset: Starting data → Preprocessing: Data Fusion → Dataset: Concatenated data → Preprocessing: 2 → Dataset: 3 → Preprocessing: Removed Outlier → Dataset: no outlier data → Preprocessing: 4-Preprocess-Step → Dataset: 5-Data

Dataset Starting data with 11175 data points and 10 variables

Variable information

Attribute	DType	NaN-Values	Mean	Median	Mode	Distinct
score_phrase	object	0			Great	11
title	object	0			Peter Jackson's King Kong	8548
url	object	0			/games/crash-twinsanity/ps2-667247	11158
platform	object	2184			nan	50
score	float64	4507	6.927	7.2		89
genre	object	21			Action	99
editors_choice	bool	0			False	2
release_year	int64	0	2006.482	2007.0		22
release_month	int64	0	7.159	8.0		12
release_day	int64	0	15.544	15.0		31

Sample data

	score_phrase	title	url	platform	score	genre	editors_choice	release_year	release_month	release_day
6321	Mediocre	Shaman King: Power of Spirit	/games/shaman-king-power-of-spirit/ps2-621374	PlayStation 2	5.0	Fighting, RPG	False	2004	11	11
10945	Great	Crosswords DS	/games/crosswords-ds/nds-949623	Nintendo DS	8.0	Puzzle, Word Game	False	2008	5	2
5785	Good	Shrek 2	/games/gba-video-shrek-2/gba-566201	Game Boy Advance	7.9	Action, Adventure	False	2004	5	4
1926	Painful	NASCAR Challenge	/games/nascar-challenge/gbc-12541	Game Boy Color	NaN	Racing	False	2000	1	4
6271	Okay	The Lord of the Rings: The Third Age	/games/the-lord-of-the-rings-the-third-age/gba-679793	Game Boy Advance	NaN	RPG	False	2004	11	2

General Plots

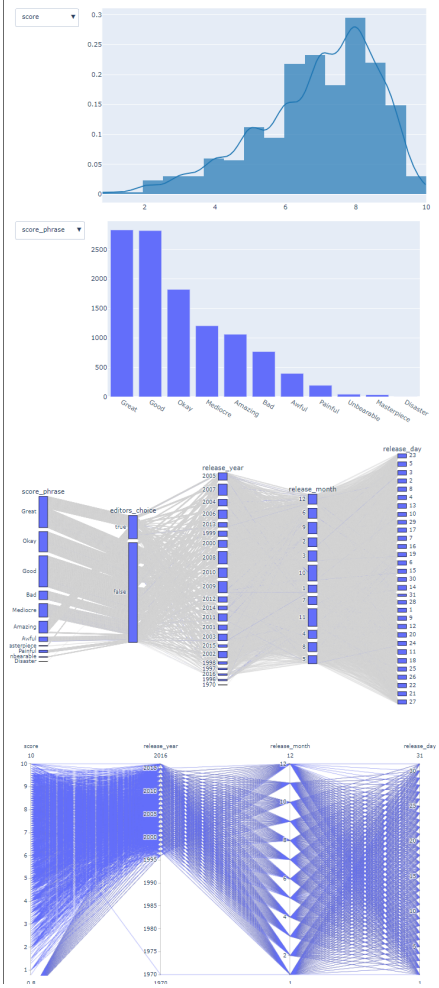


Abbildung 19: Generierte Datensatzansicht

namisch alle benötigten Steuerungselemente erstellt und die übergebenen Namen eingesetzt. Im Falle von fehlenden Namensparametern wurden numerische Ersatzwerte verwendet. Diese Elemente dienen nun, wie oben beschrieben, zum Ansteuern der Datensatz und Vorverarbeitungs spezifischen Sichten der Website. Im Falle der Datensatzansicht wurde diese erfolgreich für jeden der übergebenen Datensätze konstruiert und enthält auch die jeweiligen Übergabeparameter, wie anhand der Abbildung 19 erkennbar. Des Weiteren zeigt die Darstellung 19 exemplarisch für den ersten Datensatz die jeweiligen generierten und oben genauer erläuterten Teilbereiche dieser spezifischen Ansicht. Beginnend mit den generellen Informationen und gefolgt von den generellen Plots.

Auch die vorverarbeitungsspezifischen Ansichten wurden erfolgreich generiert und setzen sich aus den im Folgenden genannten Teilbereichen zusammen. Zur Veranschaulichung wurde hier die erste Datenintegration gewählt und die für diese generierten Informationen in den Abbildungen 20, 21 und 22, zur Übersicht dienend, dargestellt. Wieder beginnend mit der Vorverarbeitungsübersicht, folgt hier indessen die parallele Abbildung der zwei Datenansichten des Eingabe- und Ausgabe-Datensatzes der Vorverarbeitung. Diese ermöglicht, wie oben erläutert, die Erkennung von Veränderungen der zwei Datensätze. Ergänzend dazu lassen sich darunter außerdem Vergleichsdiagramme erkennen, welche wiederkehrende Attribute der zwei Datensätze in gemeinsamen Diagrammen darstellen. Abschließend finden sich nun noch der jeweiligen Vorverarbeitung entsprechende spezifische Darstellungen. Diese können variieren und werden in der folgenden Tabelle 5 für die jeweils getätigten Vorverarbeitungsschritte aufgeschlüsselt.

Beginnend mit der Datenintegration wurde hier die spezifische Contenterstellung

Vorverarbeitung	**erzeugte Visualisierung**
Datenintegration	Datenmanipulation, Nullstellen
Nullstellenentfernung	Datenmanipulation, Nullstellen, Ausreißer
Ausreißerentfernung	Datenmanipulation, Ausreißer
Oversampling	Datenmanipulation, Nullstellen, Ausreißer

Tabelle 5: Erkannte spez. Vorverarbeitungen

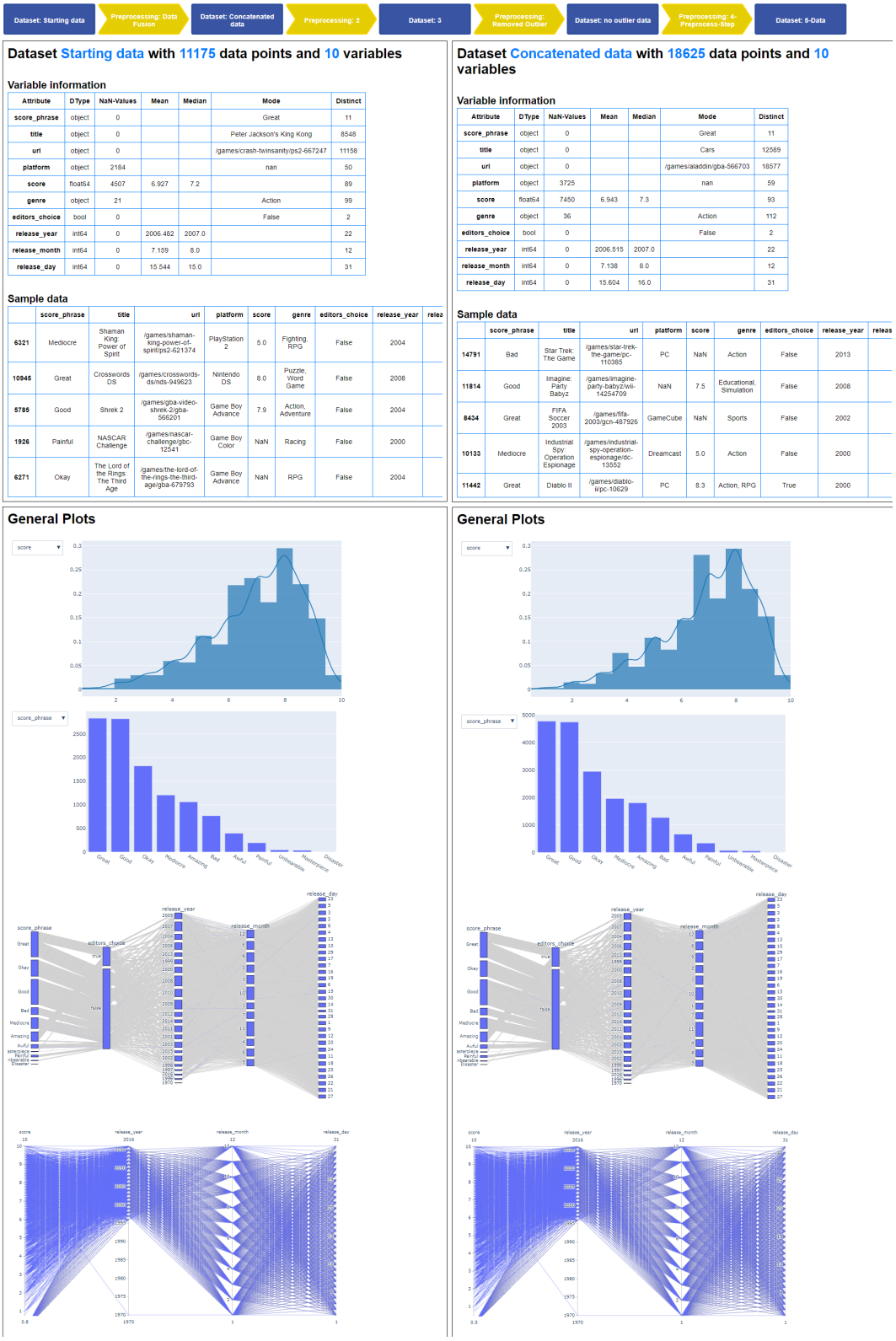


Abbildung 20: Generierte Vergleichsansicht 1

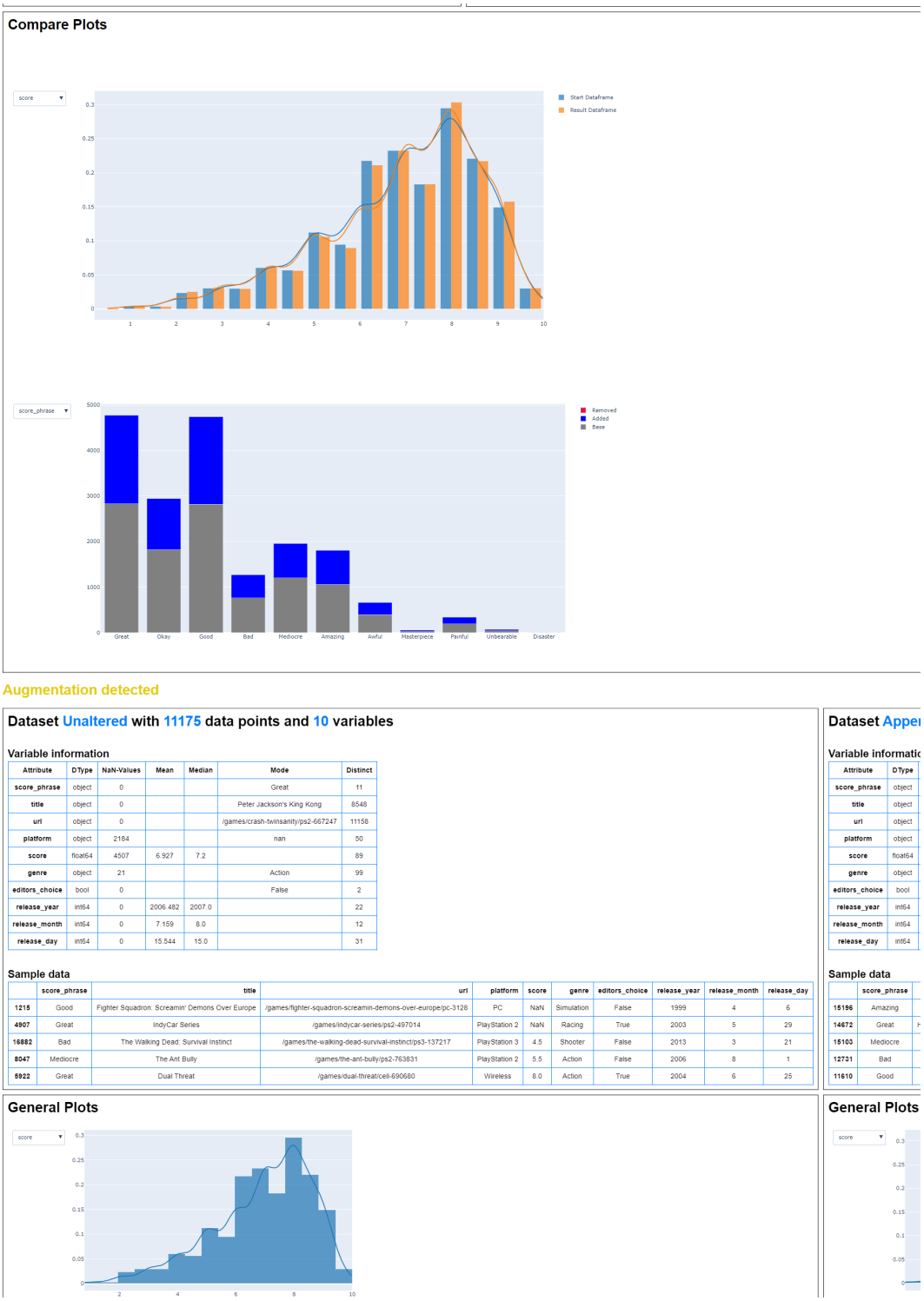


Abbildung 21: Generierte Vergleichsansicht 2

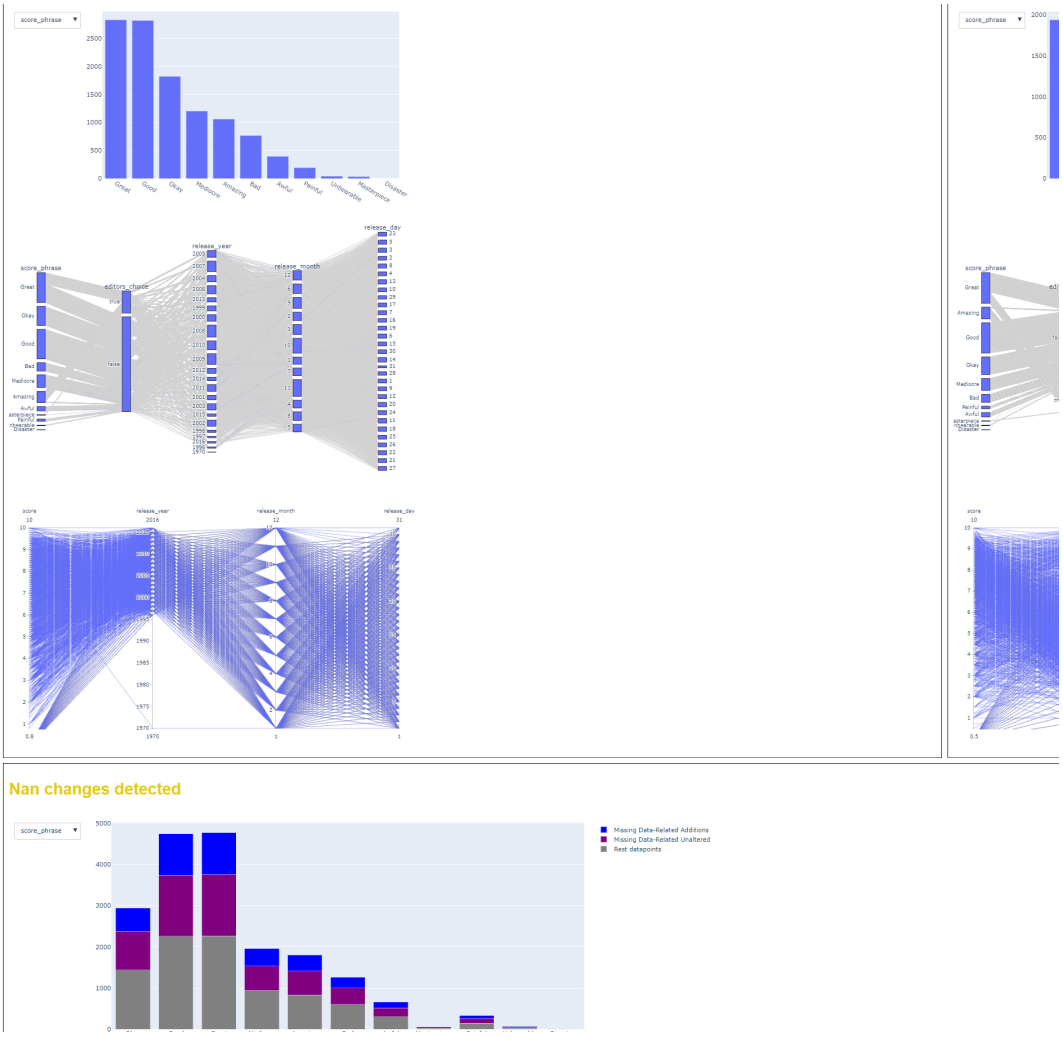


Abbildung 22: Generierte Vergleichsansicht 3

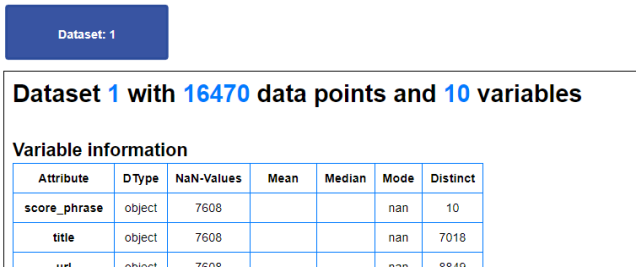


Abbildung 23: Generierte Webseite (ein Datensatz)

der Datenmanipulation und der Nullstellen ausgelöst. Während die Datenmanipulation im Hinblick auf die vorgenommene Integration logisch erscheint, so findet der Nullstellenspezifische Content nur seinen Platz, weil ebenjene in den integrierten Daten liegen. Der zweite Vorverarbeitungsschritt der Nullstellenentfernung hat alle spezifischen Contentgenerierungsmethoden ausgelöst. Während dies im Hinblick auf die Nullstellen-Contenterstellung wieder logisch scheint, lässt der erstellte Ausreißercontent erkennen, dass durch das Entfernen der Null-Wert-Reihen des Datensatzes eine starke Veränderung der Standardabweichungen der numerischen Attribute vorgenommen wurde. Die Generierung des Datenmanipulations-Contents für diese und auch alle anderen Vorverarbeitungsansichten, zeigt die fehlende Generalisierung von PP-Vis auf. So wird zum Beispiel im Falle der Nullstellenentfernung nicht spezifisch ebenjene erkannt und somit die dadurch fehlenden Reihen als eine Datenmanipulation identifiziert.

Analog zu den zwei vorausgehenden Vorverarbeitungen lässt auch der spezifische Content der anderen zwei Schlüsse zu dem genauen Vorgehen bei der Vorverarbeitung zu und dient somit schon ohne Betrachtung der spezifischen Visualisierungen als gutes Analysewerkzeug. Auch die Erstellung der zweiten Webseitenausgabe, mit nur einem eingegebenen Datensatz, wurde umgesetzt, wie anhand der Abbildung 23 zu erkennen ist. Die Datenansicht wurde erfolgreich konstruiert, aufgrund der fehlenden Vergleichsdaten jedoch keine vorverarbeitungsspezifische Sicht erzeugt.

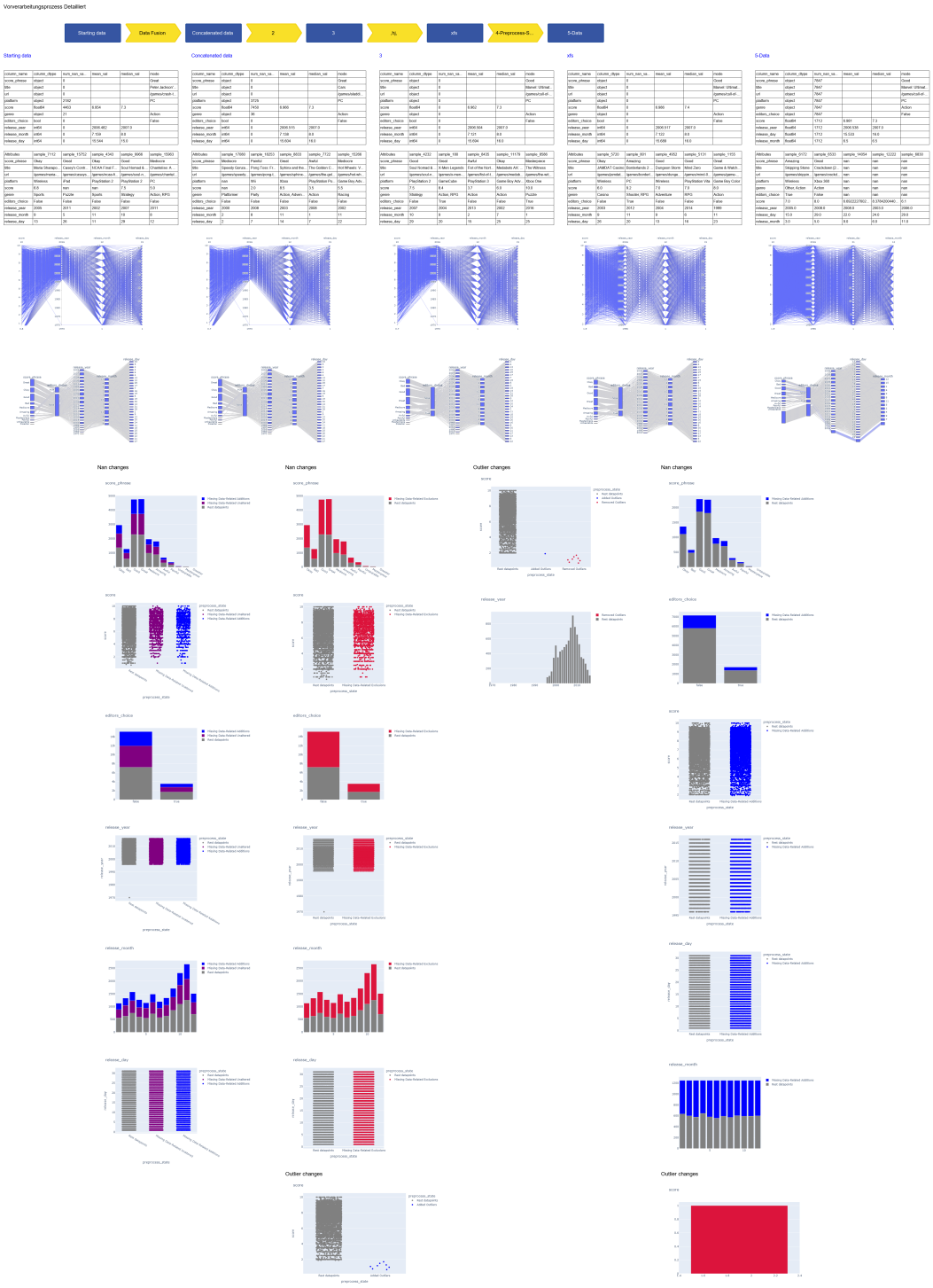


Abbildung 24: Größtmögliche Bildausgabe

6.1.4 Bild-Ausgabe

Mit Blick auf die durch diesen Test generierten Bilder lässt sich anhand der Abbildung 24 feststellen, dass auch dieser Prozess erfolgreich war (Anhang 7). Die Grafik zeigt die größtmögliche Form der durch den Prototypen erstellbaren Bilder und gleich zu Beginn ist erkennbar, dass die Vorverarbeitungsübersicht erfolgreich wiederverwendet wurde. Sie gleicht also im Hinblick auf Aussehen und dynamischen Daten exakt der durch die Webseitenausgabe präsentierten Übersicht. Darunter können für jeden der übergebenen Datensätze jeweilige Informationen und Visualisierungen erkannt werden. An oberster Stelle lassen sich die Namen der jeweiligen Datensätze als Überschrift der Spalte erkennen, gefolgt von den schon aus der Webseitenausgabe bekannten Attribut-Informationen, sowie Beispieldaten. Abgeschlossen wurde dieser generelle Inhalt der Bildausgabe mit den alle numerischen und kategorialen Attribute einschließenden, parallelen Grafiken. Darunter finden sich die jeweiligen Vorverarbeitung spezifischen Diagramme, welche jedoch aufgrund der fehlenden Interaktivität in der Bildausgabe separat für jedes betroffene Attribut untereinander abgebildet werden. Auch die Erstellung des kleinen Bildes war erfolgreich und beschränkt sich auf ausschließlich die Darstellung des ersten Bereiches dieser Abbildung 24.

6.1.5 Daten-Ausgabe

Die letzte zu analysierende Ausgabe betrifft die Generierung der Visualisierungsdaten als JSON-Datei und auch diese wurden erfolgreich konstruiert (Anhang 7). Anhand der Abbildung 25 können beispielsweise erzeugte Daten zu den Attribut-Informationen erkannt werden. Diese liegen meist in ihrer Struktur als eine Kombination von JSON-Objekten und Arrays ebenjener vor und können somit einfach programmatisch gelesen und wiederverwendet werden.

So zeigt der Codeausschnitt 7 das Lesen der Verteilungsdaten und die Überführung in Seaborn Grafiken, eine beispielhafte Verwendung dieser Daten. Auch die Daten der spezifischen Darstellungen finden in dieser Ausgabeform ihren Platz. Als Beispiel hierfür zeigt die Abbildung 26 die jeweiligen JSON-Objekte, welche Infor-

Abbildung 25: Datenausgabe Allgemein

Abbildung 26: Datenausgabe Nullstellen

mationen zu den Nullstellenveränderungen beinhalten.

```
1      import json
2      import pandas as pd
3      import seaborn as sns
4      import matplotlib.pyplot as plt
5
6      # Read JSON file and construct seaborn graph
7      data_dic = {}
8
9      # Open the json file and read the data
10     with open(".\output\MeineDaten.json", 'r') as
           json_file:
11          data = json.load(json_file)
12
13     # Get the data to visualize
14     data_dic = data
15     distribution_data = data_dic["Data: Starting data"]["
           Data_for_distribution_vis"]["score_phrase"]
16     series = pd.Series(distribution_data)
17
18     # Create the Seaborn barplot
19     plt.figure(figsize=(12, 6))
20     sns.barplot(x=series.values, y=series.index, palette="
           viridis")
```

Codebeispiel 7: Nutzung der Datenausgabe

6.2 Nutzerstudie

Nachdem die Arbeitsweise des Prototypen vorausgehend aufgeschlüsselt wurde, gilt es nun, diese durch Probanden bewerten zu lassen. Ziel soll hierbei sein, erstes Feedback zu der implementierten Anwendung zu erhalten und basierend darauf, Empfehlungen zu Verbesserungen tätigen zu können. Das frühe Erkennen von etwaigen Problemen oder Fehleinschätzungen während der Entwicklung der Anwendung kann zum einen, die weitere Arbeit an ebenjener in eine positive Richtung lenken, sowie umständliche Anpassungen in einer finalisierten Form verhindern. Aufgrund ebenjenes frühen Zeitpunkts der Evaluation wurde sich auch dazu entschieden, diesen Test als Online-Selbststudie durchzuführen, um so eine erste schnelle Einordnung des aktuellen Stands von PP-Vis zu erhalten. Alle für diese Studie benutzten Dateien lassen sich dem Anhang 7 entnehmen.

6.2.1 Methode

Um die jeweiligen Kennzahlen und subjektiven Bewertungen der Tester zu erhalten, wurde ein exemplarischer Anwendungsfall des Prototyps geschaffen, welcher einen Einblick in alle bestehenden Grundfunktionalitäten der Anwendung erlaubt. Hierbei zu beachten ist, dass auf die Bild- und Dateiausgabe in diesem Anwendungsfall verzichtet wurde, um eine tiefgreifendere Evaluation der Webseitenausgabe zu ermöglichen. Nach der Analyse der Testversion wurde den Probanden ermöglicht, ihre Einschätzung über einen Fragebogen kund zu tun, um so Erkenntnisse über die Nutzerzufriedenheit des Prototyps zu gewinnen. Diese Fragen lagen sowohl in geschlossener als auch offener Form vor. Somit wurde zum einen eine strukturierte Datenerfassung ermöglicht, wie auch Raum für individuelle Bemerkungen geschaffen. Diese erfassten Daten zur Nutzerzufriedenheit können also als abhängige Variable gesehen werden, während es sich bei dem Prototyp um die unabhängige Variable dieses Versuchs handelt.

Apparatus und Stimuli

Der konkrete zur Verfügung gestellte Anwendungsfall der Anwendung wurde mit Hilfe dreier verschiedener Elemente dem Probanden zugänglich gemacht. Hierfür gilt es vorausgehend noch einmal zu erwähnen, dass diese Studie online als Selbstexperiment stattfand, weswegen bestimmte Ausführungen eine besondere Ausführlichkeit benötigten, um keinen Raum für etwaige Fragen offenzulassen. So handelt es sich bei dem ersten dem Nutzer präsentierten Gegenstand um eine informative *README-Datei*, welche Hinweise zum Kontext der Nutzung des Prototyps in diesem spezifischen Beispielszenario darlegte. Des Weiteren liefert diese Datei eine Einweisung zu allen folgenden Elementen des Prototyp-Tests. Der darauffolgende Teil dient der Repräsentation der Eingabe in den Prototypen und liegt in Form eines Bildes einer Jupyther-Notebook-Zelle vor. Diese nutzt alle verfügbaren Eingabewege der Datensätze in den Prototypen und stößt zum Abschluss die Generierung der Webseitenausgabe an. Es wurde auf eine eigenständige Nutzung der Prototypeingabe durch den Nutzer verzichtet, um zum einen mögliche Probleme bei der Selbststudie zu verhindern. Zum anderen liegt der Prototyp, wie oben erwähnt 5.2.1, bisher nicht in einer Pip-Paket-Version vor, welche auf herkömmliche Weise installiert werden kann. Als letzter Teil wurde die generierte Website als interaktive HTML-Datei zur Verfügung gestellt, welche durch den Nutzer frei erkundet werden kann.

Vorgehen

Den ausgewählten Testpersonen wurde der oben beschriebene Apparatus in Form eines gezippten Ordners gesendet und somit die Testnutzung der Anwendung ermöglicht. Für die darauffolgenden Fragen lag dem Ordner ergänzend eine weitere HTML-Datei bei, über welche diese beantwortet werden konnten. Alle Fragen lagen entweder als geschlossene Bewertung einer Aussage auf einer Skala von 0 bis 6 oder als offene Freitextfrage vor. Beispiele für beide Arten lassen sich der Abbildung 27 entnehmen. Als Ausnahme findet sich im unten genauer beschriebenen Ausgabeteil des Fragebogens eine Frage, in welcher eine Benotung verschiedener Bereiche des Prototyps vorgenommen werden soll. Thematisch lassen sich zu Beginn des Fragebogens allgemeine, das Vorwissen der Probanden betreffende Fragen identifizieren.

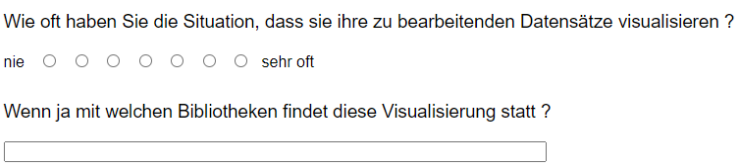


Abbildung 27: Fragenstruktur

Beispielfragen zum allgemeinen Teil:

- Wie erfahren sind Sie im Umgang mit Python? (Geschlossene Frage)
- Wie erfahren sind Sie im Umgang mit Datenvisualisierung in Python? (Geschlossene Frage)

Darauf folgt eine Kurzversion des *User Experience Questionnaire*, welcher Schlüsse zur Benutzererfahrung der evaluierten Systeme ermöglicht und somit eine wissenschaftlich fundierte Methode zur Bewertung von PP-Vis bietet. Beispielfragen zum UEQ-Teil:

- Geben Sie hier nun Ihre generelle Einschätzung des Prototyps ab.
- behindernd bis unterstützend (Geschlossene Frage)
- langweilig bis spannend (Geschlossene Frage)

Um indessen auch spezifische Aspekte der Anwendung des Prototyps zu evaluieren, wurden im Zuge der nächsten zwei thematischen Bereiche des Fragebogens, sowohl geschlossene als auch offene Fragen zur Eingabe und Ausgabe gestellt. Beispielfragen zum Eingabe und Ausgabe Teil:

- Was würden Sie an dieser Einbindung und Verwendung des Pakets verbessern? (Offene Frage)
- Erschien ihnen die Übergabe der Daten in Methodenform an das Paket schlüssig? (Geschlossene Frage)
- Wie intuitiv erschien ihnen die Navigation auf der Website? (Geschlossene Frage)
- Wie würden Sie die gewählten Visualisierungen des Prototyps verbessern? (Offene Frage)

Im die Ausgabe des Prototypen betreffenden Fragebogenbereich findet sich hier außerdem die im Vorfeld angedeutete Frage, die der Benotung der verschiedenen Bereiche der Webseitenausgabe dienen soll. Im Zuge dessen können die in Kapitel 5.2.3 definierten Bereiche, sowie die speziellen Vorverarbeitungsvisualisierungen zu Nullstellen und Ausreißern, mit den Noten 1 bis 6 bewertet werden. Abschließend folgt noch ein Teil, welcher sonstige Anmerkungen zu der Anwendung zulässt. Beispielfragen zum SSonstige Anmerkungen"Teil:

- Gibt es zusätzliche Funktionen, die Sie sich von so einer Art Prototyp wünschen würden? (Offene Frage)

Teilnehmer

Bei den insgesamt 7 Teilnehmern (2 weiblich, 5 männlich, 0 Non-binär; Durchschnittsalter: 28, Standardabweichung: 8.4) an dieser Studie handelte es sich hauptsächlich um Mitarbeiter eines Software-Unternehmens. Da dieses Unternehmen mitunter in der *KI und Data Science* tätig ist, können die Probanden hervorragend als künftige Anwender des Prototyps gesehen werden und dienen somit optimal zur Evaluation ebenjenes. Die Teilnehmer erhielten keine vorausgehenden Informationen über den Prototypen, seine Funktionsweise und das ihm innen liegenden Anwendungsfeld.

6.2.2 Ergebnisse und Diskussion

UEQ

Wie anhand des Diagramms 28 zu sehen, war die Nutzererfahrung und -zufriedenheit des Prototyps positiv. Jedoch lässt gerade die pragmatische Qualität zu Wünschen übrig, da diese mit Hinblick auf die Zielsetzung der Entwicklung der Anwendung ein wichtiger zu maximierender Faktor sein sollte. Das Ziel ist die Schaffung eines Werkzeugs, welches den Nutzer in seiner Datenanalyse unterstützt und somit vor allem als effizient und nützlich wahrgenommen werden sollte. Jedoch sind auch positive Entwicklungen in der hedonistischen Qualität wünschenswert, denn eine durchweg ästhetische und ansprechende Anwendung überzeugt den Nutzer, die Anwendung als Ersatz seiner eigenen Visualisierungen zu nutzen.

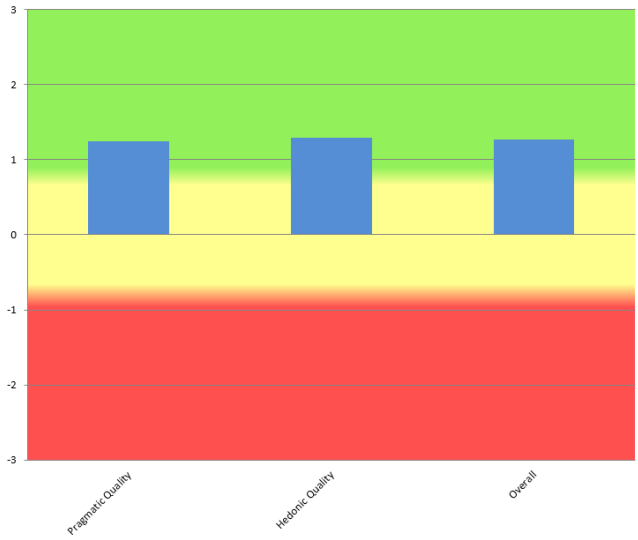


Abbildung 28: Ergebnisse UEQ

Geschlossene Fragen

Die Ergebnisse des allgemeinen Teiles der geschlossenen Fragen, welche durch die Abbildung 29 dargestellt werden, zeigen auf, dass die Probanden im Mittel sowohl ein Vorwissen zu Python und Pandas als auch der Vorverarbeitung von Datensätzen besitzen. Besonders interessant hierbei ist, dass trotz der Erfahrung im Umgang mit diesen Technologien, bei den Fragen zum Umgang mit Datenvisualisierung und der Häufigkeit der Nutzung ebenjener durchschnittlich geringe Werte erkannt werden können. Somit kann die Einschätzung getroffen werden, dass Daten zwar vorverarbeitet werden, eine ebenso häufige Visualisierung dieser jedoch ausbleibt. Dies zeigt wiederholt die oben erläuterte Lücke, die der Prototyp zu schließen versucht, auf.

Die besten Bewertungen erhielt die Eingabe der Daten in den Prototyp, dies verdeutlicht, dass die einfach und ohne großen Aufwand nutzbaren Eingabemöglichkeiten, als ein guter Ansatz gesehen werden können. Ein klarer Unterschied kann hier zwischen der Methoden- und Dekoratorform der Eingabe gesehen werden, welcher die Methodenform klar als schlüssigere Alternative darstellt. Jedoch gilt es hier zu beachten, dass die Dekorator-Methode ein bei Weitem unbekannteres Pythonkonstrukt ist, im Vergleich zur generellen Methodennutzung.

Wird abschließend nun die Bewertungen der Ausgabe von PP-Vis betrachtet, so

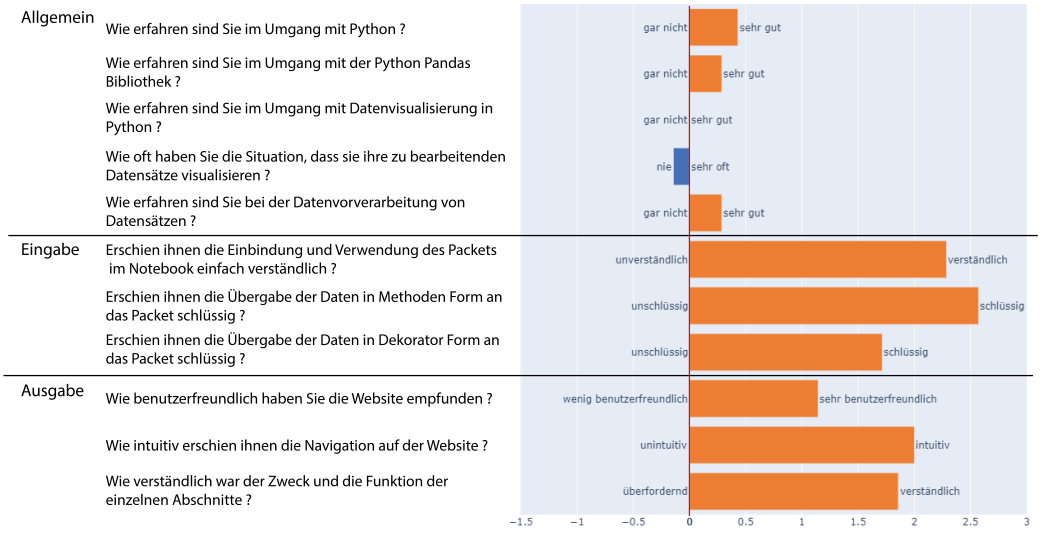


Abbildung 29: Ergebnisse geschlossene Fragen

fällt auf, dass die Steuerung als sehr intuitiv wahrgenommen wurde. Dies könnte mit der Verwendung der Workflowübersicht als zentralen Angelpunkt der Anwendung und der sonstigen, sehr einer herkömmlichen Website entsprechenden Steuerung, in Zusammenhang stehen. Auch die Unterteilung der Website in ihre Abschnitte wurde als zweckdienlich wahrgenommen und deren individuelle Bewertungen werden jetzt im Folgenden betrachtet.

Wie die Abbildung 30 erhielten die Bereiche der Vergleichs- und speziellen Vorverarbeitungsplots, im Mittel die beste Note. Das sind ausgezeichnete Ergebnisse, denn diese Bereiche stellen mit ihrer direkten Darstellung der Veränderung zweier Datensätze eine der zentralsten Funktionalitäten dar, die es durch den Prototypen zu erreichen gab. Die schlechtesten Bewertungen der Datenansicht wurden durch die generellen Plots erzielt, was an dem in einem späteren Kapitel 6.3.2 angedeutetem unübersichtlichen Verhalten der Parallelplots mit bestimmten Attributkompositionen liegen könnte. Eine ebenso im Verhältnis zu den anderen Bereichen schlechte Bewertung erhielt die Workflowübersicht, was zusammen mit den hervorragenden Ergebnissen zur intuitiven Steuerung der Anwendung bedeuten könnte, dass eine visuelle Überarbeitung dieses Bereichs vorzunehmen ist, die Funktionalitäten jedoch beibehalten werden sollten.

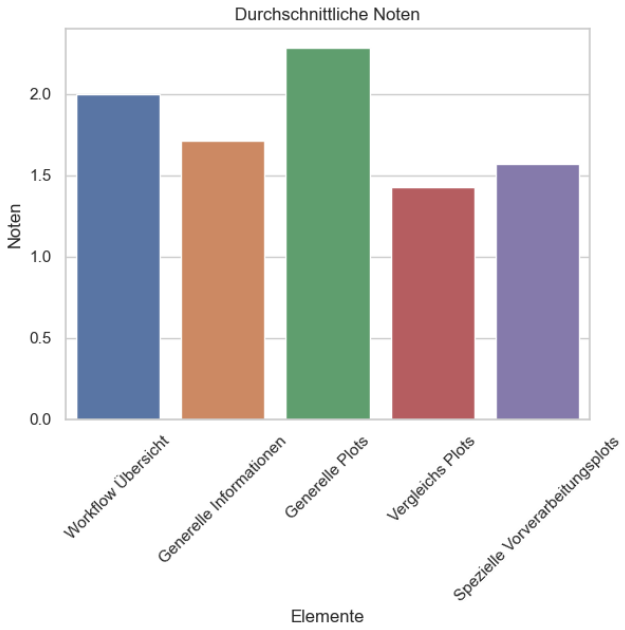


Abbildung 30: Durchschnittliche Noten der Webseitenelemente

Offene Fragen

Verbesserungsansätze zur Einbindung des Pakets, die durch die Probanden getätigt wurden, sind zum einen, dass es sich aktuell noch um eine enorme Black-Box handelt. Diese könnte durch die Möglichkeit mehrerer durch den Nutzer bestimmbarer Parameter verständlicher gestaltet werden. Zum anderen wurde die Funktionalität vorgeschlagen, die PP-Vis Funktionen standardmäßig als *pure functions* anzubieten. Das würde bedeuten, dass durch die Anwendung erzeugte Daten nicht mehr ersetzt, sondern kopiert werden, wenn sie weiterverarbeitet werden. Somit könnte die einfache Verwendung ursprünglicher, in diesem Falle Visualisierungen, gestattet werden. Ein konkretes Beispiel hierfür bietet die oben erläuterte Bibliothek Pandas, welche, wie an dem Codebeispiel 8 zu erkennen, bei der Entfernung von einer Spalte eines *Dataframes* diese nicht in dem ursprünglichen Datenkonstrukt entfernt, sondern über die Rückgabe einer bearbeiteten Kopie. Beide Verbesserungsansätze scheinen schlüssig und können einfach umgesetzt werden, weswegen sie in zukünftigen Forschungen auf jeden Fall eine lohnenswerte Ergänzung von PP-Vis darstellen könnten.

```
    # drop-Methode ohne inplace=True
    df_dropped = df.drop('A', axis=1)

    # Das ursprüngliche DataFrame wird nicht verändert
    print(df)
    # Output:
    #    A  B
    # 0  1  5
    # 1  2  6

    # Ein neues DataFrame wird erstellt
    print(df_dropped)
    # Output:
    #    B
    # 0  5
    # 1  6
```

Codebeispiel 8: Pure functions Beispiel

Bei der Frage zur Verbesserung der Benutzerfreundlichkeit und Intuitivität der Anwendung, wurde eine Anzeige des aktuell ausgewählten Schrittes im Steuerungselement genannt und eine bessere Responsivität der Anwendung für kleine Geräte als wünschenswert gesehen. Des Weiteren wurde der Vorschlag einer Hilfefunktion, welche durch zum Beispiel eine *Hoverfunktion* Beschreibungen zu den verschiedenen Bereichen einblendet, getätigt. Sowie die Verwendung klassischer Navigationselemente. Am häufigsten genannt, bei der Frage nach den besten Aspekten der Visualisierungen des Prototyps, wurden die Vergleichsdiagramme. Dies bestärkt nochmals die vorangegangenen Ergebnisse der geschlossenen Fragen. Zur Verbesserung der Visualisierungen wird in den Antworten vorgeschlagen, bestimmte Diagramme aufgrund bestimmter Datensätze und mit gegebenen Filteroptionen anzuzeigen oder auszublenden. Diese Funktion ist schon in kleinerem Umfang vorhanden, könnte aber auch auf unter anderem die parallel Plots ausgeweitet werden, welche im Rahmen dieser Frage wiederholt kritisiert wurden. Des Weiteren wurde eine gleichmäßige Achsenskalierung, sowie eine Verbesserung der Verständlichkeit der spezifischen Vorverarbeitungsdiagramme empfohlen. Zweiteres könnte durch die weiter oben genannten Hilfefunktionen mit umgesetzt werden. Im Falle zusätzlicher Visualisierungen wünschten sich die Probanden eine Korrelationsmatrix,

sowie eine detaillierte Darstellung von Datentypänderungen. Zusätzliche Funktionen betreffend, wurde die Möglichkeit einer textuellen Zusammenfassung genannt, welche die bestehenden Ausgabearten ergänzen könnte.

Zusammenfassung - Nutzerstudie

Zusammenfassend bot die Nutzerstudie gute Einblicke in bisher erreichte Ziele der Umsetzung der Anwendung und stellte gute Informationen zu möglichen Erweiterungen und Anpassungen des bestehenden Prototyps zur Verfügung. Das Gesamtbild von PP-Vis war positiv und ein Mehrwert durch die Anwendung für die Probanden wurde in jedem Fall wahrgenommen. Vor allem kann das Bedürfnis nach aussagekräftigen, dennoch einfachen und gut dokumentierten Visualisierungen erkannt werden. Vor allem die Visualisierungen, welche direkt die Veränderungen der Datensätze abbildeten, wurden als sehr positiv wahrgenommen und sollten bei weiterer Forschung zu PP-Vis ausgebaut werden.

6.3 Leistungsbewertung

Ergänzend zur Evaluation des Prototyps durch Probanden, soll im folgenden Kapitel eine Bewertung anhand von objektiv feststellbaren Metriken stattfinden. Diese Werte und Einordnungen dienen als erster fest haltbarer Ist-Zustand der Anwendung und können so als Orientierung bei der Weiterentwicklung dienen oder als Vergleichskriterium zu nachfolgenden Versionen von PP-Vis. Hierfür werden in diesem Kapitel Werte zur Laufzeit der verschiedenen Ausgabemethoden präsentiert, sowie Angaben zur Menge abdeckbarer Datensätze und Webbrowser getätigt. Zum Abschluss findet außerdem eine Auswertung des Prototyps hinsichtlich der in Kapitel 2 erwähnten Richtlinien für visuelle Anwendungen zur Analyse von Vorverarbeitungsprozessen statt.

6.3.1 Laufzeit

Für die Analyse der Laufzeit der verschiedenen Ausgabemethoden wurde der Prototyp für verschiedene Mengen an Eingabedatenpunkten getestet. Hierbei zu beach-

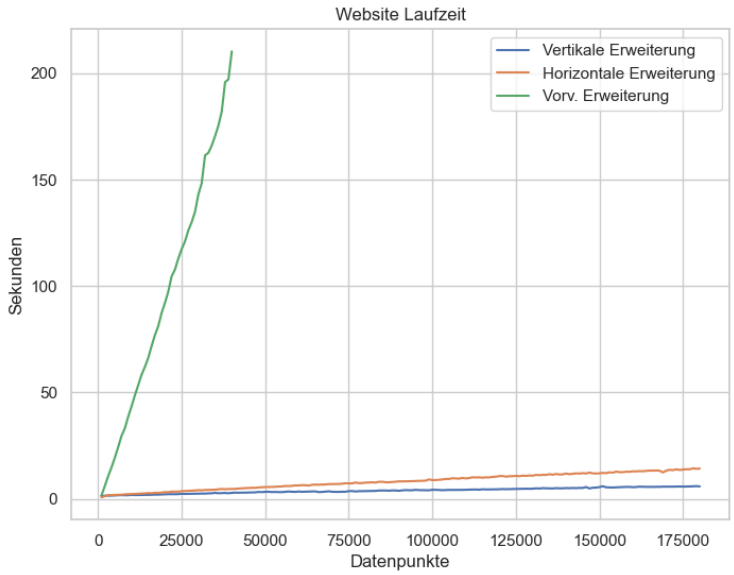


Abbildung 31: Webseitenausgabe Laufzeit

ten ist, dass sich diese Laufzeiten bezüglich der Leistungsfähigkeit des jeweiligen Endgeräts unterscheiden können, die Unterschiede ebenjener sind jedoch repräsentativ für alle Systeme. Sowohl das Einlesen der Daten, als auch Anpassungen und Vorverarbeitungen wurden im Vorfeld getätigt, um Verfälschungen der ermittelten Laufzeit aufgrund dessen zu vermeiden. Die Datei, mit deren Hilfe die Laufzeiten gesammelt wurden, kann dem Anhang 7 entnommen werden. Die Menge an Datenpunkten wurde mittels dreier verschiedener Herangehensweisen erhöht. In einem ersten Versuch wurde durch zusätzliche Datensätze, die Menge an Daten vergrößert. Dies erhöht zum einen die Anzahl zu erstellender Vorverarbeitungsschritte, wie auch die Anzahl zu verarbeitender Datenpunkte. In einem zweiten Test wurde die Menge an Attributen des Testdatensatzes erhöht, um so eine horizontale Erweiterung an Datenpunkten zu erzielen. Abschließend fand in einem letzten Versuch eine vertikale Vergrößerung des Datensatzes durch das Hinzufügen von Datenreihen statt. In jedem Fall wurden immer tausend Datenpunkte ergänzt, also etwa ein zusätzlicher Datensatz mit hundert Datenreihen und zehn Attributen oder zwei neue Datensatzspalten mit fünfhundert Reihen an Daten. Diese Verfahren wurden für Bild-, Daten- und Webseitenausgabe durchgeführt und die Ergebnisse können den Abbildungen 31, 32 und 33 entnommen werden.

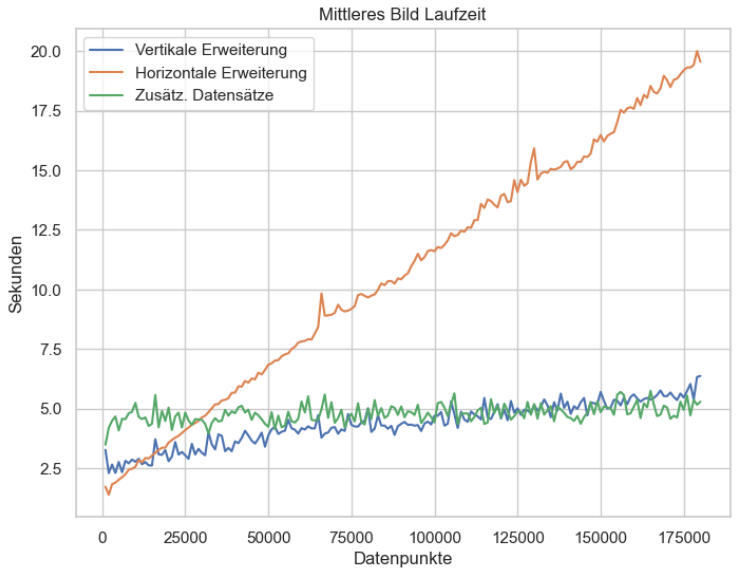


Abbildung 32: Bildausgabe Laufzeit

Beginnend mit der Betrachtung der Abbildung zu den Laufzeiten der Webseitenausgabe, kann erkannt werden, dass sich die Laufzeit im Falle der Erweiterung durch zusätzliche Datensätze sehr viel schneller erhöht, als wenn diese nur um zusätzliche Reihen oder Spalten erweitert wurden. Dies scheint im Hinblick auf die Arbeitsweise der Webseitenausgabe nachvollziehbar, da für jeden zusätzlichen Datensatz zwei komplett neue Contentelemente generiert werden. Hierbei handelt es sich um die Daten- und Vorverarbeitungsansicht, welche eine Vielzahl an Visualisierungen beinhalten. Außerdem findet zwischen zwei Datensätzen auch immer eine Erkennung aller spezifischen Vorverarbeitungen statt. Werden die anderen zwei Laufzeitentwicklungen betrachtet, so fällt auf, dass sich bei der vertikalen Erweiterung die geringste Steigung vorfinden lässt, da diese nur zusätzliche Datenpunkte für ihre Diagramme erhält. Im Falle der horizontalen Erweiterung wird jedoch eine Vielzahl zusätzlicher Diagramme konstruiert, was die leicht erhöhte Laufzeit erklärt. Werden die drei Kurven hinsichtlich ihrer *Big-O-Notation* bewertet, so kann der vertikalen und horizontalen Erweiterung klar ein linearer Aufwand $O(n)$ nachgewiesen werden. Die Erweiterung mit zusätzlichen Datensätzen weist einen quasilinearen Aufwand $O(n \log n)$ auf, was an dem Schreiben des umfangreichen HTML-Dokuments liegen könnte. Es könnte sich jedoch auch nur um einen kurzfristigen

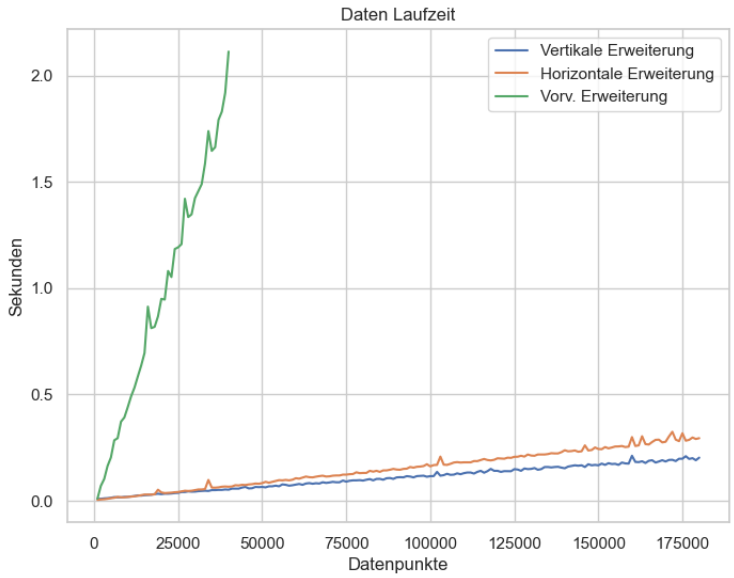


Abbildung 33: Datenausgabe Laufzeit

Trend handeln, welcher sich nach weiteren Messungen wieder einpendelt.

Fast identische Aufwände, jedoch auf einer wesentlich kleineren Skala, weist die Abbildung 33 mit den Laufzeiten zur Datenausgabe auf. Die Laufzeiten betragen an allen Stellen lediglich ein Zwanzigstel der Webseitenausgabe, was darauf hindeutet, dass ein Großteil des Rechenaufwands durch die Erstellung der Grafiken und die Transformation zu Webelementen aufgewendet wird. Durch das sonst jedoch analoge Verhalten lässt sich feststellen, dass auch hier das Schreiben in neue JSON-Objekte und die Erkennung der spezifischen Vorverarbeitungen einen Mehraufwand schaffen. In der Abbildung 32 der Bildausgabe lässt sich als erstes ein klarer Unterschied erkennen, wenn die Laufzeit bei zusätzlich ergänzten Datensätzen betrachtet wird. Erklären lässt sich dieser damit, dass als Methode die Basis-Bildausgabe gewählt wurde, welche jeweils nur Informationen zum Start- und Enddatensatz einer Vorverarbeitung generiert. Somit verursachen zusätzliche Vorverarbeitungsschritte keinen Mehraufwand und es lässt sich nur eine ganz geringe positive Tendenz aufgrund der steigenden und zu visualisierenden insgesamten Veränderungen erkennen. Auch hier verhalten sich die horizontale und vertikale Erweiterung wieder ähnlich zur Webseitenausgabe, aufgrund der veränderten Skala ist außerdem der vorausgehend erläuterte Unterschied hier nochmals besser zu erken-

nen.

6.3.2 Datensatzabdeckung und Webbrowserabdeckung

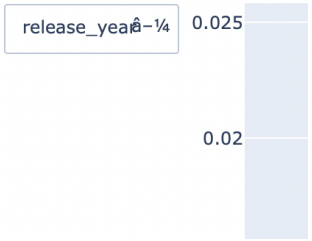


Abbildung 34: Safari SVG Fehler

Um eine Einschätzung bezüglich analysierbarer Datensätze zu erhalten, wurde der Prototyp auf vier der aktuell trendenden Datensätze der Website *Datasets - Trending Datasets* (2023) angewendet. Es wurde jeweils der Datensatz dem Prototyp übergeben, ein Vorverarbeitungsschritt vorgenommen und dann alle möglichen Ausgabearten generiert. Die Datensätze sind dem Anhang 7 zu entnehmen. Bei keinem der Anwendungsfälle konnte eine Fehlermeldung und somit ein Abbruch der Verarbeitung beobachtet werden, wobei die Existenz von durch Grenzfälle bedingtem Fehlverhalten trotzdem nicht ausgeschlossen werden kann. Alle Darstellungselemente verhielten sich, wie erwartet, jedoch können an bestimmten Stellen für unterschiedliche Datensätze Schönheitsfehler wahrgenommen werden. Ein Beispiel hierfür sind die parallelen Diagramme, welche eine Darstellung aller Attribute vornehmen und im Falle zu vieler Reihen eines Datensatzes oder zu langer Klassennamen der Attribute sehr unübersichtlich und schwer verständlich werden. Somit sollten für diese Darstellungen mehr Filter implementiert oder eine alternative Darstellung mit demselben Ziel erwogen werden.

Der zweite in diesem Kapitel vorgenommene Test betrifft verschiedene Webbrowser und hat zum Ziel, eine mindestens fehlerfreie Darstellung auf den bekanntesten festzustellen. Viele Webbrowser unterscheiden sich in ihrem Umgang mit bestimmten CSS-Klassen und im automatischen Festlegen von ebenjenen Klassen aufgrund der HTML-Tag-Selektoren. Ein Beispiel hierfür kann der Abbildung 35 entnommen werden, welche die automatischen hinzugefügten Attribute der zwei Webbrowser Chrome und Firefox darstellt. Zu erkennen ist, dass durch Chrome eine Vielzahl an Werten ergänzt wird, während Firefox darauf verzichtet. Gerade mit Hinblick auf die große Menge an durch Plotly erzeugten Browsercode sollte eine Überprüfung

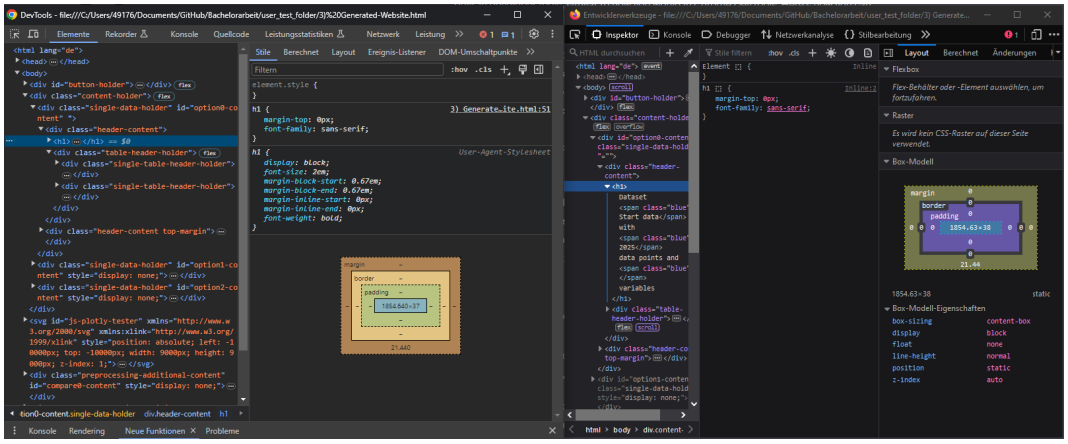


Abbildung 35: Verschiedene automatische Attribute

wichtig sein. Wie anhand der Website *Browser Market Share Worldwide* (2023) zu erkennen, können Chrome, Safari, Edge und Firefox als die vier größten Webbrowser identifiziert werden und wurden für die Darstellung der durch den Prototypen generierten Website getestet. Die Ladezeiten der generierten Website der Nutzerstudie in jedem Browser können der Tabelle 6 entnommen werden. Die Darstellung

Webbrowser	**Ladezeit**
Chrome	9.28s
Firefox	7.71s
Edge	15.17s
Safari	4.63s

Tabelle 6: Ladezeiten Webbrowser

der Website ist mit Chrome und Edge unproblematisch und es sind keine Fehler zu erkennen. Große Probleme verursachte die Darstellung im Firefox-Webbrowser, welcher in manchen Fällen eine falsche Interpretation der Diagrammhöhe vornahm und diese somit in viel zu großen Containern darstellte, was eine Überlagerung von Inhalten und der langgezogenen Darstellung mancher Grafiken zur Folge hatte. Im Falle von Safari funktionierte alles, bis auf die Darstellung der Plotlydiagramme eigenen SVG's (Abbildung 34).

Richtlinie	Bedeutung	Prototyp Umsetzung
G1: Unified	Integration with the most used tools for data analysis.	–
G2: Large Scale	Ability to work with scenarios dealing with huge volumes of data.	✓
G3: Metadata.	Ability to generate informative summaries of the preprocessing activities.	✓
G4: Data-Mining	Use of data mining methods to support preprocessing activities.	✗
G5: Statistics	Use of statistical methods to generate a detailed description of the data and to support preprocessing activities.	✓
G6: Comparison	Ability to compare the data prior and after transformations and the impacts of the preprocessing decisions.	✓
G7: Recommendation	Use of recommendation systems to propose visualizations.	✓
G8: Template	Ability to generate automatically initial visualizations or basic templates.	✓
G9: Interaction	Use of visualization interaction techniques to support flexible data exploration.	✓

Tabelle 7: Erfüllung der PrAVA-Richtlinien

6.3.3 PrAVA Richtlinien

Im letzten Teil der Leistungsbewertungen wird der Prototyp nun entsprechend der in Kapitel 2 angedeuteten Richtlinien bewertet. Diese wurden im Zuge einer Erweiterung des visuellen Analyseprozesses durch die Vorverarbeitung von Milani et al. (2021) geschaffen und können hier als guter Maßstab zur Einordnung der Anwendung dienen. Die Abbildung 7 legt die Richtlinien dar und enthält gleichzeitig eine Bewertung des Prototyps aufgrund dieser. Erkennbar ist, dass die erste Richtlinie, nach welcher eine Integration mit den meistgenutzten Datenanalysewerkzeugen möglich sein soll, nur teilweise erfüllt ist. Da der Fokus von PP-Vis auf Python liegt, ist eine Verwendung mit Programmiersprachen wie R, zwar möglich, jedoch kompliziert umsetzbar. Diese Sprachen spielen jedoch eine wichtige Rolle neben Python für das Feld der Data-Science, weswegen die Anwendung für diese Richtlinie nur eine neutrale Wertung erhält. Sowohl der Umgang mit großen Datenmengen, wie anhand der Laufzeitevaluation in Kapitel 6.3.1 ersichtlich, und die Ausgabe von Metadaten durch die Datenausgabe des Prototyps, sind möglich und können somit als erfüllt gesehen werden. Im Gegensatz dazu sind *Data-Mining* Prozesse durch die Anwendung nicht nutzbar und können nur visualisiert werden. Wieder erfüllbar durch die Anwendung sind die Richtlinien fünf und sechs, denn sowohl detaillierte Beschreibungen der Datensätze, als auch Vergleiche der transformierten Daten sind erzielbar. Abschließend können auch die letzten drei Punkte als eingehaltene Richtlinien gesehen werden, denn zum einen werden für spezifische Daten automatisch passende Visualisierungen erzeugt und somit gleichzeitig umgesetzt, wie auch vorgeschlagen. Zum anderen ist die Interaktivität der Visualisierungen eine grundlegende Anforderung an die Anwendung und daher umgesetzt.

6.3.4 Zusammenfassung - Leistungsbewertung

Abschließend legt die Leistungsbewertung in jedem Falle eine sehr gelungene erste Umsetzung des Prototypen dar. Vor allem hinsichtlich der durch Milani et al. (2021) definierten Richtlinien, erfüllt die Anwendung fast alle definierten Regeln und kann nach dieser Definition, als ein wertvolles Werkzeug für die visuelle Da-

tenanalyse gesehen werden. Die Laufzeiten, wie auch die in 6.3.2 getesteten Abdeckungen, sind für den Rahmen eines ersten Prototyps annehmbar, aber gerade die Laufzeit hinsichtlich mehrerer Vorverarbeitungsschritte sollte auf jeden Fall noch weiter verbessert werden. Das Ziel sollte sein, eine möglichst schnelle Generierung der Visualisierungen zu garantieren, um so keine großen Blockaden im Vorverarbeitungsprozess zu erzeugen. Durch eine detaillierte Laufzeitanalyse der einzelnen Bereiche von PP-Vis könnten etwa spezifische Problemstellen identifiziert werden und diese behoben. Auch eine verfrühte Ausgabe und die Generierung zusätzlicher Daten im Hintergrund wäre denkbar. Die Erweiterung der Browserabdeckung kann vorerst noch vernachlässigt werden, um keine voreiligen breiten Anpassungen von Darstellungen zu erzielen, bevor genau definiert wurde, ob diese überhaupt verwendet werden.

7 Zusammenfassung

Nach der Vorstellung aller bisher implementierten Teile von PP-Vis und der Erläuterung aller dazu relevanten Themen lässt sich hier nun ein abschließendes Résumé ziehen. Die in Kapitel 3.2 präsentierten Anforderungen konnten größtenteils durch die in Kapitel 5 erläuterte implementierte Anwendung umgesetzt werden. So wurde eine interaktive Visualisierung von Vorverarbeitungsschritten durch PP-Vis möglich gemacht, welche darüber hinaus eine gewisse Generalität bietet. Diese zeichnet sich primär durch die vielen verschiedenen Eingabedaten abhängigen Diagramme aus, sowie durch die Möglichkeit vieler verschiedener Ausgabeformen ebenjener. Auch die Anforderung nach einer Zugänglichkeit des Prototyps wurde erfüllt und so kann er in seiner bestehenden Form einfach in klassische Python Vorverarbeitungsprozesse integriert werden und im Rahmen dieser ohne viel Code visuelle Ausgaben erzeugen. Diese Ausgaben zeigen eine starke Diversität auf und unterscheiden sich vordergründig in ihrem Verwendungszweck wie auch Abstraktionsgrad. Wie geplant ist PP-Vis modular erweiterbar und kann ergänzende Informationen zu spezifischen Vorverarbeitungsschritten, ergänzend zu den allgemeinen Informationen, darstellen. Abschließend wurde der Prototyp durch zwei Evaluationen getestet und konnte somit erste Einblicke in seine Leistung geben, wie auch Einschätzungen durch Probanden.

Ebenjene ließen auch erste Schwächen der bisher bestehenden Anwendung erkennen, welche sich hauptsächlich in einer zu hohen Laufzeit, wie auch dem fehlenden Feinschliff bestimmter Komponenten und dem Einsatzzwecke in bestimmten Nutzungsumgebungen widerspiegelten. Zu beachten ist weitergehend, dass PP-Vis in seinem aktuellen Stand nicht als finalisierte nutzbare Anwendung gesehen werden kann. Dafür würden zusätzliche Tests und Anpassungen der Architektur der Anwendung benötigt, für welche die Grundstruktur jedoch hier geschaffen wurde.

Auch die Aussagekraft der umgesetzten Nutzerstudie kann als fragwürdig angesehen werden, da nur eine kleine Menge an Probanden befragt wurde. Die Ergebnisse können jedoch trotzdem als erste gute Einschätzung der geplanten Anwendung gesehen und im Rahmen einer darauf aufbauenden Studie weiter Daten gesammelt werden.

Dennoch kann die Forschungsfrage nach der Realisierbarkeit einer Anwendung für die visuelle und interaktive Erkundung von Datenänderungen in Vorverarbeitungsprozessen, durch die oben ausgeführten Kapitel als beantwortet gesehen werden. Und die in diesem Rahmen erhaltenen Evaluationsdaten können für weitere Forschung in dieser Richtung dienen. Die Realisierung weiterer die Anforderungen erfüllenden Komponenten, sowie deren Bewertung im Gesamtkontext der Implementierung, kann als nächster Anknüpfungspunkt zu dieser Arbeit gesehen werden. Basierend auf schon bestehender Systemarchitektur könnte PP-Vis um weitere Visualisierungen und Möglichkeiten der interaktiven Erkundung dieser erweitert werden. Oben genannte Korrelationsmatrizen oder *Heatmaps* wären hierfür nennenswerte Beispiele. Im gleichen Schritt könnten außerdem die Erkennung und Erstellung von vorverarbeitungsspezifischem Content erweitert werden, wobei es hier zu beachten gilt, dass dieser Schritt womöglich nie als abgeschlossen zählen wird. Es könnten auch Funktionalitäten getestet werden, welche über Visualisierungen hinausgehen. So lägen Optionen wie die Animation von Darstellungen, der Schritt in den dreidimensionalen Raum und alternative Ausgabeformen wie Text und Video hier nahe. Auch eine Konkretisierung schlussendlich benötigter Komponenten der Anwendungen und eine Optimierung dieser können als weitere Schritte der Forschung gesehen werden. So kann zum Beispiel das Verbessern der Laufzeit durch eine Optimierung der Nutzung der schon verwendeten Technologien wie Pandas einen Folgeschritt darstellen. Oder der Vergleich von Alternativen zur benutzten Plotly Visualisierungsbibliothek kann neue Möglichkeiten bezüglich Leistung und Darstellung bieten. In weiterer Ferne lägen dann die Weiterentwicklung zur Nutzung von PP-Vis in anderen Softwareumgebungen wie *R* oder die Anpassung der *Responsivness* für eine breite Menge an Endgeräten. Hier inbegriffen wäre die Tes-

tung der verschiedenen Ausgabeformen, für alle gängigen Möglichkeiten zur Darstellung ebenjener. Genau diese Breite der Thematik an vielen Stellen, kann als eine ihrer größten zu überwindenden Hürden gesehen werden, denn sowohl die Unmengen verschiedener Endgeräte und Darstellungssoftware. Aber auch die Vielzahl an Vorverarbeitungsschritten, Arten von Datensätzen und Optionen an Diagrammen verursachen eine große Anzahl zu treffender Abwägungen und abzudeckender Randfälle.

So kann zum Schluss festgehalten werden, dass der hier präsentierte Prototyp als erster Schritt in dieser umfassenden Thematik dient und somit zwar eine Grundidee festigt, jedoch auch Raum zum Ausbau bietet. Das Ziel in folgenden Arbeiten sollte es sein, ergänzend auf PP-Vis aufzubauen und weitere Facetten der Datenvorverarbeitung visuell zu unterstützen, um dieses bedeutsame Forschungsgebiet zugänglicher und übersichtlicher zu gestalten.

Literaturverzeichnis

Bernard, J., Hutter, M., Reinemuth, H., Pfeifer, H., Bors, C. & Kohlhammer, J. (2019). Visual-interactive preprocessing of multivariate time series data. In *Computer graphics forum* (Bd. 38, S. 401–412).

Bors, C., Gschwandtner, T. & Miksch, S. (2019). Capturing and visualizing provenance from data wrangling. *IEEE computer graphics and applications*, *39* (6), 61–75.

Browser market share worldwide. (2023, Okt). StatCounter. Zugriff auf `https://gs.statcounter.com/browser-market-share#monthly-202209-202309-bar`

Chen, C., Yuan, J., Lu, Y., Liu, Y., Su, H., Yuan, S. & Liu, S. (2020). Oodanalyzer: Interactive analysis of out-of-distribution samples. *IEEE transactions on visualization and computer graphics*, *27* (7), 3335–3349.

Chen, M., Mao, S. & Liu, Y. (2014). Big data: A survey. *Mobile networks and applications*, *19*, 171–209.

Chen, P., Plale, B., Cheah, Y.-W., Ghoshal, D., Jensen, S. & Luo, Y. (2012). Visualization of network data provenance. In *2012 19th international conference on high performance computing* (S. 1–9).

Dabbas, E. (2021). *Interactive dashboards and data apps with plotly and dash: Harness the power of a fully fledged frontend web framework in python–no javascript required*. Packt Publishing Ltd.

Datasets - trending datasets. (2023, Okt). Google LLC. Zugriff auf `https://www.kaggle.com/datasets?topic=trendingDataset`

Davenport, T. H. & Patil, D. (2012). Data scientist. *Harvard business review*, *90* (5), 70–76.

Dhruv, A. J., Patel, R. & Doshi, N. (2021). Python: the most advanced programming language for computer science applications. *Science and Technology Publications, Lda*, 292–299.

Embarak, D. O., Embarak & Karkal. (2018). *Data analysis and visualization using python*. Springer.

Fahad, S. A. & Yahya, A. E. (2018). Big data visualization: Allotting by r and python with gui tools. In *2018 international conference on smart computing and electronic enterprise (icscee)* (S. 1–8).

Grafberger, S., Stoyanovich, J. & Schelter, S. (2021). Lightweight inspection of data preprocessing in native machine learning pipelines. in 11 th conf. on innovative data sys. research. *Online Proceedings*.

Gschwandtner, T., Aigner, W., Miksch, S., Gärtner, J., Kriglstein, S., Pohl, M. & Suchy, N. (2014). Timecleanser: A visual analytics approach for data cleansing of time-oriented data. In *Proceedings of the 14th international conference on knowledge technologies and data-driven business* (S. 1–8).

Hardin, M., Hom, D., Perez, R. & Williams, L. (2012). Which chart or graph is right for you? *Tell Impactful Stories with Data. Tableau Software*.

Installing packages - python packaging user guide 2023. (2023, Sep). The Python Software Foundation. Zugriff auf `https://packaging.python.org/en/latest/tutorials/installing-packages/`

Jolly, K. (2018). *Hands-on data visualization with bokeh: Interactive web plotting for python using bokeh*. Packt Publishing Ltd.

Kandel, S., Parikh, R., Paepcke, A., Hellerstein, J. M. & Heer, J. (2012). Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Proceedings of the international working conference on advanced visual interfaces* (S. 547–554).

Keim, D., Kohlhammer, J., Ellis, G. & Mansmann, F. (2010). *Mastering the information age solving problems with visual analytics*. Eurographics Association.

Kitchin, R. (2014). The data revolution: Big data, open data, data infrastructures and their consequences. *The Data Revolution*, 1–240.

Krause, J., Perer, A. & Stavropoulos, H. (2015). Supporting iterative cohort construction with visual temporal queries. *IEEE transactions on visualization and computer graphics*, *22* (1), 91–100.

Lam, H., Bertini, E., Isenberg, P., Plaisant, C. & Carpendale, S. (2011). Empirical studies in information visualization: Seven scenarios. *IEEE transactions on visualization and computer graphics*, *18* (9), 1520–1536.

McKinney, W. et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th python in science conference* (Bd. 445, S. 51–56).

Milani, A. M. P., Loges, L. A., Paulovich, F. V. & Manssour, I. H. (2021). Prava: Preprocessing profiling approach for visual analytics. *Information Visualization*, *20* (2-3), 101–122.

Podrzaj, P. (2019). A brief demonstration of some python gui libraries. In *Proceedings of the 8th international conference on informatics and applications icia2019* (S. 1–6).

Razzaq, H. S. & Zaibidi, N. Z. (2023). Improving inventory management decisions by outliers detection and elimination. *Central European Management Journal*, *31* (1), 597–603.

Rousseeuw, P. J. & Hubert, M. (2011). Robust statistics for outlier detection. *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, *1* (1), 73–79.

Sacha, D., Stoffel, A., Stoffel, F., Kwon, B. C., Ellis, G. & Keim, D. A. (2014). Knowledge generation model for visual analytics. *IEEE transactions on visualization and computer graphics*, *20* (12), 1604–1613.

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N. & Diakopoulos, N. (2016). *Designing the user interface: strategies for effective human-computer interaction*. Pearson.

Siddiqui, T., Alkadri, M. & Khan, N. A. (2017). Review of programming languages and tools for big data analytics. *International Journal of Advanced Research in Computer Science*, *8* (5).

Simmhan, Y. L., Plale, B., Gannon, D. et al. (2005). A survey of data provenance techniques. *Computer Science Department, Indiana University, Bloomington IN*, *47405*, 69.

Stančin, I. & Jović, A. (2019). An overview and comparison of free python libraries for data mining and big data analysis. In *2019 42nd international convention on information and communication technology, electronics and microelectronics (mipro)* (S. 977–982).

Szoka, K. (1982). A guide to choosing the right chart type. *IEEE Transactions on Professional Communication* (2), 98–101.

Tufte, E. R. (2001). *The visual display of quantitative information* (Bd. 2). Graphics press Cheshire, CT.

VanderPlas, J., Granger, B., Heer, J., Moritz, D., Wongsuphasawat, K., Satyanarayan, A., ... Sievert, S. (2018). Altair: interactive statistical visualizations for python. *Journal of open source software*, *3* (32), 1057.

Vartak, M. & Madden, S. (2018). Modeldb: Opportunities and challenges in managing machine learning models. *IEEE Data Eng. Bull.*, *41* (4), 16–25.

Wickham, H. & Wickham, H. (2016). *Data analysis*. Springer.

Yi, J. S., ah Kang, Y., Stasko, J. & Jacko, J. A. (2007). Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics*, *13* (6), 1224–1231.

Yuan, J., Chen, C., Yang, W., Liu, M., Xia, J. & Liu, S. (2021). A survey of visual analytics techniques for machine learning. *Computational Visual Media*, *7*, 3–36.

Erklärung zur Urheberschaft

Ich habe die Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie alle Zitate und Übernahmen von fremden Aussagen kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt.

Die vorgelegten Druckexemplare und die vorgelegte digitale Version sind identisch.

Regensburg, 2023-10-16

Unterschrift

Erklärung zur Lizenzierung und Publikation dieser Arbeit

Name: Maximilian Schmerle

Titel der Arbeit: *Visualisation and interactive exploration of data changes in data engineering workflows*

Hiermit gestatte ich die Verwendung der schriftlichen Ausarbeitung zeitlich unbegrenzt und nicht-exklusiv unter folgenden Bedingungen:

- ☐ Nur zur Bewertung dieser Arbeit
- ☐ Nur innerhalb des Lehrstuhls im Rahmen von Forschung und Lehre
- ☒ Unter einer Creative-Commons-Lizenz mit den folgenden Einschränkungen:
  - ☒ BY – Namensnennung des Autors
  - ☐ NC – Nichtkommerziell
  - ☐ SA – Share-Alike, d.h. alle Änderungen müssen unter die gleiche Lizenz gestellt werden.

(An Zitaten und Abbildungen aus fremden Quellen werden keine weiteren Rechte eingeräumt.)

Außerdem gestatte ich die Verwendung des im Rahmen dieser Arbeit erstellten Quellcodes unter folgender Lizenz:

- ☐ Nur zur Bewertung dieser Arbeit
- ☐ Nur innerhalb des Lehrstuhls im Rahmen von Forschung und Lehre
- ☐ Unter der CC-0-Lizenz (= beliebige Nutzung)
- ☒ Unter der MIT-Lizenz (= Namensnennung)
- ☐ Unter der GPLv3-Lizenz (oder neuere Versionen)

(An explizit mit einer anderen Lizenz gekennzeichneten Bibliotheken und Daten werden keine weiteren Rechte eingeräumt.)

Ich willige ein, dass der Lehrstuhl für Medieninformatik diese Arbeit – falls sie besonders gut ausfällt - auf dem Publikationsserver der Universität Regensburg veröffentlichen lässt.

Ich übertrage deshalb der Universität Regensburg das Recht, die Arbeit elektronisch zu speichern und in Datennetzen öffentlich zugänglich zu machen. Ich übertrage der Universität Regensburg ferner das Recht zur Konvertierung zum Zwecke der Langzeitarchivierung unter Beachtung der Bewahrung des Inhalts (die Originalarchivierung bleibt erhalten).

Ich erkläre außerdem, dass von mir die urheber- und lizenzrechtliche Seite (Copyright) geklärt wurde und Rechte Dritter der Publikation nicht entgegenstehen.

- ☒ Ja, für die komplette Arbeit inklusive Anhang
- ☐ Ja, für eine um vertrauliche Informationen gekürzte Variante (auf dem Datenträger beigefügt)
- ☐ Nein
- ☐ Sperrvermerk bis (Datum):

Regensburg, 2023-10-16

Unterschrift

Inhalt des beigefügten Datenträgers

`/1_Ausarbeitung`	Die schriftliche Ausarbeitung als PDF
`/2_Prototyp`	Quellcode des Prototypen als Pip-Paket
`/3_Anwendungsfall`	Alle für den in Kapitel 6.1 vollführten Test verwendete Notebooks und die generierten Ausgaben
`/4_Nutzerstudie/Rohdaten`	Rohdaten der Studie im JSON-Format + Alle für die in Kapitel 6.2 verwendeten Dateien und die dadurch erhaltenen Ergebnisse
`/5_Leistungsbewertung`	Notebook zum in Kapitel 6.3.1 vollzogenen Laufzeittest und Datensatzabdeckungstest der Anwendung
`/6_Bilder und Video`	Bilder und Videos des Prototypen