Vergleichende Analyse
Python [Jupyter Notebook] versus Power BI

Arzum Kavak, Daniel Weißenberger, Jonas Sigmund, Salih Kelmendi

Mai 2024

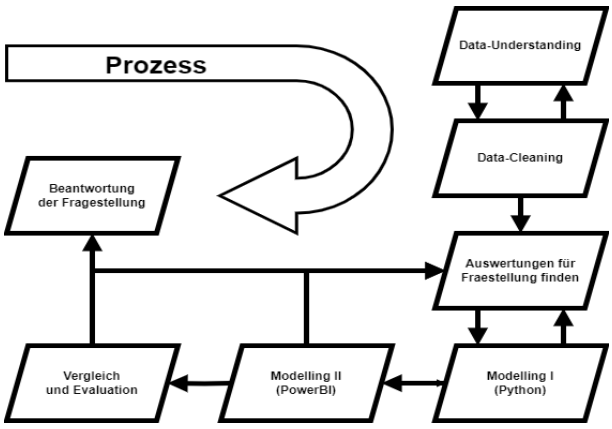
Inhaltsverzeichnis

1 Abstrakt

Die nachfolgende Ausarbeitung behandelt die grundlegenden Möglichkeiten, Einschränkungen und Unterschiede zwischen der Programmiersprache Python und der Business-Intelligence-Plattform Power BI für datengetriebene Auswertungen. Dabei soll insbesondere die Code-freie Benutzung von Power BI mit Python verglichen werden, um die unterschiedlichen Vorgehensweisen besser hervorzuheben. Mithilfe eines zuvor ausgewählten, umfassenden Datensatzes werden verschiedene und unterschiedlich komplexe Grafiken implementiert und visualisiert. Im Genaueren wird untersucht, welche Erfahrungen für die Verbraucher und die Entwickler hinsichtlich einer identischen Datenauswertung in beiden Tools entstehen. Ziel dieser Analyse ist die faire und begründete Beurteilung beider Softwarelösungen.

2 Einleitung

Um aus dem Datensatz die gewünschte Visualisierung zu erstellen, wird im Folgenden der CRISP-DM-Ansatz genutzt. CRISP-DM steht dabei für den Cross-Industry Standard Process for Data Mining und stellt ein strukturiertes Vorgehensmodell für Data-Mining-Projekte dar. Dieser wird häufig zur Analyse von Daten und der anschließenden Entwicklung von Modellen und Grafiken verwendet. Der CRISP-DM Ansatz beginnt mit dem *Business Understanding*. In dieser Phase werden (geschäftliche) Anforderungen identifiziert und spezifische Data-Mining-Fragen gebildet, um sicherzustellen, dass die Analyseergebnisse relevant sind. Es folgt das *Data Understanding*, hierbei werden Daten gesammelt und einer ersten Analyse unterzogen, um wichtige Merkmale und Muster zu erkennen. In der Phase der *Data Preparation* werden die Daten durch Säuberung, Auswahl und Transformation für die Analyse vorbereitet. Anschließend werden in der *Modeling*-Phase verschiedene statistische und maschinelle Modelle entwickelt, implementiert, getestet und bewertet, um optimale Lösungen zu identifizieren. Die *Evaluation* überprüft, ob das Modell den Zielen entspricht und nimmt gegebenenfalls Anpassungen durch Repetition der vorangegangenen Schritte vor. Die abschließende Phase *Deployment*, ist in unserem Kontext untergeordnet, da unsere Modelle nicht in bestehende (Geschäfts-) Prozesse integriert werden. Für die nachfolgende Analyse wird nun diese abgeleitete Variante des beschriebenen CRISP-DM Ansatzes verwendet:



Dieser Ansatz gewährt das beschriebene, strukturierte und flexible Vorgehen für die Analyse beider Tools und ist somit optimal für den Vergleich zwischen Python und Power BI geeignet. Obwohl das Thema Terrorismus zweifellos ein schwerwiegendes, komplexes und oft belastendes Forschungsgebiet darstellt, bietet es für den Zweck dieser Analyse eine hervorragende Grundlage. So ist es möglich, bei respektvollem Umgang mit diesem Thema und einem fokussiertem Blick auf das übergeordnete Ziel, gute Ergebnisse durch diesen Datensatz zu erzielen.

3 Datensatz

Die Global Terrorism Database (GTD) ist eine umfassende Ereignisdatenbank, die mehr als 200.000 Aufzeichnungen von weltweiten terroristischen Angriffen seit 1970 enthält. Sie wird vom National Consortium for the Study of Terrorism and Responses to Terrorism (START) und der University of Maryland verwaltet. Die GTD ist ein entscheidendes Instrument für Forscher und Analysten, um Muster, Ursachen und Konsequenzen des Terrorismus' zu untersuchen und zu verstehen. Die Datenbank ist öffentlich zugänglich, sie unterliegt jedoch den Nutzungsbedingungen der Endbenutzer-Lizenzvereinbarung.

3.1 Qualität und Vertrauenswürdigkeit

Die Datensammlung für die Global Terrorism Database (GTD) erfolgt aus einer Kombination von automatisierten Verfahren und manueller Verbesserung. Zu Beginn steht die automatisierte Erfassung und Filterung Tausender täglich veröffentlichter Medienartikel. Die Quellen werden dabei durch den Einsatz wie dem Metabase Application Programming Interface (API) von Lexis Nexis und der API von BBC Monitoring erschlossen. Nach der automatisierten Vorverarbeitung wie Natural Language Processing (NLP) und Machine Learning (ML) wird ein Subset der Artikel manuell durch das GTD-Team überprüft, um Ereignisse zu identifizieren, die den strikten Einschlusskriterien entsprechen. Diese Kriterien definieren Terrorismus als absichtliche Gewaltakte von nichtstaatlichen Akteuren, die bestimmte Ziele verfolgen und außerhalb legitimer Kriegshandlungen liegen müssen. Die Vertrauenswürdigkeit des GTD-Datensatzes wird durch seine umfassende Zitierhäufigkeit in der wissenschaftlichen Literatur und die Verleihung einer Goldauszeichnung auf Kaggle bestätigt. Die Verfügbarkeit eines detaillierten Codebooks unterstützt zudem die Transparenz und Vertrauenswürdigkeit des Datensatzes, indem es Forschern ermöglicht, den Datensatz besser zu verstehen und potenzielle Quellen für Bias oder Fehler zu identifizieren.

3.2 Umfang

Der Datensatz beinhaltet über 180.000 Einträge und umfasst über 100 Spalten die eine Vielzahl von Informationen über jedes Ereignis liefern. Darunter das Jahr, den Monat, den Tag, das Land, die Region, die Stadt, die geographischen Koordinaten und weitere spezifische Details zu jedem Vorfall. Besondere Aufmerksamkeit wird auf die geografischen Daten gelegt.

3.3 Komplexität

Die Datenbank über terroristischer Ereignisse besitzt eine große Komplexität, die sich über verschiedene Ebenen erstreckt. Sie umfasst sowohl zeitliche als auch geografische Aspekte, welche aus der geopolitischen und dynamischen Natur des Terrorismus resultieren. Die zeitliche Komponente basiert auf präzisen Datumsangaben, die saisonale Schwankungen und unterschiedliche historische Ereignisse beinhalten. Dies trägt zur Erklärung der zeitlichen Dynamik des Terrorismus bei und unterstützt die Forschung bei der Entwicklung von zielgerichtete Maßnahmen zum Schutz der Bevölkerung. Die geografische Komponente ermöglicht es hingegen, örtliche Zusammenhänge zu untersuchen und Hotspots zu identifizieren. Die GTD bietet ebenfalls eine Grundlage zur Untersuchung von kriegsbezogenen Aspekten, um die Wechselwirkungen zwischen Terrorismus und bewaffneten Konflikten zu verstehen und Maßnahmen zur Konfliktprävention zu entwickeln. Die Komplexität entsteht insbesondere aus der

Verknüpfung dieser Ebenen. Auch nationale Sicherheitsbedenken durch die lokale Politik oder die lokale Religion nimmt Einfluss auf die Komplexität.

4 Power BI

4.1 Entstehung

Die Entstehung von Power BI ist eng mit der Evolution des Business-Intelligence-Marktes verbunden, der von der zunehmenden Datenflut und dem Bedarf an fortschrittlichen Analysewerkzeugen angetrieben wird. So begann auch Microsoft, eigene BI-Lösungen zu entwickeln, wobei frühe Vorgänger wie SQL Server Reporting Services (SSRS) und Excel grundlegende Analyse- und Berichtsfunktionen boten. Die Entwicklung von Power BI begann mit der Einführung von Power Pivot, Power Query und Power View in Excel 2010, welche leistungsstarke Datenmanipulations- und Visualisierungsfunktionen in die Tabellenkalkulationssoftware integrierten. Der entscheidende Wendepunkt erfolgte 2015 mit der Einführung der eigenständigen Power BI-Plattform, die das Beste aus Excel mit neuen Cloud-basierten Services kombinierte. Dies ermöglichte Unternehmen die nahtlose Integration von Daten aus verschiedenen Quellen, die Erstellung anspruchsvoller Dashboards und Berichte sowie die Durchführung fortgeschrittener Analysen mithilfe von Funktionen wie DAX (Data Analysis Expressions) und M (Power Query Formula Language). Die kontinuierliche Weiterentwicklung von Power BI, einschließlich der Einführung von Funktionen wie Power BI Desktop, Power BI Service und Power BI Embedded, hat die Plattform zu einem Eckpfeiler im BI-Markt gemacht. Die Integration von künstlicher Intelligenz und maschinellem Lernen, sowie die Unterstützung von Big Data-Technologien wie Azure Synapse Analytics haben Power BI zu einer umfassenden Lösung für Datenanalyse und -visualisierung gemacht.

4.2 Einsatz

Die Anwendungsszenarien von Power BI sind äußerst vielfältig und durchdringen zahlreiche Branchen und Geschäftsbereiche. Die Plattform offeriert ein breites Spektrum an Funktionalitäten für Datenanalyse, Visualisierung und Reporting, was Organisationen ermöglicht, ihre Daten effizient zu nutzen und fundierte Entscheidungen zu treffen. Power BI erlaubt die nahtlose Integration von Daten aus diversen Quellen wie relationalen Datenbanken, Cloud-Services, Big-Data-Plattformen und Excel-Dateien. Mittels Datenverbindungen und Gateways können Nutzer Daten in Echtzeit abrufen und aktualisieren, um stets auf dem aktuellen Stand zu sein. Die Funktionen zur Datenmodellierung von Power BI ermöglichen es Anwendern, komplexe Datenstrukturen zu erstellen, um Daten zu organisieren, zu transformieren und zu modellieren. Dies beinhaltet die Verwendung von Abfragesprachen wie SQL und M zur Datenmanipulation sowie die Erstellung von Berechnungen und Metriken mittels DAX (Data Analysis Expressions). In Bezug auf die Datenvisualisierung stellt Power BI eine breite Palette von Visualisierungstypen wie Diagrammen, Karten, Tabellen und Matrixansichten zur Verfügung. Nutzer haben zudem die Möglichkeit, benutzerdefinierte Visuals zu erstellen und in Berichte und Dashboards zu integrieren, um Einblicke zu gewinnen und Erkenntnisse zu kommunizieren. Die Analysefunktionen von Power BI umfassen Funktionen wie Filterung, Sortierung, Gruppierung und Aggregation, die es Nutzern ermöglichen, Daten auf verschiedene Weise zu untersuchen und zu analysieren. Darüber hinaus bietet Power BI erweiterte Analysefunktionen wie Trendanalysen, Vorhersagemodelle und Clusteranalysen für anspruchsvolle Datenanalysen. Power BI bietet eine intuitive Benutzeroberfläche und eine einfache Integration in bestehende Unternehmenssysteme. Die Plattform unterstützt die nahtlose

Integration mit anderen Microsoft-Produkten wie Excel, SharePoint und Dynamics 365, um Daten aus verschiedenen Quellen zu kombinieren und erstellte Auswertungen bereitzustellen.

4.3 Preis

Es gibt drei mögliche Power BI-Varianten. Die kostenlose Variante bietet nur dir Grundfunktion, Berichte zu erstellen. Die zweite Variante *Power BI Pro* kann zusätzlich die Berichte teilen und zur gemeinsamen Benutzung freigeben. In dieser Variante wird der Speicher auf 10GB pro Benutzer begrenzt und diese Verson kostet 9,40€ pro Benutzer pro Monat. Die letzte Variante *Power BI Premium* kostet 18,70€ pro Benutzer pro Monat. Jedoch wird der Speicher auf 100 TB erweitert und es wird zudem eine erweiterte KI zur Verfügung gestellt.

4.4 Erweiterungen

Die Erweiterungen von Power BI sind eine essenzielle Ergänzung der Plattform, welche es Benutzern ermöglichen, die Funktionalität der Software durch zusätzliche Module und Integrationen zu erweitern. Diese Erweiterungen bieten vielfältige Möglichkeiten für die Anpassung und den Ausbau der Analyse- und Visualisierungsfunktionen von Power BI, um den spezifischen Anforderungen und geschäftlichen Szenarien gerecht zu werden. Entwickler gestatten es, maßgeschneiderte Visualisierungskomponenten zu erstellen und in Power BI-Berichte zu integrieren. Diese benutzerdefinierten Visuals können mittels verschiedener Programmiersprachen wie Python, JavaScript und TypeScript, entwickelt werden und bieten eine hohe Flexibilität in Bezug auf Design und Anpassung der Visualisierungen. Zusätzlich erlauben Power BI-Erweiterungen die Integration von externen Datenquellen und Analysetools in die Plattform. Dies beinhaltet die Anbindung an externe Datenbanken, Cloud-Services und Big Data-Plattformen sowie die Einbindung von maschinellem Lernen und künstlicher Intelligenz in Power BI-Berichte und Dashboards. Ein weiterer bedeutender Aspekt sind benutzerdefinierte Funktionen, welche es Anwendern ermöglichen, komplexe Berechnungen und Transformationen durchzuführen, die über die Standardfunktionalität von Power BI hinausgehen. Diese benutzerdefinierten Funktionen können mittels DAX (Data Analysis Expressions) oder M (Power Query Formula Language) erstellt werden und bieten fortgeschrittene Möglichkeiten zur Datenmanipulation und -analyse.

4.5 Hilfestellung

Die Hilfestellungen bieten Unterstützung auf verschiedenen Ebenen, angefangen von Online-Dokumentationen bis hin zu Community-Foren und Support-Diensten. Eine primäre Hilfestellung ist die umfangreiche Online-Dokumentation, welche eine Vielzahl von Informationen zu allen Aspekten der Plattform bereitstellt. Diese Dokumentation umfasst Tutorials, Leitfäden, technische Referenzen sowie Best Practices, die Benutzern dabei helfen, Power BI effektiv zu nutzen und komplexe Probleme zu lösen. Des Weiteren bietet Power BI eine Vielzahl von Schulungsressourcen, darunter Online-Kurse, Webinare und Schulungsmaterialien an, die es Benutzern ermöglichen, ihre Kenntnisse über die Plattform zu vertiefen und neue Analysetechniken zu erlernen. Diese Schulungsressourcen werden von erfahrenen Trainern und Fachleuten bereitgestellt und bieten praxisnahe Einblicke in die Anwendung von Power BI in verschiedenen Geschäftsszenarien. Ein weiteres wichtiges Hilfsmittel sind die Community-Foren und Support-Communitys von Power BI, in welchen Benutzer Fragen stellen, Probleme diskutieren und Ratschläge von anderen Nutzern und Experten erhalten können. Diese Communitys bieten eine lebendige Plattform für den Austausch von Wissen und Erfahrungen und ermöglichen es Benutzern,

von den Erfahrungen anderer zu profitieren und Lösungen für ihre eigenen Herausforderungen zu finden. Jedoch ist anzumerken, dass die Hilfestellung durch generative AI aufgrund der Bedienung durch die Benutzeroberfläche (noch) nicht sonderlich ausgereift ist.

4.6 Sicherheit

Die Sicherheit in Power BI ist ein entscheidender Aspekt, um die Vertraulichkeit und Integrität von Daten zu gewährleisten und unbefugten Zugriff zu verhindern. Power BI bietet eine umfassende Palette von Sicherheitsfunktionen und -mechanismen, die den Schutz sensibler Daten vor unbefugtem Zugriff, Missbrauch und Datenverlust gewährleisten. Trotz dieser Sicherheitsmaßnahmen besteht jedoch das Problem potenzieller Datenlecks, insbesondere bei der Nutzung der Cloud von Microsoft, auf der Power BI basiert. Zudem ist zu bedenken, dass die Daten auf einem von Microsoft betriebenem Server gespeichert werden und Microsoft wahrscheinlich technisch in der Lage wäre, diese Daten einzusehen.

5 Python

5.1 Entstehung

Die Entstehung von Python spiegelt die Wünsche nach einer benutzerfreundlichen, umfangreichen aber leistungsstarken Programmiersprache wider. Ende der 1980er Jahre begann Guido van Rossum am Centrum Wiskunde & Informatica (CWI) in den Niederlanden mit der Entwicklung von Python. Die erste Version von Python wurde 1991 veröffentlicht und markierte den Beginn einer neuen Ära in der Softwareentwicklung. Python wurde gezielt entwickelt, um eine leicht lesbare und vielseitig einsetzbare Sprache derzustellen. Van Rossum und sein Team legten großen Wert auf die Verständlichkeit der Syntax. Dies spiegelt sich in der bewussten Anlehnung an die englische Sprache wider, was die Lesbarkeit des Codes erheblich verbessert. Die Entwicklung von Python wurde stark von der Vorstellung geprägt, eine Programmiersprache zu schaffen, die gleichermaßen für Anfänger und Experten zugänglich ist. Die Python Software Foundation (PSF) spielt eine entscheidende Rolle bei der Förderung und Weiterentwicklung von Python. Die PSF koordiniert die Arbeit an der Sprache und fördert die Interessen der Python-Community. Dies trägt dazu bei, dass Python kontinuierlich verbessert und an die Bedürfnisse der Entwickler angepasst wird.

5.2 Einsatz

Python wird sowohl in der Industrie, in der Forschung sowie von Privarpersonen weltweit eingesetzt und gilt als eine der beliebtesten Programmiersprachen weltweit. Dabei wird Python häufig für die Entwicklung von Webanwendungen, die Datenanalyse, die Automatisierung und das maschinelle Lernen eingesetzt. Durch seine leistungsstarken Bibliotheken, dynamische Typisierung und Unterstützung für verschiedene Programmierparadigmen bietet Python eine solide Grundlage für die Entwicklung innovativer Anwendungen und die Durchführung anspruchsvoller (datenbasierter) Forschungsprojekte.

5.3 Preis

Python ist eine kostenlose und Open-Source-Programmiersprache. Somit ist sie für jeden ohne finanzielle Hürden frei zugänglich. Dennoch können Zusätze wie Libraries oder Schnittstellen durch

beispielsweise API-Aufrufe kostenpflichtig sein. Ein Beispiel ist hierfür die Library *Plotly*, welche eine kostenpflichtige Version für die Nutzung durch Unternehmen anbietet.

5.4 Erweiterungen

Python bietet eine breite Palette von Erweiterungen und Bibliotheken. Dies ermöglicht den Entwicklern, eine Vielzahl von Anwendungen zu entwickeln und komplexe Probleme zu lösen. Diese Erweiterungen decken verschiedenste Bereiche ab, von der Datenanalyse und dem maschinellen Lernen oder der Spieleentwicklung, bis hin zur Webentwicklung und Bildverarbeitung. Durch die Nutzung dieser Erweiterungen und Bibliotheken können Entwickler einfach auf effiziente, optimierte und vorgefertigte Funktionen zurückgreifen.

5.5 Hilfestellung

Die Python-Community umfasst Entwickler aller Erfahrungsstufen und Teilgebiete. Ihre Hilfestellungen umfassen offizielle Dokumentationen, aber auch private Tutorials und Online-Kurse, die dazu beitragen, die Lernkurve zu verringern. Die offizielle Python-Dokumentation ist eine umfassende Ressource, die eine ausführliche Beschreibung der Python-Sprache, ihrer Bibliotheken und ihrer Funktionen bietet. Aber auch Plattformen wie Stack Overflow und das offizielle Python-Forum bieten Möglichkeiten für Entwickler, Fragen zu stellen, Antworten zu geben und Wissen auszutauschen. Zudem gibt es eine Fülle von Büchern und Fachzeitschriften, die sich mit Python und seinen Anwendungsbereichen befassen. Ebenso finden sich auf GitHub zahlreiche Repositorys mit Beispielcode, Bibliotheken und Frameworks, die von der Community entwickelt und geteilt werden. Auch künstliche Intelligenz bietet eine optimale Hilfestellung für Entwickler aller Erfahrungsstufen. Sie kann den Entwickler beim Erstellen, Debuggen und Optimieren von Code unterstützen. Insbesondere Tools wie GitHubs CodePilot oder Devin optimieren diese Erfahrung durch eine direkte Integration in die Entwicklungsumgebung.

5.6 Sicherheit

Python gilt zumeist als eine sichere Programmiersprache. Dennoch müssen potenzielle Sicherheitsrisiken beachtet werden. So besteht eine potenzielle Gefahr darin, dass Entwickler unsicheren Code schreiben, der anfällig für Sicherheitslücken ist. Es ist wichtig, dass Entwickler bewährte Sicherheitspraktiken einhalten und ihren Code robust und gegen potenzielle Angriffe gestalten. Eine weitere potenzielle Gefahr besteht in der Verwendung von fehlerhaften, unsicheren oder bösartigen Bibliotheken. Obwohl die Python-Community eine Vielzahl von hochwertigen und sicheren Bibliotheken bereitstellt, kann es vorkommen, dass ein Entwickler auf entsprechende Bibliotheken zurückgreifen. Es ist wichtig, regelmäßig die Bibliotheken zu aktualisieren und sicherzustellen, sodass nur vertrauenswürdige und gewartete Bibliotheken verwendet werden. Es ist ratsam, vor der Verwendung von externen Paketen deren Quellcode zu überprüfen.

6 Jupyter Notebook

Ein Jupyter Notebook bietet zunächst eine interaktive (Web-) Umgebung, in der segmentierter Code in Python, R und Julia geschrieben, ausgeführt und anschließend visualisiert werden kann. Zudem

wird die Verwendung von Markdown-Notizen unterstützt, um die Codesegmente gleichzeitig dokumentieren zu können. Ein Jupyter Notebook kann sowohl in einer lokalen *.ipynb* Datei, als auch in der Cloud ausgeführt werden. Dies ist optimal für den Zugriff und die Zusammenarbeit verschiedener Teammitglieder. Ein weiterer Vorteil der Verwendung von Python innerhalb eines Jupyter Notebooks ist die Segmentierung des Codes, was eine bessere Organisation und Effizienz bei der Verwendung von großen Datensätzen ermöglicht. Somit bieten Jupyter Notebooks aufgrund ihrer Interaktivität, Segmentierung, Zugänglichkeit, Visualisierung und Dokumentation in der Data Science eine benutzerfreundliche und effektive Möglichkeit um Datenanalysen und Modellierungen erstellen zu können. Diese Auswertung verwendet im Weiteren ein Jupyter Notebook und alle folgenden Angaben mit "[Python]" referenzieren auf die Verwendung von Python innerhalb dieses Notebooks.

7 Realisierung

7.1 Datensatz Modifizierung

Der umfassende Datensatz bat aufgrund seiner Größe von 162.81 MB einen optimalen Anhaltspunkt, um die Möglichkeiten der LOW-LEVEL-Modifizierung des Datensatzes durch beide Tools zu vergleichen. Dabei war das Ziel, den Datensatz dynamisch auf die maximale Größe von 25 MB (für GitHub) zu zuschneiden. Dabei sollten die Spalten zunächst auf die wirklich relevanten reduziert werden. Zudem sollte im Weiteren die Anzahl der Zeilen gemäß dem Datum reduziert werden. So sollten insbesondere ältere Zeilen zuerst entfernt werden, da diese auch zumeist nur wenige Länder repräsentieren.
[**Power BI**] Das Zuschneiden des Datensatzes ist in Power BI nur bedingt möglich, normalerweise benutzt man die Filterfunktion, und blendet so ungewollte Daten aus. Wenn man aber trotzdem den Datensatz modifizieren möchte, muss man dafür Code verwenden. Dafür hat Power BI eigene Methoden, die dabei helfen können. Deshalb braucht man nicht zwingend zusätzliche Libraries, es ist jedoch auch möglich, Libraries wie Pandas zu Verwenden.
[**Python**] Im Gegensatz dazu bietet Python unter der Verwendung der Pandas Library nahezu unbegrenzte Möglichkeiten und Ressourcen zur Lösung dieser Aufgabe. Eine mögliche Ausführung wird im Abschnitt *0.2 Trim Data Elaboration.ipynb* implementiert. Dabei wird insbesondere deutlich wie einfach jegliche Information und Metadaten wie die Speichergröße eines Datensatzes beziehungsweise eines Eintrags erfasst werden können. Im Folgenden wird der Datensatz im Code neu geordnet, um zunächst die neuen Zeilen gemäß der Anforderung zu übertragen. So werden die Informationen Eintrag für Eintrag in einen neuen leeren Datensatz übertragen, bis die kritische Grenze erreicht ist. Schlussendlich wird der Datensatz entsprechend dem Datum erneut sortiert. Dabei ist anzumerken, dass in der derzeitigen Version, der verwendete Algorithmus noch äußerst ineffizient implementiert ist (voraussichtlich $O(n * n!)$). Dies verdeutlicht, dass umfangreichere Auswertungen, welche über das graphische Aufbereiten der Daten hinausgehen, problemlos möglich sind. Jedoch wird zudem deutlich, dass Operationen auf dieser Metaebene ein umfangreiches Programmierverständis voraussetzen. Dies gilt insbesondere für große Datensätze, wenn Skalierbarkeit und Effizienz unerlässlich werden. Eine entsprechende Effizienzsteigerung könnte beispielsweise durch das Divide-And-Conquer-Prinzip mit der Komplexität $O(log\_2(n))$ erzielt werden.

7.2 Data Understanding / Preperation

7.2.1 Data Reading

Für das Data Reading wird nun vergleichen, wie einfach es ist, Daten aus verschiedenen Quellen mit verschiedenen Spezifikationen (JSON, CSV, etc.) einzulesen.
[Power BI] Um die Daten, welche für die Visualisierungen gebraucht werden, zu verwenden, kann in Power BI einfach "drag and drop" verwenden werden. CSV, Excel, SQL, Dataflows, ... sind nur einige der Dateiformate, welche dafür in Frage kommen. Da Power BI ein Microsoft-Tool ist, ist es möglich mit Programmen wie Microsoft-Azure einfache und automatisierte Verbindungen zu Datenmanagementsystemen oder Cloudlösungen im allgemeinen aufzubauen. Aber auch Programme, welche nicht von Microsoft sind, können dazu verwendet werden. Solche sind zum Beispiel: Google BigQuery, Amazon Redshift, SAP oder Salesforce.
[Python] Auch das Einlesen der Daten wird in Python, durch die Pandas Library stark vereinfacht, obwohl es dennoch sehr umfangreich gestaltet werden kann. Zwar können direkt beim Einlesen bestimmte Datentypen der Spalten oder andere Spezifikationen definiert werden, jedoch macht es zumeist mehr Sinn diese Ungenauigkeiten in einem separaten Data Cleaning Schritt zu bereinigen. So reicht es, zunächst den relativen Dateipfad in der entsprechenden Methode *read_csv()* anzugeben. Zudem stellt die Library weitere Methoden für Dateiformate wie SQL, JSON und EXCEL bereit. Genau wie bei Power BI ist es auch in Python möglich, cloudbasierte Inhalte einzubinden. Diese Methode ist insbesondere dann relevant, wenn auch die Jupyter Notebook Datei übergreifend zugänglich gemacht werden soll und keine Speicherredundanz durch Dezentralisierung hervorrufen werden soll. Diese Methode ist zumeist jedoch Fehleranfälliger und Schreibevorgänge können gegebenenfalls nur bedingt erfolgen.

7.2.2 Data Cleaning

Das Data Cleaning gilt als essentieller Besandteil eines datengetrieben Analyseverfahrens. Das Ziel ist die Identifizierung, Bereinigung, Korrektur und Entfernung von ungenauen, unvollständigen, inkonsistenten oder fehlenden Daten.
[Power BI] Das Data Cleaning kann unter dem Reiter "Tabellenansicht" betrieben werden. Es können Spalten gelöscht, hinzugefügt, sortiert oder gefiltert werden. Natürlich kann man auch einzelne Zeilen löschen oder hinzufügen und mehrere Datensätze zusammenfügen. Für ein komplexeres Data Cleaning, welches nicht standardmäßig von Power BI zur Verfügung gestellt wird, kann auch Code genutzt werden. Für diesen Code gibt es eigene vorimplementierte Methoden, welche benutzt werden können.
[Python] Dagegen wird das Data Cleaning unter Python relativ schnell komplex, sobald die Korrektur kontextspezifischer wird. Zunächst ist es sehr leicht, einfache Transformationen wie das Filtern, Zusammenfassen oder Überschreiben von Werten durchzuführen. Auch fehlende Werte können mit Methoden wie beispielsweise *interpolation()* der Pandas Library leicht bereinigt werden. Handelt es sich jedoch um ein unübliches Problem, welches nicht direkt durch die Methoden der gängigen Libraries abgedeckt wird, wird die Korrektur zumeist umständlich. Insbesondere Neuordnungen oder Abänderungen, welche in Abhängigkeit zu anderen Informationen stehen erfordern Programmschritte, die ein umfangreiches Wissen über die Programmiersprache benötigen. Besonders nennenswert sind hierbei die *lambda-Ausdrücke*. Natürlich könnten die Einträge auch iterativ bereinigt werden, jedoch ist diese Anwendung (mit: *for i, row in df.iterrows()*) oftmals sehr ineffizient, da nicht auf optimierte Algorithmen zurückgegriffen wird.

7.3 Erstellvorgang

Der Erstellvorgang soll die Konstruktion jedes Diagramms durch einen Ersteller vereinheitlichen. Bei dem Ersteller handelt es sich nicht zwangsläufig auch um den späteren Verwender. (näheres in 7.4Verwendung).

[Power BI] In Power BI wird der eingegebene Datensatz zunächst am rechten Bildschirmrand entsprechend den Spalten gegliedert dargestellt. Im Folgenden werden diese Spalten in dafür vorgesehene Felder gezogen und bei Bedarf weiter angepasst. Eine solche Anpassung kann zum Beispiel das Kumulieren oder Zählen eines Wertes sein. Außerdem kann das Format, sowie die Visualisierungen angepasst und weitere Analysen, wie eine Trendlinie, hinzufügen werden. Diese Schritte sind meist intuitiv und können relativ schnell umgesetzt werden. Bei Problemen kann sich die Bearbeitungszeit jedoch stark erhöhen.

[Python] In Python ist es in der Regel vorteilhaft zu Beginn einer Auswertung einen neuen Datensatz zu erstellen, der nur die relevanten Informationen enthält. Dabei ist es sinnvoll, die Daten direkt entsprechend der verwendeten Library und dem verwendeten Diagrammtyp zu formatieren. Im Folgenden wird meist ein neues Objekt dieser Library erstellt. Diesem wird anschließend der neue Datensatz als Grundlage übergeben. Ferner kann dieses Objekt optisch strukturiert und gestaltet werden. Diese Schritte sind meist sehr intuitiv, so kann der Datensatzes durch Methoden wie beispielsweise *.groupby()* gruppiert oder anderweitig transformiert werden. Auch die folgende Erstellung des Objekts ist sehr einfach und einheitlich gestaltet. Nur das optische Aufbereiten und Strukturieren kann durch großen Dictionaries schnell unübersichtlich werden. Zudem ist es im Gegensatz zu Power BI möglich, aufgrund der programmiertechnischen Freiheit individuelle Zusätze zu ergänzen. Jedoch benötigt das Beheben von Anzeigefehlern viele unschöne "Workarounds". Der grundlegende Erstellvorgang in Python lässt sich anhand des nachfolgenden Python: Pie Chart - Weapons gut verallgemeinern:

```
#Import Libraries
import plotly.express as px

#Collect and Format Subset
df_weaptype_count = df.groupby(["weaptype1_txt"])).size().reset_index(name="count")

#Create Object
fig = px.pie(
    df_weaptype_count,
    values="count",
    labels="weaptype1_txt",
    names="weaptype1_txt",
    titel="Verteilung Waffentypen"
)

#Format Object
fig.update_traces(textposition='inside')

fig.update_layout(
    uniformtext_minsize=12,
    uniformtext_mode='hide',
    showlegend=True
```

```
)

#Show Object
fig.show()
```

7.3.1 Einfache Charts

Im folgenden wird das Erstellen von Diagrammen aufgezeigt, welche aufgrund ihrer geringen Komplexität vollumfänglich in Power BI enthalten sind.
[**Power BI**] Die "einfachen" Charts, auf die wir während der Arbeit mit Power BI gestoßen sind, waren sehr einfach zu implementieren. Diese Diagramme sind das Streudiagramm, das Kuchendiagramm, die Flächendiagramme und die Balkendiagramme. Trotzdem kam es bei einem der Balkendiagramme für die Darstellung zu kleineren Komplikationen. Das Streudiagramm PowerBI: Trend zeigt wie viele Todesopfer es pro Jahr gab, und eine steigende Trendlinie. In dieser Visualisierung wurden die Spalten *iyear* und *nkill* verwendet, wobei *nkill* in diesem Fall kumuliert die Anzahl auf das Jahr bezogen angibt. Power BI erlaubt bei manchen Diagrammen unter den Formateinstellungen auch das Hinzufügen eines Schiebereglers oder eine Trendlinie. Die Schieberegler müssen für die gewünschten Achsen, also die X- und Y-Achse, aktiviert werden. Die Trendlinie wird unter *weitere Analysen*, mit ebenfalls nur einem Klick aktiviert. Bei dem PowerBI: Pie Chart - Weapons wurde ausschließlich die Spalte *weaptype1_txt* verwendet. Zunächst für die Legende, und dann noch die Anzahl, wie oft der jeweilige Waffentyp für einen Anschlag verwendet wurde. Die beiden Flächendiagramme PowerBI: Visulize History by Country and Weapon haben fast den selben Aufbau und sie verwenden die bereits erwähnten Schiebergler und Trendlineien, daher gibt es nicht viel neues über diese zu sagen. Das erste Diagramm zeigt, die Anzahl der Anschläge mit der jeweiligen Art der Waffe pro Jahr. Das zweite Diagramm zeigt stattdessen, wann wie viele Anschläge in welchem Land verübt wurden. Die beiden Balkendiagramme PowerBI: Top Bottom u Countries zeigen die zehn Länder mit den meisten und die zehn mit den wenigsten Anschlägen. Das nur die ersten beziehungsweise die letzten zehn Länder angezeigt werden, kann mit dem integrierten Filter von Power BI erreichen werden. In diesem gibt es bereits die Funktion, nur die top n" oder bottom n" Länder anzuzeigen. Deshalb mussten wir nur die Zahl zehn in die dafür vorgesehenen Felder schreiben und das Ziel war erreicht. Jedoch gab es dann bei dem Balkendiagramm welches die "bottom 10" Länder anzeigt das zuvor angekündigte Problem, dass das Land mit den wenigsten Anschlägen immer noch unten stand, wir wollten aber das dieses Land oben steht. Zunächst haben wir gedacht es gäbe beim Filter die Funktion die Reihenfolge der Balken zu drehen, jedoch führte diese Annahme ins nichts. Als wir auch nach Recherche im Internet keine Lösung fanden, haben wir einfach rumprobiert. Wir sind dann zufällig auf die Lösung gestoßen, es gibt oben rechts von jeder Visualisierung drei Punkte, dort kann man, unter *Sort axis*, die Reihenfolge ändern.
[**Python**] Auch in Python war die Implementierung dieser Kathegorie von Charts nicht sonderlich aufwändig. Das Erstellen der Balkendiagramme Python: Top Bottom u Countries war intuitiv und schnell zu erledigen, sobald der neue Datensatz abgeleitet wurde. Nur das Kuchen-Diagramm Python: Pie Chart - Weapons hatte zunächst eine unschöne Auswucherung der kleinen Prozentzahlen. Dies ließ sich nach einiger Recherche jedoch durch die Verwendung von *textposition="inside"* und *update_layout(uniformtext_minsize=12, uniformtext_mode="hide", showlegend=True)* beheben. Das Flächendiagramm PowerBI: Visulize History by Country and Weapon konnten auch sehr einfach konstruiert werden, jedoch musste die Berechnung der Trendlinie gesondert erfolgen. Sie konnte jedoch un-

problematisch als zusätzliche *Trace* ergänzt werden. Die Trendlinie für das Scatter-Plot Python: Trend konnte hingegen ohne zusätzlichen Code direkt in das Diagramm eingefügt werden. Nur das Zusammenfügen der individuellen Diagramme auf eine Figur war nicht ganz so intuitiv wie es entsprechend der Erstellung der individuellen Charts zu erwarten gewesen wäre. Dies lag unter anderem an der sich ändernden Formatierung der Charts.

7.3.2 Komplexe Charts

Im Gegensatz zu den "einfachen" Charts sind die "komplexen" Diagramme nicht standardmäßig in Power BI enthalten und müssen entsprechend der nachfolgenden Möglichkeiten implementiert werden.
[**Power BI**] Die "komplexen" Charts sind deutlich Zeitaufwendiger, da und diese einige Probleme bereiteten. In dieser Auswertung wurden exemplarisch zwei dieser Diagramme in Power BI übernommen. Das 3D Cluster und das Violin Chart wurden nicht übernommen, da diese nicht standardmäßig enthalten sind, und daher entweder bezahlt oder mit Python-Code erstellt werden müssten. Der Python-Code wäre somit eine identische Kopie aus dem Python Code. Das übernommene Diagramm ist die Korrelationstabelle PowerBI: Correlation Table. Um diese zu erstellen, wählt man die Vorlage von Power BI für eine Visualisierung mit Python-Code aus, dann "popt" ein Fenster auf, in welches man den Code einfügen kann. Der eingelesene Datensatz wird in der Variable *dataset* gespeichert. Was jedoch Probleme bereitet, war dass die Spalte *gname* ein String ist und ein solcher nicht für die Korrelation verwendet werden kann. Die Lösung ist ein "Workaround", man erstellt eine neue Spalte welche jedem einzigartigen Wert/String der Spalte *gname* einen ganzzahligen Wert in der neuen Spalte zuweist, dafür hat Power BI auch schon integrierte Funktionen, jedoch muss man auch hierfür wieder Code verwenden. Mit dieser neuen Spalte kann dann die Korrelationstabelle ohne Probleme erstellt werden. Weiter "komplexe" Visualisierung sind die Karte PowerBI: Geo-Map. Diese existiert bereits als Vorlage. Man muss ledeglich die Spalten in die dafür vorgesehenen Felder ziehen. Die Karte zeigt für das jeweilige Land, wie häufig jede Art der Waffe für die Anschläge verwendet wurde und mit der Größe des Kuchendiagrammes wie viele Todesopfer es in diesen gab. Jedoch kann die Genauigkeit nur ganze Länder abdecken.
[**Python**] In Python unterscheiden sich die "einfachen" und "komplexen" Darstellungen zumeist nur in ihrem Konfigurationsaufwand. Insbesondere für den individuellen Aufbau der Karte Python: Geo-Map ist eine umfangreiche Konfiguration mit JSON notwendig (vergleiche. *3.4 Geo-Mapping* des Jupyter Notebooks). Der im subsection 7.3 beschriebene Prozess wird somit jedoch größtenteils beibehalten. Auch dimensionsreiche Auswertungen wie die Python: 3D Visual Cluster können beliebig angepasst und implementiert werden. Jedoch ist anzumerken, das diese komplexen Auswertungen, oft nur durch individuelle Libraries realisiert werden können. Dies benötigt somit im Gegensatz zu den "einfachen" Charts einen zusätzlichen Recherche- und Implemetierungsaufwand. Jedoch zeigt dies erneut den Umfang, den Auswertungen in Python annehmen können.

7.4 Verwendung

7.4.1 Optisches Aussehen

Das optische Aussehen beschreibt im Weiteren die Übersichtlichkeit der Diagramme.
[**Power BI**] Die, mit Power BI erstellten Diagramme sind simpel gehalten, und machen es somit einfach die dargestellten Informationen herauszulesen. Jedoch funktioniert dies bei simplen Diagrammen deutlich besser als, zum Beispiel bei der PowerBI: Geo-Map. Die simplen, aber größentechnisch starren

Kuchendiagramme können daher viel zu klein oder groß im Verhältnis zu der Karte sein. Wir konnten, während der Bearbeitung und Erstellung der Visualisierungen, keine farblichen oder schärfetechnischen Mängel feststellen. Deshalb ist die Qualität der Visualisierungen zufriedenstellend. Für Visualisierungen, welche mit Python-Code erstellt wurden, gilt das selbe wie in Python selbst.
[**Python**] In Python ist die optische Qualität der Diagramme stark abhängig von der verwendeten Library. Jedoch lässt sich verallgemeinert sagen, dass der gestalterische Umfang durch Libraries wie *Plotly* nahezu unerschöpflich ist. So lassen sich viele Dimensionen durch Farbe, Stärke, Form, etc. in einem Diagramm zusammenführen. Dabei können diese Eigenschaften frei gewählt und beliebig angepasst werden. Insgesamt wirken die Diagramme übersichtlich, modern und aufgeräumt. Nur das Python: 3D Visual Cluster wirkt aufgrund seiner Dimensionsdichte unstrukturiert, jedoch soll in diesem Diagramm lediglich verdeutlicht werden, wie viele unterschiedliche Informationen zusammengeführt und verarbeitet werden könnten. Gerade die Python: Geo-Map zeigt, wie strukturiert, modern und benutzerfreundlich eine komplexe Auswertung optisch Aufbereitet werden kann.

7.4.2 Anpassung

Die Anpassungsfähigkeit beschreibt die Möglichkeiten, die ein Verwender innehält, um die Auswertung gemäß seinen Anforderungen oder einer spezifischeren Fragestellung anzupassen. Dies kann zumeist durch die Verwendung von Filtern erreicht werden.
[**Power BI**] In Power BI ist es möglich Schieberegler für beispielsweise Zeitspannen einzubauen. Diese können in den Formateinstellungen für einige Visualisierungen aktiviert werden, dabei ist es möglich die entsprechenden Achsen zu aktivieren. Abgesehen von den Schiebereglern gibt es keine weiteren "Buttons" die zusätzlich hinzugefügt werden können. Wenn die Visualisierung jedoch eine Legende hat, kann man die einzelnen Elemente anklicken und diese werden dann hervorgehoben. Wenn eine Visualisierung mit Python-Code erstellt wird, können bei Bedarf auch andere / zusätzliche Funktionen dem Verwender zur Verfügung gestellt werden.
[**Python**] Prinzipiell ist es möglich, durch Jupyter Notebooks interaktive Elemente wie Schieberegler oder Checkboxen in die Auswertung einzubinden. Dies ermöglicht das dynamische Anpassen von Filtern. Jedoch benötigt dies einen sehr großen initialen Aufwand. Aufgrund dessen, wurde der Ansatz verfolgt, die Filter innerhalb der Variablen der Programmiersprache zu definieren. Dadurch können die Filter schnell angewandt und verändert werden. Jedoch setzt diese Lösung ein wenig Ahnung über Python voraus. Da die Auswertung jedoch in Jupyter Notbook erstellt und verbreitet wird, kann dies Variante als für die Zielgruppe angemessen betrachtet werden. Zudem bieten die Libraries zur graphischen Erstellung die Möglichkeit, nachträglich die Auswertung anzupassen. So bietet beispielsweise die Library *Plotly* mit dem Argument *animationframe* die Möglichkeit die Auswerung entlang einer Dimension zu verändern. Diese nachträgliche und diagrammspezifische Interaktivität lässt sich auch gut mit der interaktiven Python: Geo-Map verdeutlichen.

7.4.3 Laufzeit

Im Weiteren soll die anfängliche Ladezeit der Auswertung verglichen werden.
[**Power BI**] Wenn ein "Board" in Power BI geöffnet wird, kann es einige Sekunden dauern, bis alle Visualisierungen geladen wurden. Jedoch waren dass in unserem Fall immer weniger als fünf Sekunden. Sobald das "Board" vollständig geladen ist, folgen keine bemerkbaren Verzögerungen mehr.
[**Python**] In einer Jupyter Notebook Datei wird der Code Segement für Segment ausgeführt. Dies hat insbesondere den Vorteil, dass nur individuelle benötigte Codeblöcke ausgeführt werden können und so

die relevanten Teile der Ausertung noch schneller zur Verfügung stehen. Startet man die sequentielle Ausführung, so benötigt nur das Data Cleaning zu Beginn der Auswertung eine bemerkbare Zeit (ca. 30 Sekunden). Alle anderen Abschnitte werden trotz des umfangreichen Datensatzes in kurzer Zeit (¡ 3 Sekunden) ausgeführt, dass sie vernachlässigt werden können. Das Data Trimming hingegen würde bei Ausführung eine beträchtlich Zeit benötigen (ca. 50 Minuten). Jedoch liegt dies an der zuvor in 7.1Datensatz Modifizierung beschriebenen, schlechten Implementierung. Dies soll verdeutlichen, dass die Laufzeit in Python (mit Jupyter Notebook) grundsätzlich besser ist, jedoch stark von der Implementierung, beziehungsweise den Fähigkeiten des Erstellers abhängen kann.

7.4.4 Export

Im Folgenden werden die Möglichkeiten der Verbreitung der Auswertung durch beide Softwarelösungen an Dritte erläutert.
[**Power BI**] Das Ausgeben einer Datei ist meist recht kompliziert. Man kann das "Board" einfach in einer Power BI-Datei speichern. Um diese zu öffnen muss der Verwender jedoch auch über eine Software verfügen, welche diese Dateien lesen kann, oder sich online unter *app.Power BI.com* anmelden. Deswegen werden in der Praxis teilweise die "Boards" live auf Webseiten angezeigt. Somit muss der Benutzer sich nicht selbst um so eine Software (zum Beispiel Power BI Desktop) bemühen. Zudem ist es ebenfalls möglich Bilder oder PDF-Dateien von den Diagrammen zu erstellen.
[**Python**] Durch die in Python verwendete Libraries ist es ebenfalls fast immer möglich, die Grafiken als Bilddatei zu exportieren und gesondert zu verteilen. Ebenso können statische PDFs erzeugt werden, welche Code, Dokumentation und Auswertung umfassen. Natürlich kann auch das konkrete Notebook als Datei verteilt werden. Jedoch benötigt der Empfänger für eine sinnvolle Verwendung eine eingerichtete Entwicklungsumgebung mit allen verwendeten Libraries. Dadurch ist es zumeist sinnvoller, die entsprechende Datei über ein cloudbasiertes Versionierungstool wie Git bereitzustellen. Dabei verfügen insbesondere SAAS-Anbieter wie Github über eine direkte Visualisierung des Notebooks.

8 Auswertung

Im Folgenden wird die Qualität und Möglichkeit der Aussagen bewertet, welche durch die Diagramme in Python und Power BI formuliert werden konnten. Die Balkendiagramme PowerBI: Top Bottom u Countries und Python: Top Bottom u Countries weisen keine strukturellen Unterschiede auf. Demnach lassen sich die Länder mit den meisten und wenigsten Anschlägen problemlos ermitteln. Die Flächendiagramme PowerBI: Visulize History by Country and Weapon und Python: Visulize History by Country and Weapon besitzen jedoch einen deutlichen Unterschied, so kann in Python das Diagramm leicht die Summe über alle Länder bilden. Dies ermöglicht schnelle Aussagen über den globalen Terrorismus im Zusammenhang mit individuellen Ländern. Im Gegensatz dazu kann diese Summe in Power BI nicht so leicht gebildet werden. Dies lässt sich analog für beide Flächendiagramme zeigen. Das Violine-Chart aus Python: Pie Chart - Weapons lässt sich nur in Python erzeugen. Somit ließen sich Aussagen über die Verteilung der Jahre nur mit Python formulieren. Das Kuchendiagramm ist jedoch erneut identisch und ermöglicht gleichwertige Aussagen über die Verwendung der Waffentypen (vgl. PowerBI: Pie Chart - Weapons und Python: Pie Chart - Weapons). Auch aus den Korrelationstabellen PowerBI: Correlation Table und Python: Correlation Table wird gleichermaßen ersichtlich, dass nur die Spalten *country* und *iyear* in geringem Maße korreliert. Dies führt zu der Interpretation, dass zu bestimmten Zeiten, bestimmte Länder stärker von terroristischen Akten betroffen sind als

andere. Zwar ist das Python: 3D Visual Cluster in Python nur exemplarisch um die Dimensionsdicht aufzuzeigen. Jedoch lassen sich bei genauerer Betrachtung trotz der Unübersichtlichkeit spannende Zusammenhänge finden. Diese Auswertung ist in Power BI leider nicht möglich. Dagegen kann aus den Trend-Line-Charts PowerBI: Trend und PowerBI: Trend die Zunahme des globalen Terrorismus' gleichermaßen festgestellt werden. Die Python: Geo-Map in Python zeigt trotz des geringen Kontrasts zwischen Land und Wasser den genauen Ort eines Anschlages. Dabei beschreibt die Höhe die Anzahl der Todesopfer, während die Farbe den verwendeten Waffentyp repräsentiert. Insbesondere die Übersichtlichkeit für alle Skalierungen konnte in Power BI mit PowerBI: Geo-Map nicht so übernommen werden. Einen weiteren Vorteil bietet die Karte in Python durch ihre hohe Genauigkeit. Es ist jedoch zu bedenken, dass wie in 7.3.2Komplexe Charts beschrieben, alle Diagramme in Power BI auch mit Python hätten erstellt werden können. Dies hätte voraussichtlich zu identischen Aussagen geführt.

9 Diskussion

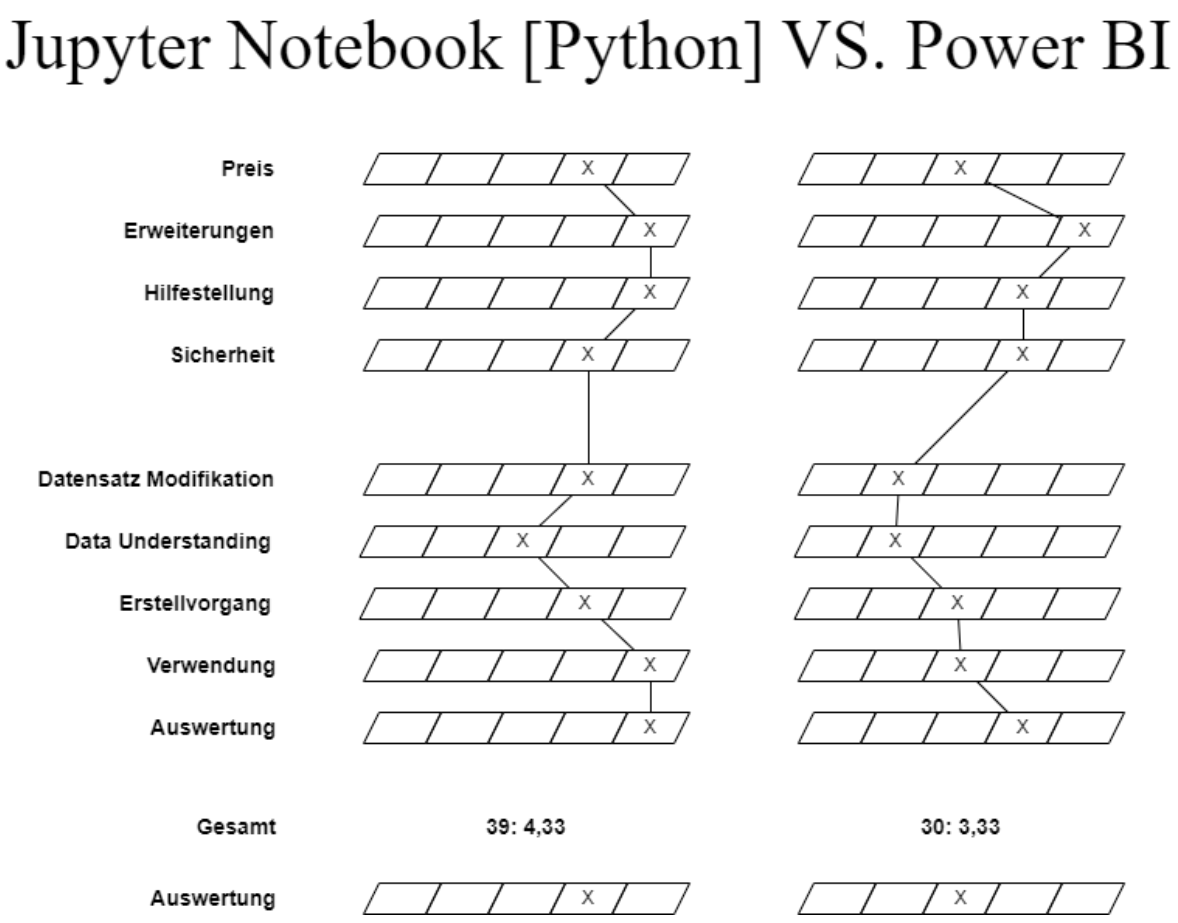
Im folgenden Abschnitt sollen nun die Aspekte aus 7Realisierung und 8Auswertung diskutiert und bewertet werden. Dies geschieht unter der Verwendung eines Punktesystems, welches zudem die Grundlage für das 11Cheat Sheet darstellt. [**Datensatz Modifizierung**]: Bei der Auswahl zwischen Power BI und Python für die Datensatzmodifizierung bietet sich nur Python an, da Power BI nahezu keine eingebauten Funktionen dafür bietet und ebenfalls größtenteils auf eigenen Code angewiesen wäre. Python bietet, insbesondere mit der Pandas Library, nahezu unbegrenzte Flexibilität bei der Datensatzmodifikation und ist ideal für komplexe, datenintensive Projekte geeignet. Es ermöglicht tiefgreifende Veränderungen und effiziente Datenreorganisation. Es erfordert jedoch ein fortgeschrittenes Verständnis für die Programmierung. Für Projekte, die eine umfassende Abänderung des Datensatztes erfordern, ist daher Python die geeignetere Wahl. **PowerBI 2; Python 4** [**Data Understanding / Preparation**]: Beim Vergleich zwischen Power BI und Python für Aufgaben, wie das Einlesen von Daten und das Data Cleaning, zeigt sich erneut, dass beide Tools ihre spezifischen Stärken haben. Power BI bietet durch seine Drag-and-Drop-Funktionalität und die nahtlose Integration mit Microsoft Azure eine benutzerfreundliche Lösung, die eine Vielzahl von Datenformaten unterstützt und automatisierte Datenverbindungen ermöglicht. Dies macht es zu einer hervorragenden Wahl für unerfahrene Nutzer. Für das Data Cleaning bietet Power BI ebenfalls eine intuitive Oberfläche, die für Standards gut geeignet ist und keine Programmierkenntnisse erfordert. Komplexe Aufgaben sind jedoch kaum ohne einen Zukauf von weiteren Leistungen möglich. Im Gegensatz dazu bietet Python mit der Pandas Library eine flexible und leistungsstarke Umgebung für komplexere Data-Cleaning-Prozesse. So können auch spezifische oder ungewöhnliche Anforderungen realisiert werden. Es setzt jedoch erneut umfassende Programmierfähigkeiten voraus. In diesem Fall lässt sich sagen, dass Python auch hier die bessere Wahl zu sein scheint. **PowerBI 2; Python 3**. [**Erstellvorgang**]: Auch hier präsentieren Power BI und Python zwei grundlegend unterschiedliche Ansätze. Power BI ermöglicht in seiner aktuellen Version das mühelose Erstellen einfacher Diagramme wie das Streu-, Kuchen-, Flächen- und Balkendiagramme. Für weitergehende Visualisierungen ist jedoch entweder der Kauf zusätzlicher Funktionen oder der Einsatz von Python erforderlich. Darüber hinaus bietet Power BI bei komplexeren Problemen, die schwer zu erläutern sind oder spezifischen Kontext erfordern, kaum Unterstützung durch Online-Troubleshooting. Jedoch ist die Unternehmenseigene Dokumentation von Microsoft sehr ausführlich und es werden einige Schulungen bereit gestellt. Python zeigt sich dennoch als flexible Alternative, da es nicht nur einfache Diagramme mit wenig Anstrengung erstellt werden können, sondern

auch die Erstellung komplexer Grafiken bei entsprechenden Vorkenntnissen möglich ist. Insbesondere bei der Verwendung geografischer Daten zeigt sich Python als überlegen, denn ohne finanzielle Investitionen ist Power BI hier kaum einsetzbar. Python erweist sich somit einmal mehr als das überlegene Werkzeug in dieser Gegenüberstellung. [**PowerBI 3; Python 4**] **Verwendung**: Durch die Verwendung vielfältiger Libraries ermöglicht Python eine nahezu unbegrenzte kreative Freiheit in der Darstellung von Daten. Nutzer können Farben, Formen, Stärken und viele weitere Dimensionen in ihren Diagrammen individuell anpassen, was zu modernen, übersichtlichen und aufgeräumten Visualisierungen führt. Obwohl Power BI für seine Einfachheit und das schnelle Erstellen einfacher Diagramme bekannt ist, bietet es nicht dieselbe Flexibilität und Detailtiefe. [**PowerBI 3, Python 5**] **Auswertung**: Die Möglichkeit, Erkenntnisse aus den Visualisierungen herauszulesen ist, wie in 8Auswertung beschrieben in Python deutlich umfangreicher. Jedoch bietet auch Power BI mit seinen grundlegenden Funktionen eine starke Aussagekraft. [**PowerBI 4, Python 5**] So steht es fest, dass für dieses Auswertung Python Power BI in nahezu allen Belangen voraus ist. Der Hauptgrund könnte dafür sein, dass Power BI nicht auf die wissenschaftliche Arbeit ausgelegt ist. Während es sicherlich nicht unmöglich ist, liegen seine Vorteile mehr im Reporting oder der Anwendung in Unternehmen.

10 Fazit

Im Rahmen unserer Datenanalyse standen uns sowohl Power BI als auch Python als leistungsfähige Werkzeuge zur Verfügung, die jeweils ihre spezifischen Stärken und Schwächen aufweisen. Microsofts Power BI bietet eine nahtlose Integration in andere Microsoft-Produkte, was die umfassende Datenintegration erleichtert. Seine intuitive Benutzeroberfläche, inklusive Drag-and-Drop-Funktionen macht es möglich, schnell ansprechende Visuals zu erstellen. Im Gegensatz dazu benötigt Python fundierte Programmierkenntnisse mit einer steileren Lernkurve. Python ist jedoch äußerst flexibel und erlaubt es, spezifische Analyseprozesse präzise zu steuern und zu erweitern. Dies eignet sich insbesondere für komplexere Datenmanipulation und tiefgreifende Analysen. Python punktet mit seiner kostenfreien Verfügbarkeit und der Vielzahl an Bibliotheken. Dies ist besonders attraktiv für kleinere Organisationen oder Einzelpersonen. Power BI kann hingegen kostspielig sein, bietet aber eine stabile Plattform mit professionellem Support durch Microsoft. Als Cloud-basierte Lösung ermöglicht Power BI das einfache Arbeiten im Team und eine gute Skalierbarkeit von Projekten. Dies birgt jedoch auch den theoretischen Verlust der exklusiven Datenhoheit. Python, das auf individuellen Systemen ausgeführt wird, bietet mehr Kontrolle über die Sicherheitsaspekte, wobei die Verantwortung für die Sicherheit jedoch stark vom Endnutzer abhängt. Beide Plattformen profitieren von starken ihren aktiven Communities. Python bietet jedoch umfangreiche Ressourcen und eine breite Unterstützung durch eine weltweite Community, die schnelle Hilfe und zahlreiche Lernmaterialien zur Verfügung stellt. Python übertrifft Power BI in der Flexibilität der Anpassung und der Tiefe der analytischen Möglichkeiten. Es unterstützt komplexe statistische Berechnungen und maschinelles Lernen durch umfangreiche Bibliotheken wie NumPy, Pandas und scikit-learn. Power BI bietet zwar ebenfalls fortschrittliche Analysefunktionen, diese sind jedoch insbesondere in der kostenlosen Testversion in ihrem Umfang und ihrer Tiefe im Vergleich zu Python deutlich eingeschränkt. Zusammenfassend lässt sich sagen, dass die Wahl zwischen Power BI und Python stark von den spezifischen Bedürfnissen des Projekts, den technischen Fähigkeiten des Teams und den gewünschten Ergebnissen abhängt. Während Power BI ideal für die schnelle Erstellung von Berichten und Dashboards in einer Microsoft-zentrierten Umgebung ist, bietet Python eine robuste Lösung für tiefgreifende Datenanalyse und maßgeschneiderte Datenauswertungen.

11 Cheat Sheet



12 Aufteilung Autoren

1Abstrakt Arzum
2Einleitung Arzum
3Datensatz Salih
4Power BI Salih
5Python Salih
6Jupyter Notebook Daniel
7Realisierung Daniel
7Realisierung [**Power BI**] Jonas
7Realisierung [**Python**] Daniel
8Auswertung Daniel; Jonas
9Diskussion Arzum
10Fazit Arzum
11Cheat Sheet Daniel
Codeing [**Power BI**] Jonas
Codeing [**Python**] Daniel

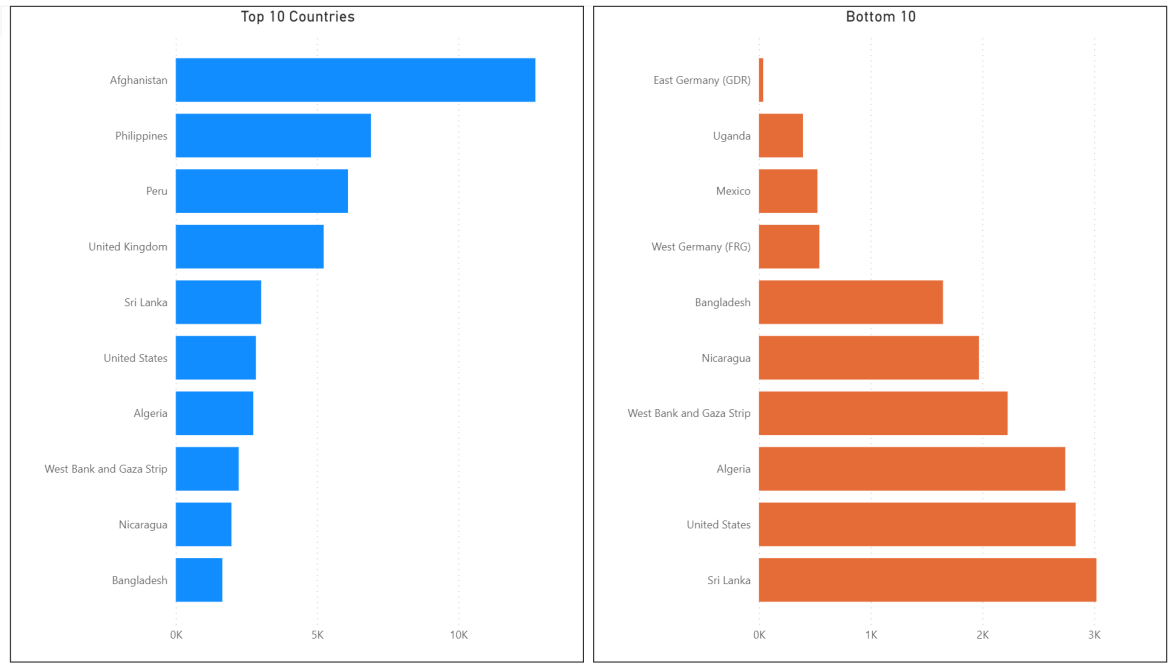


Figure 1: PowerBI: Top Bottom u Countries

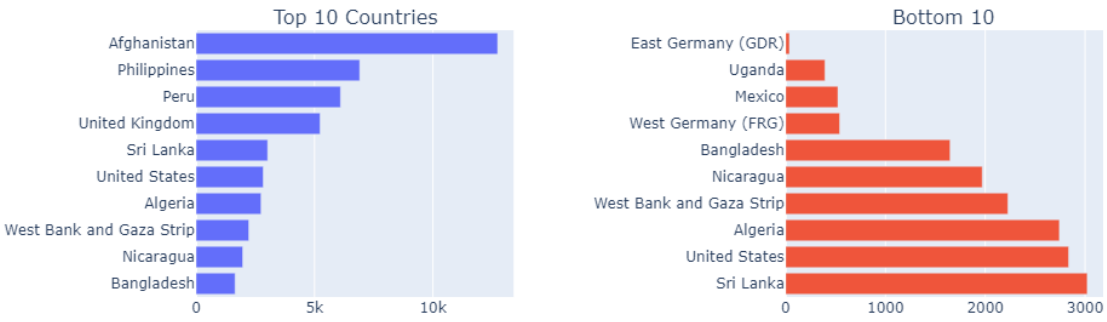


Figure 2: Python: Top Bottom u Countries

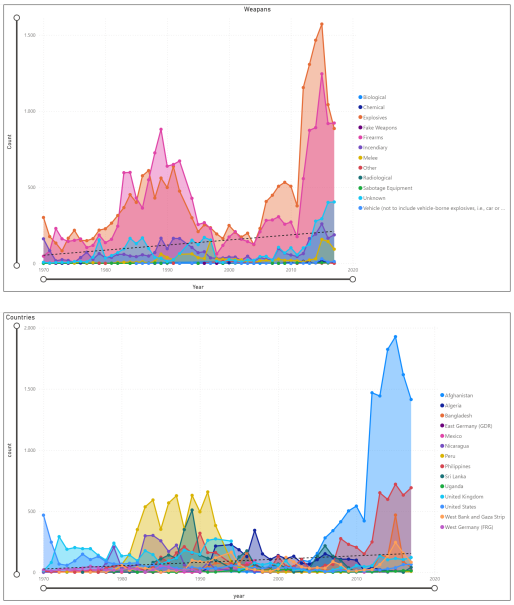


Figure 3: PowerBI: Visulize History by Country and Weapon

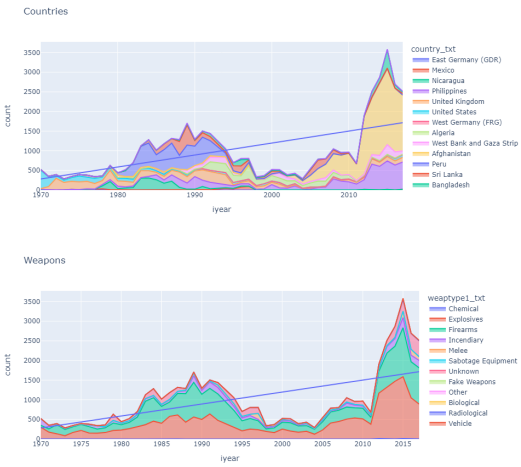


Figure 4: Python: Visulize History by Country and Weapon

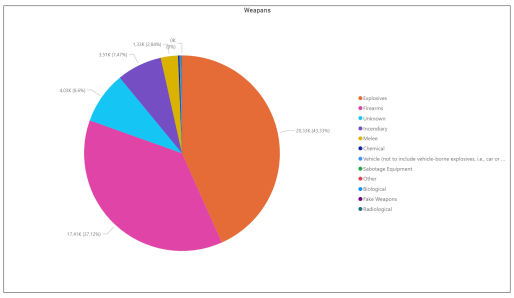


Figure 5: PowerBI: Pie Chart - Weapons

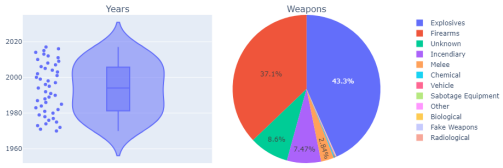


Figure 6: Python: Pie Chart - Weapons

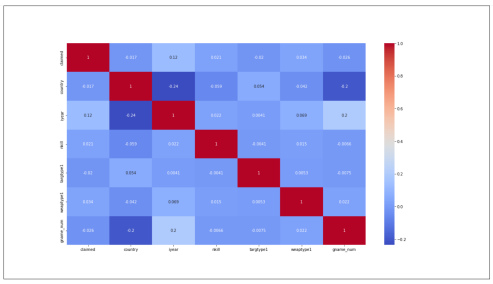


Figure 7: PowerBI: Correlation Table

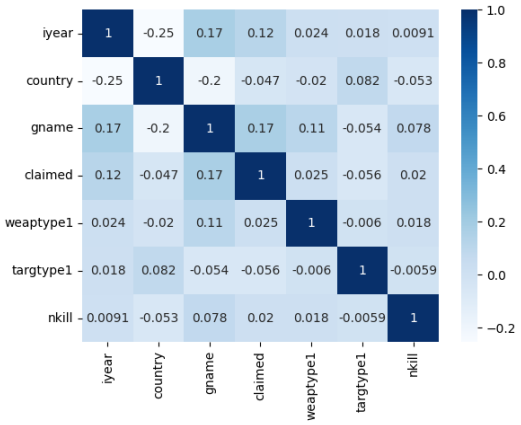


Figure 8: Python: Correlation Table

Figure 9: Python: 3D Visual Cluster

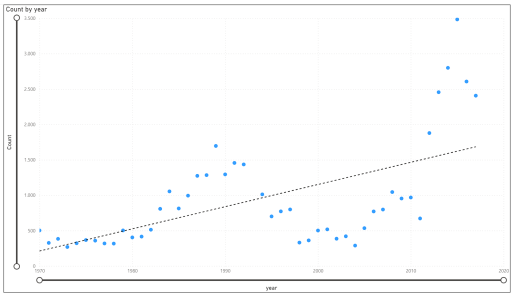


Figure 10: PowerBI: Trend

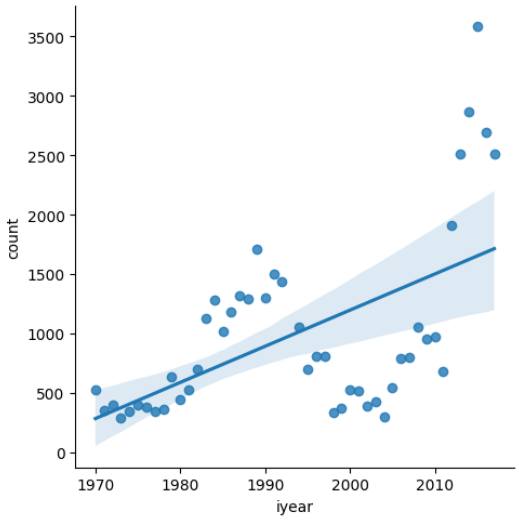


Figure 11: Python: Trend

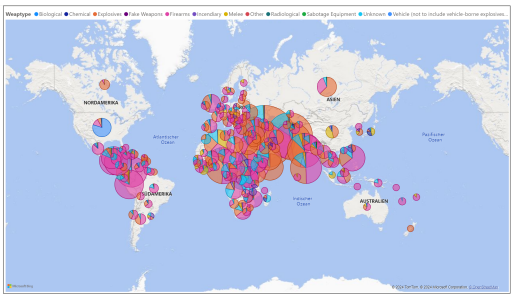


Figure 12: PowerBI: Geo-Map

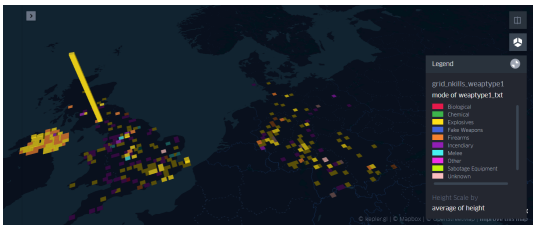


Figure 13: Python: Geo-Map