Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Moderne Arbeit
Ein Blick auf Gehälter im Bereich Data Science

Portfolio

im Modul

Data Visualization

Im Rahmen der Prüfung zum Bachelor of Science (B. Sc.)

Verfasserin/Verfasser:	**Benjamin Esch 1240633**
	Vincent Merkel 8163313
	Alina Müller 7560885
	Eduardo Stein Mössner 7561136
Kurs:	**WWI23B4**
Dozentin/Dozent:	**Prof. Dr. Jennifer Schoch**
Abgabedatum:	**10.05.2024**

Inhaltsverzeichnis

1 Einleitung

Als Wirtschaftsinformatik-Studenten mit Schwerpunkt Data Science ist es entscheidend, die Tools zur Datenvisualisierung zu kennen sowie die Vor- und Nachteile der verschiedenen Lösungen zu verstehen. Auf diese Weise können wir das beste Tool auswählen und effizient sowie zielgerichtet arbeiten.

Im Rahmen dieser Arbeiten haben wir das Tool Orange Data Mining analysiert und es mit Python als Visualisierungstool verglichen. Dabei haben wir die jeweiligen Stärken und Schwächen der Tools herausgearbeitet. Hierfür wurden sowohl für Orange Data Mining als auch für Python eine SWOT-Analyse und eine Nutzwertanalyse durchgeführt.

Für unsere Vorgehensweise haben wir einen praktischen Ansatz gewählt. Die vier Mitglieder unserer Arbeitsgruppe haben jeweils eine Fragestellung formuliert und versucht, sie mithilfe von Orange Data Mining und Python bestmöglich zu beantworten. Auf diese Weise konnten wir die Vor- und Nachteile der beiden Tools erkennen und in die entsprechenden Analysen einfließen lassen.

2 Über Orange

Als Data Visualization Tool wurde in diesem Portfolio die Software „Orange Data-Mining“ gewählt. Verwendet wird Version 3.36.2 des Tools. Die Entwicklung fand durch die Forschungsgruppe „Bioinformatics Laboratory” der „Faculty of Computer and Information Science” an der University of Ljubljana statt. Insgesamt arbeiteten sechzehn Labor-Mitglieder an dem Projekt. Mittlerweile können auch zusätzlich die Endnutzer zur Erweiterung von Orange beitragen. Der wesentliche Zweck des Programms ist es, dem Anwender einfache und überblickbare Python-Skripte zu ermöglichen. Dies basiert auf C++ Implementierungen mit rechenintensiven Aufgaben. Der Kern der Software besteht aus fast 200 C++ Klassen. Die Anwender-Zielgruppe von Orange sind erfahrene Programmierer, die in einem professionellen Rahmen arbeiten möchten, sowie Studenten, welche sich mit Data Mining befassen. Die Software läuft auf Windows, Mac OS X und Linux. Sie kann aus dem Python Package Index Repository (pip) installiert werden. Auf der offiziellen Website des Tools steht auch ein Link zum Download zur Verfügung. Der

Code wird auf BitBucket gehostet. Da Orange Open Source ist und eine hohe Distribution hat, ist das Programm für eine große Bandbreite an Benutzern zugänglich.

3 Quelle des Datensatzes

Der für die Prüfungsleistung verwendete Datensatz beschäftigt sich mit den Berufen und Gehältern im Bereich Data Science. Gefunden wurde der Datensatz zuerst auf kaggle.com unter dem Namen „Jobs and Salaries in Data Science“. Er wurde von Hummaam Qaasim dort hochgeladen. Das letzte Update liegt, zum Stand März 2024, drei Monate zurück. Ursprünglich stammt der Datensatz jedoch von einer anderen Website. Erhoben wurden die Daten auf der Website Ai-jobs.net. Von dort wurden sie nach Kaggle überführt. Ai-jobs.net gibt an, dass der Datensatz auf ihrer Webseite für gewöhnlich wöchentlich aktualisiert wird. Für das Projekt wurde der weniger aktuelle Datensatz von Kaggle verwendet, da in diesem die Jobs manuell nach Kategorien sortiert wurden. Der Arbeitsaufwand beim Data Cleaning konnte so eingeschränkt werden. Die Daten auf Ai-jobs.net kann man als vertrauenswürdig sehen, da die Daten hier selbst von Personen übermittelt werden, die im Data Science Feld tätig sind. Da die Daten aber auf der Website eingetragen werden, die exklusiv Data Science Jobs anzeigt, ist davon auszugehen, dass überwiegend Personen im Data Science Feld die Eingabemaske benutzen. Somit ist von einer relativ hohen Datenqualität auszugehen. Die Website ist im Besitz von Foorilla LLC. Foorilla LLC ist ein Ein-Personen-Unternehmen für digitale Medien und Technologie mit Sitz in Zürich in der Schweiz (Quelle 5). Sich selbst beschreibt die Firma als „tiny global digital media & technology company”. (Quelle 4)

Insgesamt hat der Datensatz 14.200 Zeilen und 12 Spalten. Die benötigten Spalten für die vier Data Visualizations sind: experience_level, salary_in_usd, company_size, company_location, job_category. Alle anderen Spalten werden nicht verwendet und können daher entfernt werden.

4 Bewertung der Tools

Zur Bewertung des genutzten Tools in Gegenüberstellung zur Datenvisualisierung in Python wird eine gleichgewichtige Nutzwertanalyse genutzt. Dabei wird in 4 Kategorien unterschieden. Betrachtet werden Zeitaufwand, Anwenderfreundlichkeit, Ladedauer und Funktionalität. Je Kategorie werden bis zu 3 Punkte vergeben. Am Ende können Orange und Python anhand der jeweiligen Gesamtpunktzahlen verglichen werden. Beim Zeitaufwand ist primär die Programmierarbeit beziehungsweise das Erstellen der Workflows von Interesse. Dabei werden Unterbrechungen nicht mit eingerechnet, es wird nur um die Bearbeitungszeit betrachtet. In Sachen Anwenderfreundlichkeit wird beurteilt, wie gut Erstbenutzer mit Orange und Python zurechtkommen und wie die Einarbeitung verläuft. Außerdem wird die Ladedauer gemessen und verglichen, wie lange die Ausführung des Codes oder Workflows dauert. Zuletzt werden die Visualisierungen gegenübergestellt. Welches Tool bietet mehr Personalisierungsmöglichkeiten und was ist bei gleicher Zielsetzung das tatsächliche Ergebnis auch im Hinblick auf die Optionsvielfalt?

5 Einlesen des Datensatzes

Der erste Schritt, um mit einem Datensatz zu arbeiten und aus diesem Visualisierungen zu erstellen, ist das Einlesen der Daten. Wichtig hervorzuheben ist, dass der Datensatz „Jobs and Salaries in Data Science“ insgesamt etwas mehr als 1660 KB groß ist. Zudem liegt er als CSV-Datei vor. Deshalb ist es möglich, dass das Einlesen von größeren Datensätzen oder Datensätzen anderen Formates einen größeren Zeitaufwand hat als in diesem Projekt.

Um die CSV-Datei in Python einlesen zu können, muss zuerst die „pandas“ Bibliothek importiert werden. Das Einlesen des Datensatzes erfolgt danach mit einer Geschwindigkeit von etwa 22 Millisekunden.

Um in Orange Data Mining den Datensatz zu importieren, reicht es, einen einzigen Knoten zu erzeugen. Dieser heißt „CSV File Import“. Innerhalb des Knotens muss der Datensatz aus den gespeicherten Dateien ausgewählt werden. Mit diesen zwei Schritten wird

der Datensatz erfolgreich importiert. Im selben Moment bekommt man schon eine Vorschau der Daten in Tabellen-Format. Die Ladedauer des Datensatzes ließ sich nicht genau bestimmen, sie beträgt schätzungsweise weniger als 30 Millisekunden.

Um nun beide Methoden zum Einlesen des Datensatzes zu vergleichen, wird ein Blick auf die Zugänglichkeit und die Leistung geworfen. Das Einlesen von Datensätzen in Python erfordert spezifische Kenntnisse, die beim Nutzer erstmal vorhanden sein müssen. Dies ist als Einstiegsbarriere zu betrachten. In dieser Hinsicht macht Orange den Einstieg viel einfacher, da kein fachspezifisches Wissen nötig ist. Die Leistung war in beiden Fällen zufriedenstellend. Jedoch ist die Zeit, welche nötig ist, um den Python Code zu schreiben, zusätzliche Arbeit. Mit Orange ist dies nicht nötig. Daher bekommt Orange von uns für das Einlesen des Datensatzes für beide Kriterien einen Punkt, insgesamt damit zwei Punkte. Python erhält nur einen Punkt für Leistung.

6 Data Cleaning

Für die Visualisierung werden nicht alle Spalten des Datensatzes benötigt. Daher können diese ohne Bedenken entfernt werden. Zudem wird geprüft, ob Daten fehlen. Ebenso wollen wir feststellen, welche Dateitypen die einzelnen Spalten haben.

Zuerst wird der Datensatz auf Vollständigkeit geprüft. Um dies in Python zu machen, wurden 7 Zeilen Code geschrieben, was einen Zeitaufwand von 5 Minuten erfordert. Daher wurde ganz detailliert festgestellt, dass keine Daten gefehlt haben. Um denselben Prozess in Orange durchzuführen, benutzt man „Feature Statistics". Hierbei werden in der letzten Zeile die Anzahl und der prozentuale Anteil an fehlenden Werten angezeigt. Das Ergebnis beider Untersuchungen ergibt, dass dem Datensatz keine Werte fehlen, er also vollständig ist.

Falls es doch fehlende Werte geben sollte, ist es in Python möglich, diese Zeilen zu entfernen oder durch einen anderen Wert zu ersetzen. Dafür ist ein Zeitaufwand von circa 10 Minuten nötig. In Orange hingegen ist es, soweit im Umfang dieser Arbeit recherchiert wurde, nicht möglich, einzelne fehlende Werte einzufügen. Daher muss man schon vor der Bearbeitung in Orange den Datensatz mit einem externen Programm wie zum Beispiel

Excel vorbereiten. Um Zeilen mit fehlenden Werten zu entfernen, wird „Preprocess" benötigt. Dort wird zuerst „Impute Missing Values" und dann „Remove rows with missing values" ausgewählt.

Als nächstes wird geprüft, welche Spalte welche Datentypen enthält. Dies wird in Python mit der Methode „Data.dtypes" aus der Pandas Bibliothek gemacht. Das Ganze erfordert eine Zeile Code, mit einem Schreibaufwand von 20 Sekunden. Das Ergebnis ist, dass die Spalten des Datensatzes aus int64 und Objekt-Datentypen bestehen. Um diese Information in Orange zu erhalten, wird „Edit Domain" verwendet. Dort werden einem sofort die einzelnen Datentypen gezeigt. In Orange sind diese Datentypen jedoch anders benannt. Hier gibt es folgende Datentypen: „Categorical", „Numeric", „Text", „Time". In diesem Knoten hat man auch die Möglichkeit, den Datentyp einer Spalte zu ändern.

Eine Spalte aus einem Datensatz in Python zu entfernen, erfordert einen Arbeitsaufwand von ca. 5 Minuten. In Orange hingegen ist es nicht möglich, eine Spalte zu entfernen, allerdings können Spalten ausgeblendet werden. Dafür wird der Knoten „Select Columns" genutzt. Diese Funktion zu implementieren erfordert keine Programmierkenntnisse und wird wie jede andere Funktion in Orange hinzugefügt. Dies zu tun, beträgt einen Arbeitsaufwand von ca. 1 Minute. Man zieht jede Spalte, die man verwenden möchte, von dem Feld „Ignored" zu „Features".

In Summe verfügen beide Tools über die wichtigsten benötigten Funktionen, um die wesentliche Meta-Information aus dem Datensatz zu erhalten. Der Arbeitsaufwand um diese zu implementieren ist in Orange geringer. Im Gegensatz zu Python sind auch keine Programmierkenntnisse erforderlich. Jedoch ist es in Orange nicht möglich, fehlende Werte zu ergänzen. Das heißt, sollte es in diesem Datensatz eine Spalte geben, in der der Wert für die Jobkategorie fehlt, kann man diesen nicht selber ergänzen. Leider ist dies eine der wichtigsten Funktionen, die benötigt werden, um Data Cleaning zu betreiben. Daraus folgt, dass Orange Data-Mining sich nicht gut für Data Cleaning eignet.

7 Visualisierungen zu den Fragestellungen

7.1 Wie steht die Firmengröße in Relation zum Gehalt?

Eine Fragestellung, für die sich der Datensatz ideal eignet, ist das Untersuchen des Gehalts in Relation zur Firmengröße. Eine solche Auswertung wird durch die Spalte „company_size" ermöglicht. Diese Spalte beinhaltet die Größenordnung der Arbeit gebenden Firma in den 3 Klassifikationen „S", „M" und „L". Foorila gibt an, dass S (= small) kleine Firmen mit weniger als 50 Mitarbeitern klassifiziert, M (= medium) mittelgroße Firmen mit 50 bis 250 Mitarbeitern und L (= large) größere Unternehmen mit mehr als 250 Mitarbeitern, eingeteilt ist. (s. Quelle 2).

Um die oben genannte Fragestellung zu beantworten, bietet sich ein sogenanntes Bar Chart bzw. ein Balkendiagramm an. Mit einem solchen Diagramm können unter anderem Unterschiede in Zahlen zwischen verschiedenen Kategorien angezeigt werden. In der x-Achse steht in diesem Fall die Firmengröße in der oben beschriebenen Codierung, die y-Achse enthält den Median des Gehaltes. Somit beantwortet das Diagramm die Frage, welche Gehälter für verschiedene Größenordnungen von Firmen typischerweise erwartet werden können. Um das Diagramm schlüssiger zu machen, ist es außerdem sinnvoll, die Balken in der Reihenfolge S, M, L nach Größe sortiert darzustellen.

Um das Balkendiagramm in Python zu erstellen, werden zuerst die nötigen Bibliotheken eingebunden. Besonders hervorzuheben ist an dieser Stelle „MatPlotLib", da mit dieser Bibliothek das Diagramm erstellt wird. Zuerst wird wie im Kapitel „Datensatz einlesen" der gesäuberte Datensatz eingebunden. Daraufhin müssen die Daten gruppiert werden, wobei auch direkt in derselben Zeile durch Anhang von „.median()" der Median berechnet wird. Im letzten Schritt wird nun das Balkendiagramm erstellt. Dabei müssen in 8 Codezeilen verschiedene Parameter wie Farbe, Beschriftungen, Diagramminhalt und Größe beschrieben werden. Die Reihenfolge der Balken war ohne weitere Einstellungen richtig von „S" zu „L" geordnet.

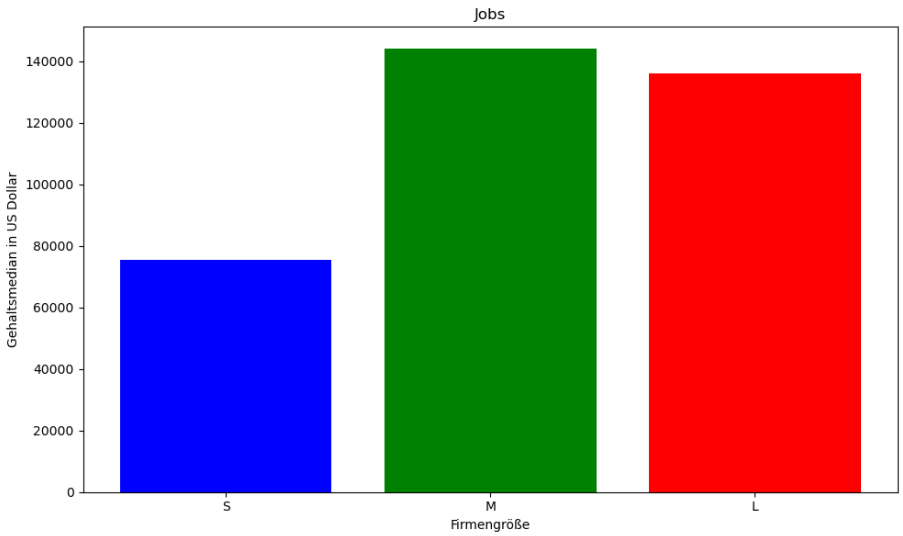

Abbildung 1Bar Chart in Python

Die Erstellung des Balkendiagramms ist in Python zwar sehr kompakt, war aber nicht einfach umzusetzen. Die ersten Schritte waren intuitiv und mit Vorerfahrung ohne Probleme lösbar. Auch bei der Erstellung von (Balken)Diagrammen lagen Grundkenntnisse vor. Trotzdem kamen einige kleinere Fehl Formatierungen auf, die nur durch konsultieren von einer Recherche (u.a. ChatGPT, Stackoverflow) gelöst werden konnte. Auch die Python-Dokumentation wurde in Betracht gezogen, die Verwendung der anderen Quellen war allerdings intuitiver. Für die Anwenderfreundlichkeit wird somit beim Balkendiagramm 1 von 3 Punkten vergeben. Der Zeitaufwand für die Programmierung war mit etwa 45 Minuten recht hoch, Python bekommt somit 1 Punkt für Zeitaufwand. Die Ausführung war ohne menschliches Erkennen vorüber, somit werden 3 Punkte für Ladedauer vergeben. Die Funktionalität bekommt 3 Punkte, denn im konkreten Anwendungsfall haben keine wichtigen Funktionen gefehlt, das Diagramm ist gut lesbar und die Achsen waren beliebig benennbar.

Der Bar Chart ist Teil des Grundpakets von Orange und somit ohne Installation von Addons nutzbar. Der Aufbau der Knotenpunkte erscheint logisch: Zuerst wird per „CSV File Import" der Datensatz importiert. Daraufhin werden mit „Select Columns" die beiden relevanten Spalten „company_size" und „salary_in_usd" als Unterdatensatz erstellt. Dieser wird dann in einem „Group By" Knotenpunkt nach der Firmengröße gruppiert und daraufhin wird der Median aller Gehälter einer Firmengröße ausrechnet. Diese Daten werden nun mit einem „Color" Knotenpunkt farblich codiert und daraufhin mit dem „Bar

Chart” Knotenpunkt ausgegeben. In diesem kann auch die Beschriftung und Farbdatei (im vorherigen Knotenpunkt erstellt) ausgewählt werden. Es wurde bei einer Recherche kein Weg gefunden, bei nominal skalierten Werten wie der Firmengröße mit den Werten S, M oder L die Reihenfolge zu ändern. Deshalb ist sie im Diagramm nicht entsprechend sortiert.

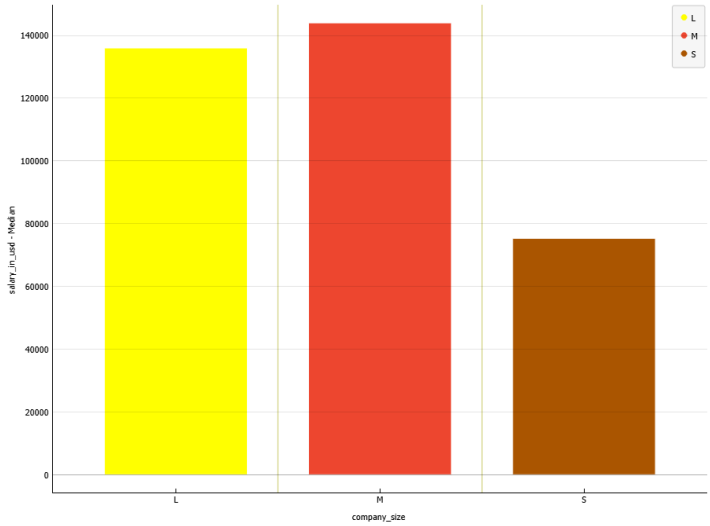

Abbildung 2Bar Chart in Orange

Diese Auswertung in Orange war nach der Beurteilung des Autors intuitiv und einfach zu bewältigen. Um die Benutzerfreundlichkeit von Orange zu beurteilen, hat der Autor ohne Vorerfahrungen mit Orange und ohne Anleitung probiert, die Auswertung zu erstellen. Dies war durch Ausprobieren nach 15 Minuten erledigt. Somit bekommt Orange 3 Punkte für die Kategorie Zeitaufwand. Ein kleineres Problem lag aber vor, als nach dem Hinzufügen des „Color” Knotenpunkts die Balken im Diagramm unsichtbar waren. Dies konnte durch Absprache mit den Teamkollegen gelöst werden, in der Dokumentation wird aber ebenfalls beschrieben, wie der Farb-Knotenpunkt richtig eingebunden werden kann. Außerdem erwähnenswert ist die Möglichkeit, den Durchschnitt direkt in dem „Bar Chart” Knotenpunkt zu erstellen. Es wurde aber als intuitiver erachtet, den „Group By” Knotenpunkt zu verwenden. Orange wird mit 3 Punkten für Anwenderfreundlichkeit bewertet, denn der gesamte Prozess war im Anwendungsfall intuitiv und, im Vergleich zu Python, reibungslos. Die Ladedauer war im Testfall ebenfalls stets nicht vom Nutzer erkennbar und wird somit mit 3 Punkten bewertet. Da die Beschriftung des Diagramms

nicht änderbar ist, erhält Orange für das Ergebnis 2 Punkte, da immerhin die Farben einstellbar sind.

7.2 Durchschnittsgehälter nach Jobkategorie

Eine wichtige Frage, die sich stellt, ist, welche Berufe in der Data Science Branche am besten bezahlt werden. Um diese Frage zu beantworten, eignen sich eine ganze Reihe von unterschiedlichen Visualisierungen. Für einen bestmöglichen Überblick, werden in diesem Abschnitt zwei Visualisierungen zu dieser Frage erstellt. Zur Veranschaulichung in Orange wurde wie folgt vorgegangen:

Nachdem die Daten erfolgreich importiert und das „Data Cleaning“ durchgeführt wurde, bietet sich an, die benötigten Daten in eine weitere Tabelle zu extrahieren. Dafür wurde ein Prozess in dem Tool gestartet, welches „Select Columns“ heißt. Dafür wird dieser Prozess aus dem Reiter „Transform“ in das Arbeitsblatt hineingezogen, nachfolgend wird die Verbindung mit der generellen Datentabelle hergestellt. Diese Verbindungen werden in Orange Data Mining hergestellt, indem von der Seite eines Prozesses, beziehungsweise Icons, die Verbindung zum nächsten Prozess gezogen wird. Als nächstes wurden in dem Prozess „Select Columns“ die Spalten „salary_in_usd“ und „job_category“ selektiert, da diese beiden benötigt werden, um die Datenvisualisierungen zu erstellen. Das Selektieren von Spalten wird durchgeführt, indem den gewünschten Spalten von „Ignored“ zu „Featured“ hinzugefügt werden. Das Ergebnis kann veranschaulicht werden, indem ein weiterer Prozess bzw. Icon hinzugefügt wird. Dieser Prozess heißt „Data Tale“ und befindet sich im Reiter „Data“.

Die erste Visualisierung, die erstellt wurde, ist eine sogenannte „Violin Plot“ Visualisierung. Dafür wird vom Reiter „Visualize“ das Icon „Violin Plot“ in das Arbeitsblatt hineingezogen und die benötigte Verbindung mit der vorherigen Datentabelle wird hergestellt. Der „Violin Plot“ wird konfiguriert, indem man das entsprechende Icon doppelt anklickt und so das Fenster öffnet. In dem Fenster wird als Variable die Spalte „salary_in_usd“ ausgewählt, da diese die Original skalierten Gehaltsdaten enthält. Die Kategorien werden festgestellt, indem die Spalte „job_category“ in dem entsprechenden Abschnitt selektiert wird. Als Resultat wird eine Visualisierung generiert, die aussagekräftig ist. Die entsprechenden Achsenabschnitte sind automatisch korrekt gelabelt. Allerdings

besteht nicht die Möglichkeit diese zu editieren. Der Versuch, der Visualisierung einen Titel zu geben, ist auch aufgrund fehlender Funktion gescheitert. Es ist jedoch möglich, die Farbpalette zu verändern. Der ganze Prozess hat einen Arbeitsaufwand von 30 Minuten.

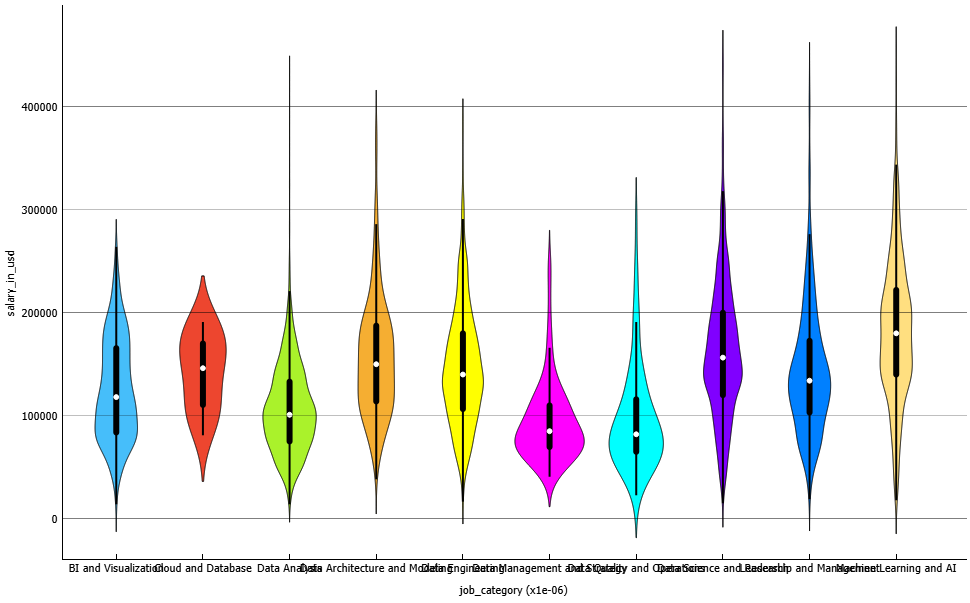

Abbildung 3Violin Plot in Orange

Als zweite Visualisierung wird ein Box Plot erstellt. Dies ist relativ einfach zu machen. Von dem Reiter „Visualize“ wird das Icon „Box Plot“ in das Arbeitsblatt hineingezogen und danach entsprechend mit der Datentabelle verbunden. Um den Box Plot zu konfigurieren, wird genauso vorgegangen wie im Violin Plot. Das Ergebnis ist aussagekräftig. Die entsprechenden Achsenabschnitte sind aber nicht gelabelt und die Möglichkeit diese zu labeln wurde auch nicht gefunden. Ein Titel konnte der Visualisierung auch nicht hinzugefügt werden. Der Box Plot ist ebenfalls, genauso wie der Violin Plot, nur ein einfarbiger Plot. Eine Farbpalette kann nicht hinzugefügt werden. Die Visualisierung ist jedoch aussagekräftig und sieht professionell aus.

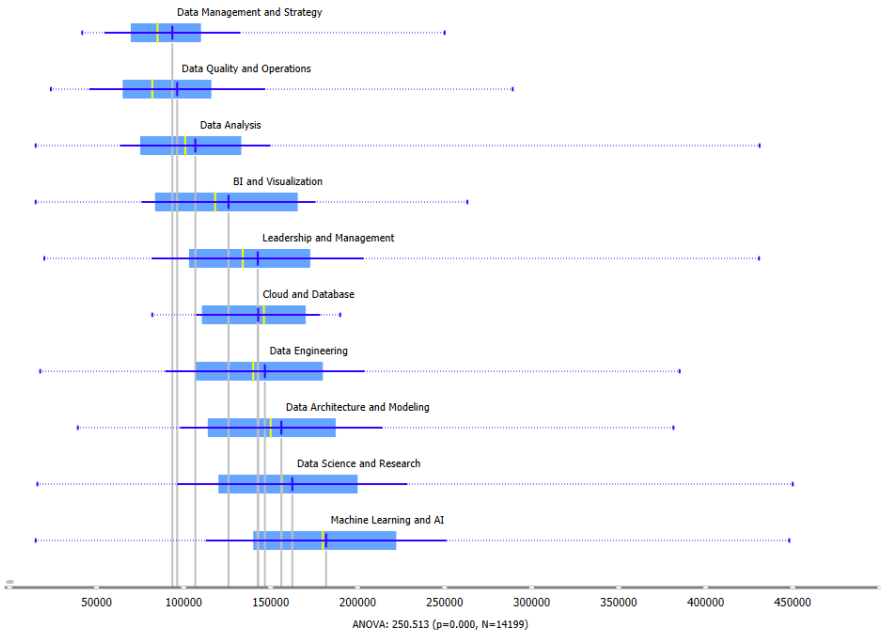

Abbildung 4Box Plot in Orange

Um in Python beide Visualisierungen zu erstellen, gibt es viele Bibliotheken, die benutzt werden können. In diesem Fall wird die Seaborn Bibliothek benutzt. Im Code muss diese Bibliothek erstmal importiert werden und danach kann der Plot mit unterschiedlichen Parametern erzeugt werden. Dafür werden 14 Zeilen Code geschrieben mit einem Arbeitsaufwand von 30 Minuten.

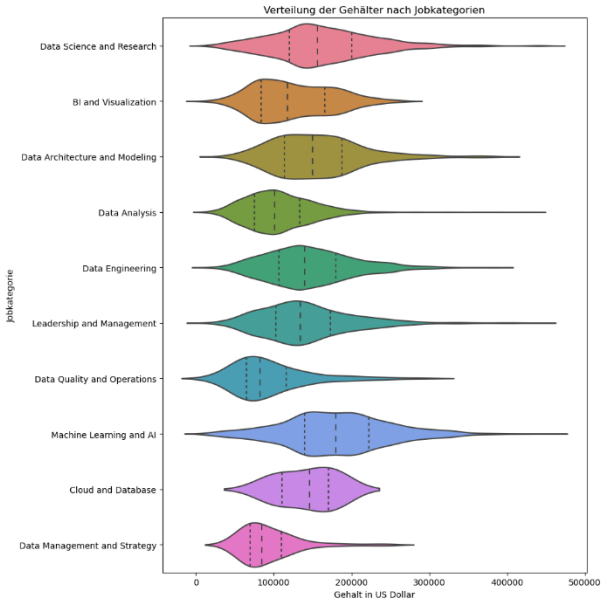

Abbildung 5Violin Plot in Python

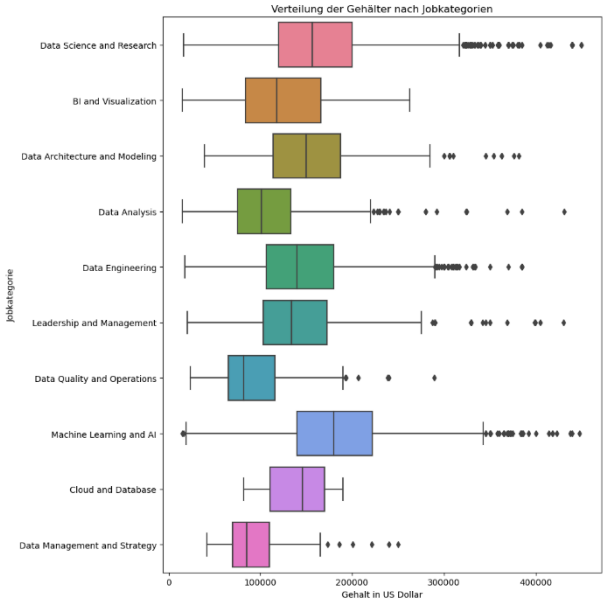

Abbildung 6Box Plot in Python

Alle vier Visualisierungen sind sehr aussagekräftig und beantworten die Fragestellung gleich gut. Es gibt jedoch einige Unterschiede. Ein negativer Aspekt von dem Tool „Orange Data Mining“ ist, dass es zum Stand dieser Arbeit nicht möglich ist, die Achsenabschnitte selbst zu beschriften. Dies ist nicht so optimal, da die Achsenabschnitte von den Namen der Spalten in der Tabelle übernommen werden. Jedoch sind diese in der Regel nicht ideal für die Achsenbeschriftung. Im Python hingegen ist es jedoch möglich, die Achsenabschnitte beliebig zu beschriften und somit die Visualisierungen ein wenig weiter zu personalisieren. Ein wesentlicher Unterschied, der auch wichtig zu betrachten ist, ist die Einstiegsbarriere von Python. Denn um in Python Visualisierungen zu erstellen, sind Programmiervorkenntnisse erforderlich. Im „Orange“ hingegen sind keine Erfahrungen im Programmieren nötig. Ein weiterer relevanter Aspekt, der beachtet werden sollte, ist der Prozess zum Erzeugen der Visualisierung. Denn in Python ist die Übersicht nicht gut. Vor allem wenn der Code von einem nicht Sachverständigen Dritten betrachtet wird. In Orange Data Mining ist die Übersicht hingegen sehr gut. Die Aussage „sehr gut” bedeutet hier, dass auf den ersten Blick die einzelnen Schritte zum Erstellen des Ergebnisses schon erkennbar sind. Das ist besonders wichtig, wenn solche Prozesse für, zum Beispiel, automatisierte Visualisierungen gespeichert oder überarbeitet werden sollen. In Summe sind Violin Plots und Box Plots in Orange Data Mining mit einem Arbeitsaufwand von einer halben Stunde zu erstellen. Sie sehen gut aus und beantworten die Fragestellung. Aber es fehlen Optionen, um diese Plots ein wenig weiter zu personalisieren. In Python

hingegen ist es möglich, fast alles am Plot zu personalisieren. Mit der Seaborn Bibliothek sind nur wenige Zeilen Code nötig, um ein Violin Plot beziehungsweise einen Box Plot zu erstellen. Die Plots beantworten die Fragestellung, aber mit dem Vorteil, diese ein wenig weiter personalisieren zu können. Für jemand mit Programmiervorkenntnisse ist es daher empfehlenswert in Python diese Plots zu erstellen. Aber für jemanden ohne diese Erfahrungen ist es besser in Orange Data Mining, da das Tool es ermöglicht, solche Plots mit wenig Aufwand und ohne benötigte Vorkenntnisse zu erstellen.

Da der Zeitaufwand für das Erstellen des Codes, wie oben beschrieben, ziemlich groß ist, erhält Python hier nur 1 Punkt. In Orange war die Zeit für das Erstellen des Workflows ziemlich ähnlich wie in Python, daher erhält Orange hier auch nur 1 Punkt. Als Anfänger ist die Lernkurve in Python deutlich steiler als bei anderen Tools. Allerdings ist die Community in Python sehr groß, wodurch man Unmengen an hilfreichen Materialien findet. Deswegen erhält Python hier 2 Punkte. In Orange ist die Lernkurve viel geringer als in Python. Alles ist ziemlich intuitiv und auf Oranges Webseite ist auch gute Dokumentation zu finden, weshalb Orange in dieser Kategorie 3 Punkte bekommt. Die Ladedauer war in diesem Beispiel immer unter 30 Millisekunden, weshalb Python hier 3 Punkte bekommt. Bei Orange es ebenfalls unter 30 Millisekunden, daher erhält Orange ebenfalls 3 Punkte. In Python ist die Anzahl der Bibliotheken und Personalisierungsmöglichkeiten praktisch unendlich, was es möglich macht, die Visualisierungen so gut wie gewünscht zu personalisieren. Deshalb erhält Python auch in dieser Kategorie 3 Punkte. In Orange sind die Möglichkeiten sehr beschränkt. Es gibt nur wenige Visualisierungsarten, die erstellt werden können, und diese lassen sich auch nur schwer personalisieren. Aus diesen Gründen bekommt Orange in dieser Kategorie 1 Punkt. Somit bekommt Python insgesamt 9 Punkte. Orange bekommt eine Gesamtpunktzahl von 8 Punkte.

7.3 Arbeitsumfeld und Jobkategorie

Sehr interessant ist auch die Frage, wie sich das Arbeitsumfeld der Angestellten innerhalb der verschiedenen Job Kategorien unterscheidet. Dafür eignet sich ein Kreisdiagramm, beziehungsweise Pie Chart besonders gut, da alle Angestellten im Datensatz einem von nur 3 Work Settings zugeordnet sind, nämlich „Hybrid", „in Person" oder „Remote". Aus diesem Grund kann ein Pie Chart für eine sehr gute Übersicht sorgen.

Zur Erstellung der Plots in Python bietet sich die Spyder IDE an, die im Anaconda Navigator enthalten ist. Das Codieren des gewünschten Ergebnisses war nicht sehr einfach, aber die Seite w3schools.com bietet gute Hilfestellungen zum Umgang mit Plots in Python. Neben den standardmäßigen Imports und dem Einlesen der Datei im Code muss ein Data Frame erstellt werden, in dem alle Kombinationsmöglichkeiten von Job Kategorien und Work Settings gespeichert werden. Danach wird für jede dieser Optionen die Anzahl der Einträge, also der Mitarbeiter gezählt. Nun können 10 Pie Charts ausgegeben werden, für jede Kategorie einer, in denen die Anzahl an Mitarbeitern pro Work Setting die Größe der Segmente des jeweiligen Plots bestimmt.

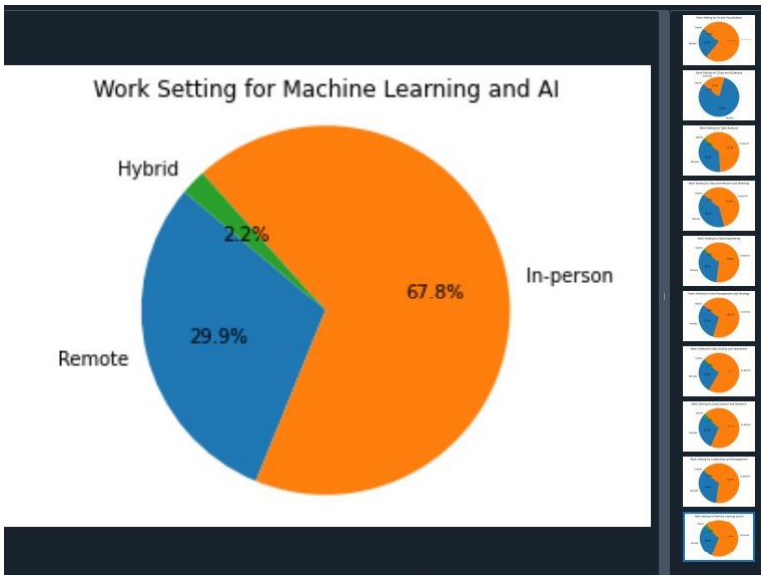

Abbildung 7Pie Chart in Python

Je nachdem, wie fortgeschritten man in Python ist, braucht man für diese Plots unterschiedlich lange. In diesem Projekt wurde bis zur Fertigstellung des Codes ein Zeitaufwand von mehr als 4 Stunden nötig. Allerdings wurden mehrere Ansätze verfolgt. Der Umgang mit der Spyder IDE ist einfach, das Programm unterstützt den Nutzer beim Coden und gibt ausführliche Fehlermeldungen zurück. Zu bemängeln ist, dass die Plots separat gespeichert werden. Das ist einerseits gut, da man jeden Plot einzeln betrachten kann und eine bessere Übersicht hat als beispielsweise in Jupyter Notebook, andererseits ist es Nachteilhaft, denn bei jeder Ausführung des Codes bleiben die alten Plots erhalten und müssen manuell gelöscht werden. Die Ladedauer des Codes bis zur Ausgabe beträgt auf dem Testsystem und in der Spyder Umgebung etwa 0,5 Sekunden.

In Orange ist der Pie Chart nicht im Standardpaket enthalten. Er ist Bestandteil des „Educational" Add-ons, welches sich sehr leicht über den „Options" Reiter in der Orange Oberfläche installieren lässt. Einmal installiert, ist der Pie Chart einfach zu verwenden. Man kann ihn direkt mit dem importierten Datensatz verknüpfen und hat dann innerhalb des Knotens die Möglichkeit, nach allen beliebigen im Datensatz enthaltenen Kategorien zu sortieren. So kann man beispielsweise wie in Python für die Segmente „work setting" und für die Plots „job_category" wählen, um dieselben Plots zu erhalten, nur deutlich übersichtlicher, nebeneinander gelistet. Statt sich informieren zu müssen, wie genau man den Code zu schreiben hat für das gewünschte Ergebnis, benötigt man in Orange höchstens 2 bis 3 Knoten und ein kleines bisschen Verständnis für die Funktionsweise des Tools. Der Arbeitsaufwand in Orange belief sich auf maximal 10 Minuten.

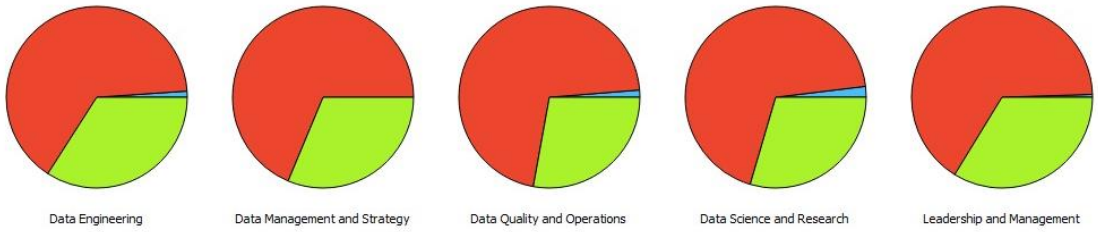

Abbildung 8Pie Chart in Orange

Das Ergebnis in Orange schlägt den Output in Python allerdings um Längen. Es gibt die Möglichkeit, per Checkbox die Segmente der Plots voneinander zu trennen, um für Übersicht zu sorgen. Außerdem lässt sich nicht nur die ursprüngliche Fragestellung beantworten, sondern durch wenige Selektionen lassen sich auch Erfahrungslevel und Firmengröße, Firmenstandort und Angestelltenverhältnis und die anderen Spalten des Datensatzes visualisieren und vergleichen.

Auch bei den Ladezeiten triumphiert Orange, diese sind auf dem Testsystem praktisch nicht vorhanden, was aber auch daran liegen könnte, dass der Pie Chart nicht sehr aufwändig ist. Andere Plots, die getestet wurden, hatten Ladezeiten von bis zu 0,2 Sekunden

Nachteile des Pie Charts in Orange gegenüber schlichtem Python Code gibt es auch. So kann man die Größe der Plots beispielsweise nicht ändern, dies passiert auch nicht automatisch, wenn man den Frame, in dem die Plots ausgegeben werden, manuell vergrößert.

Zusätzlich hat man nicht die Möglichkeit, numerische Daten zu nutzen, ausschließlich kategorische Daten sind zugelassen. Die große Einfachheit schränkt letztendlich die Freiheit bei der Nutzung ein.

Stellt man Orange und Python zur Beantwortung dieser konkreten Fragestellung gegenüber, ist der Sieger ganz klar Orange. In Sachen Bearbeitungszeit konnte das Tool sich gegenüber Python mit einem 20-fach geringeren Aufwand beweisen, hier gehen 3 Punkte an Orange und 1 Punkt an Python, wegen des zu langen Prozesses. Auch im Hinblick auf die Anwenderfreundlichkeit stellt Orange Python in den Schatten. In dieser Kategorie kam Python mit seinem herkömmlichen Code nicht gegen die Technik von Orange an, bei der Knoten mit bestimmten Funktionalitäten miteinander verbunden werden. Auch hier gehen 3 Punkte an Orange und 1 Punkt an Python. Bei der Ladezeit waren die Ergebnisse in etwa gleich gut. Die Ausführung hat jeweils weniger als 1 Sekunde gedauert, weshalb sowohl Python als auch Orange 3 Punkte erhalten. Allerdings liegt bei den Visualisierungen wieder Orange vorne. Nach der Ausgabe der gewünschten Visualisierung ist hier nicht Schluss. Durch wenige Klicks, kann man noch ganz andere Vergleiche ziehen und die Darstellung anpassen. Hier staubt das Tool volle 3 Punkte ab. In Python hingegen sind die Plots eher sperrig, es lässt sich nicht viel damit anfangen und man muss selbst bei kleinen Veränderungen große Teile des Codes überarbeiten. Für die größere Vielfalt an Optionen für die Plots bekommt Python 2 Punkte. Insgesamt steht Orange Python also mit 12 Punkten zu 7 Punkten gegenüber und eignet sich daher deutlich besser für den Pie Chart.

7.4 Verteilung der durchschnittlichen Gehälter nach Unternehmensstandort

Um darzustellen, wie die Verteilung der Durchschnittsgehälter ist, wird eine Choroplethenkarte, oder auch Flächenkartogramm, verwendet. Durch diese Visualisierung wird anschaulich, welche Länder, beziehungsweise welche Regionen das größte Gehalt für Berufe im Bereich Data Science bezahlen. Dies erkennt man, wenn man auf die Verfärbung des jeweiligen Landes schaut. Somit kann man die Länder und Regionen auf einen Blick miteinander vergleichen. Zur besseren Veranschaulichung werden jeweils zwei Karten erstellt. In der ersten werden alle Länder veranschaulicht. Bei der zweiten

Karte wurde das Land Katar entfernt. Grund hierfür ist, dass das durchschnittliche Gehalt dort in einem zu großen Abstand zu den anderen liegt. Dadurch ist der Großteil der Länder im unteren Bereich der Farbskalierung und somit sind diese farblich nur schwer zu unterscheiden. Ohne Katar entzieht sich die Farbverteilung. Zudem ist Katar auf einer Karte, die den ganzen Globus zeigt, nicht zu erkennen, es sei denn man vergrößert die Karte. Unter dieser Voraussetzung wurde beschlossen, Katar in einer zweiten Veranschaulichung zu entfernen.

Um ein Flächenkartogramm in Python zu erstellen, muss man zuerst „geopandas" und „matplotlib" installieren. Nachdem man den Datensatz eingelesen hat, überprüft man, ob alle Ländernamen erkannt werden. Hierfür importiert man „geopandas". Es wird nun eine Liste „unique_countries_data" erstellt, die alle Werte aus der Spalte company-location enthält. Danach wird in der Variable „world" durch „naturalearth_lowres"-Daten die Weltkarte als Geodatenrahmen geladen. Eine weitere Liste „unique_countries_map" erhält alle Ländernamen aus der Spalte „name" von dem Geodatenrahmen „world". Mit Hilfe einer for-Schleife werden alle Ländernamen durchgegangen. Wenn ein Land aus dem Datensatz nicht in dem Geodatenrahmen vorkommt, wird dieser ausgegeben. Als nächstes werden nicht erkannte Ländernamen korrigiert. Dazu werden erst alle Länder in der Spalte „name" von „world" durch „sorted()" alphabetisch sortiert und ausgegeben. Durch „.replace()" korrigiert man alle nicht erkannten Länder in dem Datensatz. Jedoch ist es der Fall, dass nicht alle Länder in „world" enthalten sind. Da diese nur kleine Inselstaaten sind, die auf der Weltkarte nicht zu erkennen sind, werden sie ignoriert. Als nächstes wird „matplotlib.pyplot" importiert. Das Durchschnittsgehalt wird gebildet, indem nach company_location gruppiert und der Durchschnitt von salary_in_usd für jede Gruppe berechnet wird. Das Ergebnis wird in „mean_salary" gespeichert. Mit Hilfe von „name" in „world" und company_location in „mean_salary", wird durch „merge()" „world" mit „mean_salary" verbunden. Die left-Methode sorgt dafür, dass alle Ländernamen in „world" beibehalten werden, auch wenn es kein Durchschnittsgehalt gibt. Alle Länder ohne Gehaltsdaten bekommen dort den Wert Null, damit die Länder später trotzdem abgebildet werden. Jetzt wird der Plot erstellt. Die plot()-Methode von „world" wird verwendet, um die Karte zu zeichnen und das Durchschnittsgehalt findet Verwendung als Parameter für die Farbskalierung. Zuletzt setzt man die Linienbreite, Farbe der Farbskalierung und Kantenfarbe der Länder fest und blendet die Achsen aus. Damit die gleiche

Karte ohne Katar gebildet wird, muss man vor dem Import von „matplotlib.pyplot" die Spalten mit Katar als company_location löschen. Dies wird erreicht durch „.drop()". Danach folgen dieselben Schritte wie bei der Erstellung der ersten Karte.

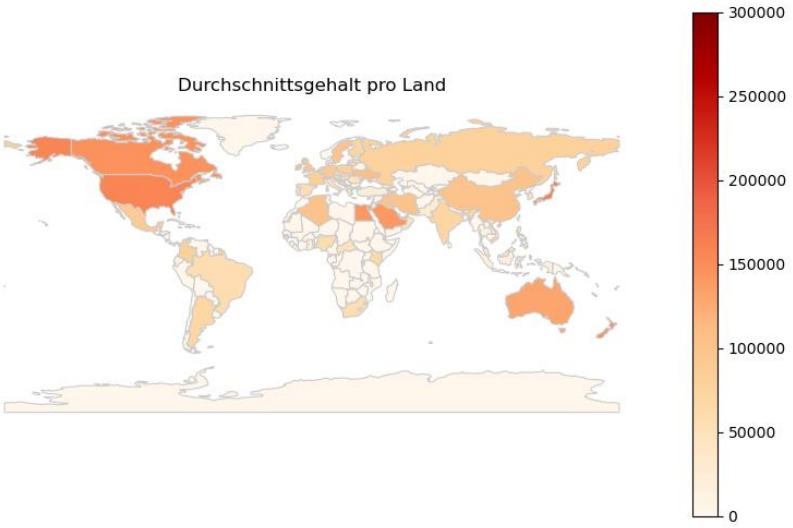

Abbildung 9Karte in Python mit Katar

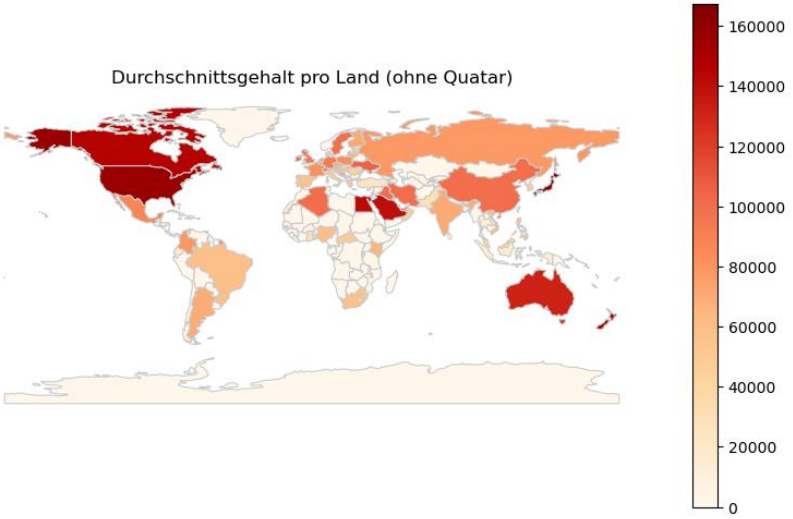

Abbildung 10Karte in Python ohne Katar

Um eine Choroplethenkarte in Orange zu erstellen, braucht man das neue Add-on „Geo". Dieses kann installiert werden, indem man zuerst Orange öffnet. Danach klickt man in der Menüleiste auf Optionen. Dort drückt man auf Add-ons. Nun wird ein Fenster geöffnet, in dem alle Add-ons aufgelistet sind. Man wählt nur noch „Geo" aus und drückt auf „OK". Jetzt wird das Add-on installiert. Zur Erstellung einer Karte braucht man Daten

zum Längen- und Breitengrad. In der verwendeten CSV-Datei gibt es diese Werte nicht. Jedoch stehen in der Spalte company_location die Ländernamen. Daher verbindet man „CSV File Import" mit „Geocoding". Dies findet man in dem neu installierten Add-on „Geo". Wenn sich das Fenster von „Geocoding" öffnet, wählt man „Encode region names into georaphical coordinates". „Region identifier" ist company-location und „Identifier type" ist „Country name". Das Programm erkennt die Länder nun automatisch und ordnet ihnen Koordinaten zu. Somit entstehen neue Spalten mit Längen- und Breitengrad in der Datentabelle. Wenn ein Name nicht erkannt wird, wird dies angezeigt. Im Fall dieses Datensatzes „Türkiye". Nun kann man daneben den Namen benutzerdefiniert ändern. Aus „Türkiye" wird nun „Turkey". Falls man wissen möchte, welcher Stand des Globus verwendet wird, benutzt man „Geo Transform". Zur Visualisierung dieses Datensatzes wird die Karte von Slovenia 1996 von Slovene National Grid benutzt. Zur Erstellung der Karte mit Katar wird zum Workflow „Choropleth Map" angehängt. In dem geöffneten Fenster werden beim „Map setting" die vorhin automatisch generierten Längen- und Breitengrade erkannt und gleich ausgewählt. Bei „Control" wählt man bei „Value" salary-in-usd und für „Show" den „Mean". Jetzt wird die Karte erstellt. Um die gleiche Karte ohne Katar zu erstellen, fügt man vor „Choropleth Map" die Funktion „Select Rows" in den Workflow ein. Hier wählt man „company_location is not Qatar" aus. Nun werden alle Zeilen, in denen company_location Katar ist, entfernt. Falls man eine andere Farbskala haben möchte, schaltet man „Color" zwischen „Geocoding" und „Choropleth Map". Dort kann man bei salary_in_usd zwischen verschiedenen vorbestimmten Farbverläufen wählen.

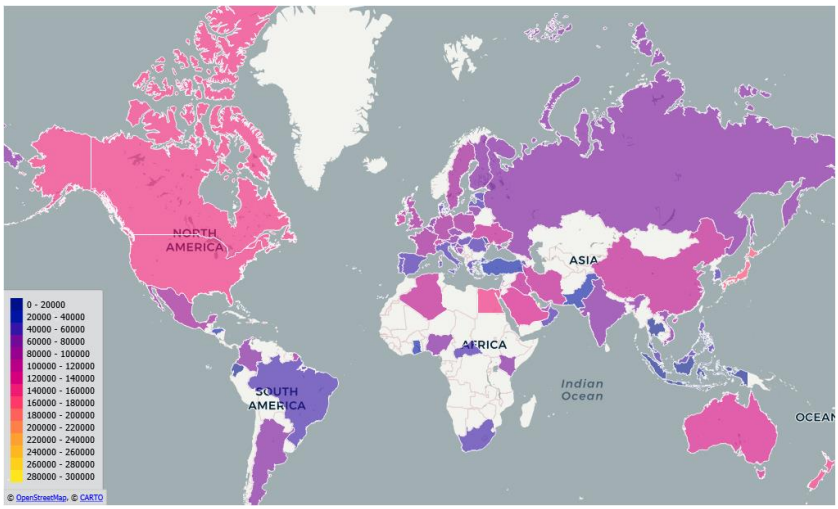

Abbildung 11Karte in Orange mit Katar

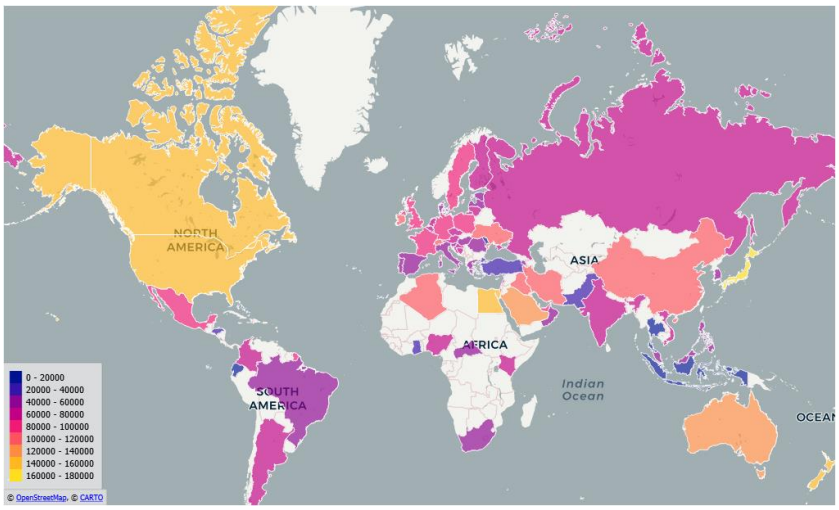

Abbildung 12Karte in Orange ohne Katar

Vergleicht man Python mit Orange fällt auf, dass man für Python mehr Anwendungen installieren muss. In Orange reichte ein Add-on und in Python brauchte man neben „pandas" auch „geopandas" und „matplotlib.pyplot". Im Aspekt des Zeitaufwandes war Orange um einiges schneller. Für Orange benötigte man ungefähr 15 Minuten, während es in Python 2 Stunden dauerte. Beide Zeitangaben entstanden ohne Vorkenntnisse zur Visualisierung. Grund hierfür ist, dass man nur die richtigen Knoten in den Workflow einbinden muss und die richtigen Variablen aus einer Auswahlliste wählt. In Python muss man mehrere Zeilen Code ausschreiben. Dies kostet mehr Zeit. Zudem ist ein großer Nachteil, dass man mit Python drei zusätzliche Schritte ausführen muss, um die falschen Ländernamen zu korrigieren. Bei Orange erledigt sich dies alles mit dem Schritt „Geocoding". Zudem hat Orange erkannt, welche Karte in dem Datensatz verwendet wurde, während in Python nur eine zur Verfügung stand. Deshalb muss man in Python mehr Länder korrigieren. Mit dem Blick auf die Anwenderfreundlichkeit ist Orange einfacher zu handhaben. Selbst mit wenigen Vorkenntnissen, wie man die Choroplethenkarte erstellt, findet man dies mit ein wenig Ausprobieren heraus. In Python jedoch braucht man die Vorkenntnisse, um zu wissen, welche Methoden man benutzen muss. Wenn man nicht schon vorher weiß, wie man so eine Karte erstellt, ist Recherche erforderlich. Bei der Ladedauer der Karten braucht Python ein paar Sekunden, während sie bei Orange sofort geladen ist.

Ob die entstandene Karte in beiden Fällen der Wirklichkeit entspricht, lässt sich leider nicht nachprüfen. Grund hierfür ist, dass man keine Werte für das durchschnittliche Gehalt findet, welches alle Bereiche im Berufsfeld Data Science einschließt. Zu finden sind nur Mittelwerte aus den einzelnen Ländern mit gewissen Berufsbezeichnungen, wie zum Beispiel Data Engineer. Ob die Karte nun die wirklichen Durchschnittsgehälter auf der Weltkarte zeigt, kann damit nicht auf einfache Art und Weise nachgewiesen werden.

Beiden Tools werden nun jeweils Punkte vergeben. In der Kategorie Zeitaufwand bekommt Python 1 Punkt und Orange 2 Punkte. in Python brauchte es mehrere Überarbeitungen des Codes um zu dem gewünschten Ergebnis zu kommen. Zu dem musste mehrere Bibliotheken importiert werden. In Orange brauchte das Installieren des Add-ons etwas Zeit. Das Finden der richtigen Knoten ging jedoch schnell. Bei der Anwenderfreundlichkeit bekommt Python nur 1 Punkt, da ein Einsteiger Schwierigkeiten gehabt hätte. In Orange kann ein Beginner schnell durch Ausprobieren eine Karte erstellen, weshalb Orange 3 Punkte bekommt. In der Ladedauer sind beide Tools schnell, jedoch brauchte Python beim ersten Laden etwas länger. Deshalb bekommt Python 2 und Orange 3 Punkte. Beide Tools haben im Bereich der Funktionalität Einschränkungen. Python hat diese bei der Wahl der Karte (gemeint ist aus welchem Jahr und Land die Karte stammt) und Orange in der Wahl der Farbskala. Aus diesem Grund bekommen beide 2 Punkte. Insgesamt erhält Python 6 und Orange 10 Punkte. Anhand dessen eignet sich Orange besser zur Erstellung einer Choroplethenkarte.

8 SWOT Analyse Orange

Kommen wir nun zur SWOT Analyse von unserem Visualisierungstool Orange Data Mining. Dieses hat einige Stärken vorzuweisen. Beispielsweise benötigt man zur Arbeit mit Orange praktisch keine programmiertechnischen Vorkenntnisse. Die übersichtliche Gestaltung des Tools in Kombination mit den online zur Verfügung gestellten Tutorial-Videos ermöglichen ein gutes Verständnis der Funktionalität von Orange. Außerdem hat man als Nutzer die Option, zusätzliche Add-ons zu installieren, die den Umfang des Tools massiv erweitern. In Orange kann man gut als Team zusammenarbeiten, da die Workflows so einfach wie ein Bilderbuch zu lesen sind, wodurch die Reproduzierbarkeit steigt.

Natürlich sollte man erwähnen, dass das gesamte Tool vollumfänglich kostenlos zur Verfügung steht, es keine versteckten Gebühren gibt und es zusätzlich open Source ist. Zu allen Knoten und Add-ons existieren ausführliche Erklärungen bereits im Tool selbst, sodass der Arbeitsaufwand minimiert wird. Die Arbeit mit dem Tool führt aufgrund der Einfachheit zu ständigen Erfolgserlebnissen.

Doch Orange bringt auch einige Schwächen mit sich. Das Programm startet beim ersten Mal ziemlich langsam, es braucht selbst auf schnellen Systemen zur Initialisierung bis zu 10 Sekunden. Zudem ist Orange mehr oder weniger unbekannt, es existiert kaum Literatur oder Informationen im Internet, anders als bei anderen Tools. Wenn die Erklärungen in Orange selbst und die Videos online nicht ausreichen, hat man Schwierigkeiten, ein Problem zu lösen. Der Fokus liegt auch eindeutig auf Data Mining und Machine Learning, nicht aber auf Visualisierungen. Diese sind zwar auch ein großer Aspekt, allerdings bieten andere Tools in diesem Segment mehr Optionen. Schade ist auch, dass man in Orange nicht gemeinsam arbeiten kann, es lässt nur einen gleichzeitigen Nutzer zu. Daher muss man sich beim gemeinsamen Arbeiten die Workflows hin und her schicken.

Welche Chancen Orange bietet, lässt sich schnell beantworten. Machine Learning gepaart mit Simplifikation sollte ausreichen, um das Potential eines solchen Tools zu verdeutlichen. Entwurf und Training von ML - Modellen ist möglich und auch die Visualisierungsoptionen lassen sich einfach ausbauen. Wenn mehr Nutzer zu Orange dazu stoßen und die Community wächst, werden außerdem neue Add-ons und Patches entwickelt.

Orange steht unter Konkurrenzdruck. Es gibt viele gute Optionen zur Erstellung von Visualisierungen wie Power BI oder Tableau. Die Frage ist, wie lange Orange noch die Nase vorne haben kann. Dadurch, dass das Tool open Source ist, gibt es leider keinen Support, somit ist man bei der Nutzung auf sich gestellt.

9 SWOT-Analyse Python

Python erweist als Visualisierungstool verschiedene Vor- und Nachteile, diese werden nun in einer SWOT-Analyse zusammengefasst.

Wir konnten feststellen, dass Python verschiedene Stärken als Visualisierungstool besitzt. Die Flexibilität beim Einlesen von Datensätzen ist sehr hoch; es ist möglich, die unterschiedlichsten Formate einzulesen, dafür ist nur die richtige Bibliothek nötig. Dies führt auch zu einem weiteren Punkt: die umfangreichen Bibliotheken, die in Python zur Verfügung stehen. Dank der sehr großen Community gibt es Unmengen an Bibliotheken, die uns zu unserem Zweck helfen können und uns die Arbeit sehr erleichtern, da nicht alles selbst programmiert werden muss. Denn die Methoden der Bibliotheken stehen einem zur Verfügung. Beispiele dafür sind die Matplotlib- oder Seaborn-Bibliotheken. Ein weiterer wichtiger Vorteil ist, dass Python als Programmiersprache kostenlos ist, ebenso wie die meisten der verfügbaren Bibliotheken. Dies stellt einen ökonomischen Vorteil dar, der natürlich wichtig zu beachten ist.

Python als Tool besitzt jedoch auch einige Schwächen. Die erste und wichtigste ist wahrscheinlich, dass Programmierkenntnisse erforderlich sind, da Python im Grunde genommen eine Programmiersprache ist. Demzufolge müssen die benötigten Befehle programmiert werden, was für jemanden, der die Programmiersprache nicht kennt, eine große Einstiegsbarriere darstellen kann und möglicherweise zu einem sehr hohen Arbeitsaufwand führt. Ein weiterer negativer Punkt ist die geringe Übersichtlichkeit. Da alles in Code geschrieben wird, ist es leicht für einen normalen Studenten, den Überblick zu verlieren, besonders nach 100+ Zeilen Code. Wie bei jeder Programmiersprache ist die Fehleranfälligkeit sehr groß, sowohl in Bezug auf den geschriebenen Code als auch auf das Auffinden und Verbessern von Fehlern. Dies ist oft sehr aufwendig und gilt daher als negativer Punkt.

Python bringt jedoch auch einige Chancen mit sich. Zum einen ist es als Standardprogrammiersprache für Data Analysis sehr gut in verschiedene andere Anwendungen zu integrieren. Dies ist auch im Hinblick auf das Maschinellen Lernen und die Integration mit KI sehr wichtig und zukunftsrelevant. Eine weitere Chance, die genutzt werden kann, ist die Möglichkeit, sehr personalisierte und Echtzeit-Dashboards zu erstellen, da im Grunde in Python alles programmiert werden kann. Dies öffnet die Tür zur Entwicklung eigener Bibliotheken, die im eigenen Unternehmen verwendet werden können, um somit den Stil des Unternehmens bestmöglich zu repräsentieren.

Als Bedrohung ist zu erkennen, dass, wenn spezialisierte Software mit geringem Arbeitsaufwand dasselbe wie Python leisten kann, sich der Arbeitsaufwand, eigene Visualisierungen in Python zu erstellen, möglicherweise nicht lohnt. Auch ist zu beachten, dass im Gegensatz zu anderen Cloudbasierten Services die Performance von Python von der eigenen Hardware abhängt. Dies bedeutet, dass bei großen Datenmengen eine erhebliche Investition in Hardware nötig sein kann, um möglichst effizient arbeiten zu können.

10 Fazit

Um nun abschließend eine quantifizierte Aussage zu treffen, welches Tool besser ist, können die Punkte der gleichgewichtigen Nutzwertanalyse zusammengezählt werden. Bei einer Skala von 1-3 Punkten pro Kategorie mit 4 Kategorien und je einer Auswertung von 4 verschiedenen Testern, können beide Tools jeweils maximal 48 Punkte, und minimal 16 Punkte erreichen. Pro Kategorie können 12 Punkte erreicht werden. Um diese recht krummen Zahlen besser lesbar zu machen, wird stattdessen eine gerundete Prozentzahl der möglichen Punkte errechnet. Python erreicht insgesamt 63% der Gesamtpunkte, Orange erreicht 85%. Somit ist im Test mit 4 verschiedenen Data Science Studenten mit ein paar Vorkenntnissen in Python und keinen Vorkenntnissen in Orange, trotzdem Orange um 22% besser. Über die 4 Kategorien hinweg sind auch einige Erkenntnisse zu machen, so erreicht Python beispielsweise bei Zeitaufwand der Programmierung bei allen Auswertungen nur die Mindestpunktzahl mit 33% und Orange aber 75%. Außerdem hervorzuheben ist, dass Orange sowohl in der Kategorie Anwenderfreundlichkeit als auch Ladedauer 100% erreicht. Die Anwenderfreundlichkeit ist bei Python hingegen mit 42% nur knapp über der Mindestpunktzahl. In der Kategorie Funktionalität schneidet Python aber mit 83% gegen 67% besser ab.

Somit lässt sich in unserem Anwendungsfall von Studenten mit eher mäßigen Vorkenntnissen Orange gut empfehlen. Generell auch für Fälle, in denen schnell einfache Visualisierungen ohne besondere Vorkenntnisse erstellt werden sollen, lässt sich Orange gut einsetzen. Aufgrund der größeren Funktionalität von Python ist für umfangreiche Analysen von Experten im Bereich Data Science aber Python vorzuziehen, da es mehr Möglichkeiten bietet.

11 Quellen

Über Orange:

1) https://fri.uni-lj.si/en/laboratory/biolab-27 (Autor unbekannt, ohne Jahr, letzter Zugriff 09.05.2024)
2) https://orangedatamining.com/citation/ (Autor unbekannt, ohne Jahr, letzter Zugriff 09.05.2024)
3) https://jmlr.org/papers/volume14/demsar13a/demsar13a.pdf (Janes Demsar et al., 2013, letzter Zugriff 09.05.2024)

Quelle Datensatz:

1) https://www.kaggle.com/datasets/hummaamqaasim/jobs-in-data (Hummaam Qaasim, 2024, letzter Zugriff 09.05.2024)
2) https://ai-jobs.net/salaries/download/ (Autor unbekannt, ohne Jahr, letzter Zugriff 09.05.2024)
3) https://ai-jobs.net/about/ (Autor unbekannt, ohne Jahr, letzter Zugriff 09.05.2024)
4) https://foorilla.com/ (Autor unbekannt, ohne Jahr, letzter Zugriff 09.05.2024)
5) https://www.apollo.io/companies/foorilla/5da4bc4193d8fb00015570b2 (Autor unbekannt, ohne Jahr, letzter Zugriff 09.05.2024)