Dokumentation – Verbrechen in Baltimore

Philipp Nowak | Matr.-Nr.: 2174358 | GitHub-Benutzername: phinowa

Einleitung und Projektidee

Im Rahmen dieses Projektes möchte ich die Entwicklung von Verbrechen beziehungsweise der Sicherheitslage in Baltimore in den letzten Jahren untersuchen.

Grundlage für das Projekt ist eine CSV-Datei der US-Regierung (http://catalog.data.gov/dataset/crime-safety-2010-2013), die Auskunft über die Kriminalitätsrate und verschiedene Verbrechen wie Gewalt, Diebstahl, häusliche Gewalt und Schusswaffentötung in sämtlichen Bezirken von Baltimore gibt.

Die 620.000 Einwohner umfassende Ostküstenstadt der USA „hat unter den US-amerikanischen Großstädten am stärksten mit Armut, Verwahrlosung, Drogenabhängigkeit und Suburbanisierung zu kämpfen“ (o. A., 2016) und ist für diese Untersuchung deshalb besonders interessant.

Unter anderem möchte ich Fragen beantworten wie: Hat die Verbrechenslage im Laufe der letzten Jahre zu- oder abgenommen? In welchen Stadtbezirken lebt es sich am sichersten, in welchen am gefährlichsten? Wie ist die Entwicklung im „sichersten“ und „gefährlichsten“ Bezirk hinsichtlich mehrerer Faktoren? Und wie sieht die Entwicklung bezüglich der Faktoren in ganz Baltimore über die Jahre hinweg aus?

Am Ende folgt eine abschließende Zusammenfassung und Diskussion der Ergebnisse, in der ich mögliche Ursachen für diese betrachte, indem ich für auffällige Bezirke geografische Hintergrundchecks durchgeführt habe.

Definitionen, Hintergrundinformationen, Begründungen

Die Webseite http://catalog.data.gov/dataset habe ich im Laufe des Projektseminars Informetrie kennengelernt, weshalb ich für dieses Projektseminar die Idee hatte, eine Auswertung zu einem interessanten Datensatz durchzuführen. Auf der Seite befinden sich tausende Datensätze, die von der US-Regierung öffentlich zur Verfügung gestellt wurden.

Für die Darstellung von Diagrammen bin ich letztes Semester im Projektseminar Informetrie mit matplotlib in Berührung gekommen, weshalb ich mich für die Darstellung in diesem Projekt in ein anderes System reingearbeitet habe. Im Projektplan hatte ich beispielhaft Pandas (http://pandas.pydata.org/) und Seaborn (https://stanford.edu/~mwaskom/software/seaborn/) angegeben, letztendlich habe ich mich für zweiteres entschieden.

Im Seminar stellte sich heraus, dass die Ursprungsidee vom Umfang her nicht reichen würde, weshalb ich den Projektplan um weitere Berechnungsfunktionen für mehr Diagramme und eine Einbettung der Auswertung und Diagramme in eine selbst erstellte Benutzeroberfläche erweitert hatte. Für die GUI hatte ich im Projektplan noch an Tkinter (https://wiki.python.org/moin/TkInter) gedacht, schlussendlich habe ich es jedoch mit JavaFX (http://docs.oracle.com/javase/8/javase-clienttechnologies.htm) umgesetzt, wovon ich mir mit mehr Aufwand ein schöneres Ergebnis versprochen habe. Den Scene Builder habe ich dabei nicht genutzt, sondern „von Hand“ programmiert.

Die Diagramme habe ich dementsprechend mit Python programmiert, die Benutzeroberfläche mit Java.

Als Editoren haben dabei Geany (https://www.geany.org/) für Python und IntelliJ Idea (https://www.jetbrains.com/idea/) für Java fungiert.

Das Bild im Startbildschirm der Benutzeroberfläche ist ein Ausschnitt einer Zeichnung des Hafens von Baltimore von Calvert Koerber namens „*Batimore Inner Harbor*“ (http://images.fineartamerica.com/images-medium-large-5/batimore-inner-harbor-calvert-koerber.jpg).

Projektplan/Projektaufbau

Python-Programm

Zuallererst musste die CSV-Datei eingelesen werden [Code Zeilen 199-203], wofür der „import csv“ vonnöten war.

Um mit dem Datei-Inhalt arbeiten zu können, mussten noch ein paar Schritte getan werden:

- Listen definieren, mit denen ich arbeiten möchte [142-196] (Diese sind zunächst noch leer)
- Listen füllen [206-260] (Auf welche Spalte der CSV-Datei möchte ich für welche Berechnung zugreifen können?)
- Störende Spaltenüberschriften entfernen für die Listen [263-316] (Wenn ich Datensätze möchte, kann ich die Spaltenüberschrift für die Berechnung nicht gebrauchen)
- Umwandeln der Strings in Gleitkommazahl oder Integer, damit mit ihnen gerechnet werden kann [324-382]

Die Listen sind nun bereit für Berechnungen. Folgende 12 Funktionen [7-139] wurden dafür geschrieben:

1. Durchschnitt
   ➔ Berechnet den Durchschnittswert der Werte einer Liste.
2. Median
   ➔ Berechnet den Median der Werte einer Liste, d.h. den sich in der Mitte befindenden Wert.
3. Modalwert
   ➔ Berechnet den Modalwert der Werte einer Liste, d.h. den am häufigsten vorkommenden Wert.
4. Varianz
   ➔ Berechnet die Varianz der Werte einer Liste, d.h. ein Maß für die Abweichung vom Mittelwert.
5. Standardabweichung
   ➔ Berechnet die Standardabweichung der Werte einer Liste. Als Grundlage dient die Varianz, heraus kommt, wie hoch die Werte durchschnittlich vom Mittelwert entfernt liegen – egal ob nach unten oder oben hin.
6. Spannweite
   ➔ Berechnet die Spannweite der Werte einer Liste, d.h. den Abstand zwischen dem kleinsten und größten Werts.
7. Prozentzahl der Werte einer Liste unter Wert „50“
   ➔ Berechnet, wie viel Prozent der Werte einer Liste unter dem Wert „50“ liegen. Dies war angedacht, um die Entwicklung über die Jahre hinweg herauszufinden, wie viel Prozent der Bezirke pro Jahr eine Kriminalitätsrate als 50 haben, was ein recht kleiner Wert ist.

8. Top-10 der Bezirke
   ➔ Berechnet die Top-10 der Werte einer Liste, d.h. welche Bezirke mit den dazugehörigen Werten die höchsten Werte haben.
9. Flop-10 der Bezirke
   ➔ Berechnet die Flop-10 der Werte einer Liste, d.h. welche Bezirke mit den dazugehörigen Werten die kleinsten Werte haben. Bei der Top- und Flop-Funktion werden zum ersten Mal bei den Funktionen nicht nur die Werte einer Liste eingelesen, sondern auch die dazugehörigen Namen der Bezirke. Die beiden Funktionen entstanden in Zusammenarbeit mit Karoline Jüttner.
10. Verbrechensrate der Bezirke gestiegen
    ➔ Berechnet, wie oft die Werte einer Liste sich im Vergleich zum jeweiligen „Vorjahr“ erhöhen, d.h. es werden fünf Listen eingelesen und wenn sich ein Wert einer Liste an einer bestimmten Position bei der zu vergleichenden Liste jeweils erhöht, wird der Counter erhöht.
11. Verbrechensrate der Bezirke gesunken
    ➔ Berechnet, wie oft die Werte einer Liste sich im Vergleich zum jeweiligen „Vorjahr“ senken, d.h. es werden fünf Listen eingelesen und wenn sich ein Wert einer Liste an einer bestimmten Position bei der zu vergleichenden Liste jeweils senkt, wird der Counter erhöht.
12. Prozentuale Veränderung
    ➔ Berechnet die prozentuale Veränderung der Werte zweier Listen, d.h. z.B. um wie viel Prozent die Kriminalitätsrate in Bezirk X im Jahr 2014 im Vergleich zum Jahr 2010 gestiegen/gesunken ist.

Nach dem Schreiben der Funktionen können die Berechnungen angestellt werden [385-527]. Dabei werden die Ergebnis-Listen von einer Funktion teilweise in eine andere Funktion eingesetzt, beispielsweise um eine Top-10-Liste der prozentualen Veränderung der Werte zweier Listen herauszufinden.

Nun können die Diagramme endlich erstellt werden [530-1088]. Als Imports werden numpy (für Umrechnung von Liste in Array), sowie seaborn und matplotlib für die Darstellung des Diagramms benötigt. Den Code-Aufbau werde ich anhand eines Balkendiagramms kurz beschreiben:

1. Stil-Einstellung und Anlegen des Diagramms, mit Angabe der Größe in Breite und Höhe [540-541]
2. Labels für die X-Achse sowie die „neu errechneten“ Werte einlesen, die dargestellt werden sollen [544-545]
3. Liste in Array umwandeln, da die Werte in normalen Listen mit Seaborn nicht kompatibel sind [548]
4. Informationen für Diagramm einlesen (Labels, Werte, Farben) [549]
5. Überschrift definieren und über das Diagramm setzen [550-551]
6. Letzte Design-Einstellungen [554-555]
7. Fertig!

Bei Liniendiagrammen [Bsp. 968-1006] werden die Schritte 2, 3 (auch für Labels) und 4 für jeden einzelnen Datensatz separat ausgeführt.

Die Namen in der CSV-Datei der Bezirke musste ich kürzen – teilweise waren mehrere zusammengefasst – damit auf den Diagrammen nichts abgeschnitten wurde [319-321]. Per *plt.xticks(rotation= _)* wurde die Neigung der Bezirks-Namen bei den Diagrammen erzeugt.

Ganz zum Schluss erfolgt die Ausgabe der Diagramme [1092]. Dabei werden die insgesamt 30 Diagramme in der Reihenfolge ausgegeben, wie sie programmiert wurden – was nicht die Reihenfolge ist, in der ich sie bei meiner Auswertung genutzt habe.

Java-Programm

Mein Programm, das die Benutzeroberfläche für die Auswertung der zuvor erstellten Diagramme darstellt, lässt sich in drei Arten von Java-Klassen einteilen:

- Main-Klasse
- Master-Klasse
- Menü-Klassen

Mit der Main-Klasse wird das Programm gestartet, einmalig die Fenstergröße und Einstellungen wie der Fenstertitel und dass die Größe fix ist, d.h. nicht verändert werden kann, definiert. Außerdem bezieht die Main-Klasse die Instanzen aller Menü-Klassen, damit sie angezeigt werden.

Die Master-Klasse wiederum verwaltet sämtliche getter und setter aller Menü-Klassen. Die setter nimmt sich die Main-Klasse, die getter werden bei den Menüklassen gebraucht, damit zwischen ihnen hin- und hergeschaltet werden kann.

Es gibt insgesamt 34 Menü-Klassen. Eins davon bildet das Start-Fenster, in das das zuvor erwähnte Bild eingebunden wurde. Als Layout habe ich die GridPane gewählt, um individuell die Inhalte setzen zu können, damit sie mit dem Hintergrundbild harmonieren. Nach dem Start-Fenster (Menu) folgt eine Einleitung in das Thema (Menu_einleitung), ehe die Diagramme nacheinander ausgewertet werden (Menu_1 bis Menu_30). Den Schluss rundet das zweiteilige Fazit ab (Menu_fazit_1 und Menu_fazit_2).

Der Aufbau der Menü-Klassen ähnelt sich: Nach den nötigen Imports folgt die Initialisierung von Inhalten wie Pane, Labels und Buttons, das Füllen derer mit Inhalten, die eventuelle ID-Setzung für die CSS-Datei z.B. für Überschriften, das Ordnen von Komponenten, der Zugriff auf besagte CSS-Datei aus dem Ressourcen-Ordner, und zum Schluss die Registrierung von Listenern.

Die mit dem Python-Programm erstellten Diagramme wurden als Bilddateien gespeichert, sodass das Java-Programm auf sie im Ressourcen-Ordner zugreifen kann.

Die restlichen Menüs nutzen keine GridPane, sondern eine VBox (Vertical Box), d.h. die Inhalte sind untereinander angeordnet.

Anleitung zur Ausführung/Benutzung

Die Benutzeroberfläche muss nur mit einem Doppelklick gestartet werden. Sie liegt als JAR-Datei vor, weshalb zum Starten Java (https://www.java.com/de/download/) auf dem Rechner installiert sein sollte. Ist das Programm gestartet, wird per Mausklick auf die Buttons navigiert. Auf „Weiter“ klickt man sich durch die Auswertung bis hin zum zweiten Teil des Fazits, wo man wieder zum Start-Bildschirm gelangt. Das dortige „Beenden“ schließt das Programm. Mit den „Zurueck“-Buttons hat man jederzeit die Möglichkeit, nicht nur nach vorne, sondern auch zurück zu navigieren.

Wer sich die für die Auswertung erstellten Diagramme ausgeben lassen möchte, benötigt Python 2 (https://www.python.org/downloads/). Da die Diagramme mit

Seaborn erstellt wurden, muss dieses vorher in cmd mit folgendem Befehl installiert werden: *pip install seaborn*

Starten lässt sich die Python-Datei dann beispielsweise per Öffnen mit Geany und der Aktion "Ausführen".

Reflexion

Das Arbeiten mit zwei verschiedenen Programmiersprachen – Python und Java – war für mich eine gute Auffrischung und hat mir Spaß bereitet, ebenso wie die eigenständige Auswertung und das Aufstellen von Schlussfolgerungen rund um ein Thema, das mich sehr interessiert hat.

Das Arbeiten am Python-Programm lief verhältnismäßig problemlos. Gut war, dass ich die meisten Schritte, was für die Verarbeitung getan werden muss, bereits im Projektplan detailliert aufgeschrieben hatte. Für den GUI-Teil war dort noch nichts festgehalten, weshalb bei der Programmierung mit Java eher das eine oder andere kleine Problem auftrat, die jedoch alle durch Ausprobieren und Internetrecherche gelöst werden konnten.

Mein Ziel, die Entwicklung von Verbrechen beziehungsweise der Sicherheitslage in Baltimore in den letzten Jahren auf Grundlage von einer CSV-Datei mithilfe von selbst erstellten Diagrammen und einer ordentlichen Benutzeroberfläche zu untersuchen, habe ich erreicht. Alle selbst gestellten Antworten konnte ich beantworten und sogar Erkenntnisse darüber hinaus erzielen. Ich habe viele Stunden und Tage in das Projekt investiert und bin jetzt glücklich, dass ich es fertig geschafft habe.

Literaturverzeichnis

Wikipedia.de (2016). *Baltimore*. Abgerufen am 23.09.2016, von https://de.wikipedia.org/wiki/Baltimore.