TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

# Aplikace pro hledání skrytých závislostí v datech o kriminalitě

## Bakalářská práce

*Studijní program:* N2612 – Elektrotechnika a informatika
*Studijní obor:* 1802T007 – Informační technologie

*Autor práce:* **Filip Prádler**
*Vedoucí práce:* Ing. Bc. Marián Lamr Ph.D.

Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

**Zadání bakalářské práce**

# Aplikace pro hledání skrytých závislostí v datech o kriminalitě

| | |
|---|---|
| *Jméno a příjmení:* | **Filip Prádler** |
| *Osobní číslo:* | M19000038 |
| *Studijní program:* | B0613A140005 Informační technologie |
| *Specializace:* | Aplikovaná informatika |
| *Zadávající katedra:* | Ústav mechatroniky a technické informatiky |
| *Akademický rok:* | 2022/2023 |

**Zásady pro vypracování:**

1. Vyberte na základě rešerše zahraničních existujících projektů nejvhodnější postupy a algoritmy k analýze a hledání skrytých závislostí v datech o kriminalitě.
2. Navrhněte a realizujte způsob uložení a distribuce dat pro rozšíření datové matice o další atributy. Využijte veřejně dostupné datové zdroje a API pro rozšíření stávajících dat.
3. Naprogramujte aplikaci pro analýzu dat a hledání skrytých závislostí v kriminálních datech. Aplikace bude podporovat import, export dat, manipulaci s daty a vhodné grafické výstupy.
4. Do aplikace implementujte vybrané vhodné algoritmy pro hledání skrytých závislostí v datech o kriminalitě.

| | |
|---|---|
| *Rozsah grafických prací:* | dle potřeby dokumentace |
| *Rozsah pracovní zprávy:* | 30-40 stran |
| *Forma zpracování práce:* | tištěná/elektronická |
| *Jazyk práce:* | Čeština |

**Seznam odborné literatury:**


[1] WITTEN, I. H. a Frank EIBE. Data mining: practical machine learning tools and techniques. Fourth Edition. Cambrige: Morgan Kaufmann, 2017. ISBN 9780128042915.

[2] BERKA, Petr. Dobývání znalostí z databází. Praha: Academia, 2005. ISBN 8020010629.

[3] SUMMERFIELD, Mark. Rapid GUI programming with Python and Qt: the definitive guide to PyQt programming. Upper Saddle River, NJ: Prentice Hall, c2007. ISBN 978-0-13-235418-9.

[4] RAMALHO, Luciano. Fluent Python: clear, concise, and effective programming. Second edition. Sebastopol: O' Reilly, 2022. ISBN 978-1-492-0563


| | |
|---|---|
| *Vedoucí práce:* | Ing. Marián Lamr, Ph.D.<br>Ústav mechatroniky a technické informatiky |
| *Konzultant práce:* | Ing. Pavel Tyl<br>Ústav mechatroniky a technické informatiky |
| *Datum zadání práce:* | 12. října 2022 |
| *Předpokládaný termín odevzdání:* | 15. května 2023 |

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Černohorský, Ph.D.
vedoucí ústavu

V Liberci dne 12. října 2022

# Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

22. května 2023 Filip Prádler

# Aplikace pro hledání skrytých závislostí v datech o kriminalitě

## Abstrakt

Tato bakalářská práce se zabývá průzkumem využití data miningových nástrojů pro hledání skrytých závislostí v datech o kriminalitě, a následným vývojem aplikace, která tyto nástroje používá.

Teoretická část obsahuje přehled o využití kriminálních dat v zahraničí, kde jsou popsány principy prediktivních systémů.

V praktické části byla na základě tohoto přehledu naprogramována desktopová aplikace, ve které jsou implementovány nejpoužívanější postupy a algoritmy pro hledání skrytých závislostí v datech o kriminalitě.

Součástí práce je popsán i obecný pohled na vývoj desktopových aplikací v programovacím jazyce Python.

**Klíčová slova:** kriminalita, data mining, PyQt, Python, folium

# Application for finding hidden dependencies in criminal data

## Abstract

This bachelor thesis explores the use of data mining tools to find hidden dependencies in crime data, and the subsequent development of an application that uses these tools.

The theoretical part contains an overview of the use of crime data abroad, where the principles of predictive systems are described.

In the practical part, based on this review, a desktop application was programmed that implements the most commonly used procedures and algorithms for finding hidden dependencies in crime data.

The thesis also describes a general view of desktop application development in the Python programming language.

**Key words:** criminality, data mining, PyQt, Python, folium

# Seznam obrázků

# Seznam tabulek

# Seznam zdrojových kódů

# Obsah

# Seznam zkratek

**API** Sbírka procedur, funkcí, tříd a protokolů knihovny

**DBSCAN** Shlukování metodou hustoty bodů

**GUI** Grafické uživatelské rozhraní

**IDE** Vývojové prostředí

**KMEANS** Shlukování metodou nejbližších středů

**UI** User Interface

**UIC** User Interface Compiler

**QML** Programovací jazyk Qt Modeling Language

# 1 Úvod

V současné době dochází k rychlému rozvoji a vzniku nových technologií, což můžeme pozorovat postupným zaváděním výpočetní techniky do prakticky všech aspektů našeho života. Implementace výpočetní techniky je ve většině odvětvích vítána, protože s sebou přináší nové možnosti a urychluje proces řešení již existujících problémů. Díky neustálému kontaktu široké veřejnosti s těmito technologiemi dochází k hromadění velkého množství dat (big data), které obsahují velmi cenné skryté závislosti, ze kterých je možné vyvozovat netriviální závěry o celém datasetu. Obor zabývající se analýzou big data pomocí specializovaných algoritmů, se nazývá data-mining.

Jednou z velmi atraktivních aplikací data-miningu je analýza dat o kriminalitě, jejíž studium a rozvoj může přinášet dalekosáhlé dopady (jak pozitivní, tak negativní) na společnost jako takovou. Proto je tato problematika zaměřením mnoha studií a soukromých firem po celém světe. Přestože díky prudkému rozvoji data-miningu a příbuzných oborů, např. umělá inteligence, vznikají snahy celý proces analýzy kriminálních dat automatizovat, stále je potřeba znalého odborníka, který dokáže upravená data konzistentně interpretovat a vyvodit správný závěr.

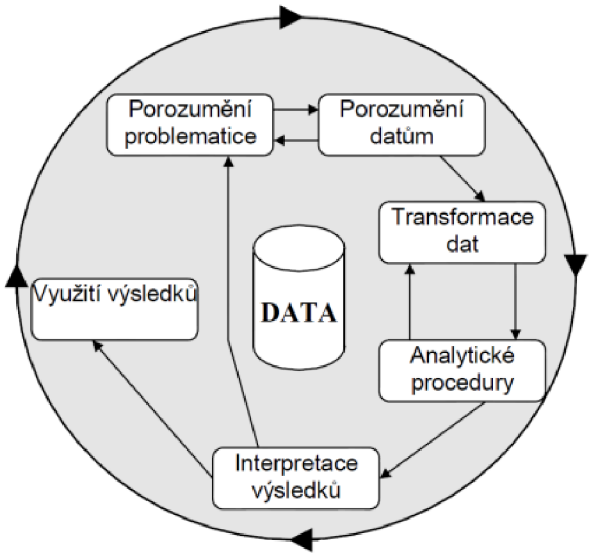
Cílem této práce je získat ucelený přehled o nejpoužívanějších nástrojích data-miningu pro hledání skrytých závislostí v datech o kriminalitě v celosvětovém měřítku a následně naprogramovat aplikaci, která tyto nejvyužívanější nástroje dokáže využívat. Již existující aplikace se vyznačují přílišnou obecností (nebyly vytvořené přímo pro hledání skrytých závislostí v datech o kriminalitě), tudíž při jejich užívání vyžadují pokročilé technické znalosti. Výsledkem programování by tak měla být v ideálním případě aplikace, která bude sofistikovanější a uživatelsky přívětivější pro uživatele bez hlubšího porozumění algoritmů, které jsou k analýze využívány.

# 2 Hledání skrytých závislostí v datech o kriminalitě

Hledání skrytých závislostí (data-mining) je analytická činnost, při které se zpracovávají data. Účelem této činnosti je vyhledávání skrytých závislostí, vzorů a trendů za pomoci matematických a statistických algoritmů, či algoritmů strojového učení. Data mining pracuje s velkým množstvím dat různého formátu.

## 2.1 Metodika CRISP-DM

Data mining je komplexní záležitostí, kde se postup analýzy liší případ od případu. Existuje však metodika, která tento proces obecně popisuje [13].




Obrázek 2.1: Proces data-miningu dle CRISP-DM [14]

### 2.1.1 Porozumění a transformace dat

Nejčastější formou dat, se kterou se můžeme u data miningu setkat, jsou databáze, datové sklady nebo transakční data [3]. Všechna tato data se nazývají dataset, nebo datová matice. Entitou databáze je datový objekt. Aby byl dataset validní pro

analýzu, je nutné, aby datový objekt vlastnil alespoň jeden atribut, který uchovává nenulovou hodnotu. Informace vyplývající z hodnoty atributu závisí nejen na datovém typu, ale i na základě diskrétnosti hodnot. Existují čtyři typy atributů, kterými jsou: nominální, binární, ordinální a numerické. Výběr správného datového typu je důležitým krokem, který je nutné provést ještě před analýzou.

Stejně jako u statistické analýzy, tak i u data miningu je důležitý výběr dat. Počet dat by neměl být nízký a obecně data musí mít určitou kvalitu, aby byl výsledek analýzy co nejvíce směrodatný. Kvalita dat závisí na vhodném filtrování a počtu atributů.

V data miningu se na začátku často pracuje s obrovským množstvím dat. Tato data nazýváme big data. Kromě velké velikosti bývají data také nestrukturované nebo nekompletní. Z tohoto důvodu je nutné tato data před analýzou nejprve transformovat.

Součástí transformování dat je odstranění záznamů s chybějícími hodnotami v atributech nebo naopak doplnění těchto chybějících hodnot. Dále dochází k odstranění redundantních záznamů nejen z důvodu optimalizace, ale i ke zlepšení výsledků. Data lze rozšířit o další atributy například pomocí API, nebo extrakcí dat.

### 2.1.2 Analytické procedury

Součástí analytické procedury je tvorba modelu. Při tvorbě se vybírá modelovací technika, dále pak parametry pro tuto techniku.

Existují dva typy modelu. Nazývají se prediktivní a deskriptivní. Výběr typu modelu vychází z představy analytika o výsledku analýzy, respektive z celkového porozumění problematiky.

Prediktivní model pracuje s historickými daty a hledá v těchto datech vzory. Tyto vzory se později aplikují na aktuální data. Dále tento model pracuje například s klasifikací, regresí a časovými řadami.

Deskriptivní model je spíše statistickou záležitostí. Vytvoří závěr z dat, který je pak pro člověka lehce čitelný. Tento model pracuje například se shlukováním nebo s asociační analýzou.

Výběr vhodného algoritmu vychází z typu modelu. Tyto algoritmy se dále dělí na klasifikační, asociační, regresivní a shlukovací. Dohromady jsou tyto algoritmy nástroji strojového učení.

### 2.1.3 Strojové učení

Strojové učení je odvětvím umělé inteligence, které se mimo jiné využívá i při data-miningu. Vyznačuje se efektivitou ve zpracování velkého množství dat. Funguje na principu učení, kdy se počítač nejprve učí z dat, a následně pak naučené závislosti aplikuje v nadcházejících datech. Strojové učení se dělí na: učení se s učitelem, nebo učení se bez učitele [18].

V kontextu strojového učení se také mluví o hlubokém čtení (deep learning), kde pomocí neuronových sítí se počítač snaží napodobit lidský mozek.

## 2.2 Asociační analýza

Asociační analýza se zabývá hledáním asociačních pravidel, nebo-li vzorců a trendy v datech.

Asociační pravidla se nejčastěji vyskytují v oblastech marketingu, nicméně i při hledání skrytých závislostí v datech o kriminalitě. Data jsou zpracována pomocí asociačních algoritmů. Mezi algoritmy asociační analýzy se řadí algoritmus Apriori.

### 2.2.1 Apriori algoritmus

Algoritmus Apriori funguje na principu hledání nejčastějších sad prvků, které se vyskytují v celé datové matici [12].

Princip algoritmu je popsán v těchto krocích:

1. Definujeme minimální podporu a minimální spolehlivost. Minimální podpora je minimální procento transakcí v datové matici, které obsahují určitou sadu prvků, aby sada byla považována za častou sadu prvků. Minimální spolehlivost je minimální procento transakcí v datové matici, které obsahují určitou sadu prvků, aby byla považována za pravidlo.

2. Algoritmus začíná vytvořením prvního seznamu jednoprvkových sad. Poté vypočítává podporu pro každou sadu prvků v seznamu. Pokud je podpora větší nebo rovna minimální podpoře, pak je sada prvků považována za častou sadu.

3. Dalším krokem je kombinace častých sad prvků pro vytvoření nové sady prvků obsahujících více než jeden prvek. Opět se vypočítává podpora pro každou novou sadu prvků.

4. Postupně se takto vytvářejí stále větší a větší sady prvků, dokud již nelze dále vytvářet nové sady prvků. Tyto sady prvků jsou poté použity v následujícím kroku pro tvoření pravidel.

5. Pravidla jsou vytvořena z kombinace prvků v častých sadách prvků. Podporu a spolehlivost každého pravidla je opět vypočítána. Pokud je spolehlivost pravidla větší nebo rovna minimální spolehlivosti, pravidlo je považováno za platné.

Algoritmus je ukončen, když už nelze vytvořit novou sadu prvků nebo nové pravidlo.

## 2.3 Mapová analýza

Při mapové analýze se zpracovávají data, která obsahují prostorovou informaci, přičemž typicky se jedná o souřadnice v podobě zeměpisné šířky a délky. K těmto souřadnicím jsou mimo jiné přiřazeny další atributy nesoucí informace vztahující se právě k tomuto místu. Výsledkem analýzy vzniká podmnožina dat, ze které jsou

pak jednotlivé položky graficky znázorněny, například na rastrové, nebo vektorové mapě.

Ze studie Mapy budoucnosti [16] vyplývá, že existuje šest metod a postupů mapové analýzy pro predikci kriminality. Všechny metody a postupy jsou popsány v tomto seznamu:

- Hot-spot analýza,
- teplotní mapa,
- repeat victimisation,
- risk terrain modeling,
- geografické profilování.

### 2.3.1 Hot-spot analýza

Metodou hot-spot analýzy vznikají oblasti, které nazýváme shluky. Na mapě jsou graficky znázorněny jako místa, ve kterých je koncentrace páchání zločinů nejčastější, nebo kde je riziko spáchání zločinu největší.

Prvním krokem je použití algoritmu provádějící shlukovou analýzu, která z dat vytvoří shluky. Poté se pomocí metody identifikace hot-spotů určí hranice každého vytvořeného shluku. Nakonec se podle účelu hot-spot analýzy určí relativnost každého hot-spotu a dle potřeby se rozliší na mapě.

Hot-spot analýzou je možné získat dva různé výstupy, které se liší svojí vypovídající hodnotou [17]. Jedná se o „repeat places hot spots“ a „repeat victimization hot spots“.

#### Repeat places hot spots

Je to nejčastější forma hot-spotů. Každý hot-spot v tomto případě reprezentuje jednu oblast na mapě, ve které dochází k častějšímu výskytu trestní činnosti, avšak typ této kriminální činnosti je irelevantní. Z této analýzy vyplývá, že konání trestné činnosti v oblasti nemá vliv na charakter oběti [17].

#### Repeat victimization hot spots

Druhou formu nazýváme mapování opakované viktimizace, při které se hledá vztah mezi druhem zločinu a o jakou oběť se jedná. Hot-spot je vytvořen v místech, kde je vysoká koncentrace těchto potenciálních obětí.

Například když bude druh zločinu kapsářství a obětí důchodce, tak se předpokládá, že pachatelem bude znovu vybrán senior. Nepředpokládá se tedy, že by se tyto scénáře odehrávaly častěji na nějakém konkretním místě. Z toho důvodu hot-spoty vznikají tam, kde se tyto oběti (senioři) nejčastěji nacházejí [17].

#### Tvorba shluků

Shluky jsou výstupem shlukovacích algoritmů. Jsou to množiny, do kterých jsou zločiny vkládány na základě předem určeného pravidla. Mezi pravidly je například přiřazení ke shluku podle eukleidovské vzdálenosti. Ke shluku se přiřadí zločin, který je zrovna nejblíže, nebo naopak ten nejdále.

Shlukovacích algoritmů je mnoho, liší se vstupními parametry. Většina z nich vyžadují jako parametr vzdálenost.

### 2.3.2 KMEANS algoritmus

KMEANS je nehierarchický shlukovací algoritmus. Někdy tento algoritmus nazýváme jako metoda shlukování nejbližších středů. Od tohoto názvu je zřejmé, že se jednotlivé body přiřazují ke nejbližšímu středu shluku. Středy shluků také nazýváme centroidy. Přiřazením bodu mění shluk svůj tvar tvořený z bodů, a tudíž i střed. Prvotní souřadnice středu shluků jsou zvoleny náhodně, a proto více iterací tohoto algoritmu může mít rozdílný výsledek.

KMEANS vyžaduje jeden parametr, a to počet shluků.


Obrázek 2.2: KMEANS algoritmus se 3 shluky [20]

### 2.3.3 DBSCAN algoritmus

DBSCAN je zkratka pro algoritmus zvaný „Density-based spatial clustering of applications with noise“. Jedná se o shlukovací algoritmus, který stejně jako KMEANS shlukuje data do shluků. Shluky nemění tvar a tudíž vznikají i body, které se

nepřiřadí do žádného shluku. Tyto body nazýváme jako šum [11] a pro analýzu jsou nerelevantní. Šum (noise) je často klasifikován jako shluk s číslem -1 a je značen černou barvou.

DBSCAN vyžaduje dva paremetry:

- minimální počet bodů – tento parametr udává minimální počet bodů, aby vznikl shluk.
- $\epsilon$ – parametr udává maximální hodnotu vzdálenosti, do které jsou body považovány jako sousedící body, z kterých je vytvořen shluk.

DBSCAN narozdíl od KMEANS algoritmu nevyžaduje počet shluků jako parametr. Z tohoto důvodu je počet shluků součástí výsledku analýzy, což může vést ke správné interpretaci dat. Skutečnost, že DBSCAN dokáže určit hraniční body, které nezařazuje do žádného shluku, snižuje požadavky na kvalitu vstupních dat. Protože data, jejichž přítomnost je irelevantní mají menší šanci ovlivnit výsledek analýzy.

### Porovnání shlukovacích algoritmů

V nigérijské univerzitě proběhla studie, ve které autoři použili kriminální data od policie nacházející se v hlavním městě Nigérie Abuja. Výsledkem bylo porovnání tří shlukovacích algoritmů [15]. Mezi těmito algoritmy patří KMEANS, hierarchické shlukování a EM algoritmus.

Všechny analýzy proběhly v analyzačním programu WEKA, a cílem bylo porovnat přesnost a časovou náročnost těchto zmíněných algoritmů. Ze studie vyšlo najevo, že nejpřesnějším algoritmem je EM algoritmus, avšak za cenu časové náročnosti. Naopak dva zbylé algoritmy provedly analýzu mnohem rychleji, nicméně s mnohem menší přesností.

Tabulka 2.1: Porovnání časové náročnosti a přesnosti použitých algoritmů

| **Použitý algoritmus** | **Přesnost** | **Časová náročnost** |
|---|---|---|
| EM algoritmus | 90,5% | 8,5s |
| K-Means | 62,6% | 0,09s |
| Hierarchické shlukování | 51,9% | 0,11s |

### Tvorba hot-spotů

Hot-spoty je možné vytvořit vícero způsoby. Liší se ve tvaru, který může být jak pravidelný, tak i naopak nepravidelný [16].

Mezi tyto způsoby patří:

- Kernel Density Estimation.
- elipsy směrodatné odchylky,
- kvadrantová metoda.

### 2.3.4 Teplotní mapa

Teplotní mapy jsou grafická znázornění dat, která pomocí barev znázorňují hustotu datových bodů v určité oblasti. Tepelné mapy lze použít k vizualizaci ohnisek kriminality a identifikaci vzorů trestné činnosti.

Teplotní mapy zobrazují koncentraci trestné činnosti na konkrétních místech, ve kterých pak policie zvýší hlídkovou činnost.

Pomocí teplotní mapy a časové osy lze identifikovat změny v kriminálním chování a podle toho upravit strategii policejních složek.

# 3 Využití kriminálních dat v zahraničí

## 3.1 Systémy predictive policing

K analýze kriminálních dat mohou orgány činné v trestním řízení a další interní bezpečnostní složky využívat systém predictive policing, a to k předvídání trestné činnosti. Jejím cílem je buď dopadnout podezřelého, nebo zabránit spáchání trestního činu. Dle účelu analýzy toho se systém dělí do tří kategorií [1]:

- Predikce trestného činu,
- predikce podezřelého,
- predikce oběti.

Zvolená kategorie závisí na vstupních datech. Dle toho lze vybrat nejvhodnější systém, který použije jednu z analytických metod a snaží se co nejpřesněji vytvořit hot-spoty, nebo nalézt spojitosti (pattern) mezi již spáchanými trestními činy [2].

### 3.1.1 PredPol

PredPol je nejpoužívanější predictive policing systém, který na základě historických dat o kriminalitě přiřazuje oblastem 150 m x 150 m pravděpodobnost, že se zde v následujících několika hodinách odehraje konkrétní kriminální aktivita. Oblasti s největší přidělenou pravděpodobností jsou před začátkem každé hlídky vizualizovány na mapě. Poté jsou přiděleny policistům, kteří jsou v těchto oblastech pobídnuti trávit více času a dbát zvýšené pozornosti. Systém se snaží předcházet zločinům v těchto oblastech tím, že zvýší přítomnost policistů, čímž dochází k odrazu potenciálního pachatele [5].

PredPol aplikuje strojové učení na historickou databázi a vytváří prognózu z těchto čtyř faktorů:

1. Unikátní ID zločinu,
2. typ zločinu,
3. souřadnice,
4. časový úsek, ve kterém se zločin odehrál.

Model, pomocí kterého strojové učení vytvoří předpověď, je založen na třech předpokladech o chování pachatele:

1. Repeat victimization – v oblastech, kde již byl trestný čin spáchán, je větší pravděpodobnost, že bude spáchán v blízké budoucnosti znovu.

2. Near-repeat victimization – pokud se v oblasti odehrál určitý zločin, zvyšuje se pravděpodobnost odehrání tohoto zločinu i v sousedních oblastech.

3. Local search – předpokládá se, že se pachatel pohybuje v blízkosti místa svého zaměstnání nebo bydliště. V důsledku toho se stejné trestné činy obvykle vyskytují v těsné blízkosti.

V [4] je tento model přirovnáván k seismologii. Konkrétní zločin vysílá do svého okolí vlny, které při průchodu dalšími oblastmi zvyšují riziko opakování tohoto zločinu. Vlny mají dočasný efekt a v průběhu časem slábnou. Tím se pravděpodobnost zločinu v oblastech vrací na normální (předešlou) hodnotu.

### 3.1.2 HunchLab

HunchLab od společnosti Azavea patří mezi predictive policing systémy, který stejně jako systém PredPol využívá strojového učení za účelem vytvoření hot-spotů. Systém je navržen tak, aby přijímal nespočet vstupních parametrů. Ty sice zvyšují přesnost predikce, ale zároveň znatelně zhoršují standardizaci vstupu. Parametry jsou rozděleny na povinné/nepovinné a jsou klasifikovány do několika kategorií [7].

Mezi povinné parametry patří:

- Označení jurisdikce/okrsku,
- výsledná data o trestné činnosti (výstup této predikce),
- unikátní identifikační číslo zločinu, typ zločinu, souřadnice, adresa, ... .

Mezi nepovinné parametry patří:

- Souřadnice zajímavých míst (nemocnice, restaurace, autobus. zastávky),
- společensko-ekonomické indikátory (chudoba, hustota bezdomovců),
- čas (dny v týdnu, roční období),
- speciální událost (státní svátky, sportovní zápasy),
- počasí.

Součástí postupu je rozdělení oblasti do 150 m x 150 m kvadrantů [16], kde je pro každý kvadrant vytvořen samostatný model. Model je vytvořen pomocí „gradient boosting“ regrese za pomoci optimalizační funkce AdaBoost. Regrese vytvoří rozhodovací strom (decision tree) na základě toho, jestli k trestné činnosti došlo či nikoliv. Pokud ano, rozhodovací strom se začne rozšiřovat aby zjistil příčinu vzniku činu. Tento celý proces je opakován hned několikrát, aby došlo k co nejpřesnějšímu výsledku. Postup lze shrnout do těchto kroků [7]:

1. Selekce trénovacích dat,
2. obohacení variabilními vstupy,
3. vytvoření modelu,
4. hodnocení výsledku,
5. výběr nejlepšího vzoru.

Po výběru nejlepšího vzoru vznikne model, který obsahuje informace o tom, zda za určitých podmínek dojde, nebo nedojde k trestné činnosti. Na tento model je následně použito Poissonovo rozdělení. Výsledkem je míra pravděpodobnosti každého typu zločinu pro každý kvadrant.

### 3.1.3 RAIDS

RAIDS je systém od společnosti BAIR Analytics. Pomocí strojového učení a použití historických dat systém definuje hot-spoty s vysokým výskytem kriminální činnosti. Systém také předpovídá typ spáchaných zločinů na následujících 30 dní. Policie na tyto oblasti reaguje zvýšenou hlídkovou činností [19].

RAIDS využívá tato data:

- Lokace spáchané činnosti,
- informace o podezřelém (rasa, použitá zbraň, pohlaví),
- data o počasí,
- demografická data,
- socioekonomická data.

## 3.2 Využití v jednotlivých zemí

### 3.2.1 Spojené státy americké

Spojené státy americké (dále USA) patří mezi nejvyspělejší státy světa. Není tedy divu, že první implementace modelů předpovídajících kriminalitu se odehrály právě

v tomto státě. Díky tomuto prvenství nyní většina největších společností, které se zabývají vývojem a prodejem prediktivních modelů, sídlí právě v USA. Každé policejní oddělení trápí specifické problémy spjaté s jurisdikcí města jejich působení, proto je využití prediktivních algoritmů velmi různorodé. V této sekci se zaměříme na tři největší poskytovatele predictive policing technologií, a to PredPol, RTM a HunchLab (systémy, které tyto společnosti nabízejí, mají často stejná jména jako společnosti samotné) [4].

V roce 2015 byla ve městě Los Angeles provedena studie trvající 117 dní. Po tuto dobu měl tým kriminálních analytiků policejního oddělení Los Angeles za úkol předpovědět místo a čas co nejvíce zločinů, za pomoci dosavadních klasických metod. Na konci experimentu byly výsledky analytiků porovnány předpovědí PredPolu. Ukázalo se, že analytici předpověděli úspěšně 2,1 % zločinů, zatímco PredPol jich předpověděl 4,7 % [4]. Objektivnost této studie je ovšem rozporuplná, jelikož se na ní podíleli členové vedení společnosti PredPol [6].

### 3.2.2 Německo

Spolková republika Německo se začala zajímat o systémy predikující kriminalitu již v roce 2014. V roce 2018 systém pak v Německu fungoval až v šesti spolkových zemích. Kvůli rozdílné jurisdikci mezi spolkovými zeměmi je v každém spolku použit jiný systém [8].

#### Bavorsko a Bádensko-Württembersko

Tyto spolkové země začali již od roku 2015 používat systém Precobs, který se zaměřuje na predikci vloupání. Systém potřebuje pouze policejní databázi se záznamy o předchozích vloupáních. Celý systém poté funguje na dvou pilířích. Prvním pilířem je definice předmětu. Předmět může být buď místo, nebo informace o jakou odcizenou věc se jedná. Druhým pilířem je filtrování oblastí, ve kterých se stal podobný čin. Tyto oblasti jsou vybírány na základě tzv. „near repeat data“, které se následně včetně tohoto záznamu znovu použijí. Výsledkem předpovědi je pro každou oblast o velikosti 250 m x 250 m znázorněno riziko podle vypočítané pravděpodobnosti. Tento proces je však nepoužitelný pro méně zalidněná města a vesnice. Pro tyto oblasti je použit tzv. „far repeat approach“, který místo predikce vloupání monitoruje možný pohyb pachatele mezi regiony [8].

Švýcarský deník Aargauer Zeitung v roce 2015 uvedl, že díky systému Precobs došlo v těchto spolkových zemích až o 30 % méně vloupání oproti minulému roku 2014 [9].

#### Berlín

Státní úřad pro vyšetřování trestních činů za podpory Microsoftu začal v roce 2016 vyvíjet prediktivní systém KrimPro. Narozdíl od bavorského Precobs se také zaměřuje na krádeže. KrimPro oproti Precobsu bere v potaz také společensko-ekonomická data a úroveň infrastruktury v dané oblasti. Systém je schopný předpovídat krádeže a vloupání pro oblasti o velikosti až 400 m x 400 m na tři dny dopředu [8].

#### Hesensko

V Hesensku je od roku 2017 používán systém KLB-operativ. System se opět snaží předpovídat co nejvíce loupeží. Pro predikci je použita kriminální a společensko-ekonomická databáze. Systém funguje na principu „near repeat approach“, kde na začátku dne policista obdrží mapu s hot-spoty (oblasti) možného výskytu nedovolené činnosti. Celý systém je v podobě mobilní aplikace, která mimo hot-spotů umí i filtrovat a zobrazovat podezřelé činnosti na interaktivní mapě [8].

#### Dolní Sasko

V roce 2016 místní policie spustila aplikaci PreMAP (Predictive Mobile Analytics for Police). PreMap za pomocí „near repeat data“ předává policejním hlídkám informace v podobě heat mapy, na které jsou znázorněny oblasti možných krádeží. Jaká data policie pro tento systém používá, autor v tomto článku neuvádí [8].

#### Severní Porýní-Vestfálsko

STCPONW (The State Office of Criminal Investigations in North Rhine Westphalia) spustilo v roce 2015 systém SKALA. Tento systém podobně jako KLB-operativ využívá kriminální a společensko-ekonomická data. Dále také dodatečné informace o lokaci, jako například úroveň infrastruktury. Cílem je opět co nejlépe předpovídat vloupání, která byla blíže specifikována jako domácí vloupání. Analýza je provedena v data miningovém programu SPSS Modeler. Výsledek analýzy je vizualizován na mapě společně s heat mapou, která znázorňuje oblasti s největší pravděpodobností výskytu domácího vloupání [8].

### 3.2.3 Francie

Ke konci roku 2017 obdrželo jedenáct oddělení francouzské vojenské policie (Gendarmerie) predictive policing systém PAVED [10]. Jako většina predictive policing systémů, je PAVED pouze dalším nástrojem, který má zefektivnit a usnadnit práci policejním oddělením. Jeho použití proto není povinné.

PAVED byl vyvinut odnoží francouzského ministerstva vnitra. Systém je založen na „near repeat data“ teorii a jako vstup vyžaduje dva typy dat:

1. Historická data o zločinech z databáze francouzského ministerstva vnitra (veřejně nepřístupná) – přesnou podobu dat z této databáze [10].

2. Společensko-ekonomická data, která dodává francouzský národní institut statistiky a ekonomických studií – například množství domů v určité čtvrti, nebo počet automobilů spadajících pod danou domácnost [10].

Hlavním zaměřením systému PAVED je predikce vloupání a krádeží aut. Pomocí strojového učení je na základě výše uvedených dat vytvořena hotspot mapa, která znázorňuje oblasti s vysokým rizikem odehrání určitého zločinu. Tato mapa je pro uživatele přístupná pouze z jejich pracovních počítačů, čímž se liší například od

systému PredPol nebo Hunchlab, který umožňuje policejním hlídkám přístup k výsledkům analýzy přímo z jejich vozidla.

# 4 Vývoj desktopové aplikace v Pythonu

Nejvhodnější verzí programovacího jazyka Python pro tvorbu desktopové aplikace je verze 3.8. Podporuje nejen nejznámější, v budoucnu zmíněné, knihovny grafického uživatelského rozhraní, dále ale také všechny populární knihovny pro manipulaci a analýzu dat jako je pandas, folium nebo maptlotlib.

## 4.1 Grafické uživatelské rozhraní

Součástí každé moderní aplikace je grafické uživatelské rozhraní (zkráceně GUI), pomocí něhož je umožněno uživateli aplikaci mnohem lépe a komfortněji ovládat.

Mezi nejpopulárnějšími GUI knihovnami pro Python se řadí PyQt, PySide, Kivy nebo PySimpleGUI.

Při výběru vhodné knihovny bylo bráno v potaz několik faktorů. Typ licence, rozmanitost a srozumitelnost referencí a míra upravitelnosti všech objektů, ze kterých je GUI tvořen. Cílem bylo najít kompromis mezi všemi dostupnými GUI knihovnami, aby bylo možné vytvořit dobře vypadající moderní aplikaci.

### Tkinter a PySimpleGUI

Principem těchto knihoven je pokládání widgetů. Tyto knihovny obsahují však jen některé základní widgety, jelikož se jedná o knihovny pro tvorbu spíše odlehčeného (light-weight) grafického rozhraní. Jsou open-source. Výhodou těchto knihoven je dostupnost mezi všemi operačními systémy a hardwarovými platformami podporující jazyk Python. Hlavní nevýhodou těchto knihoven je absence moderního vývojového prostředí obsahující designer.

### Kivy

Multi-platformní GUI knihovna. Opět je hlavním elementem widget. Součástí je „Kv language", díky kterému je možné tyto widgety do detailu rozložit a upravit. Nevýhodou je nižší popularita, a tudíž chudá dokumentace a nedostatek příkladů.

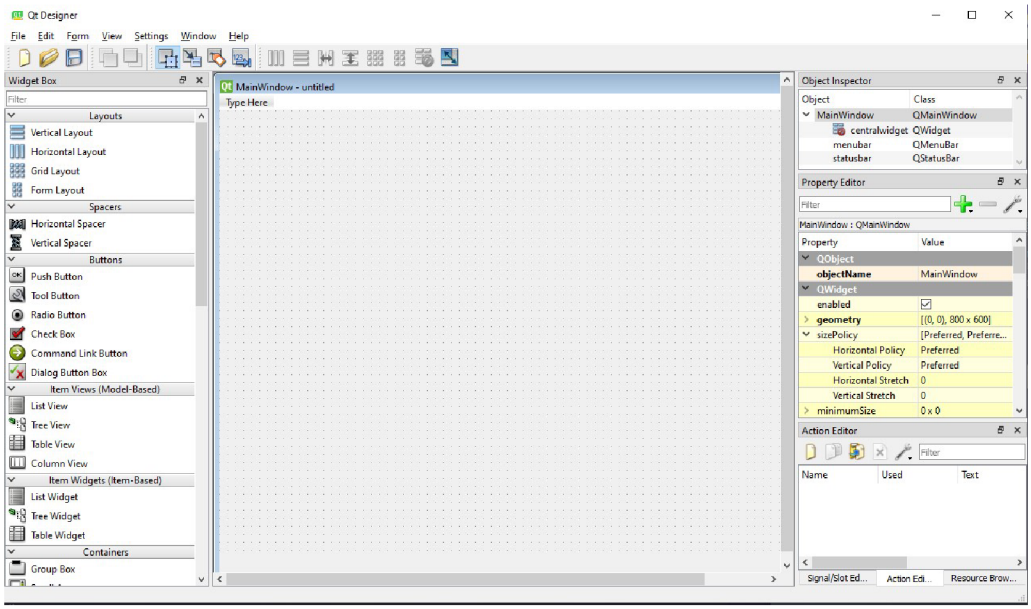
### PyQt a PySide

Knihovna PyQt je součástí volně dostupného frameworku Qt, který umožňuje použití tohoto frameworku v kombinaci s Pythonem. Největší výhodou této knihovny je kompatibilita napříč všemi systémy, bohatá dokumentace a dostupnost rozšiřujících

modulů. Dále také existuje pro tuto knihovnu více designerů, které rapidně zjednoduší tvorbu front-endové části aplikace. Existuje také jiná varianta této knihovny s názvem PySide, jež se liší v použití jiné licence, umožňující komerční použití aplikace bez poplatků.

### 4.1.1 Qt Designer

Qt Designer je jeden z vývojových prostředí pro Qt. Tento designový nástroj umožňuje tvořit okna a do nich vkládat widgety pomocí techniky táhni a pusť (drag and drop). Tímto způsobem lze bez ručního kódování vytvořit plně funkční grafické uživatelské rozhraní. Výstupem je soubor ve formátu ui, který je tímto nástrojem vygenerován v XML jazyce.




Obrázek 4.1: Ukázka Qt Designeru

#### Rozložení aplikace Qt Designer

Editor se skládá z několika částí.

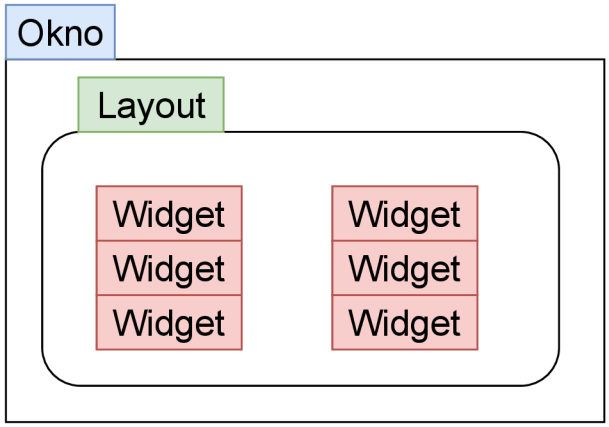
V levé části se nachází „Widget Box“, ve kterém je seznam všech widgetů. Widget, který bude použit, se přetáhne do požadovaného okna, které se nachází uprostřed editoru. Zobrazená okna se dynamicky mění na základě úprav, a po implementaci představují totožnou podobu, která je shodná se samotnou aplikací. V tomto editoru lze zobrazit a upravovat více oken zároveň.

Seznam všech oken a již položených widgetů se nachází vpravo nahoře v „Object Inspector“. Je zde také znázorněno jaký objekt dědí od kterého.

Každý widget má své vlastní atributy, které jsou zobrazeny níže v „Property Editor“. Druh a četnost atributů závisí na tom, o který widget se jedná.

### Obecné schéma aplikace

Každá vytvořená aplikace v Qt Designeru obsahuje několik vrstev.




Obrázek 4.2: Schéma aplikace s mřížkovým rozložením

### Okno

První vrstvou je okno, na které se pokládají další vrstvy widgetů, nebo dalších oken. Existují dva druhy oken. Základním oknem je „MainWindow“, které slouží jako úvodní okno. Obsahuje totiž prostředky pro spuštění aplikace a připravuje prostor pro navigační menu, panel nástrojů a další panely.

Druhým oknem je dialogové okno. Toto okno slouží jako prázdný vzor a neobsahuje žádné, již zmíněné, prostředky nacházející se v „MainWindow“. Typicky toto okno slouží pro zadávání vstupů, informování nebo upozornění uživatele.

### Rozložení widgetů

Další vrstvou je rozložení. Slouží k rozvržení plochy pro pozice, do kterých se budou vkládat widgety. Zajišťují rovnoměrnou vzdálenost mezi widgety nebo okrajem okna. Další využití je škálovatelnost celého okna v závislosti na rozlišení. Widgety jsou poté responsivní a celá aplikace se stane adaptivní aplikací. Rozložení se dělí na tyto tři druhy:

- Vertikální rozložení,
- horizontální rozložení,
- mřížkové rozložení.

### Widget

Poslední vrstvou jsou samotné widgety. Jednotlivé widgety mohou být navzájem odděleny pomocí rozpěrek. Rozpěrky se dále dělí na vertikální a horizontální. Widgetem je každý další objekt v okně. Nejčastěji se jedná o textové pole, různá tlačítka nebo kontejnery pro další widgety.

### 4.1.2 Implementace grafického uživatelského rozhraní

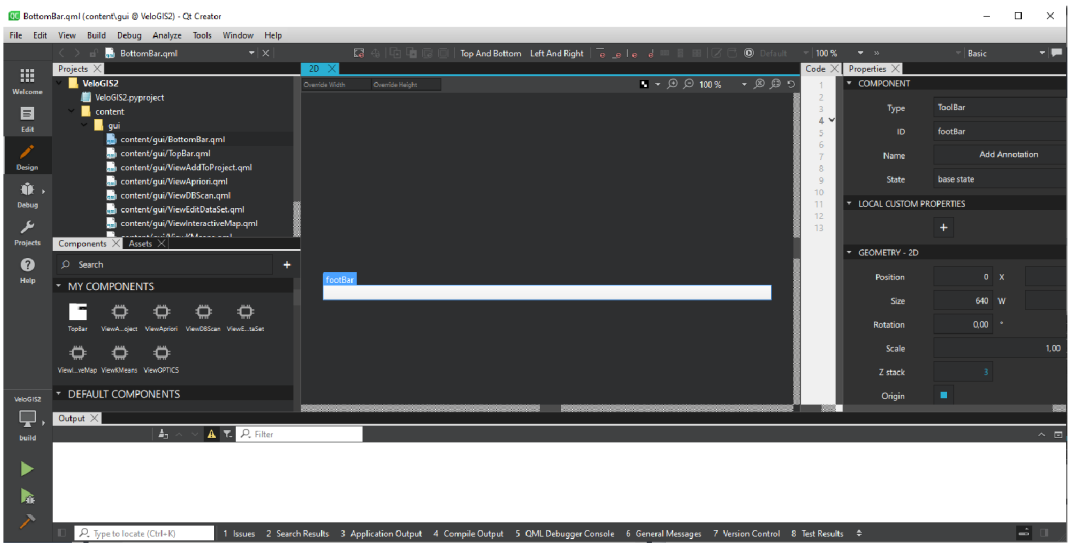
Qt Designer exportuje vytvořené GUI do souboru ve formátu ui. Po exportu následuje implementace do back-endu. Existuje více způsobů implementace.

Nejjednodušším způsobem je všechny soubory formátu ui načíst pomocí modulu UIC (User Interface Compiler). Jedná se o překladač umožňující překlad z XML jazyka do C++ nebo Pythonu. Nevýhodou tohoto způsobu je nefunkčnost napovídače v používaném vývojovém prostředí. Soubory jsou přečteny až při spuštění aplikace, a tudíž vývojové prostředí nemůže vědět, jaké widgety se v souborech nacházejí. Z tohoto důvodu je tento způsob doporučený pro znalejší vývojáře, které již mají s Qt frameworkem znalosti.

Dalším způsobem je použití sofistikovaného programu nebo skriptu pyuic5. Stejně jako uic se jedná o kompilátor s rozdílem toho, že k překladu dochází ještě před spuštěním celého programu. Vývojář manuálně pomocí tohoto skriptu přeloží XML soubor do Python jazyka. Výsledkem je tedy py soubor, který stačí do aplikace naimportovat. Implementace proběhne vytvořením instance této třídy, se kterou se pak dále pracuje. Díky tomuto je pak napovídač plně funkční a výsledek je téměř identický jako kdyby bylo grafické rozhraní napsáno ručně v jazyce Python, a tudíž je napovídač plně funkční.

## 4.2 Qt Creator

Qt Creator obsahuje vývojové prostředí nejen pro Python, ale také i pro C++. Součástí je designer, který je však v kombinaci s jazykem Python nestabilní, pomalý a často nefunkční. Qt Creator umí stejně jako Qt Designer nejen vytvářet UI soubory, ale také i modernější QML soubory.




Obrázek 4.3: Ukázka Qt Creatoru

#### QML

QML je deklarativní jazyk, díky kterému lze za pomocí značek vytvářet komplexní uživatelská rozhraní. QML nabízí jednoduchou syntaxi, je moderní, rychlý a reaktivní. Další výhodou je multiplatformnost mezi zařízeními.

## 4.3 Qt Design Studio

Nejnovějším nástrojem pro Qt je Qt Design Studio. Plně podporuje tvorbu aplikací i na mobilním zařízení. Nevýhodou je nutná znalost QML jazyka, tedy nemožnost vygenerovat ui soubor.

## 4.4 Vývojové prostředí pro Python

Výběr vývojového prostředí (IDE) bývá spíše osobní preferencí. Nicméně cílem bylo najít takové vývojové prostředí, ve kterém by bylo možné co nejjednodušeji implementovat soubory grafického uživatelského rozhraní.

### 4.4.1 Vývojové prostředí Visual Studio

Visual Studio je vývojové prostředí od firmy Microsoft. Jedná se o populární IDE podporující mnoho pluginů a modulů, které mimo jiné také dovolují použití Visual Studia spolu s jazykem Python. Spárování s Qt Designerem je možné pomocí rozšíření „Qt for Python“ vytvořené autorem Shuang Wu. Rozšíření však nebylo odzkoušeno z důvodu technických problémů spjatých s Visual Studiem.

### 4.4.2 Vývojové prostředí PyCharm

Vývojové prostředí PyCharm od firmy JetBrains je moderní vývojové prostředí pro Python. Umožňuje spárování s Qt Designerem, avšak spárování je neintuitivní. Neexistuje totiž žádné rozšíření a je nutné všechno nastavit ručně. Nevýhodou PyCharmu je občasná potíž importovat některé knihovny. Mnoho pluginů, včetně pluginu podporující QML, bývají placené.

### 4.4.3 Editor Visual Studio Code

Univerzální editor použitelný pro spousty jazyků. Funkce tradičního IDE jsou nahrazovány v podobě pluginů. Pro Qt framework existuje nespočet rozšíření a díky jednoduchosti a nenáročnosti na hardware je Visual Studio Code dobrou volbou. Narozdíl od PyCharmu je zde lepší synchronizace editoru s Python knihovnami.

## 4.5 Kompilace zdrojového kódu

Kompilace komplexního projektu vytvořeného v jazyce Python do spustitelného exe souboru je složitou záležitostí. Přestože existuje spoustu nástrojů, tak program, který pracuje s větším počtem knihoven, způsobuje těmto nástrojům problémy, které vyúsťují v konečné nefunkčnosti a obecné nestabilitě aplikace. Program se kompiluje do exe souboru, aby bylo umožněno spuštění aplikace na více zařízeních nezávisle na tom, zda zařízení disponuje nainstalovaným Pythonem nebo přítomností příslušných knihoven.

### 4.5.1 Kompilace v PyInstaller

Jednou z variant je použití modulu s názvem PyInstaller. Jednoduchým příkazem v příkazové řádce přetvoří Python skript do exe souboru, včetně importovaných knihoven nacházející se v tomto skriptu. Předností tohoto modulu je tendence neimportovat nepotřebné knihovny, které by v překompilovaném programu pouze překážely a pouze by zbytečně zvyšovaly velikost programu na disku. Při této snaze však často dochází k nesprávné detekci potřebných knihoven. Pravděpodobně se jedná o knihovny, které jsou vyžadovanými dalšími knihovnami.

Další potíže nastávají ve chvíli, kdy se v projektu nachází více py souborů. Nehledě na to, že jsou tyto py soubory importovány, PyInstaller je ignoruje. Stejně na tom jsou i další soubory, které slouží jako assety. Mezi nimi jsou například ikony, obrázky nebo UI soubory. V důsledku toho se stává překompilovaná aplikace nestabilní, nebo dokonce i nespustitelná. Možným řešením těchto potíží je, mimo manuálního přesouvaní souborů, před kompilací vytvořit konfigurační soubor, ve kterém lze definovat které soubory zahrnout do kompilace. Při kompilaci lze také určit, zda spuštění aplikace proběhne přes příkazový řádek nebo jestli má být překompilovaný projekt uchován v co nejméně souborech.

Existuje také varianta PyInstaller s grafickým uživatelským rozhraním s názvem auto-py-to-exe, která slouží k jednoduššímu konfigurování následné kompilace.

### 4.5.2 Kompilace v cx_Freeze

Další variantou je cx_Freeze. Oproti PyInstaller je konfigurační soubor nutnou podmínkou pro kompilaci. V konfiguračním souboru je nezbytné určit systémovou platformu, pro kterou bude aplikace určena. Dalším nastavením je například definice souborů, které se při kompilaci kopírují. Konfigurační soubor obsahuje seznam knihoven, které má při kompilaci zahrnout, a které naopak vyloučit. Z tohoto důvodu nezbývá nic jiného než knihovny ručně zapsat do konfigurace. Přestože se do seznamu zahrnutých knihoven definují pouze ty skutečně potřebné, kompilátor převede i některé navíc i přesto, že je aplikace nevyžaduje. Možným řešením je definováním nechtěných knihoven do seznamu vyloučených, nebo izolovat projekt do virtuálního prostředí, a tím minimalizovat šanci pro naimportování nepotřebných knihoven. Nicméně tato chyba nemá žádný vliv na chod programu.

Pro Qt framework je dostupná upravená varianta konfiguračního souboru, ve které je přednastavený seznam knihoven nutnými pro Qt projekt.

### 4.5.3 Porovnání PyInstalleru a cx_Freeze

Při porovnávání těchto tvou kompilátorů vznikl závěr, že PyInstaller je ideálním kompilátorem k použití na projekt menších rozměrů, kde se předpokládá že neobsahuje větší počet knihoven. Tím dochází k menší pravděpodobnosti nekompletního přesunutí knihoven. Ačkoli lze určitým způsobem tento problém vyřešit, pro tento projekt je vhodnějším kompilátorem cx_Freeze.

### 4.5.4 Virtuální prostředí

Zejména při použití kompilátoru cx_Freeze je doporučeno vytvořit Python projekt ve virtuálním prostředí. Ve virtuálním prostředí se projekt izoluje od ostatních knihoven, a tím zamezí nesprávné přesunutí knihoven, které se, jak již bylo řečeno výše, týká zmiňovaného kompilátoru.

K tomu dostatečně stačí modul venv, který je již součástí každého instalačního balíčku Pythonu.

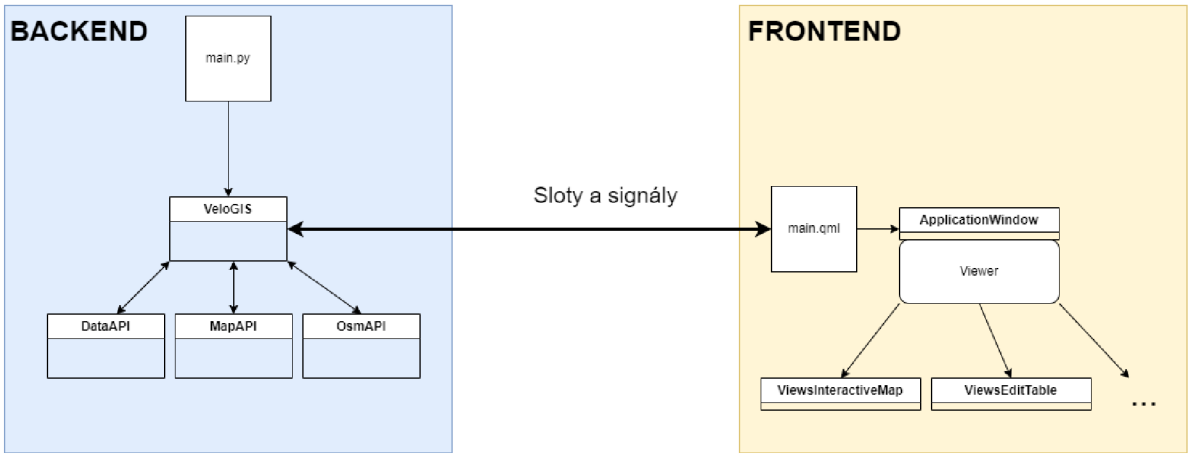
# 5 Aplikace VeloGIS

Pro moderní aplikaci s grafickým rozhraním, která manipuluje se spousty dat bylo nutné aplikaci rozdělit na front-endovou a back-endovou část. Back-endová část komunikuje s daty, provádí na těchto datech různé logické operace. Skrze front-endovou část uživatel program ovládá a zadává vstupy (viz obr. 5.1).

Back-end je napsán v jazyce Python ve verzi 3.8.10. Jako vývojové prostředí byl použit PyCharm od společnosti JetBrains. Pro grafické uživatelské rozhraní byl vybrán Qt framework.

Aplikace nepracuje s žádnou relační databází, veškerá data jsou uložena v DataFrame od pandas.

Front-end byl vytvořen v jazyce QML za pomocí designeru Qt Creator. Součástí QML jazyka je i JavaScript.




Obrázek 5.1: Obecné schéma aplikace

Hlavním elementem back-endu je třída VeloGIS. Třída je singletonem, tedy při spuštění aplikace se vytvoří pouze jedna instance této třídy. Tato třída uchovává většinu atributů, jako je například právě používaný dataset, načtená data, nebo reference na všechna dostupná API. Její hlavní funkcí je komunikace mezi front-endem, back-endem a všech implementovaných API.

## 5.1 Komunikace back-end a front-end

Komunikace mezi front-endem a back-endem probíhá skrze sloty a signály (viz kód 5.1). Slot zajišťuje komunikaci z front-endu do back-endu a signál naopak. Aby byla komunikace možná, musí třída, ve které jsou signály a sloty definovány, dědit z třídy QObject. Tato dědící třída se nastaví jako součást kontextu (ContextProperty) aplikace.

```
1 class Velogis(QObject):
2     ...
3
4 backend = VeloGIS()
5 engine = QQmlApplicationEngine()
6 engine.rootContext().setContextProperty("backend", backend)
```

Zdrojový kód 5.1: Definování kontextu aplikace

Back-end je uložen pod klíčovým názvem v podobě textového řetězce "backend".

V Pythonu se deklaruje slot skrze anotaci před příslušnou funkcí. V této anotaci je nutné definovat datové typy všech parametrů. Datový typ musí být buď klasického datového typu (str, int, list, ...), nebo datového typu definovanou knihovnou PyQt/-PySide (QAbstractTableModel, QObject, ...). Tělo funkce je pak stejného formátu, dle syntaxe jazyka Python.

Signál je narozdíl od slotu třída. Datové typy jsou vloženy jako parametry konstruktoru. Při inicializaci se třída uloží do proměnné. Název proměnné je důležitý, protože dle konvencí „Handler Event“ systému bude funkce v QML definována jako JavaScript funkce s obdobným názvem jako proměnná (viz kód 5.3).

```
1 optics_result = Signal()
2
3 @Slot(float, int)
4     def optics(self, max_distance, min_points_count):
5         Clustering.my_optics(self._current_dataset.get_df(),
6                              max_distance, min_points_count)
7         self.optics_result.emit()
```

Zdrojový kód 5.2: Ukázka komunikace mezi front-endem a back-endem

Tato metoda (viz kód 5.2) obdrží dva nutné parametry zadané uživatelem pro provedení OPTICS algoritmu. Metoda nevrací žádný výstup do front-endu, pouze vygeneruje graf a pošle prázdný signál informující aplikaci o úspěšném vytvoření.

Ve zdrojovém kódu (viz kód 5.2) je definována metoda „optics“ vyžadující dva zmíněné parametry, které jsou datového typu float a integer. Tyto parametry jsou dále zpracovány jinou metodou „my_optics“. Jelikož výstup této metody není uložen do proměnné, tak i signál, tedy odpověď pro front-end, bude bez parametrů. Signál je poslán pomocí funkce „emit“.

```
Connections {
        target: backend

        function onOptics_result() {
            image.source = '../../tmp/optics.png'
        }
    }
```

Zdrojový kód 5.3: Reference na back-end v QML

Ve front-endu je pak signál nadefinován jako funkce v objektu Connections (viz kód 5.3). V tomto objektu je nejprve nutné určit cíl, tedy referenci, kam má front-end naslouchat pro případný příchozí signál. V tomto případě je cíl „backend“, jak bylo popsáno v textu výše. Název funkce musí mít přesný formát, a to „on[Název signálu definovaný v back-endu začínající velkým písmenem]“. Pokud byl signál definován jako „optics_result“, tak název funkce v QML bude nazvána jako „onOptics_result“.

## 5.2 Způsob uložení dat

### DataFrame

DataFrame je datová struktura, nahrazující relační databázi. Data jsou uložena ve sloupcích a řádcích, tedy ve dvou osách. Každý sloupec má vlastní datový typ. Výhodou DataFramu je vysoký výkon při transformaci dat a zadávání funkcí a výrazů. Další výhodou je využití při analýze dat z důvodu toho, že mnoho těchto knihoven vyžaduje DataFrame jako parametr pro své metody.

Pro autora práce bylo použití DataFrame místo relační databáze vhodnější alternativou z důvodu jednodušší implementace právě do zmíněných analytických knihoven, nebo již stávající znalost této datové struktury.

### TableModel

TableModel je vlastní třída, která vychází z abstraktní třídy QAbstractTable. Každý TableModel obsahuje referenci na jednu instanci třídy DataFrame. Implementace této třídy je nutná k tomu, aby bylo možné data zobrazit ve front-endu jako tabulku. Z důvodu toho, že pro vytvoření dynamické tabulky je potřeba znát počet řádků, nebo na základě indexu řádku a sloupce hodnotu na této pozici, je nutné v těchto předdefinovaných metod doprogramovat logiku na základě datové struktury, ve které se data nacházejí. Mezi povinné metody patří: rowCount, columnCount, data a headerData. Pro názornou ukázku se níže nachází implementace metody „data“ (viz kód 5.4).

```
    def data(self, index, role=Qt.DisplayRole):
        if index.isValid():
            if role == Qt.DisplayRole:
                return str(self._df.iloc[index.row(), index.column
                    ()])
        return None
```

Zdrojový kód 5.4: Implementace metody „data“ pro QTableModel

### Dataset

TableModel obsahuje referenci vždy na jeden DataFrame. Třída Dataset obsahuje referenci na tento TableModel a má přístup jak TableModelu, tak i k DataFramu (viz kód 5.5).

Každý Dataset má svoje vlastní unikátní identifikační číslo, které slouží při selektování tohoto Datasetu.

```
1  class Dataset:
2      _id = 0
3      _name = ""
4      _delimiter = ""
5      _coding = ""
6      _header = True
7      _data_path = ""
8      _file_suffix = ""
9      _table_model = None
10     _backup_df = None
11     _important_attributes = {}
```

Zdrojový kód 5.5: Seznam atributů třídy Dataset

Kódování, hlavička nebo oddělovač je zadán dle toho, z jakého souborového formátu jsou data načítána. Seznam podporujících souborých formátů se nachází v tabulce 5.1.

Tabulka 5.1: Podporované souborové formáty

| Formát souboru | Hlavička | Oddělovač |
|---|---|---|
| csv | ano | ano |
| xls | ne | ne |
| xlsx | ne | ne |
| ods | ne | ne |

### DataAPI

DataAPI je třída obsahující statické metody. Toto API se stará o veškerou manipulaci s daty, ať už již načtených dat v paměti, tak i dat ze souboru.

K datům se přistupuje, a s nimi pak dále manipuluje, pomocí funkcí knihovny pandas.

```
1  @staticmethod
2      def remove_attribute(df, attribute):
3          df.drop(attribute, axis=1, inplace=True)
```

Zdrojový kód 5.6: Odstranění atributu v DataFrame

Statická metoda „remove_attribute“ (viz kód 5.6) odstraní v DataFrame definovaný atribut. K tomu poslouží funkce „drop“, která je pro DataFrame již implementována. Aby k odstranění atributu došlo, je nutné určit, na jaké ose se atribut

nachází. Osa číslo jedna se vztahuje na sloupce, naopak osa číslo nula na řádky. Parametr „inplace“ vynutí provedení této funkce na zvolený DataFrame. V opačném případě by došlo k vytvoření nového DataFrame.

Pro porovnání se zde (viz kód 5.7) nachází obdobná funkce napsána v jazyce SQL pro relační databáze, kde z databáze „criminality“ dojde k odstranění atributu „Date“.

```
1 ALTER TABLE criminality
2 DROP COLUMN Date;
```

Zdrojový kód 5.7: Odstranění atributu v SQL jazyce

## 5.3 Použitá data a rozšíření dat

Pro kvalitnější analýzu je vhodné datovou matici rozšířit o další atributy. K tomu lze využít volně dostupná API, nebo nové atributy vytvořit z již existujících atributů, na základě pravidel.

### 5.3.1 Použitá data

Přestože aplikace umí načíst jakákoliv data v rámci podporovaných souborových formátů, tak pro testování byly použity data z Mapa kriminality (dostupná z https://kriminalita.policie.cz/). Tyto data jsou neveřejná, respektive data lze zadarmo získat pouze s falešnými hodnotami. Pro správné hodnoty je nutné zažádat o přístup. Data je možné stáhnout manuálně skrze webový prohlížeč nebo pomocí API. Toto API využívá autentifikaci pomocí bearer tokenu a HTTP metody GET. Jako argument je právě token, časový interval a identifikační číslo okrsku.

Data jsou ve souborovém formátu CSV a struktura dat je popsána v tabulce 5.2.

Tabulka 5.2: Struktura dat získaných z Mapy kriminality

| Atribut | Popis |
|---|---|
| id | identifikační číslo zločinu |
| x | zeměpisná šířka |
| y | zeměpisná délka |
| mp | ? |
| date | datum a čas |
| state | identifikační číslo závěru |
| relevance | relevance souřadnic |
| types | identifikační číslo typu zločinu |

### 5.3.2 Veřejné zdroje dat

Jako veřejný zdroj dat je použita webová stránka Open-Meteo (dostupné z: https://open-meteo.com/).

Pomocí API a HTTP metody GET jsou z této stránky získávány data o počasí. Jako argument jsou použity souřadnice a datum s časem. Pro zredukování počtů žádostí GET jsou data o počasí získána jako celek, v celém časovém intervalu dat a jako souřadnice je vypočítán centroid (střed) tvořený ze všech těchto dat.

Jako relevantní rozšíření o data počasí byly určeny tyto atributy:

- Minimální teplota,
- maximální teplota,
- průměrná teplota,
- celkové množství srážek,
- celkové množství napadaného sněhu.

### 5.3.3 Extrakce dat

Jednou z možností, jak rozšířit datovou matici je extrakcí dat (feature engineering). Nové atributy a hodnoty těchto atributů jsou získávány z existujících atributů, pro které se definuje pravidlo, na základě kterého budou do nového atributu vkládány hodnoty.

Jako příklad (viz kód 5.8) bude popsáno extrahování dat pro tvorbu nového atributu „roční období“. Jak je známo, roční období dělí rok do 4 části. Tyto části jsou definovány časovým intervalem. Jelikož kriminální záznamy již obsahují hodnoty s datem, je možné toto pravidlo na tyto data aplikovat.

```
1  def extend_data_season(data_set):
2      df = data_set.get_df()
3      date_column = data_set.get_important_attributes().get('date')
4      date_column = to_datetime(df[date_column], format="%d-%m-%Y")
5      def get_season(row):
6          if 80 <= row.day_of_year < 172:
7              return 'Jaro'
8          elif 172 <= row.day_of_year < 264:
9              return 'Leto'
10         elif 264 <= row.day_of_year < 355:
11             return 'Podzim'
12         else:
13             return 'Zima'
14     df['season'] = date_column.apply(lambda row: get_season(row))
```

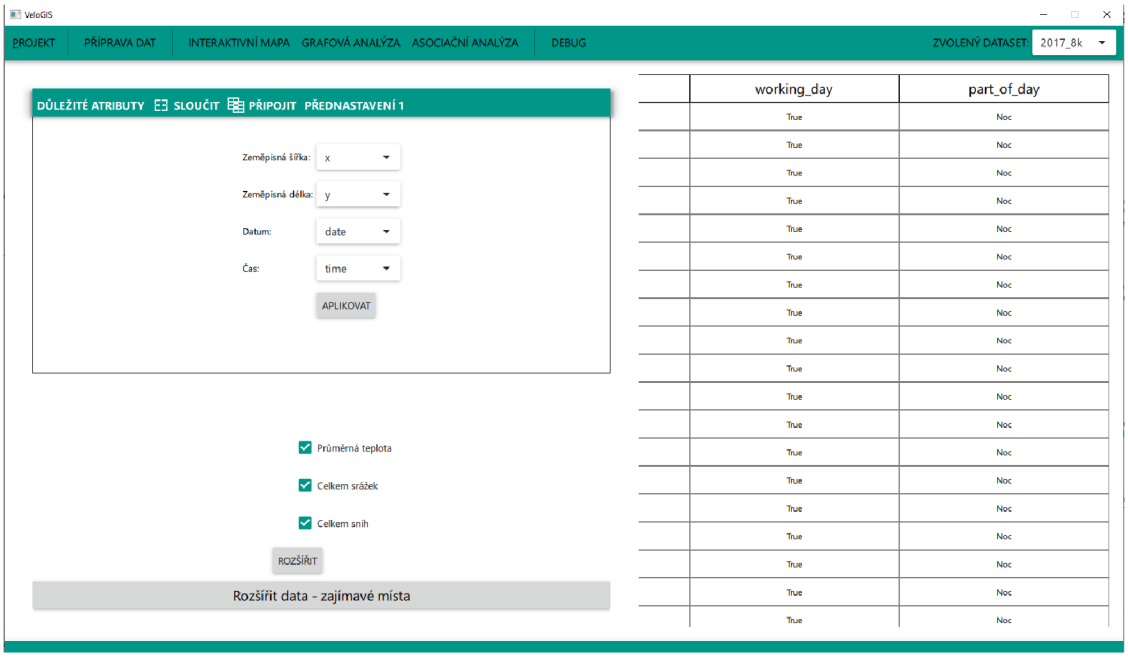
Zdrojový kód 5.8: Rozšíření datasetu o hodnoty ročního období

Z datumu je nejprve získán den. Den v měsíci je přepočítán na den v roce, který je následně poslán jako parametr do vnitřní funkce „get_season“. Vnitřní funkce vrací textový řetězec s hodnotou ročního období. Z důvodu toho, že roční období začíná a končí v různých dnech v měsíci, je tato implementace na den v roce jednodušší.

Tato aplikace obsahuje tento seznam rozšíření extrakcí dat:

- Den v týdnu (pondělí, úterý, ...),

- měsíc (leden, únor, ...),
- roční období (jaro, léto, ...),
- pracovní den (vč. státních svátků),
- část dne (noc, den).




Obrázek 5.2: Ukázka datasetu po extrakci dat

## 5.4 Interaktivní mapa

Mapa je součástí knihovny folium, která za pomocí knihovny Leaflet.js, vytvořenou JavaScriptem, umožňuje tvorbu interaktivní mapy.

Interaktivní mapou je však jen do jisté míry. Do mapy sice lze dynamicky přidávat různé objekty jako jsou body, vyskakovací okna, polygony a další. Nicméně výstupem této knihovny je mapa, která je vytvořena statickým HTML a JavaScript kódem. Z důvodu toho, že mapa obsahuje pouze omezené množství callbacků, tak pro úplnou dynamiku je nutné mapu vždy vygenerovat znovu. Existuje i alternativa ke knihovně využívající Leaflet.js nesoucí název ipyleaflet. Tato knihovna sice nabízí callbacky ke kompletní interaktivitě, ale je o mnoho chudší v podobě pluginů, nástrojů a objektů, které lze na mapu umístit.

Po selekci Datasetu v aplikaci dostane MapAPI signál, aby došlo k vytvoření nové mapy. Nejprve se inicializuje instance třídy Map. Při inicializaci se mapa vycentruje na centroid (střed). Centroid je aritmetickým průměrem všech souřadnic (viz kód 5.9).

```
x_column = self._important_attributes.get('x')
y_column = self._important_attributes.get('y')
xy_count = len(self._df.index)

centroid = [sum(self._df[x_column]) / xy_count,
            sum(self._df[y_column]) / xy_count]

self._m = Map(location=centroid, zoom_start=12, prefer_canvas=True,
    tiles=None)
```

Zdrojový kód 5.9: Výpočet středu ze všech souřadnic

Do mapy je pak vloženo několik mapových podkladů (TileLayer) dostupných z OpenStreetMap. Mapa pracuje pouze s daty, která jsou zrovna vybrána.

Poté dle konfigurace uživatele v ovládacím panelu jsou do mapy postupně přidávány objekty. Objekty jsou vkládány do FeatureGroup, které se pak do mapy vloží jako skupina objektů. Pomocí LayerControlu lze pak u těchto skupin vypínat a zapínat vizualitu.

Níže je ukázka (viz kód 5.10) přidání bodů ve shlucích, které byly přiděleny DBSCAN algoritmem.

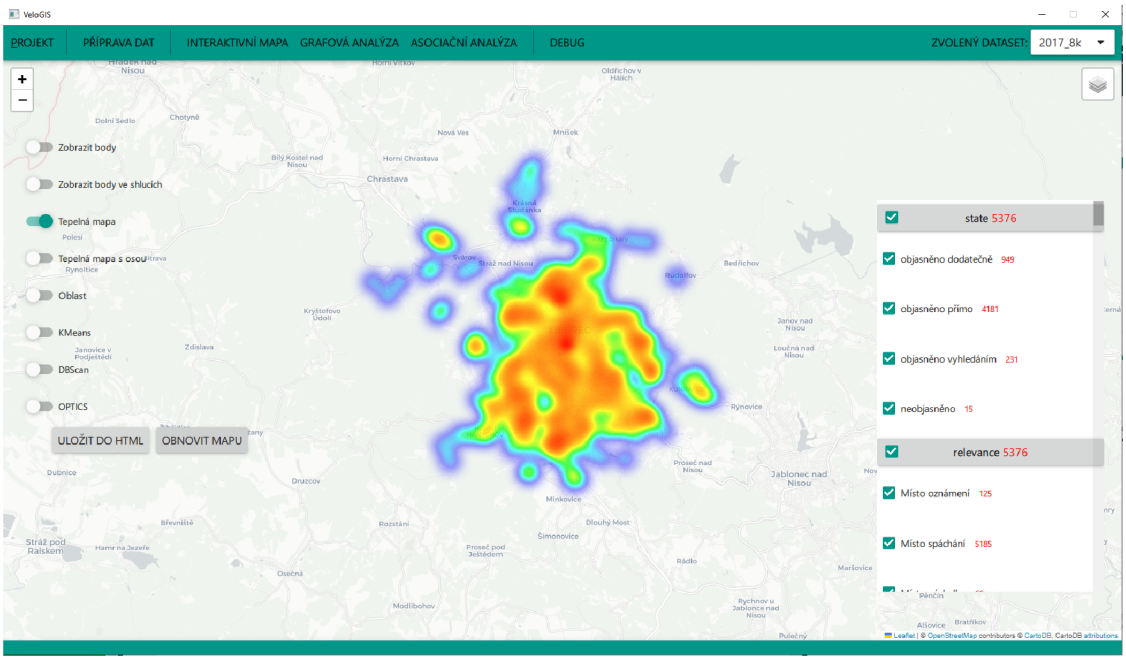
```
def dbscan(self):
    Clustering.my_dbscan(self._df, self._dbscan_settings['epsilon'
       ], self._dbscan_settings['min_points'])

    def random_color():
        return "#" + ''.join([random.choice('0123456789ABCDEF') for
            _ in range(6)])

    clusters = list(self._df['shluk_DBScan'].unique())
    colors = {}
    for i in range(0, len(clusters)):
        if clusters[i] == -1:
            colors[clusters[i]] = "black"
        else:
            colors[clusters[i]] = random_color()
    points = FeatureGroup(name="DBSCAN shluky")
    for index, row in self._df.iterrows():
        if row['shluk_DBScan'] != -1:
            circle = CircleMarker(location=(row['x'], row['y']),
                radius=4, color=colors[row['shluk_DBScan']])
            points.add_child(circle)
    points.add_to(self._m)
```

Zdrojový kód 5.10: Přiřazení DBSCAN shluků a barev k datům v DataFrame

Nejprve se provede samotný DBSCAN algoritmus, který je dostupný v knihovně sklearn. Metoda přijme Dataset a ke každému záznamu přiřadí celé číslo reprezentující o jaký shluk se jedná. Záznam obsahující číslo -1 byl vyhodnocen jako šum (noise). Následně se pro každý shluk vygeneruje náhodná barva, avšak pro šum je rezervována černá barva. Nakonec se body jako kruhové objekty vkládají do mapy.

Front-end interaktivní mapy se skládá ze tří částí. Levého panelu, ve kterém uživatel vybírá pomocí přepínačů a příslušných vstupů pro parametry různé druhy

analýzy. Pravého panelu, ve kterém se nachází seznam všech atributů a možných hodnot. Uživatel pomocí tohoto panelu vybírá podmnožinu z vybrané datové matice, na kterém probíhá analýza. Zbytek okna tvoří samotná interaktivní mapa (viz obr. 5.4).




Obrázek 5.3: Ukázka interaktivní mapy

### Levý panel

Levý panel obsahuje řadu přepínačů s dvěma stavy. Zapnutý stav přidá do vygenerované mapy další operaci analýzy v podobě nové vrstvy. Pokud to operace vyžaduje, tak potřebné parametry při zapnutém stavu se zobrazí u příslušného přepínače. Panel obsahuje tyto přepínače:

1. Body
2. Body ve shlucích
3. Tepelná mapa
4. Tepelná mapa s osou
5. Oblasti
6. DBScan
7. KMeans
8. OPTICS

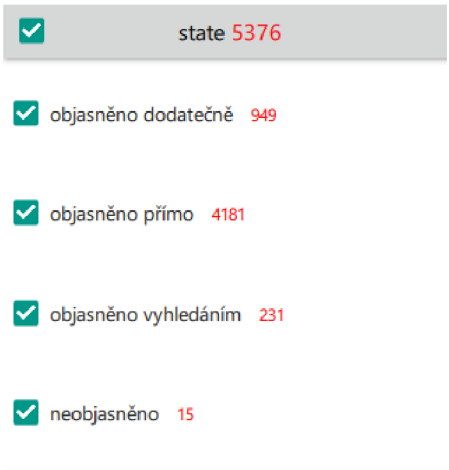
Pod levým panelem se nacházejí dvě tlačítka.

První tlačítko „Obnovit mapu“ aplikuje nové nastavení a vyjde k vygenerování mapy. Nově vygenerovaná mapa nahradí tu předešlou.

Druhé tlačítko „Vytvořit do HTML“ uloží vygenerovanou mapu jako HTML soubor, který lze otevřít všemi webovými prohlížeči. Cestu uložení určí sám uživatel pomocí průzkumníka Windows.

#### Pravý panel

Pravý panel obsahuje filtr záznamů. Nachází se v něm názvy všech nominálních atributů a všechny možné hodnoty těchto atributů. Pomocí zaklikávacích políček lze záznamy odstranit, nebo přidat. Ke každému záznamu je vždy červenou barvou znázorněn počet záznamů s touto hodnotou (viz obr. 5.4). Program očekává velké množství atributů, a tudíž je dostupná i rolovací lišta.




Obrázek 5.4: Pravý panel interaktivní mapy

### 5.4.1 Mřížka

Na tvorbu mřížky (grid) neexistuje žádný dostupný plugin, a proto bylo nutné tuto funkcionalitu naprogramovat od základu.

Cílem bylo rozdělit mapový úsek, ve kterém se nacházejí zločiny, do kvadrantů, tedy rovnoměrných oblastí dle parametrů, které určí uživatel. Každý kvadrant musel být plně interaktivní, tedy měnit svojí barvu dle počtu záznamů v této oblasti nebo po kliknutí zobrazit pop-up. Z toho důvodu je potřeba mřížku vytvořit jako geo-json, nikoliv jako skupinu uspořádaných úseček. Algoritmus lze popsat do těchto tří kroků:

1. Vytvořit geo-json mřížku složenou z kvadrantů po celé rozloze města,
2. přiřadit všechny kriminální zločiny ke svému kvadrantu,
3. pro každý kvadrant vypočítat odchylku od průměrného počtu, a tím určit intenzitu barvy kvadrantu.

### Tvorba geo-json mřížky

Mřížka je listem kvadrantů, které jako celek tvoří čtverec. Parametry pro vytvoření mřížky jsou: počet kvadrantů v řádku nebo sloupci (dle toho se vypočítá délka jednoho kvadrantu) a souřadnice nejjihozápadnějšího bodu a nejseverovýchodnějšího bodu.

Zjištění souřadnic těchto dvou hraničních bodů lze pomocí funkce min a max (viz kód 5.11).

```
1 x_column = df[self._important_attributes['x']]
2 y_column = df[self._important_attributes['y']]
3 # Nejjihozapadnejsi bod
4 lower_left = [x_column.min(), y_column.min()]
5 # Nejseverovychodnejsi bod
6 upper_right = [x_column.max(), y_column.max()]
```

Zdrojový kód 5.11: Detekce dvou hraničních bodů z útvaru tvořeného ze souřadnic

Posledním parametrem je počet kvadrantů ve sloupci. Protože je pro uživatele příjemnější určit vzdálenost v metrických jednotkách, je potřeba počet kvadrantů vykalkulovat. Města mají rozměr délky nebo šířky v jednotkách až desítkách kilometrů, a proto je vhodnou jednotkou metr.

Počet kvadrantů se rovná:

$$početKvadrantů = \frac{délkaMřížky}{délkaKvadrantu}$$

Dle délky a šířky se určí krok, nebo-li velikost kvadrantu, po které cyklus iteruje a vždy z pozice nejjihozápadnějšího vrcholu zjišťuje pomocí tohto kroku zbylé 3 vrcholy (viz kód 5.12).

```
1  x_steps = np.linspace(lower_left[0], upper_right[0], n+1)
2  y_steps = np.linspace(lower_left[1], upper_right[1], n+1)
3  x_size = x_steps[1] - x_steps[0]
4  y_size = y_steps[1] - y_steps[0]
5
6  for x in x_steps[:-1]:
7      for y in y_steps[:-1]:
8          upper_left = [y, x + x_size]
9          upper_right = [y + y_size, x + x_size]
10         lower_right = [y + y_size, x]
11         lower_left = [y, x]
```

Zdrojový kód 5.12: Výpočet velikosti kvadrantu a jeho zbylé souřadnice

### Přiřazení zločinů ke kvadrantům

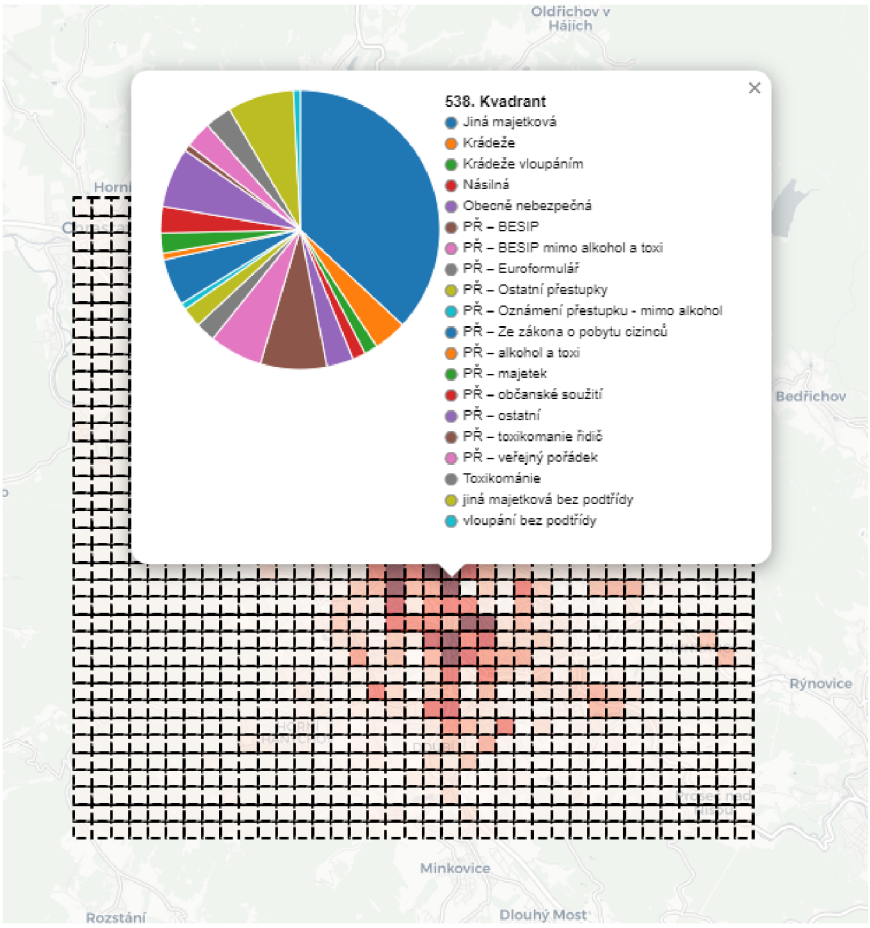
V této fázi jsou všechny kvadranty uložené v listu ve formě čtyř souřadnic. Na každý kvadrant se definuje maska, po její aplikování vznikne podmnožina obsahující všechny kriminální záznamy nacházející se v příslušném kvadrantu. Z důvodu optimalizace jsou všechny tyto body pro další iteraci odstraněny, protože každý záznam může být pouze v jednom kvadrantu.

Při iterování je mimo jiné ukládán počet záznamů v kvadrantu, aby bylo možné stanovit měřítko, podle kterého bude u vizuálního zobrazení kvadrantu určena intenzitou červené barvy. Největší intenzita kvadrantu znázorňuje kvadrant s nejhorší kriminalitou a s nejmenší intenzitou naopak s nejnižší kriminalitou (viz kód 5.13).

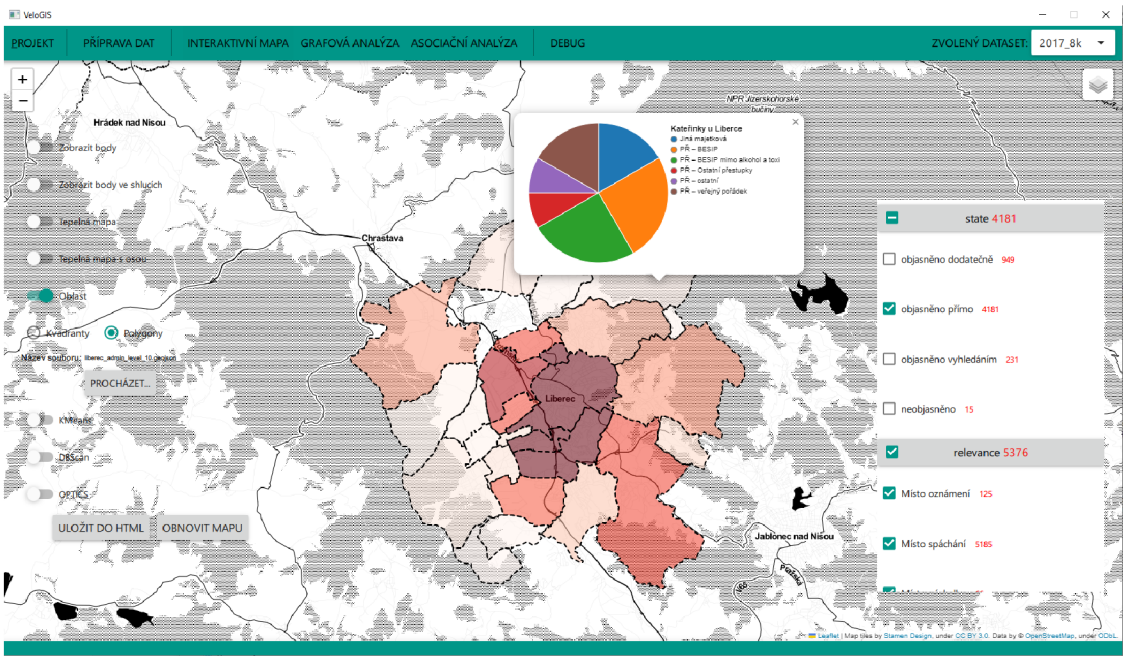
Pro každý kvadrant je také vytvořeno rozklikávací okénko, ve kterého se nachází kruhový diagram (koláčový graf), který ukazuje procentuální počet kriminálních záznamů, které jsou rozděleny dle typu zločinu.

```
worst_quadrant = max(quadrant_counts) / 5
for i, geo_json in enumerate(grid):
    color = plt.cm.Reds(quadrant_counts[i] / worst_quadrant)
    color = mpl.colors.to_hex(color)
    gj = GeoJson(data=geo_json, popup=popups[i],
                 style_function=lambda feature, color=color:
                 {'fillColor': color, 'color': "black", 'weight':
                    2,
                 'dashArray': '5, 5', 'fillOpacity': 0.55})
    grid_group.add_child(gj)
```

Zdrojový kód 5.13: Výpočet intenzity červené barvy pro jednotlivý kvadrant




Obrázek 5.5: Ukázka mapy s 300x300m kvadranty




Obrázek 5.6: Ukázka mapy s polygony

## 5.5 WebEngine

WebEngine je plugin pro framework Qt obsahující webový prohlížeč, jehož jádro je Chromium. Díky němu je možné plně zobrazit HTML kód spolu s CSS a Javascriptem. V této aplikaci je použit na zobrazení interaktivní mapy, který je v back-endu vygenerován jako HTML soubor.

Implementace vlastního prohlížeče je složitou záležitostí. Ve výchozím nastavení prohlížeč podává špatný výkon a je nestabilní. Pro správné fungování mapy, spolu s dalšími tisíci objekty je prohlížeč nejprve nutné nakonfigurovat. Jelikož se jedná o prohlížeč s jádrem Chromium, je plně kompatibilní s příkazy a flags s například Google Chrome. Tento fakt usnadňuje nastavování, protože je možné použít dokumentaci a zdroje všech prohlížečů se stejným jádrem. Pro maximální výkon je nutné zapnout všechny možné akcelerátory, které vynutí použití co nejvíce hardwarové akcelerace a naopak co nejméně softwarové akcelerace.

Prvním krokem je inicializace QtWebEnginu v back-endu. Tento krok je nutný provést před inicializací enginu aplikace, typicky QQmlApplicationEngine.

```
1 QtWebEngineQuick.initialize()
2 engine = QQmlApplicationEngine()
```

Zdrojový kód 5.14: Inicializace QtWebEngine

Poté přichází konfigurace pomocí Chromium flags a systémových proměnných vztahující se k WebEnginu.

```
1 environ["QT_DEBUG_PLUGINS"] = "0"
```

```
2 environ["QTWEBENGINE_CHROMIUM_FLAGS"] = "--disable-backing-store-
    limit --disable-web-security --ignore-gpu-blacklist --enable-
    accelerated-2d-canvas --enable-gpu-rasterization --no-sandbox"
```

Zdrojový kód 5.15: Konfigurace systémových proměnných pro QtWebEngine

Pro debugování je nutné do textového řetězce přidat „–remote-debugging-port=8080“, kde číselný parametr je port pro debug.

Po inicializaci je možné WebEngine vložit do front-endu. V QML se tento objekt nazývá WebEngineView. I ve front-endu lze webengine dále konfigurovat.

```
1  WebEngineView {
2                  id: webView
3                  anchors.fill: parent
4                  onLoadingChanged: zoomFactor = 1.0
5                  settings.javascriptEnabled: true
6                  settings.focusOnNavigationEnabled: false
7                  settings.pluginsEnabled: true
8                  settings.allowRunningInsecureContent: true
9                  settings.allowWindowActivationFromJavaScript: true
10                 settings.javascriptCanPaste: true
11                 settings.localContentCanAccessFileUrls: true
12                 settings.localContentCanAccessRemoteUrls: true
13                 settings.javascriptCanOpenWindows: true
14                 settings.webGLEnabled: true
15                 settings.allowGeolocationOnInsecureOrigins: true
16             }
```

Zdrojový kód 5.16: Konfigurace QtWebEngine v QML

HTML lze načíst ze souboru, nebo jako textový řetězec. Nicméně je doporučeno HTML načítat jako soubor. Vývojář se vyvaruje bugům a pomalého načítání stránek.

Níže se nachází funkce, která načítá mapu do webového prohlížeče. V zakomentovaném řádku (řádek č. 3) se nachází varianta načtení HTML jako textový řetězec. Aby bylo možné mapu obnovit, tedy přepsat celý HTML kód, je vždy nutné předem zavřít objekt Dokument.

```
1 function onMap(html) {
2             webView.runJavaScript("document.close();");
3             //webView.loadHtml(html)
4             webView.url = Qt.resolvedUrl("file:///tmp/test_map.html
                  ")
5             console.log("map loaded")
6         }
```

Zdrojový kód 5.17: Načtení HTML souboru QtWebEnginem

# 6 Závěr

Byla provedena rešerše zahraničních projektů z USA, Německa a Francie, které hledají skryté závislosti v datech o kriminalitě. Pro prevenci kriminality se v těchto státech používají predictive policing systémy, které se na základě vstupních dat snaží kriminalitu předpovídat. Mezi predictive policing systémy patří například PredPol nebo Hunchlab, které jsou používány ve státech napříč USA. Oba systémy používají při predikci kriminality stejné parametry, mezi nimiž jsou například data o počasí nebo společensko-ekonomické indikátory a jejich výstupem jsou hot-spoty vizualizované na mapě. Predictive policing systémy v Německu se spíše specializují na vloupání. Tyto systémy zde od roku 2018 fungují až v šesti spolkových zemích. Mezi zmíněné systémy patří PreMAP používaný ve spolkové zemi Dolní Sasko nebo SKALA, který používají policisté v Severním Porýní. Ve Francii používají systém PAVED, který kromě vloupání predikuje také krádeže aut.

Všechny tyto predictive policing systémy používají pro predikci mapovou nebo asociační analýzu. Součástí mapové analýzy jsou shlukovací algoritmy, mezi které patří například: DBSCAN, KMEANS. Po vytvoření shluku jsou data vizualizována na teplotní mapě, která umožňuje snadné nalezení hot-spotů. Nejpoužívanějším algoritmem v asociační analýze je algoritmus Apriori, který slouží ke konkrétnímu identifikování vzorů mezi atributy kriminálních dat.

Z předchozí analýzy vyplynulo, že aplikace bude muset být schopná manipulovat s velkým množstvím dat a zároveň podporovat výše zmíněné algoritmy k analýze těchto dat, což vedlo k napsání aplikace v programovacím jazyce Python. Tento programovací jazyk obsahuje mnoho knihoven zajišťujících implementaci grafického uživatelského rozhraní. Pro tuto konkrétní aplikaci byla zvolena knihovna PyQt.

Aplikace byla vytvořena tak, aby v ní uživatel dokázal co nejjednodušeji načíst data, manipulovat s nimi, následně použít některý z analytických postupů nebo algoritmů a výsledek analýzy zobrazit buď na mapě, nebo v tabulce. Aplikace dokáže data načíst ze souboru nebo díky API aplikace Mapa kriminality. Data nejsou ukládána do relační databáze, ale jsou netradičně ukládána v datovém typu DataFrame od knihovny pandas. Tento postup byl zvolen proto, protože dalším rozšířením této aplikace je implementace algoritmů strojového učení, které přijímají DataFrame jako vstupní parametr. Data je možné mazat, upravovat nebo je dále rozšiřovat. Dataset je možné rozšířit o data o počasí pomocí implementovaného Open-Meteo API. A nebo je možné použít techniku extrakce z dat.

Z knihovny sklearn byly jako data-miningové nástroje implementovány algoritmy: DBSCAN, KMEANS a OPTICS jako prostředky mapové analýzy s grafickým výstupem a algoritmus Apriori pro asociační analýzu, jejíž výsledek je vyobrazen ve

formě tabulky. Grafický výstup algoritmů mapové analýzy je zajištěn interaktivní mapou z knihovny folium. Tato mapa je generována jako kombinace HTML a JavaScript kódu, proto bylo zapotřebí do aplikace implementovat webový prohlížeč QtWebEngine.

Hlavním výsledkem této práce bylo vyvinutí aplikace VeloGIS, která umožňuje přímé načítání dat v nejpoužívanějších formátech, manipulaci a případné rozšíření těchto dat a následné provedení analýzy s grafickou vizualizací. Oproti programům podobného typu nevyžaduje aplikace VeloGIS po uživateli hluboké porozumění analyzujících algoritmů pro provedení jednoduché analýzy.

# Použitá literatura


[1] PERRY, Walter L., Brian MCINNIS a Carter C. PRICE. Predictive Policing: The Role of Crime Forecasting in Law Enforcement Operations. RAND Corporation [online]. 2013, 186 [cit. 2023-05-15]. Dostupné z: https://www.rand.org/pubs/research_reports/RR233.html

[2] BENBOUZID, Bilel. To predict and to manage. Predictive policing in the United States. Big Data Society [online]. 2019, 6(1) [cit. 2023-05-15]. ISSN 2053-9517. Dostupné z: https://www.researchgate.net/publication/334378627_To_predict_and_to_manage_Predictive_policing_in_the_United_States

[3] Data mining: Úvod do problematiky. Itnetwork.cz [online]. 2022 [cit. 2023-05-20]. Dostupné z: https://www.itnetwork.cz/python/data-mining/data-mining-uvod-do-problematiky

[4] FERGUSON, Andrew. Policing Predictive Policing. Law Reviews Other Academic Journals [online]. 2017, 81 [cit. 2023-05-15]. Dostupné z: https://digitalcommons.wcl.american.edu/facsch_lawrev/749

[5] PredPol - O nás [online]. PredPol, 2020 [cit. 2023-05-15]. Dostupné z:https://www.predpol.com/about/

[6] PredPol · Predictive Policing. (n.d.). PredPol. Srpen 30., 2021. Dostupné z:https://teamupturn.gitbooks.io/predictive-policing/content/systems/predpol.html

[7] Predictive Policing. Teamupturn.gitbooks.io [online]. [cit. 2023-05-20]. Dostupné z: https://teamupturn.gitbooks.io/predictive-policing/content/systems/hunchlab.html

[8] SEIDENSTICKER, Kai, Felix BODE, Florian STOFFEL, srpen 2018. Predictive Policing in Germany

[9] Autor, A. (2020, November 18). «Precobs» - Spezial-Software: Polizei verhindert Einbrüche, bevor sie geschehen. Aargauer Zeitung.https://www.aargauerzeitung.ch/aargau/kanton-aargau/spezial-software-polizei-verhindert-einbruche-bevor-sie-geschehen-ld.1683059

[10] Cécile, Godé; Sébastien, Brion; and Amélie, Bohas, "THE AFFORDANCE-ACTUALIZATION PROCESS IN A PREDICTIVE POLICING CONTEXT: INSIGHTS FROM THE FRENCH MILITARY POLICE" (2020). In Proceedings of the 28th European Conference on Information Systems (ECIS), An Online AIS Conference, Červen 15.-17., 2020. https://aisel.aisnet.org/ecis2020_rp/167

[11] Cluster analysis. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-05-20]. Dostupné z: https://en.wikipedia.org/wiki/Cluster_analysis#Density-based_clustering

[12] Apriori Algorithm in Data Mining: Implementation With Examples. (2021, August 27). Software Testing Help. https://www.softwaretestinghelp.com/apriori-algorithm/

[13] Wirth, R. Hipp, Jochen. (2000). CRISP-DM: Towards a standard process model for data mining. Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining.

[14] VYSKOT, Tomáš. Analýza časových řad pomocí metod data miningu. Pardubice, 2010. Bakalářská práce (Bc.). Univerzita Pardubice, Fakulta ekonomicko-správní.

[15] ADEYIGA, J.A., S.O. OLABIYISI a E.O. OMIDIORA. A comparative analysis of selected clustering algorithms for criminal profiling. Nigerian Journal of Technology [online]. 2020, 39(2), 464-471 [cit. 2023-05-15]. ISSN 2467-8821.

[16] HRUŠKA TVRDÝ, Lubor, Radek FUJAK a Jiří ŠEVČÍK. Mapy budoucnosti - moderní nástroj ke zvýšení efektivity a kvality výkonu veřejné správy v oblasti prevence kriminality založený na analýze a predikci kriminality. Moravská Ostrava: Accendo při vědecko-výzkumném ústavu Accendo - Centrum pro vědu a výzkum, z.ú, 2016. ISBN 978-80-87955-06-2.

[17] Eck, John Chainey, Spencer Cameron, James Leitner, Michael Wilson, Ronald. (2005). Mapping Crime: Understanding Hot Spots. Eck, J.E. and Chainey, S. and Cameron, J.G. and Leitner, M. and Wilson, R.E. (2005) Mapping crime: understanding hot spots. Research report. National Institute of Justice, US.

[18] CO JE STROJOVÉ UČENÍ A JAK SOUVISÍ S UMĚLOU INTELIGENCÍ? [online]. Praha: Rascasone, s.r.o, 2021 [cit. 2023-05-15]. Dostupné z: https://www.rascasone.com/cs/blog/strojove-uceni-ml-metody-klasifikace

[19] RAIDS ONLINE CRIME MAPPING [online]. Swatara Township: Swatara Township Police Department, 2023 [cit. 2023-05-15]. Dostupné z: https://dauphin.crimewatchpa.com/swatarapd/3028/content/raids-online-crime-mapping

[20] K-means clustering. In: Machine Learning for Biostatistics [online]. [cit. 2023-05-21]. Dostupné z: https://bookdown.org/tpinto_home/Unsupervised-learning/k-means-clustering.html