HAW HAMBURG

BACHELORTHESIS
Timur Varli

Crime Prediction mit Machine Learning

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Timur Varli

Crime Prediction mit Machine Learning

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Wirtschaftsinformatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stefan Sarstedt
Zweitgutachter: Prof. Dr. Marina Tropmann-Frick

Eingereicht am: 03. Juni 2025

Timur Varli

Thema der Arbeit

Crime Prediction mit Machine Learning

Stichworte

Machine Learning, Crime Prediction, Predictive Policing

Kurzzusammenfassung

Angesichts zunehmender Verfügbarkeit von Kriminalitätsdaten bieten maschinelle Lernverfahren neue Möglichkeiten zur Identifizierung von Kriminalitätsschwerpunkten und Planung präventiver Maßnahmen. Diese Bachelorarbeit beschäftigt sich mit der Anwendung und dem Vergleich verschiedener Machine-Learning-Algorithmen wie Decision Tree, Random Forest, XGBoost und K-Nearest Neighbors zur Vorhersage von Verbrechen auf Basis des öffentlich zugänglichen San Francisco Crime Datasets. Ziel der Arbeit ist es, die Leistungsfähigkeit unterschiedlicher Klassifikationsmodelle zu bewerten und deren Stärken und Schwächen bei der Klassifikation multiklassiger, unausgewogener Daten zu analysieren. Die besten Ergebnisse wurden mit XGBoost mit Beschränkung auf die zehn häufigsten Verbrechenskategorien erzielt (Accuracy: ~33 %). Dennoch zeigen die insgesamt niedrigen F1-Scores, dass seltene Klassen nur unzureichend vorhergesagt werden. Die Ergebnisse machen deutlich, dass herkömmliche Modelle bei unausgeglichenen Klassenzuweisungen an ihre Grenzen stoßen. Zur Verbesserung der Modellleistung wurde Feature Engineering eingesetzt. Zusätzlich wird auf ethische Herausforderungen und datenschutzrechtliche Probleme eingegangen.

Timur Varli

Title of Thesis

Crime Prediction with Machine Learning

Keywords

Machine Learning, Crime Prediction, Predictive Policing

Abstract

In view of the increasing availability of historical crime data, machine learning methods offer new possibilities for identifying crime hotspots and planning preventive measures. This bachelor thesis deals with the application and comparison of different machine learning algorithms for predicting crime based on the publicly available San Francisco Crime Dataset. The aim of the thesis is to evaluate the performance of different classification models and to analyse their strengths and weaknesses in the classification of multiclass, unbalanced data. The best results were achieved with XGBoost when restricted to the ten most frequent crime categories (Accuracy: ~33 %). Nevertheless, the overall low F1 scores show that rare classes are insufficiently predicted. The results show that conventional models reach their limits with unbalanced class assignments. Feature engineering was used to improve model performance. In addition, ethical challenges and data protection issues are addressed.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

SMOTE	Synthetic Minority Oversampling Technique
ML	Machine-Learning
XGBoost	Extreme Gradient Boosting

Glossar

Modell	Ein mathematisches Konstrukt, das die Beziehungen zwischen verschiedenen Variablen in den Daten beschreibt.
Algorithmus	Ein Verfahren oder eine Abfolge von Regeln, die das Modell erstellen und verbessern.
Training	Der Prozess, bei dem das Modell aus vorhandenen Daten lernt.
Testen	Die Überprüfung der Leistung des Modells anhand neuer, bisher ungesehener Daten.

1 Einleitung

Kriminalität zu verhindern ist ein wichtiger Aspekt in stabilen Gesellschaften. Diese Aufgabe beinhaltet viele Variablen, die nicht unbedingt messbar sind, da Taten mit vielen Punkten, wie mehreren Individuen, kennen sich Opfer und Täter, soziale und wirtschaftliche Lagen und weiteres zusammenhängt. Die Vorhersage ob und wo Kriminalfälle auftreten, und diesen Prozess zu verbessern, ist somit eine wichtige Aufgabe für die Verantwortlichen. Durch die Analyse historischer Kriminalitätsdaten und die Anwendung von Machine-Learning-Modellen können Polizeibehörden Kriminalitätsschwerpunkte identifizieren und vorbeugen. Durch Fortschritte im Bereich des maschinellen Lernens bieten sich neue Möglichkeiten diese Prozesse zu verbessern und genauer zu gestalten. Mithilfe von Crime-Prediction-Modellen, können mit historischen Daten, Muster erkannt werden. Diese können dazu benutzt werden, Polizeipräsenz effizienter zu planen oder Risikogebiete früher zu identifizieren.

Ziel dieser Arbeit ist es, die Klassifikationsleistung verschiedener ML-Algorithmen im Kontext von Crime Prediction zu analysieren, zu vergleichen und anhand geeigneter Metriken zu bewerten.

1.1 Motivation

Kriminalität stellt nicht nur ein großes Problem für die Individuen dar, die Opfer von Kriminalität werden, sondern hat auch soziale und wirtschaftliche Auswirkungen. Mittlerweile werden umfangreiche Daten von der Polizei und Einsatzkräften gesammelt. Damit besteht die Möglichkeit Muster und „Pattern“ deutlich zu machen. In einem Feld, welches sehr vom Zufall geprägt ist. Es stehen auch öffentliche, anonymisierte Datensätze zur Verfügung, vor allem in bestimmten Ländern, wie den USA. Klassische Methoden stoßen bei der Analyse großer Datenmengen und der frühzeitigen Erkennung von Mustern oft an Grenzen. Maschinelle Lernverfahren können helfen, neue Erkenntnisse zu gewinnen.

1.2 Aufbau der Arbeit

Diese Arbeit bearbeitet einen Crime-Datensatz aus San Francisco und wendet an diesem eine Datenanalyse an und zeigt einen Überblick über Auffälligkeiten und Muster. Dann werden vier Algorithmen angewendet und verglichen (Decision Tree, Random Forest, Knn, XGBoost) und diese evaluiert und nach Möglichkeit die Leistungen verbessert. Dann wird ein Überblick über ethische und datenschutzrechtliche Themen gegeben.

- **Kapitel 2** führt in die theoretischen Grundlagen der genutzten Machine-Learning-Algorithmen ein und stellt die konventionelle Kriminalitätsanalyse vor.

- **Kapitel 3** beschreibt den benutzten Datensatz und die Vorverarbeitung der Daten. Außerdem wird eine explorative Datenanalyse durchgeführt.

- **Kapitel 4** stellt die angewendeten Algorithmen vor und beschreibt die Motivation der Wahl der Modelle. Dann werden diese im Modelltraining angewendet und stetig angepasst und nach Möglichkeit verbessert und die Evaluationsmetriken vorgestellt.

- **Kapitel 5** vergleicht die Ergebnisse der Algorithmen anhand der Evaluationsmetriken und diskutiert die Herausforderungen und Chancen der Modelle, sowie die Anpassungen und individuellen Vor- und Nachteile.

- **Kapitel 6** widmet sich den ethischen und datenschutzrechtlichen Aspekten.

- **Kapitel 7** fasst die Ergebnisse zusammen und zeigt einen Ausblick auf die mögliche Entwicklung von Machine Learning im Bereich Crime Prediction auf.

2 Grundlagen

In diesem Kapitel wird ein Überblick über Machine Learning und die verschiedenen Kategorien Supervised, Unsupervised und Reinforcement Learning gegeben. Auf den Decision-Tree-Algorithmus und den KNN-Algorithmus wird genauer eingegangen, da diese in Kapitel 4 angewendet werden.

Außerdem werden traditionelle Methoden und die Bedeutung der Kriminalitätsanalyse vorgestellt.

2.1 Einführung in Machine Learning

Maschinelles Lernen (Machine Learning, ML) ist ein Teilgebiet der Künstlichen Intelligenz (KI), dass sich mit der Entwicklung von Algorithmen und Modellen befasst, die es Computern ermöglichen, aus Daten zu lernen und auf dieser Grundlage Entscheidungen zu treffen oder Vorhersagen zu machen, ohne explizit dafür programmiert zu sein. Anstatt fest programmierte Anweisungen zu befolgen, erkennen ML-Algorithmen Muster in Daten und verbessern ihre Leistung durch Erfahrung (Simon *et al.*, 2015). Machine Learning ist also eine Methode eines Computer Systems, durch Beispiele zu lernen (McClendon and Meghanathan, 2015).

Machine Learning kann in drei Kategorien aufgeteilt werden

(i) Supervised Learning
(ii) Unsupervised Learning
(iii) Reinforcement Learning

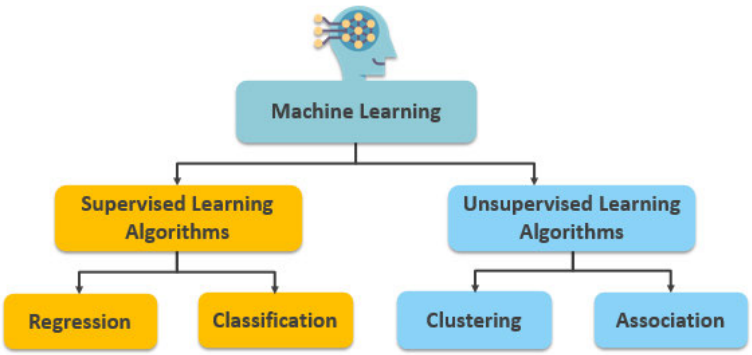

Abbildung 1: Überblick Machine Learning

2.1.1 Supervised Learning

Überwachtes Lernen ist eine Art von ML-Algorithmus, der mit gelabelten Daten lernt.

Gelabelte Daten sind Daten, die mit einer richtigen Antwort oder Klassifizierung gekennzeichnet sind. Dazu ist ein Supervisor als Lehrer anwesend. Jeder Trainingsdatensatz enthält Eingabedaten (Merkmale) und dazugehörige Ausgabedaten (Zielwert). (Liang *et al.*, 2019)

Überwachtes Lernen hat zwei wichtige Kategorien:

1. Klassifizierung
   Bei der Klassifizierung wird ein Algorithmus trainiert, um Eingabedaten in vordefinierte Kategorien oder Klassen einzuordnen. Das Ziel besteht darin, ein Modell zu erstellen, das anhand der Merkmale der Eingabedaten die richtige Klasse vorhersagen kann.
2. Regression
   Regression ist ein Prozess des überwachten Lernens, bei dem ein Algorithmus trainiert wird, um kontinuierliche Zielwerte (anstatt diskreter Klassen) basierend auf Eingabedaten vorherzusagen. Das Ziel der Regression besteht darin, die Beziehung zwischen den Eingabemerkmalen und der Zielvariable zu modellieren und eine Funktion zu finden, die die Zielvariable so genau wie möglich vorhersagt.

Wichtige Algorithmen sind Lineare Regression, Logistische Regression, Random Forest und Naive Bayes.

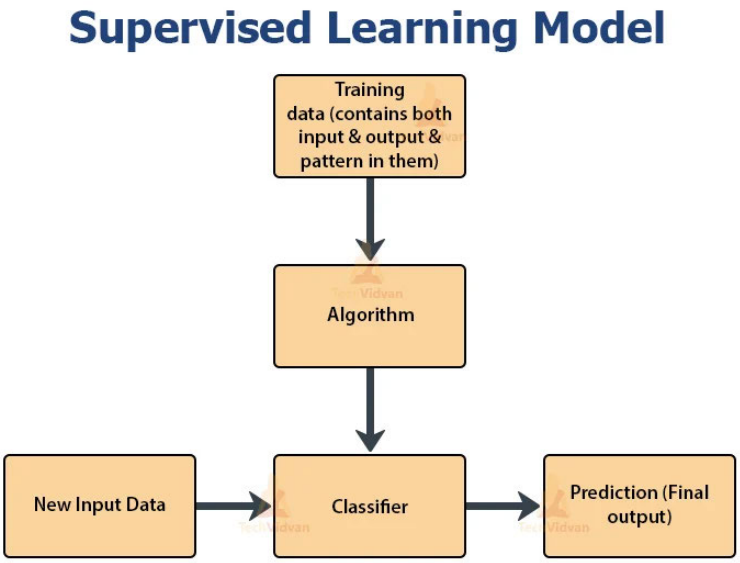

Abbildung 2: Überblick Supervised Learning

Decision-Tree-Algorithmus:

Der Decision-Tree-Algorithmus (Entscheidungsbaum) ist ein Machine-Learning-Algorithmus, der für klassifikatorische und regressive Aufgaben verwendet wird. Er stellt Entscheidungen hierarchisch in Form eines baumartigen Diagramms dar, das aus Knoten und Verzweigungen besteht.

Aufbau eines Decision Trees:

1. **Root Node:**
    - Der Ausgangspunkt des Baums.
    - Hier wird die erste Entscheidungsregel getroffen.
2. **Interne Knoten:**
    - Repräsentieren Zwischenentscheidungen basierend auf bestimmten Merkmalen.
    - Sie teilen die Daten in kleinere Gruppen auf.
3. **Leaf Nodes:**
    - Repräsentieren die Endergebnisse (z. B. eine Klasse oder einen numerischen Wert).

Funktionsweise:

In jedem Schritt werden die Daten mithilfe von zwei Funktionen aufgeteilt:

Gini Impurity und Information Gain.

Die Gini Impurity (Gini Unreinheit) misst die Wahrscheinlichkeit, dass ein zufällig ausgewähltes Element falsch klassifiziert wird:

$$I_G(p) = \sum_{i=1}^{k} p_i(1 - p_i)$$

Information Gain hilft zu entscheiden nach welchen Features aufgeteilt wird. Dafür benutzt es die Entropy Formel (Unordnung):

$$H(T) = I_E = -\sum_{i=1}^{k} p_i log_2 p_i \ \ldots$$

Information Gain wird dann mit folgender Formel berechnet:

```
IG = Entropy(parent) - Weighed Sum of Entropy(children)
```

Die Daten werden aufgeteilt, bis entweder:

- eine maximale Tiefe erreicht ist,
- die Daten nicht weiter aufteilbar sind,
- oder eine bestimmte Mindestanzahl an Datenpunkten pro Knoten vorliegt.

Vorteile:

- Gut geeignet für visuelle Darstellungen.
- Kann sowohl für Klassifikation als auch Regression verwendet werden.
- Keine Skalierung der Daten erforderlich**:** Unempfindlich gegenüber Daten-Normalisierung.

Nachteile:

- Overfitting**:** Bei komplexen Bäumen kann der Algorithmus zu stark an die Trainingsdaten angepasst sein.

- Instabilität**:** Kleine Änderungen in den Daten können zu einem komplett anderen Baum führen.
- Ineffizienz bei hohen Dimensionalitäten**:** Kann bei sehr vielen Merkmalen oder Datenpunkten ineffizient werden.

Anwendung in der Crime Prediction: Ein Decision Tree kann verwendet werden, um Vorhersagen zu treffen, z. B. ob in einer bestimmten Region eine Straftat passieren könnte, basierend auf Merkmalen wie Bevölkerungsdichte, Einkommen, Kriminalstatistiken etc.

k-Nearest-Neighbor-Algorithmus (k-NN)

Der k-Nearest-Neighbor-Algorithmus (k-NN) ist ein einfacher und intuitiver Machine-Learning-Algorithmus, der sowohl für Klassifikations- als auch für Regressionsprobleme verwendet werden kann. Sein Prinzip basiert auf der Annahme, dass ähnliche Datenpunkte in einem Raum nahe beieinander liegen.

$$P(x \in C_i) = \frac{\sum_{j \in C_i} {n_j}/{d_j}}{\sum_{l=1}^{k} {n_l}/{d_l}}$$

Funktionsweise

1. Datenrepräsentation:
    - Jeder Datenpunkt wird durch seine Merkmale (Features) in einem n-dimensionalen Raum dargestellt.

2. Berechnung der Distanz:

    - Um Ähnlichkeiten zwischen Punkten zu bewerten, wird die Distanz zwischen Punkten berechnet. Typische Distanzmetriken sind:

        - **Euklidische Distanz**

        - **Manhattan-Distanz**

3. Parameter kk**:**

    - Der Parameter kk bestimmt, wie viele Nachbarn berücksichtigt werden.
    - Zum Beispiel: Bei k=3k=3 wird auf die drei nächstgelegenen Datenpunkte geschaut.

4. Klassifikation (bei Klassifikationsproblemen):

    - Der Algorithmus zählt, zu welcher Klasse die kk nächsten Nachbarn gehören.
    - Die Klasse, die am häufigsten vorkommt, wird dem neuen Datenpunkt zugewiesen (Mehrheitsentscheidung).

5. Regression (bei Regressionsproblemen):

    - Der Algorithmus berechnet den Durchschnitt (oder eine andere Aggregation) der Zielwerte der kk nächsten Nachbarn.

Vorteile

- Einfach zu verstehen und zu implementieren.
- Keine Annahmen über die Datenverteilung erforderlich.
- Funktioniert gut bei kleinen bis mittelgroßen Datensätzen.

Nachteile

- Rechenaufwand: Bei großen Datensätzen kann die Distanzberechnung für jeden Punkt zeitaufwendig sein.
- Speicherbedarf: Der Algorithmus speichert den gesamten Trainingsdatensatz.
- Empfindlichkeit gegenüber Rauschen: Einzelne Ausreißer können die Vorhersage beeinflussen.
- Wahl von kk: Die Wahl des optimalen kk-Werts kann herausfordernd sein. kk sollte weder zu klein (übermäßige Sensibilität gegenüber Rauschen) noch zu groß (zu starke Glättung) sein.

Beispiel für Crime Prediction

Im Kontext von Crime Prediction kann der k-NN-Algorithmus verwendet werden, um festzustellen, ob ein Gebiet mit bestimmten Merkmalen (z. B. Bevölkerungsdichte, Einkommensniveau, vorherige Kriminalitätsraten) als "hohes Risiko" klassifiziert wird. Hierbei werden ähnliche Gebiete mit historischen Kriminalitätsraten als Nachbarn betrachtet.

2.1.2 Unsupervised Learning

Bei unüberwachtem Lernen hingegen werden die Daten vorher nicht gelabelt. Das Ziel besteht darin, Muster, Strukturen oder Beziehungen in den Eingabedaten zu erkennen, ohne dass es eine spezifische Zielvariable gibt. Anders als beim überwachten Lernen wird also kein Lehrer bereitgestellt und die Maschine nicht trainiert. (Liang *et al.*, 2019)

Unüberwachtes Lernen wird in zwei Kategorien eingeteilt:

1. Clustering
   Clustering ist eine Art des unüberwachtem Lernens, mit dem ähnliche Datenpunkte gruppiert werden. Die Algorithmen funktionieren, indem sie Datenpunkte iterativ näher an ihre Clusterzentren und weiter weg von Datenpunkten in anderen Clustern bewegen.

2. Assoziation
   Assoziationsalgorithmen werden zum Erkennen von Mustern verwendet. Sie funktionieren, in dem sie Beziehungen zwischen verschiedenen Elementen in einem Datensatz finden.

Wichtige Algorithmen sind Principal Component Analysis (PCA), Hierarchical clustering und K-means clustering

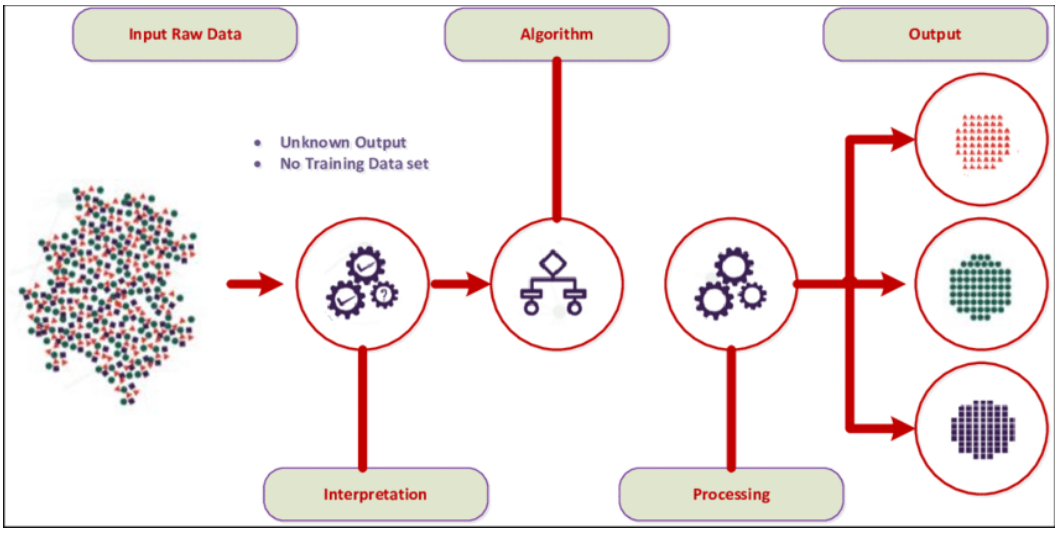

Abbildung 3: Unsupervised Learning im Überblick

2.1.3 Reinforcement Learning

Reinforcement Learning (RL) ist eine Art des maschinellen Lernens, bei dem ein Agent lernt, in einer Umgebung durch trial-and-error zu handeln, um eine maximale kumulative Belohnung zu erzielen. Unter „Agent“ kann man ein lernendes System, das Entscheidungen trifft verstehen. Der Agent trifft Entscheidungen basierend auf dem aktuellen Zustand der Umgebung (Environment) und erhält Rückmeldungen in Form von Belohnungen oder Bestrafungen. Ziel ist es, eine optimale Strategie oder Politik (Policy) zu erlernen, die die langfristige Belohnung maximiert. (Vimala Devi and Kavitha, 2023)

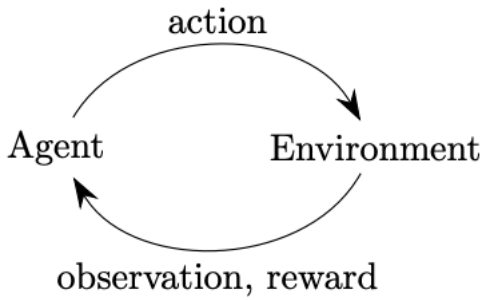

Abbildung 4: Überblick Reinforcement Learning

RL-Algorithmen haben signifikante Erfolge in der Lösung von Spielen erzielt, wie Schach, Go und Videospielen. Außerdem werden sie in der Robotik, bei der Entwicklung von Algorithmen zur Steuerung von autonomen Autos und im Finanzwesen benutzt. (Liang *et al.*, 2019)

Wichtige Algorithmen sind Q-Learning und Deep Q-Networks (DQN)

2.2 Kriminalitätsanalyse

Hier wird ein Überblick über die konventionelle Kriminalitätsanalyse gegeben.

Eine Definition wird vorgestellt und die Ziele dargestellt. Dann werden traditionelle Methoden, welche in der Kriminalitätsanalyse angewendet werden, aufgezeigt.

2.2.1 Definition und Bedeutung

Definition:

Kriminalitätsanalyse ist der Prozess der systematischen Untersuchung von Kriminalitätsdaten, um Muster und Trends zu identifizieren. Diese Analyse hilft dabei, das Verständnis für die Verbreitung und Häufigkeit von Verbrechen zu verbessern und unterstützt die Entwicklung von Strategien zur Verhinderung und Bekämpfung von Kriminalität.(Bogomolov *et al.*, 2014)

Ziele:

Das Ziel der Kriminalitätsanalyse ist es, Entscheidungsträgern und Strafverfolgungsbehörden wertvolle Einblicke zu bieten, die dazu beitragen, öffentliche Sicherheit zu verbessern. Hier sind fünf Punkte, um die Ziele der Kriminalitätsanalyse zu verdeutlichen:

1. Prävention von Verbrechen: Durch die Identifizierung von Mustern und Hotspots können präventive Maßnahmen eingesetzt werden, um die Wahrscheinlichkeit von Straftaten zu verringern.
2. Ressourcenzuweisung: Analysen helfen bei der effizienten Zuweisung von Ressourcen, wie Polizeikräften und Überwachungsmaßnahmen, um Bereiche mit hoher Kriminalitätsrate zu überwachen.

3. Strategische Planung: Langfristige Strategien und Polizeipläne können basierend auf den Analyseergebnissen entwickelt werden, um spezifische Kriminalitätsprobleme anzugehen.
4. Aufklärung und Verfolgung: Die Analyse von Kriminalitätsmustern kann Hinweise zur Aufklärung von Verbrechen liefern und die Ermittlungsarbeit unterstützen.
5. Öffentliche Sicherheit: Durch die Bereitstellung von Informationen an die Öffentlichkeit kann das Sicherheitsgefühl der Bürger gestärkt und die Zusammenarbeit zwischen Gemeinschaft und Strafverfolgung verbessert werden.

1.1.2 Traditionelle Methoden der Kriminalitätsvorhersage

1. Hotspot-Analyse:

Beschreibung: Diese Methode identifiziert geografische Bereiche mit hoher Kriminalitätsrate, sogenannte "Hotspots".

Techniken: Nutzung von Heatmaps und Clustering-Methoden zur Visualisierung und Analyse von Kriminalitätsdaten.

Anwendung: Polizeipatrouillen können gezielt in diesen Bereichen verstärkt werden, um die Kriminalität zu reduzieren. (Sherman, Gartin and Buerger, 1989)

2. Zeitreihenanalyse:

Beschreibung: Analyse der Kriminalitätsrate über einen bestimmten Zeitraum, um Trends und saisonale Muster zu erkennen.

Techniken: Statistische Methoden wie Moving Averages, Autoregressive Modelle (AR), und Seasonality Analysis.

Anwendung: Vorhersage von Kriminalitätsraten für zukünftige Zeiträume und Planung von Polizeieinsätzen. (Chen, Yuan and Shu, 2008)

3. Soziodemografische Analysen:

Beschreibung: Untersuchung des Einflusses soziodemografischer Faktoren wie Alter, Geschlecht, Einkommen und Bildungsstand auf die Kriminalitätsrate.

Techniken: Regression Analysen und Korrelationen zwischen Kriminalitätsdaten und soziodemografischen Variablen.

Anwendung: Identifizierung von Risikogruppen und Entwicklung gezielter Präventionsmaßnahmen.

4. Routine-Aktivitäten-Theorie (Routine Activity Approach)

Beschreibung: Diese Theorie besagt, dass Kriminalität wahrscheinlich dort auftritt, wo sich geeignete Ziele, fehlende Aufsicht und motivierte Täter treffen.

Techniken: Analyse der täglichen Routinen und Bewegungsmuster von Menschen.

Anwendung: Einsatz von Überwachungsmaßnahmen und Polizeipräsenz in Bereichen und zu Zeiten, in denen Kriminalität besonders wahrscheinlich ist. (Cohen and Felson, 2024)

5. Kriminalgeographisches Profiling:

Beschreibung: Analyse der geografischen Muster von Straftaten, um Rückschlüsse auf den Aufenthaltsort eines Täters zu ziehen.

Techniken: Nutzung von GIS (Geografischen Informationssystemen) und geostatistischen Methoden.

Anwendung: Unterstützung von Ermittlungen und Fokussierung auf bestimmte Bereiche zur Täterermittlung.

Diese traditionellen Methoden der Kriminalitätsvorhersage bieten eine solide Grundlage für die Entwicklung und Anwendung moderner Machine Learning-Techniken, die genauere und dynamischere Vorhersagen ermöglichen können.

3 Datenvorverarbeitung und explorative Analyse

In diesem Kapitel wird ein Überblick über die untersuchte Datenmenge gegeben.

Außerdem wird eine Vorverarbeitung (Preprocessing) der Daten vorgenommen. So wird sichergestellt, dass eine Modellierung und Analyse der Daten möglich sind. Dann wird eine explorative Datenanalyse durchgeführt, um Eigenschaften und Strukturen der Daten aufzubereiten, um Erkenntnisse über die Daten zu gewinnen.

3.1 JupyterLab

JupyterLab ist eine webbasierte, interaktive Entwicklungsumgebung, die im Bereich des maschinellen Lernens benutzt wird. In JupyterLab werden auf JSON basierende Notebooks(*.ipynb) verwendet, um Code, Text, Diagramme und Ergebnisse in einer übersichtlichen Form darzustellen und zu bearbeiten. JupyterLab unterstützt durch Kernels viele Programmiersprachen, in diesem Projekt wird Python verwendet.[1]

Um die Crime-Daten vorzubereiten und einen Überblick zu geben wird ein Jupyter Notebook benutzt. Man kann Code ausführen und die Ergebnisse unmittelbar Betrachten, wodurch Datenanalysen effizient und intuitiv ausgeführt werden können.

[1] *What is Jupyter? — Jupyter Documentation 4.1.1 alpha documentation*

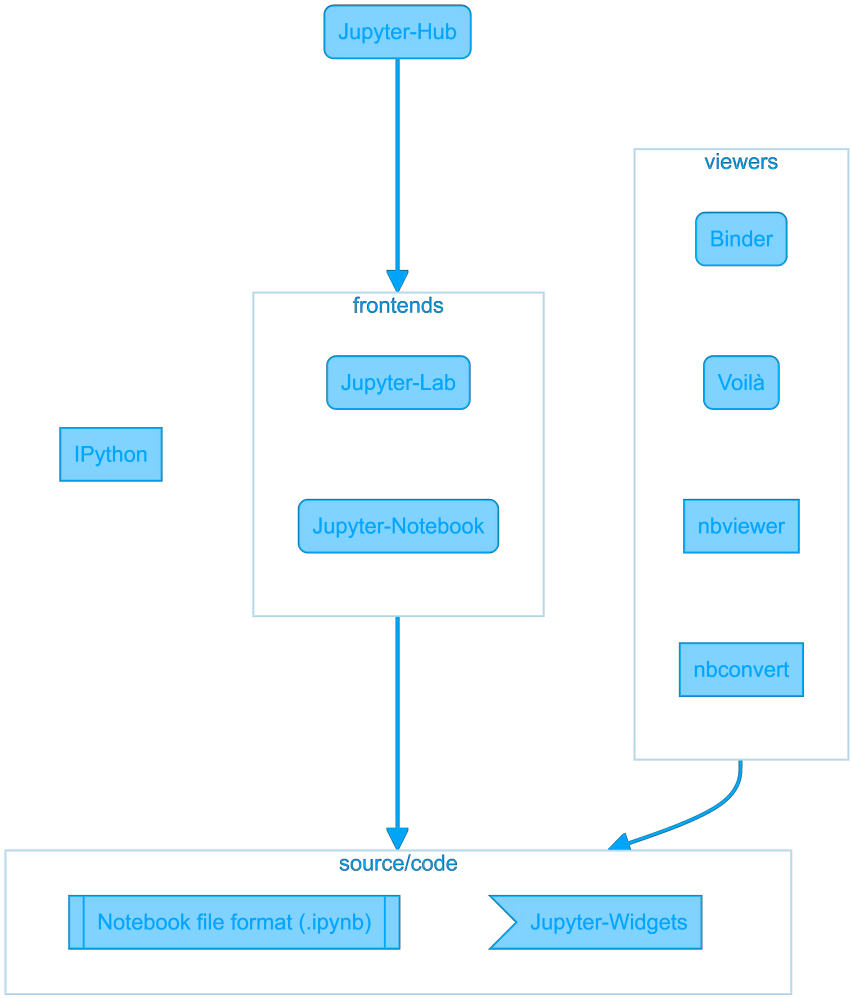

Abbildung 5: Aufbau von Jupyter

3.2 Überblick Datenquellen

Es wird folgender Datensatz für die Untersuchungen benutzt:

- San Francisco, California, Crime Data[2] von 2003 bis Mai, 2018

Es ist ein csv Datensatz mit 2129525 Zeilen. Dieser enthält Informationen über Kriminalfälle vom 01.01.2003 bis zum Mai 2018.

	PdId	IncidntNum	Incident Code	Category	Descript	DayOfWeek	Date
1	4133422003074	041334220	03074	ROBBERY	BBERY, BODILY FORCE	Monday	11/22/2004
2	5118535807021	051185358	07021	VEHICLE THEFT	STOLEN AUTOMOBILE	Tuesday	10/18/2005
3	4018830907021	040188309	07021	VEHICLE THEFT	STOLEN AUTOMOBILE	Sunday	02/15/2004
4	11014543126030	110145431	26030	ARSON	ARSON	Friday	02/18/2011
5	10108108004134	101081080	04134	ASSAULT	BATTERY	Sunday	11/21/2010
6	13027069804134	130270698	04134	ASSAULT	BATTERY	Tuesday	04/02/2013
7	17063991304134	170639913	04134	ASSAULT	BATTERY	Sunday	08/06/2017
8	16020415607020	160204156	07020	VEHICLE THEFT	RECOVERED VEHICLE	Thursday	03/03/2016
9	6068579904134	060685799	04134	ASSAULT	BATTERY	Saturday	06/17/2006
10	5134166327195	051341663	27195	TRESPASS	TRESPASSING	Monday	11/28/2005
11	9098843805041	090988438	05041	BURGLARY	NCE, FORCIBLE ENTRY	Thursday	09/24/2009
12	12620267506244	126202675	06244	LARCENY/THEFT	T FROM LOCKED AUTO	Friday	12/21/2012

Abbildung 6: Ein Beispielausschnitt aus dem Datensatz in Jupyter Lab

[2] https://data.sfgov.org/Public-Safety/Police-Department-Incident-Reports-Historical-2003/tmnf-yvry/about_data

Tabelle 1: Die Attribute des Datensatzes

Category	Art eines Verbrechens (37 Kategorien)
Descript	Beschreibung eines Verbrechens
DayOfWeek	Tag der Woche
Date	Datum
Time	Uhrzeit (Timestamp)
PdDistrict	Name des Polizeidistrikts (10 Distrikte)
Resolution	Auflösung des Verbrechens (12 Ergebnisse)
X	X-Breite des Ortes des Verbrechens
Y	Y-Länge des Ortes des Verbrechens
Location	Ort des Verbrechens

Tabelle 2: Die Häufigkeit der Top 17 Kategorien

Kategorie	**Häufigkeit**
LARCENY/THEFT	477975
OTHER OFFENSES	301874
NON-CRIMINAL	236928
ASSAULT	167042
VEHICLE THEFT	126228
DRUG/NARCOTIC	117821
VANDALISM	114718
WARRANTS	99821
BURGLARY	91067
SUSPICIOUS OCC	79087
ROBBERY	54467
MISSING PERSON	44268
FRAUD	41348
FORGERY/COUNTERFEITING	22995
SECONDARY CODES	22378
WEAPON LAWS	21004
TRESPASS	19194

Es gibt 37 verschiede Arten an Kategorien. Diese werden als Klassen bezeichnet und somit ist ein „multi class problem“ vorhanden.

3.3 Explorative Datenanalyse

Um einen genaueren Überblick über die Daten und die Zusammenhänge zu geben werden diese visualisiert. Dafür wird die seaborn library benutzt, welche ein Python-Toolkit für Datenvisualisierung ist. (Waskom, 2021)

So kann man Muster, Trends und Ausreißer visuell erkennen und das Feature Engineering vorbereiten.

Hier werden die Häufigkeiten der Verbrechenskategorien dargestellt. Man erkennt gut die Top 3: (Saeed and Abdulmohsin, 2023)

- Larcency/Theft (Diebstahl)
- Other Offenses (Sonstige Straftaten)
- Non-Criminal (Nicht-strafrechtlich)

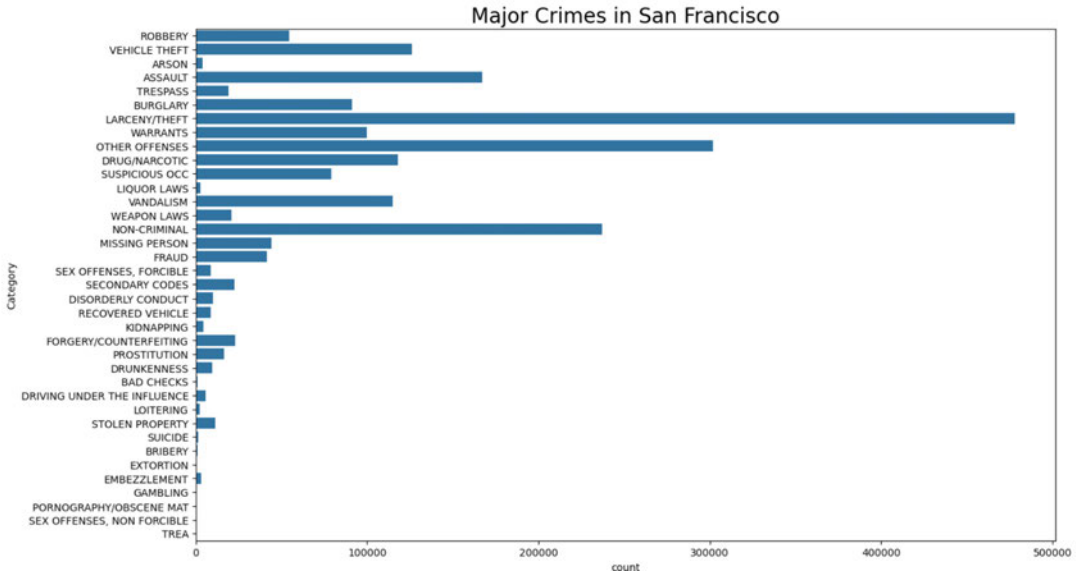

Abbildung 7: Die Häufigkeit der Crime-Katrgorien

In dieser Grafik werden Anzahl der Verbrechen je nach Polizeidistrikt dargestellt.

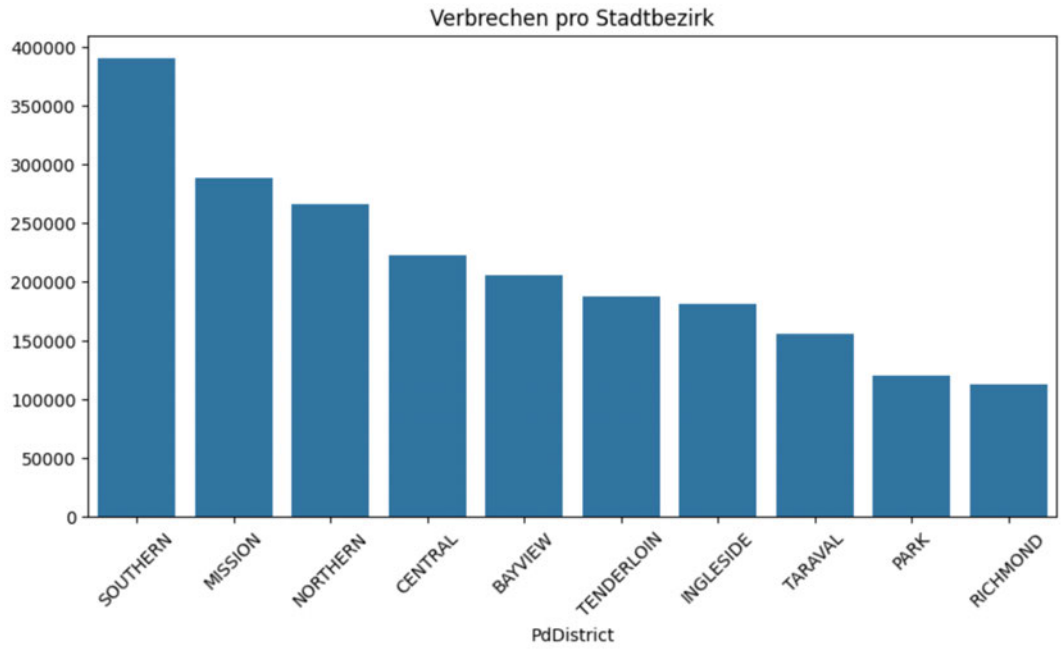

Abbildung 8: Verbrechen nach Polizeidistrikt

Einige Bezirke wie "Southern", "Mission" oder "Northern" weisen mehr Verbrechen auf als andere. Diese Bezirke sind tendenziell zentral gelegen und zeichnen sich durch höhere Bevölkerungsdichte und Nachtleben aus, was die Kriminalitätsrate verstärken kann.

Die räumliche Verteilung der Verbrechen ist nicht gleichmäßig, sondern konzentriert sich auf bestimmte Stadtteile. Diese Information kann für regionale Prognosemodelle oder Hotspot-Erkennung verwendet werden.

Hier wird die Kriminalitätsrate nach Wochentag (Montag bis Sonntag) dargestellt.

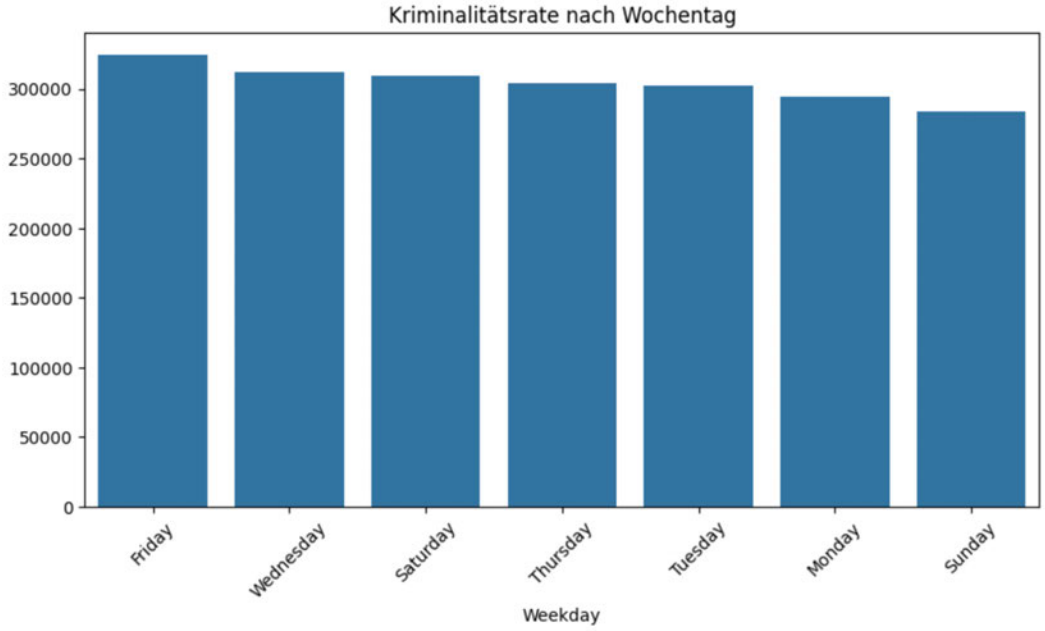

Abbildung 9: Kriminalitätsrate nach Wochentag

3.4 Datenvorverarbeitung

Für die Datenvorverarbeitung ist die Datenbereinigung ein essenzieller Schritt.

Dazu gehört fehlende Werte entfernen oder ersetzen, Duplikate entfernen, Ausreißer erkennen und Datentypen prüfen und konvertieren.

3.4.1 Datenbereinigung

Für das Preprocessing der Daten wird kontrolliert ob null-Werte vorhanden sind:

```
print(df.isnull().sum())
```

```
PdId            0
IncidntNum      0
Incident Code   0
Category        0
Descript        0
DayOfWeek       0
Date            0
Time            0
PdDistrict      1
Resolution      0
Address         0
X               0
Y               0
location        0
```

Abbildung 7: Die Daten vor der Bereinigung.

Man kann sehen, dass in der Kategorie „PdDistrict“ null-Werte vorzufinden sind.

Mit dem Befehl:

```
df["PdDistrict"].fillna(df["PdDistrict"].mode()[0],inplace=True)
```

werden die null-Werte durch die in der Kategorie am häufigsten vorkommende Werte aufgefüllt (Imputation). Danach sieht man, dass die null-Werte nicht mehr vorhanden sind:

```
print(df.isnull().sum())
```

```
PdId            0
IncidntNum      0
Incident Code   0
Category        0
Descript        0
DayOfWeek       0
Date            0
Time            0
PdDistrict      0
Resolution      0
Address         0
X               0
Y               0
location        0
```

Abbildung 8: Die Daten nach der Bereinigung.

Außerdem werden Duplikate entfernt mit:

```
df.drop_duplicates(inplace=True)
```

3.4.2 Kategorische Features encoden

Machine-Learning Modelle können nur mit numerischen Daten arbeiten.

Um Machine-Learning Algorithmen auf den Datensatz anwenden zu können, müssen erst Kategorische Variablen wie zum Beispiel: *PdDistrict*, *Category* und *DayOfWeek* die vom String Typ sind, in numerische Daten umgewandelt werden.[3]

Zu den gängigen Methoden gehören Label Encoding und One Hot Encoding.

Beim Label Encoding wird jeder eindeutigen Kategorie ein ganzzahliger Wert zugewiesen. Diese Methode ist speicher- und recheneffizient, da nur eine Spalte erzeugt wird. Sie ist sinnvoll für ordinal skalierte Merkmale, bei denen eine natürliche Reihenfolge besteht.

Das One Hot Encoding transformiert jede Kategorie in eine separate binäre Spalte (0 oder 1). Diese Methode ist besonders für nominal skalierte Merkmale geeignet, da sie keine künstliche Rangordnung einführt.

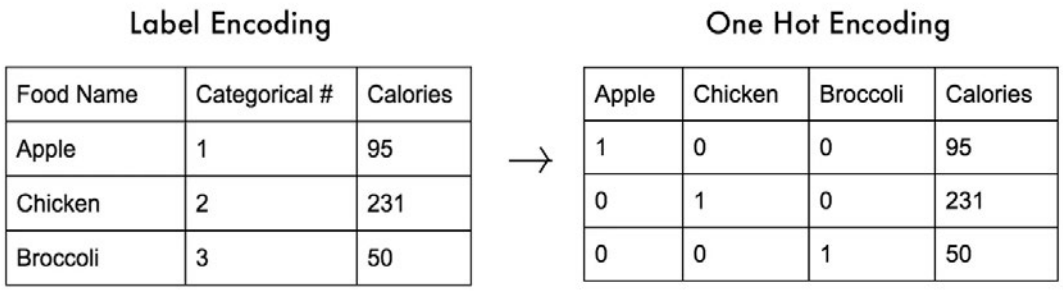

Abbildung 10: Label Encoding und One Hot Encoding

[3] *What is One Hot Encoding and How to Do It | by Michael DelSole | Medium*)

Für die Zielvariable *Category* wird Label Encoding benutzt und für *PdDistrict* und *DayOfWeek* One Hot Encoding.

3.4.3 Feature-Engineering

Mit Feature-Engineering werden aus den Daten, relevante Merkmale erstellt, umgewandelt und kombiniert, um die Machine-Learning Algorithmen leistungsfähiger zu machen. Dies ist ein wichtiger Schritt der Datenvorverarbeitung um die Algorithmen zielführend anzuwenden.

Aus der Spalte „Date“ werden die Features *Year*, *Month*, *Day* und *Weekday* extrahiert.

Außerdem wird ein Feature *NightCrime* erstellt, mit der Tatzeit zwischen 22 und 5 Uhr.

Die Motivation für die Einführung dieses Merkmals liegt in der Annahme, dass bestimmte Verbrechen bevorzugt in den Nachtstunden auftreten, zum Beispiel Einbrüche oder Vandalismus. Durch die explizite Modellierung dieses Verhaltensmusters sollte den Algorithmen eine verbesserte Trennung zwischen verschiedenen Kriminalitätskategorien ermöglicht werden.

```
df["Date"] = pd.to_datetime(df["Date"])  # Datum umwandeln
df["Year"] = df["Date"].dt.year
df["Month"] = df["Date"].dt.month
df["Day"] = df["Date"].dt.day
df["Weekday"] = df["Date"].dt.dayofweek  # Montag=0, Sonntag=6

df["NightCrime"] = df["Hour"].apply(lambda h: 1 if h >= 22 or h <= 5 else
0)
```

Die Spalte Time enthält die Uhrzeit eines Vorfalls (Format: HH:MM). Daraus wird die Stunde (*Hour*) extrahiert:

```
df["Hour"] = pd.to_datetime(df["Time"], format="%H:%M").dt.hour
```

4 Algorithmen auf die Crime Daten anwenden

In diesem Kapitel werden die Machine-Learning Modelle welche auf die Crime Daten angewendet werden ausgewählt und vorgestellt. Diese werden in Jupyter Lab trainiert und hier mit Codeabschnitten vorgestellt. Dann werden die benutzten Evaluationsmetriken vorgestellt.

4.1 Auswahl der Machine-Learning Modelle

Die Auswahl geeigneter Machine-Learning-Modelle ist ein zentraler Bestandteil jeder prädiktiven Analyse, da unterschiedliche Algorithmen unterschiedliche Stärken im Umgang mit Datenstrukturen und Problemstellungen aufweisen.

Ziel dieser Arbeit ist die Klassifikation von Kriminalitätsarten auf Basis zeitlicher, geografischer und kategorischer Merkmale.

Dementsprechend wurden mehrere Modelle in Betracht gezogen, die sich für die Klassifikation in mehr als zwei Klassen eignen und mit gemischten Datentypen umgehen können.

Zu den evaluierten Modellen zählen:

1. Decision Tree: Dient als interpretierbares Basismodell. Es trifft Entscheidungen durch rekursives Aufteilen der Eingabedaten anhand der Merkmale.
2. Random Forest Classifier: Ein ensemblebasiertes Entscheidungsbaumverfahren, das mehrere Decision Trees kombiniert. Random Forests sind zudem interpretierbar und liefern Feature-Importances.
3. XGBoost Classifier: Ein leistungsstarkes Boosting-Verfahren, das oft bei strukturierten Klassifikationsaufgaben angewendet wird. Durch sukzessives

Lernen aus Fehlern vorheriger Modelle erzielt XGBoost häufig sehr hohe Genauigkeiten.

4. K-Nearest Neighbors (KNN): Ein Verfahren, das die Zielklasse basierend auf den ähnlichsten Trainingsbeispielen vorhersagt. Besonders bei niedrigdimensionalen Daten kann KNN nützlich sein, ist jedoch rechenintensiv bei größeren Datensätzen.

Die Auswahlkriterien für die genannten Modelle basieren auf:

- der Fähigkeit, mit multiklassigen Zielvariablen umzugehen,
- der Effizienz bei größeren Datenmengen,
- der Robustheit gegenüber Ausreißern und irrelevanten Features sowie
- der Interpretierbarkeit der Ergebnisse

Die Modelle wurden auf Basis einheitlicher Trainings- und Testdaten trainiert und mithilfe geeigneter Metriken (Accuracy und F1-Score) verglichen. Ziel des Modelltrainings ist es, mithilfe verschiedener überwachter Machine-Learning-Algorithmen Kriminalitätsmuster zu erkennen und zukünftige Vorfälle zu prognostizieren.

4.2 Modelltraining

Nach der Auswahl der vier Klassifikationsmodelle – Decision Tree, Random Forest, XGBoost und K-Nearest Neighbors (KNN) – erfolgt das Training jeweils auf demselben, zuvor bereinigten und vorverarbeiteten Datensatz. Die Daten wurden dabei in Trainings- und Testdaten im Verhältnis 80:20 aufgeteilt.

4.2.1 Decision Tree

Der Decision Tree Algorithmus wird mit der Python-Bibliothek Scikit-Learn implementiert.

Die Zielvariable sind die Verbrechenskategorien (Category) und die wichtigen Merkmale umfassen Year, Month, Day, Hour und PdDistrict. Der Datensatz wird in Training (80%) und Testen (20%) geteilt.

```
# Features und Zielvariable definieren
X = df[["Year", "Month", "Day", "Hour", "Nightcrime"]]
y = df["Category"]

# Datensatz splitten
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Modell trainieren
dt_model = DecisionTreeClassifier(max_depth=5, random_state=42)
dt_model.fit(X_train, y_train)

# Vorhersage und Evaluation
y_pred = dt_model.predict(X_test)
```

```
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Das Decision Tree model ist mit einer maximalen Tiefe von 5 trainiert um overfitting zu vermeiden und Gini Impurity wird an jedem Knoten zur Trennung der Daten benutzt.

4.2.2 Random Forest

Um Ungenauigkeiten, zum Beispiel durch Overfitting zu vermeiden, kann der Random Forest Algorithmus genutzt werden. Es ist ein Ensemble-Learning-Ansatz, der mehrere Entscheidungsbäume kombiniert und so genauere Vorhersagen trifft.

Das Random Forest Modell wurde mit 100 Entscheidungsbäumen (n_estimators=100) und einer maximalen Tiefe von 10 trainiert.

```
# Random Forest Modell trainieren
rf_model = RandomForestClassifier(n_estimators=100, max_depth=10, ran-
dom_state=42)
rf_model.fit(X_train, y_train)

# Vorhersagen & Evaluierung
y_pred_rf = rf_model.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
```

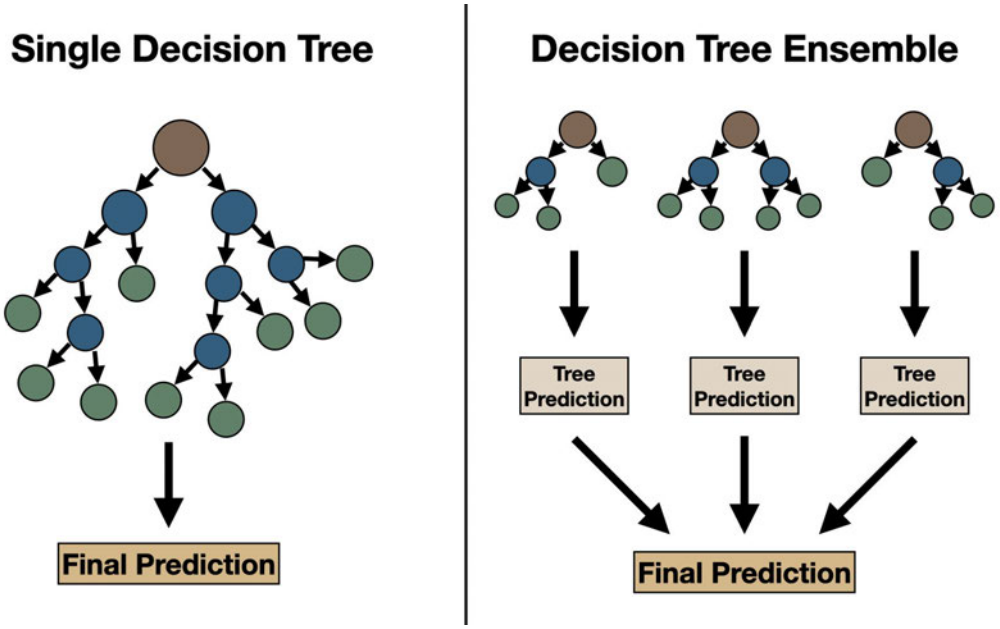

Abbildung 11: Single Decision Tree vs Decision Tree Ensamble

4.2.3 XGBoost (Extreme Gradient Boosting)

Darauf aufbauend wird das XGBoost Verfahren angewandt.

Dieses ist ein leistungsfähiges Ensemble-Verfahren, das zur Gruppe der gradientenbasierten Boosting-Algorithmen zählt. Es wurde mit dem Ziel entwickelt, sowohl effizient als auch hochgradig performant zu sein.

Das Grundprinzip von XGBoost basiert auf Gradient Boosting Decision Trees (GBDT). Dabei werden viele eher schwache Modelle (zum Beispiel: Entscheidungsbäume mit geringer Tiefe) nacheinander trainiert. Jeder neue Baum versucht, die Fehler der vorherigen Modelle zu minimieren. Dadurch entsteht ein starkes Gesamtmodell, dass auch nichtlineare Zusammenhänge und Interaktionen zwischen Merkmalen abbilden kann.(Talebi, 2023)

Im Vergleich zu einem einzelnen Decision Tree oder Random- Forest Modellen bietet XGBoost einige Vorteile:

- Paralleles Training und Regularisierung

- Effiziente Speicher- und Zeitnutzung
- Umgang mit fehlenden Werten

```
# Train-Test-Split (80% Training, 20% Test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# XGBoost Modell erstellen
xgb_model = xgb.XGBClassifier(n_estimators=200, learning_rate=0.1,
max_depth=6, random_state=42)

# Modell trainieren
xgb_model.fit(X_train, y_train)

# Vorhersage treffen
y_pred_xgb = xgb_model.predict(X_test)

# Evaluierung
print("XGBoost Accuracy:", accuracy_score(y_test, y_pred_xgb))
print(classification_report(y_test, y_pred_xgb))
```

4.2.4 KNN-Algorithmus

Neben den vorherigen baumbasierten Verfahren wird auch der K-Nearest-Neighbors (KNN)-Algorithmus als instanzbasiertes Klassifikationsverfahren eingesetzt. KNN zählt zu den einfacheren Verfahren im Bereich des überwachten Lernens. Die Vorhersage erfolgt bei KNN durch einen Vergleich mit den k nächsten Trainingsbeispielen.

Das Modell wurde mit k = 5 Nachbarn und dem euklidischen Abstand als Distanzmaß konfiguriert. Da KNN auf Abständen basiert, wurde vor dem Training eine Standardisierung der Eingabemerkmale mittels StandardScaler durchgeführt, um Verzerrungen durch unterschiedlich skalierte Features zu vermeiden.

Als Eingabemerkmale wurden in dieser Arbeit geografische Koordinaten (X, Y) sowie die Tatzeit (Hour) verwendet. Die Zielvariable wurde zuvor mit Label Encoding in numerische Klassen transformiert. Der Datensatz wurde in einem Verhältnis von 80 % zu 20 % in Trainings- und Testdaten aufgeteilt.

```
# Feature Engineering: Wichtige Merkmale extrahieren
df["Hour"] = pd.to_datetime(df["Time"], format="%H:%M").dt.hour  # Stunde
extrahieren
df = df.dropna(subset=["X", "Y"])  # Fehlende Werte entfernen

# Zielvariable (CrimeType) codieren
le = LabelEncoder()
df["CrimeType"] = le.fit_transform(df["Category"])

# Features und Zielvariable definieren
X = df[["X", "Y", "Hour"]]  # Eingangsmerkmale
y = df["CrimeType"]  # Zielvariable (Verbrechenskategorie)

# Trainings- und Testdaten aufteilen (80% Training, 20% Test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
radom_state=42)
```

```
# KNN-Modell mit k=5 trainieren
knn = KNeighborsClassifier(n_neighbors=5, metric="euclidean")
knn.fit(X_train, y_train)

# Vorhersagen treffen
y_pred_knn = knn.predict(X_test)

# Modell evaluieren
accuracy = accuracy_score(y_test, y_pred_knn)
print("KNN Genauigkeit:", accuracy)
print(classification_report(y_test, y_pred_knn))
```

4.3 Evaluationsmetriken

Um die Algorithmen zu evaluieren und vergleichen werden vier Metriken angewendet die in den folgenden Unterkapiteln vorgestellt werden und im Jupyter Notebook angewandt werden. (*F1 Score in Machine Learning: Intro & Calculation*)

4.3.1 Accuracy

Accuracy (Genauigkeit) beschreibt den Anteil der korrekt vorhergesagten Klassen an der Gesamtzahl aller Vorhersagen.

$$\text{Accuracy} = \frac{\text{Anzahl korrekter Vorhersagen}}{\text{Gesamtanzahl der Vorhersagen}}$$

Die Accuracy ist besonders nützlich bei ausgeglichenen Klassenverteilungen, da sie in solchen Fällen eine gute Gesamtbewertung der Modellleistung liefert. In stark unausgeglichenen Szenarien – wie im vorliegenden Datensatz mit 37 unterschiedlich häufigen Verbrechenskategorien – kann Accuracy jedoch ein verzerrtes Bild liefern, da ein Modell allein durch das Vorhersagen der häufigsten Klassen eine hohe Genauigkeit erreichen kann, ohne die selteneren korrekt zu klassifizieren.

4.3.2 Precision

Precision misst, wie viele der vom Modell als zu einer bestimmten Klasse gehörig vorhergesagten Beispiele tatsächlich korrekt sind.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Diese Metrik ist besonders dann relevant, wenn falsche positive Vorhersagen schwerwiegende Konsequenzen haben, z. B. in sensiblen Bereichen wie der Strafverfolgung oder medizinischen Diagnostik.

Eine hohe Precision bedeutet, dass das Modell bei positiven Vorhersagen verlässlich ist. In dieser Arbeit liefert Precision wichtige Einblicke darin, wie zielgerichtet ein Modell bestimmte Verbrechenskategorien identifiziert, ohne zu viele falsche Treffer zu erzeugen.

4.3.3 Recall

Recall (auch Sensitivität oder True Positive Rate) misst den Anteil der korrekt erkannten tatsächlichen positiven Fälle.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Ein hoher Recall zeigt, dass das Modell viele der tatsächlich vorkommenden Fälle einer Klasse auch wirklich erkennt. Diese Metrik ist besonders entscheidend in Kontexten, in denen das Übersehen relevanter Fälle gravierend ist – beispielsweise bei der Erkennung von Gewaltverbrechen.

Im Rahmen dieser Arbeit dient Recall dazu, zu bewerten, wie zuverlässig ein Modell tatsächliche Kriminalitätsarten erkennt, insbesondere bei Klassen mit geringerer Häufigkeit.

4.3.4 F1-Score

Der F1-Score ist das harmonische Mittel aus Precision und Recall und bietet eine zusammengefasste Bewertung der Modellleistung. (*F1 Score in Machine Learning: Intro & Calculation*, zuletzt aufgerufen: 25.04.2025)

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

Der F1-Score ist besonders nützlich bei unausgeglichenen Datensätzen, da er sowohl die Genauigkeit der Vorhersagen (Precision) als auch die Fähigkeit zur Erkennung (Recall) berücksichtigt.

In dieser Arbeit spielt der F1-Score eine zentrale Rolle zur Bewertung der Modelle, da er eine ausgewogene Sicht auf die Klassifikationsleistung bei häufigen und seltenen Kriminalitätsarten bietet. Der F1-Score wird in dieser Arbeit noch in Macro Average und Weighed Average unterschieden, wobei der Macro Average den ungewichteten Mittelwert unabhängig von der Häufigkeit der Klassen berechnet und beim Weighed Average häufige Klassen stärker gewichtet werden.

5 Ergebnisse und Diskussion

Nach dem Training der vier Klassifikationsmodelle wurde deren Leistung anhand des identischen Testdatensatzes evaluiert. Dabei wurden die Metriken Accuracy, Precision, Recall und der F1-Score berechnet, um die Modellqualität sowohl hinsichtlich der Gesamtleistung als auch im Hinblick auf das Verhalten gegenüber einzelnen Klassen zu bewerten.

5.1 Evaluation der Modelle

5.1.1 Decision Tree

Das Decision-Tree-Modell erreichte auf dem gesamten Datensatz eine Accuracy von 27 %. Bei näherer Betrachtung der weiteren Evaluationsmetriken zeigt sich, dass die Leistung des Modells bei seltenen Klassen schwach ist. Der F1-Score im Macro Average liegt bei 10 %, was auf eine sehr schwache Klassifikationsleistung über alle Klassen hinweg hinweist.

Der Weighted Average F1-Score liegt bei 24 %, was darauf zurückzuführen ist, dass häufig auftretende Kriminalitätsarten wie „Larceny/Theft“ verhältnismäßig gut erkannt werden. Diese häufig vertretenen Klassen verzerren das Gesamtbild jedoch und verdecken die schlechten Ergebnisse bei den selteneren Kategorien.

Die Ergebnisse machen deutlich, dass der Decision Tree trotz seiner guten Interpretierbarkeit mit der hohen Klassenanzahl und dem starken Ungleichgewicht der Datenverteilung nur schwer umgehen kann. Das Modell neigt dazu, die häufigsten Klassen zu bevorzugen, wodurch eine faire und differenzierte Vorhersage aller Klassen erschwert wird.

Trotz dieser Einschränkungen bietet der Decision Tree als baseline-Modell einen guten Ausgangspunkt für den Vergleich mit komplexeren Verfahren wie Random Forest oder XGBoost. Außerdem lassen sich durch zusätzliche Maßnahmen wie Feature Engineering oder Balancing-Methoden wie SMOTE potenziell Verbesserungen erzielen.

Tabelle 3: Ergebnisse des Decison Trees

Metrik	**Ergebnis**
Accuracy	27%
F1-Score (macro avg)	10%
F1-Score (weighed avg)	24%

5.1.2 Random Forest

Im Anschluss an das Decision-Tree-Modell wurde ein Random-Forest-Klassifikator auf dasselbe Datenset angewendet. Mit einer Accuracy von etwa 27,6 % erzielte der Random Forest ein leicht besseres Ergebnis als der einfache Entscheidungsbaum. Wie zu erwarten war, profitiert das Modell durch das Ensemble-Verfahren von einer geringeren Varianz und stabileren Vorhersagen.

Der macro-averaged F1-Score bleibt jedoch insgesamt niedrig, was auf die weiterhin bestehende Schwierigkeit hinweist, seltene Verbrechenskategorien zuverlässig zu erkennen. Auch hier zeigt sich, dass häufige Klassen wie Larceny/Theft deutlich besser vorhergesagt werden als unterrepräsentierte Klassen. Der Vorteil des Random Forest gegenüber dem Decision Tree liegt vor allem in der besseren Generalisierungsfähigkeit, was sich in einer insgesamt leicht erhöhten Klassifikationsqualität widerspiegelt.

Besonders hervorzuheben ist, dass der Random Forest deutlich robuster gegenüber Overfitting ist und sich gut für erste prädiktive Ansätze auf komplexen, realweltlichen Datensätzen eignet. Dennoch bleibt die Herausforderung bestehen, unausgewogene Klassendaten adäquat zu

behandeln, was eine Ergänzung durch Balancing-Verfahren wie SMOTE oder die Einschränkung auf häufige Klassen weiterhin sinnvoll macht.

5.1.3 XGBoost

Der Einsatz des XGBoost-Algorithmus auf das vollständige Datenset mit 37 Verbrechenskategorien erzielte eine Accuracy von 22,9 %. Damit lag XGBoost auf einem ähnlichen Niveau wie Decision Tree und Random Forest, obwohl es sich um ein deutlich leistungsfähigeres Verfahren handelt. Die Ursache liegt in der stark unausgeglichenen Verteilung der Zielklassen, die auch bei diesem Modell zu einer Bevorzugung häufiger Klassen führte.

Ein detaillierter Blick auf den classification report zeigt, dass XGBoost insbesondere bei sehr häufig auftretenden Klassen wie „Larceny/Theft“ vergleichsweise hohe Recall-Werte erreichen konnte. Gleichzeitig blieben seltene Klassen nahezu unberücksichtigt, was zu niedrigen F1-Scores in vielen Kategorien führte. Der insgesamt schwache macro-averaged F1-Score verdeutlicht, dass das Modell in seiner Standardkonfiguration nicht gut mit der Klassenungleichverteilung umgehen kann.

Trotzdem besitzt XGBoost großes Potenzial: Es zeichnet sich durch effizientes Lernen, gute Skalierbarkeit und die Möglichkeit zur Feinjustierung vieler Hyperparameter aus.

Im folgenden Kapitel wird die Reduktion auf die 10 Häufigsten Klassen durchgeführt.

5.1.4 XGBoost: Top 10

Um der Problematik stark unausgewogener Klassen entgegenzuwirken, wurde der XGBoost-Algorithmus zusätzlich auf eine reduzierte Zielvariable mit den zehn häufigsten Verbrechenskategorien trainiert. Durch diese Einschränkung können die selteneren, kaum vorherzusagenden Klassen ausgeschlossen und die Vorhersagegenauigkeit verbessert werden.

Das Modell erreichte eine Accuracy von 33 %, womit XGBoost im Vergleich zu allen anderen getesteten Verfahren die beste Leistung erzielte. Auch der macro-averaged F1-Score stieg

deutlich an, was auf eine insgesamt ausgewogenere Erkennungsleistung über die verbliebenen Klassen hinweist. Besonders gut wurde die Hauptkategorie „Larceny/Theft“ erkannt, jedoch verbesserte sich auch die Vorhersagequalität für weitere Delikte wie „Assault“ oder „Vehicle Theft“.

Die Reduktion auf die Top-10-Kategorien erwies sich somit als wirksame Strategie zur Steigerung der Modellgüte, auch wenn dadurch ein Teil der Informationen verloren geht. Dennoch bietet dieser Ansatz eine Lösung, insbesondere wenn der Fokus auf die präventive Analyse häufiger Kriminalitätsformen liegt.

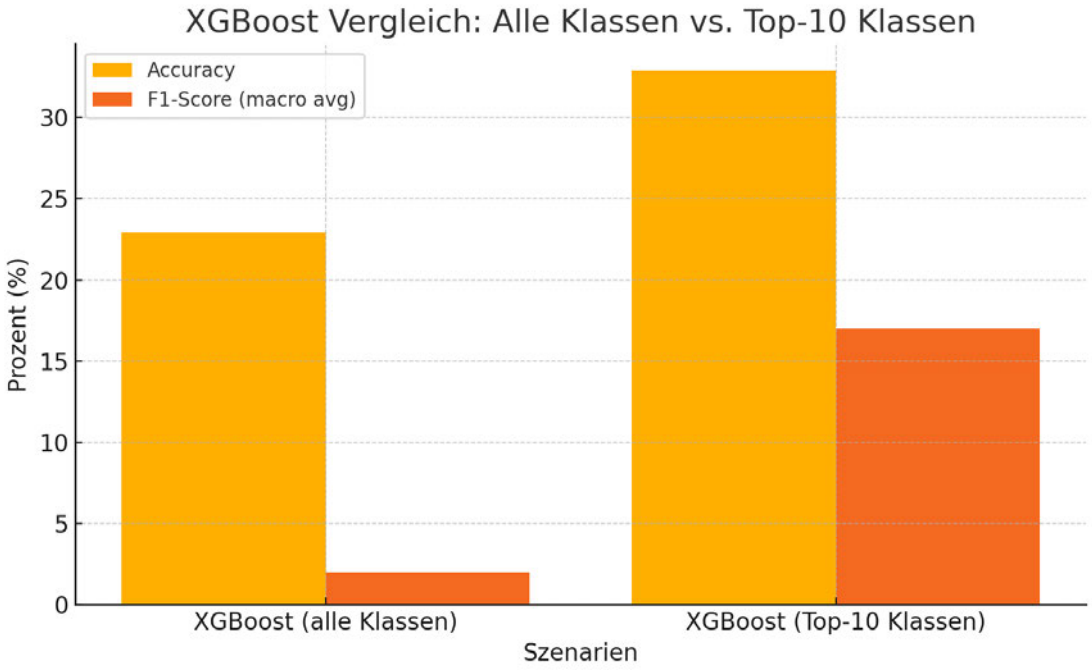

Abbildung 12: XGBoost Vergleich

5.1.5 KNN Algorithmus

Der K-Nearest-Neighbors-Algorithmus (KNN) erreichte eine Accuracy von 24,1 %.

Aufgrund der instanzbasierten Natur von KNN ist das Modell in der Lage, lokale Muster gut zu erfassen, insbesondere in dicht besiedelten Kriminalitätsbereichen.

Der F1-Score im Macro Average betrug 9 %, der Weighted Average F1-Score 22 %.

Während KNN bei häufigen Klassen akzeptable Ergebnisse erzielte, zeigt sich auch hier eine klare Schwäche bei der Erkennung seltener Verbrechen. Zudem wirkt sich die hohe Dimensionalität der Features negativ auf die Modellleistung aus, da der euklidische Abstand bei vielen Merkmalen an Aussagekraft verliert (*Curse of Dimensionality in Machine Learning*).

Insgesamt bietet KNN eine einfache, aber effektive Methode für stark clusterbasierte Daten, ist aber gegenüber komplexeren Verfahren wie XGBoost unterlegen.

5.2 Vergleich der Modelle mit dem Feature Nightcrime

Um die Leistungsfähigkeit der Modelle für die Kriminalitätsvorhersage zu bestimmen, werden die Klassifikatoren anhand der Accuracy und F1-Score verglichen.

Modell	**Accuracy**	**F1-Score(macro avg)**	**F1-Score(weighed avg)**
Decision Tree	27 %	10 %	24 %
Random Forest	28 %	10 %	25 %
XGBoost	29 %	7 %	21 %
KNN	24 %	9 %	22 %
XGBoost Neu	33 %	16 %	25 %

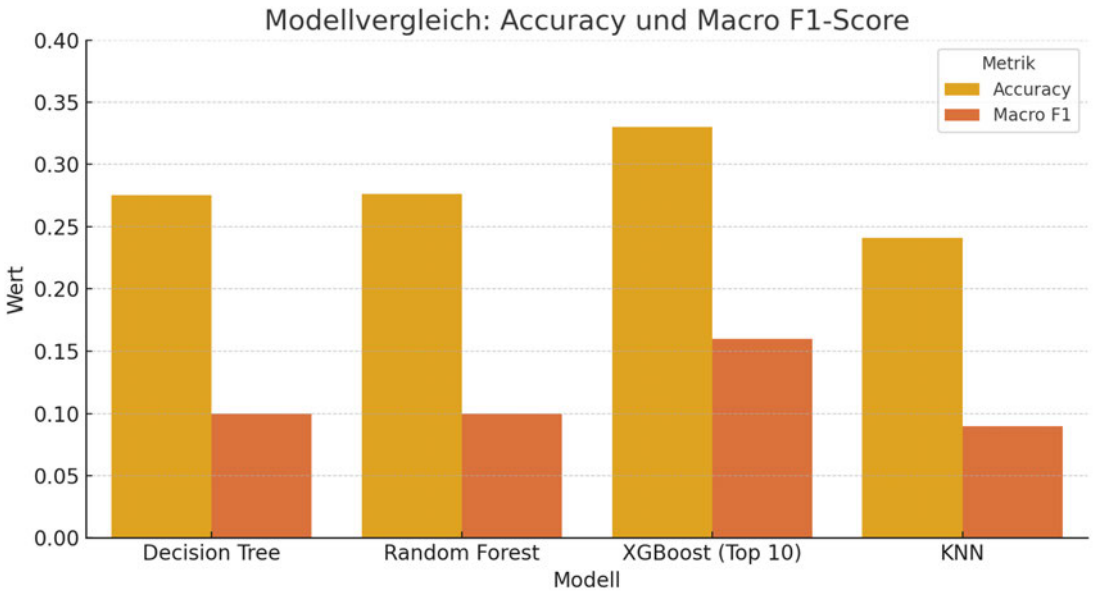

Abbildung 13: Vergleich der Algorithmen

SMOTE (Synthetic Minority Oversampling Technique**)**

Eine weitere Möglichkeit, um Klassen-Ungleichgewichtung in Datensätzen auszugleichen, wie sie in diesem Datensatz vorhanden ist, auszugleichen besteht darin, Oversampling Techniken wie SMOTE anzuwenden.[4]

SMOTE generiert synthetische Datenpunkte für die Minderheitsklassen, um die Anzahl der Stichproben zu erhöhen und so das Modell ausgewogener zu trainieren.[5]

Dabei werden die unterrepräsentierten Daten identifiziert und für jeden Datenpunkt der Minderheitsklasse, die k-nearest neighbors ermittelt. Zwischen dem ursprünglichen Datenpunkt und einem seiner Nachbarn wird ein neuer, synthetischer Datenpunkt erzeugt. Dieser wird dann der Minderheitsklasse hinzugefügt. Der Prozess wird dann für alle Datenpunkte der Minderheitsklasse wiederholt.

Dadurch wird die Modellgenauigkeit bei Minderheitsklassen verbessert. In diesem Projekt wird die imbalanced-learn Bibliothek im jupyter-Notebook verwendet, um SMOTE auf den **XGBoost** Algorithmus anzuwenden:

```
from imblearn.over_sampling import SMOTE

# SMOTE anwenden (nur auf Trainingsdaten)
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

# XGBoost-Modell erstellen und trainieren
model = XGBClassifier(
    n_estimators=200,
    learning_rate=0.1,
    max_depth=6,
```

[4] (*SMOTE — Version 0.13.0*)

[5] (‘Klassen-Imbalance (Machine Learning)’)

```
    use_label_encoder=False,
    eval_metric='mlogloss',
    random_state=42
)
model.fit(X_train_smote, y_train_smote)

# Vorhersagen & Evaluation
y_pred = model.predict(X_test)
print("Accuracy (mit SMOTE):", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Ergebnis:

Kennzahl	Ohne SMOTE	Mit SMOTE
Accuracy	33 %	25 %
F1-Score(macro)	17 %	21 %
F1-Score(weighed)	25 %	26 %

Man sieht am Ergebnis, dass die Accuracy sinkt. Das hängt damit zusammen, dass das Modell sich nicht mehr nur auf die häufigsten Klassen fokussiert. Dafür ist der F1-Score(macro) gestiegen, was zeigt, dass seltener vorkommende Klassen besser erkannt werden. Auch der weighed F1-Score ist leicht gestiegen.

Somit hat SMOTE das Modell faierer gemacht und erkennt mehrere, unterrepräsentierte Klassen besser.

6 Ethik und Datenschutz

Die Anwendung von maschinellem Lernen (ML) in der Kriminalitätsvorhersage ist mit ethischen und datenschutzrechtlichen Herausforderungen verbunden. Während die Modelle dazu beitragen können, Ressourcen effizienter einzusetzen und präventive Maßnahmen zu verbessern, birgt ihr Einsatz auch das Risiko der Diskriminierung, Stigmatisierung und ungewollten Verstärkung bestehender gesellschaftlicher Ungleichheiten.

6.1 Verzerrungen in den Daten (Bias)

Ein zentrales ethisches Problem besteht in der Möglichkeit, dass ML-Modelle bestehende Verzerrungen (Bias) in den Trainingsdaten übernehmen oder sogar verstärken. Kriminalitätsstatistiken spiegeln können durch überproportionale Überwachung bestimmter Bezirke oder soziale Vorurteile beeinflusst sein. Werden solche verzerrten Daten ungefiltert für das Training verwendet, können daraus diskriminierende Vorhersagen resultieren. (Mühlhoff, 2023)

Ein Beispiel hierfür ist der sogenannte "Runaway Feedback Loop": Wenn ein Modell wiederholt bestimmte Stadtteile als Hochrisikogebiete klassifiziert, erhöht sich die Polizeipräsenz dort. Dies führt zu mehr registrierten Straftaten und somit zur weiteren Verstärkung des Risikoprofils unabhängig von der tatsächlichen Kriminalität. Auf diese Weise kann es zu einer systematischen Benachteiligung bestimmter Bevölkerungsgruppen kommen (Ensign *et al.*, 2018).

Schon die Datensammlung von PolizeibeamtInnen beinhaltet immer eine Auswahl. Wenn Einsatzkräfte Kriminaldaten an einem Tatort sammeln, dann suchen sie nach Merkmalen, die es ihnen erlauben, den vorliegenden Tatbestand zu beschreiben und zu katalogisieren. Die Kategorien für Kriminaldaten sind dabei schon im Vorfeld durch das Klassifikationsschema der Datenbank festgelegt, deren Teil sie im späteren Verlauf werden. Beobachtungen, die nicht diesem Schema entsprechen, werden nicht Teil des produzierten Datensatzes.

Außerdem können bei der Datengenerierung Fehler passieren. Am Tatort können Dinge übersehen werden, es können sich Ungenauigkeiten bei der Dateneingabe einschleichen oder Beobachtungen können falsch klassifiziert werden. Fehlerhafte Daten können zu fehlerhaften Risikoprognosen führen. (Leese, 2020)

6.2 Umgang mit Prognosen

Ein weiteres Risiko besteht in der unkritischen Übernahme von Modellvorhersagen als objektive Entscheidungsgrundlage. Maschinelle Lernmodelle können die menschliche Entscheidung nicht ersetzen, höchstens unterstützen. Es sollte immer eine fachliche Einordnung durch Menschen erfolgen, um Fehlinterpretationen zu vermeiden. Auch muss klar definiert sein, wie mit Vorhersagen umgegangen wird:

Dürfen sie Grundlage für Polizeieinsätze sein oder dienen sie nur zur strategischen Ressourcenplanung?

6.3 Datenschutz

Kriminalitätsdaten enthalten häufig sensible Informationen wie geografische Koordinaten, Zeitangaben, Tatbeschreibungen oder Polizeidistrikte. In Einzelfällen könnten sich daraus potenziell Rückschlüsse auf einzelne Personen oder Personengruppen ziehen lassen. Daher ist es essenziell, dass bei der Datenverarbeitung strenge Datenschutzvorgaben eingehalten werden.

In Europa gilt hierbei insbesondere die Datenschutz-Grundverordnung (DSGVO), die unter anderem Datenminimierung, Zweckbindung und Transparenz vorschreibt. (Leese, 2020)

In dieser Arbeit wurden ausschließlich öffentlich zugängliche, anonymisierte Daten aus der Stadt San Francisco verwendet.

7 Fazit und Ausblick

In dieser Arbeit wurde ein Überblick über Crime Prediction im Kontext von Machine Learning und der Entwicklung zu immer mehr Datensammlung gegeben. Es wurde im Hauptteil untersucht, inwieweit sich verschiedene Machine-Learning-Algorithmen zur Vorhersage von Kriminalität eignen. Es wurden diese vier Algorithmen ausgewählt:

- Decision Tree
- Random Forest
- XGBoost
- KNN

Dazu wurden die Daten mit Datenbereinigung, Feature Engineering und explorativer Datenanalyse aufbereitet.

Dann wurden die Ergebnisse mithilfe verschiedener Metriken evaluiert und Vor- und Nachteile diskutiert und Verbesserungen wie zum Beispiel die Behandlung unbalancierter Daten mit SMOTE erarbeitet.

Im Anschluss wurden ethische und datenschutzrechtliche Herausforderung diskutiert.

Die durchgeführte Analyse zeigt, dass sich Machine-Learning Algorithmen grundsätzlich zur Vorhersage von Kriminalitätskategorien eignen aber noch viele Verbesserungen nötig sind. Insbesondere die stark unausgeglichenen Klasssenverteilungen sind eine Herausforderung. Hier wurde ein Ansatz von SMOTE zur Ausbalancierung der Trainingsdaten und eine Anpassung des XGBoost Modells auf die Top-10 Klassen eingesetzt.

Die Accuracy als Metrik ist nur bedingt aussagekräftig, da sie diese Ungleichheiten nicht gut in der Evaluation abbildet. Der F1-Score wurde ergänzend benutzt, um ein genaueres Bild zu liefern.

Insgesamt konnte ein fundierter Überblick über die Stärken und Grenzen aktueller Machine-Learning Verfahren im Bereich der Crime Prediction erreicht werden.

Um die Ergebnisse in Zukunft weiter zu verbessern, kann das Feature-Engineering weiter ausgebaut werden. Auch können weitere Ensemble Ansätze und Modelle wie Zeitreihenanalyse und Clustering angewendet werden. Da Kriminalität mit vielen Faktoren zusammenhängt, können noch Datenquellen zu Wetter, speziellen Events oder sozialen Komponenten hinzugenommen werden.

Künftige Arbeiten können noch aussagekräftigere Datenquellen, zum Beispiel durch weitere Städte und größere Zeiträume abbilden.

Es lässt sich abschließend sagen, dass datengestützte Analyseverfahren bei der Verbrechensprävention helfen können, aber auch Risiken für Grundrechte bestehen. Ein Problem ist auch eine fehlende Transparenz, bei einem gesellschaftlich wichtigen Thema. Eine öffentliche Diskussion ist in Zukunft wichtig.

Echtzeit-Vorhersagen können sich durch Anbindung an Live-Daten durch Social Media und Verkehr können mit zunehmender Digitalisierung der Gesellschaft und Konnektivität von Städten (Smart Cities) weiterentwickeln.

Literaturverzeichnis

Bogomolov, A. *et al.* (2014) ‘Once Upon a Crime: Towards Crime Prediction from Demographics and Mobile Data’, in *Proceedings of the 16th International Conference on Multimodal Interaction*. *ICMI ’14: INTERNATIONAL CONFERENCE ON MULTIMODAL INTERACTION*, Istanbul Turkey: ACM, pp. 427–434. Available at: https://doi.org/10.1145/2663204.2663254.

Chen, P., Yuan, H. and Shu, X. (2008) ‘Forecasting Crime Using the ARIMA Model’, in *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*. *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Jinan Shandong, China: IEEE, pp. 627–630. Available at: https://doi.org/10.1109/FSKD.2008.222.

Cohen, L.E. and Felson, M. (2024) ‘Social Change and Crime Rate Trends: A Routine Activity Approach’.

Curse of Dimensionality in Machine Learning (17:27:26+00:00) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/curse-of-dimensionality-in-machine-learning/ (Accessed: 28 April 2025).

Ensign, D. *et al.* (no date) ‘Runaway Feedback Loops in Predictive Policing’.

F1 Score in Machine Learning: Intro & Calculation (no date). Available at: https://www.v7labs.com/blog/f1-score-guide (Accessed: 28 April 2025).

‘Klassen-Imbalance (Machine Learning)’ (no date) *Marini Systems*. Available at: https://marini.systems/de/glossar/klassen-imbalance-machine-learning/ (Accessed: 11 May 2025).

Leese, M. (2020) ‘Predictive Policing: Umsicht ist das Gebot der Stunde’, p. 4 p. Available at: https://doi.org/10.3929/ETHZ-B-000453441.

Liang, F. *et al.* (2019) ‘Machine Learning for Security and the Internet of Things: The Good, the Bad, and the Ugly’, *IEEE Access*, 7, pp. 158126–158147. Available at: https://doi.org/10.1109/ACCESS.2019.2948912.

McClendon, L. and Meghanathan, N. (2015) ‘Using Machine Learning Algorithms to Analyze Crime Data’, *Machine Learning and Applications: An International Journal*, 2(1), pp. 1–12. Available at: https://doi.org/10.5121/mlaij.2015.2101.

Mühlhoff, R. (2023) ‘Predictive privacy: Collective data protection in the context of artificial intelligence and big data’, *Big Data & Society*, 10(1), p. 20539517231166886. Available at: https://doi.org/10.1177/20539517231166886.

Saeed, R.M. and Abdulmohsin, H.A. (2023) ‘A study on predicting crime rates through machine learning and data mining using text’, *Journal of Intelligent Systems*, 32(1), p. 20220223. Available at: https://doi.org/10.1515/jisys-2022-0223.

Sherman, L.W., Gartin, P.R. and Buerger, M.E. (1989) ‘HOT SPOTS OF PREDATORY CRIME: ROUTINE ACTIVITIES AND THE CRIMINOLOGY OF PLACE*’, *Criminology*, 27(1), pp. 27–56. Available at: https://doi.org/10.1111/j.1745-9125.1989.tb00862.x.

Simon, A. *et al.* (no date) ‘An Overview of Machine Learning and its Applications’.

SMOTE — Version 0.13.0 (no date). Available at: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html (Accessed: 11 May 2025).

Talebi, S. (2023) ‘10 Decision Trees are Better Than 1’, *Towards Data Science*, 27 February. Available at: https://towardsdatascience.com/10-decision-trees-are-better-than-1-719406680564/ (Accessed: 24 May 2025).

Vimala Devi, J. and Kavitha, K.S. (2023) ‘Adaptive deep Q learning network with reinforcement learning for crime prediction’, *Evolutionary Intelligence*, 16(2), pp. 685–696. Available at: https://doi.org/10.1007/s12065-021-00694-8.

Waskom, M. (2021) ‘seaborn: statistical data visualization’, *Journal of Open Source Software*, 6(60), p. 3021. Available at: https://doi.org/10.21105/joss.03021.

What is One Hot Encoding and How to Do It | by Michael DelSole | Medium (no date). Available at: https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179 (Accessed: 5 March 2025).

A Anhang

1. Diese Arbeit als PDF-Datei: Bachelorarbeit_Timur_Varli.pdf
2. Der Quellcode aus dem jupyter Notebook: Crime_Prediction.ipynb

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original