FOM Hochschule für Oekonomie & Management

Hochschulzentrum Stuttgart

Bachelor-Thesis

im Studiengang Informatik

zur Erlangung des Grades eines

Bachelor of Science (B.Sc.)

über das Thema

Einsatz von Machine Learning zur Erkennung von Kreditkartenbetrug: Eine Analyse mit Python und TensorFlow

von

Matthias Kroll

Erstgutachter:	Maher Hamid M.Sc.
Matrikelnummer:	559078
Abgabedatum:	04.05.2024

Bachelor-Thesis *Bachelor Thesis*	9	*1,0 (sehr gut)*
Kolloquium *Colloquium*	3	*1,0 (sehr gut)*

Titel der Bachelor-Thesis *Title of the Bachelor Thesis*

Einsatz von Machine Learning zur Erkennung von Kreditkartenbetrug: Eine Analyse mit Python und TensorFlow

Gutachter/in *Supervisor* Maher Hamid M.Sc.

Zweitgutachter/in *Second Supervisor* Prof. Dr. Peter Preuss

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

Symbolverzeichnis

$\hat{f}$	Vorhergesagter Wert
i	Index i
j	Index j
ωj	Modellparameter
ω_0	Bias-Term
$\omega_1, \omega_2, \dots, \omega_n$	gewichtete Merkmale
a_i	i. Wert des Merkmales
n	Anzahl Merkmale
$P(c\|x)$	Endgültige Klassenwahrscheinlichkeit
$P(x\|c)$	Wahrscheinlichkeit von Merkmalen
$P(c)$	Vorherige Klassenwahrscheinlichkeit
$P(x)$	Vorherige Merkmalswahrscheinlichkeit

Abkürzungsverzeichnis

CRISP-DM	Cross-Industry Standard Process for Data Mining
TDSP	Team Data Science Process
KI	Künstliche Intelligenz
ML	Machine Learning
DL	Deep Learning
NN	Neuronale Netze
VM	Virtuelle Maschine
CPU	Central Processing Unit, auch Prozessor genannt
GPU	Graphics Processing Unit, auch Grafikkarte genannt
SMOTE	Synthetic Minority Over-sampling Technique

1 Einleitung

Kreditkartenbetrug ist ein globales und nationales Problem, das durch das Wachstum der E-Commerce-Branche und -Gemeinschaft drastisch zunehmen wird. Im Jahr 2018 beliefen sich die weltweiten Kosten für Kreditkartenbetrug auf 24,26 Milliarden US-Dollar. Der Anteil der Vorfälle stieg allein im Jahr 2018 um 18,4 % zum Vorjahr und stellte den Kreditkartenbetrug somit auf Platz 1 der Identitätsdiebstähle.[1]

Laut einer Studie von Juniper Research im Juli 2021 werden die Verluste durch Online-Zahlungsbetrug von 2021 bis 2025 kumulativ 206 Milliarden US-Dollar übersteigen. Diese Zahl ist mehr als das Zehnfache des Nettoeinkommens von Amazon im Finanzjahr 2020 vor dieser Studie. Es ist daher dringend notwendig, Betrugsbekämpfung durch den Einsatz von Machine-Learning-Technologien zu priorisieren.[2]

Auch in Deutschland sprechen die Zahlen und Fakten für sich. Laut Bundeskriminalamt sind schätzungsweise mehr als 130 Millionen Zahlungskarten im Gebrauch. Auf internationaler Ebene sind die Inhaber dieser Zahlungskarten aufgrund ihrer hohen Bonität ein bevorzugtes Ziel von Straftätergruppen.[3] Obwohl die Betrugsquoten in der Regel unter 0,1 % liegen, ist der wirtschaftliche Schaden enorm. Im Jahr 2016 beliefen sich die Schäden in Deutschland auf 132 Millionen Euro und es gab über 800.000 Fälle von betrügerischen Kartenzahlungen. Obwohl der Schaden nur einen geringen Anteil an der Gesamtsumme von 643 Milliarden Euro in Deutschland ausmacht, sollte der absolute Schaden dennoch ausreichend Ansporn sein, um mehr Anstrengungen in die Betrugsbekämpfung zu investieren.[4]

Der Großteil der Straftaten wird nicht zur Anzeige gebracht, da die Beträge größtenteils durch die Banken reguliert werden.[5]

Abbildung 1 zeigt eine Statistik des Bundeskriminalamts über die in Deutschland polizeilich erfassten Fälle von EC-Kartenbetrug zwischen 2011 und 2022.

---

[1] vgl. *Shift Credit Card Processing*, 2021.
[2] vgl. *Juniper Research*, 2024.
[3] vgl. *BKA - Bundeskriminalamt*, 2024.
[4] vgl. *Deutsche Bank Research*, 2018.
[5] vgl. *BKA - Bundeskriminalamt*, 2024.

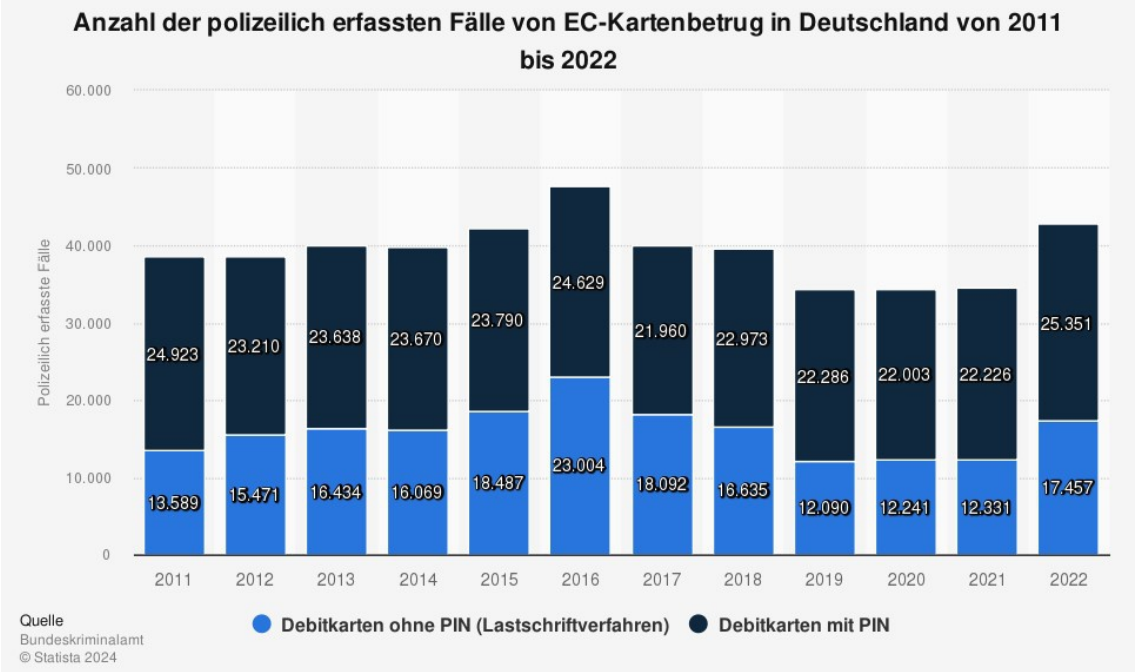


Abbildung 1: Anzahl polizeilich erfasste Fälle von Kartenbetrug in Deutschland[6]

Kreditkartenbetrug ist eine Form des Wirtschaftsbetrugs, die neben der physischen Entwendung der Karte auch im digitalen Zeitalter eine Vielzahl elektronischer Methoden aufweist. Diese umfassen Phishing-E-Mails, gefälschte Internetdienste und -shops, die Nutzung von Datenlecks und Sicherheitslücken sowie manipulierte Lesegeräte in Banken. Es gibt nahezu endlose Möglichkeiten, um Kreditkartenbetrug zu begehen.[7]

Um gegen den Kreditkartenbetrug vorzugehen, werden hauptsächlich zwei Mechanismen eingesetzt: einer, der darauf abzielt, Betrug zu verhindern, und der andere, der darauf abzielt, Betrug zu erkennen.[8]

Die vorliegende Arbeit beschäftigt sich mit der Erkennung von Betrug. Zur Entwicklung eines Computersystems zur Betrugserkennung werden Machine-Learning-Technologien verwendet. Diese Technologie ist eine Anwendung der künstlichen Intelligenz (KI), die es Systemen ermöglicht, durch Erfahrungen zu lernen und sich zu verbessern.

---

[6] *Statista für Bundeskriminalamt*, 2024.
[7] vgl. *Wikipedia*, 2024.
[8] vgl. *Aisha Abdallah, Mohd Aizaini Maarof, Anazida Zainal*, 2016, S. 92.

Machine Learning (ML) konzentriert sich dabei auf die Entwicklung von Computerprogrammen, die durch Daten selbstständig Lernergebnisse erzielen können. Es basiert auf Trainingsdaten, um Verbindungen zwischen ihnen zu verstehen.[9]

1.1 Problemstellung

Obwohl der Einsatz von ML-Technologien vielversprechend ist, ergeben sich bei der Erstellung solcher Modelle spezifische Herausforderungen. Eine erste Herausforderung besteht in der Anonymisierung von Kreditkartendaten, was die Qualität der Daten schwer einschätzbar macht, da wir als Menschen keinerlei Merkmale oder Muster in den Daten ohne Überschriften erkennen können. Die zweite Herausforderung besteht darin, aus der Vielzahl von Methoden und Modellen die geeignetsten Ansätze auszuwählen. Als letzte Herausforderung sind Kreditkartendaten nur ein Auszug aus allen Transaktionen, was, wie bereits in Abschnitt 1 erwähnt, nur etwa 0,1 % der Betrugsfälle ausmacht. Das bedeutet, dass eine starke Datenungleichheit besteht.

Diese Arbeit untersucht die Leitfrage, wie der Einsatz von ML-Technologien, insbesondere unter Verwendung von Python und TensorFlow, zur Aufdeckung und Prävention von Kreditkartenbetrug im Finanzsektor beiträgt.

1.2 Vorgehensweise

Im Vorfeld wird systematische Literaturrecherche nach der Literatur von Brocke et al. (2009) betrieben. Die Recherche erfolgt in fünf Phasen. In Phase eins werden Forschungsfragen und Forschungsgegenstände festgelegt. In Phase zwei erfolgt die Analyse und Definition der Datenbanken. In Phase drei wird Literatur über ‚Backward Search' und ‚Forward Search' gesucht. Phase vier dient der Evaluation der Literatur und Phase fünf der Analyse und Dokumentation.

Als Methodik für die Umsetzung dieser Arbeit stehen CRISP-DM (Cross-Industry Standard Process for Data-Mining), der Data Science Lifecycle und TDSP (Team Data Science Process) zur Auswahl. In dieser Arbeit wird TDSP verwendet, da es besonders für moderne, agile Projekte geeignet ist, insbesondere im Vergleich zu traditionelleren Rahmenwerken wie CRISP-DM.

---

[9] vgl. *Expert.ai*, 2022.

In dieser Arbeit wird im ersten Kapitel mit der Definition des Problems begonnen. Anschließend wird das Themenfeld ML sowie Fraud Detection betrachtet und ein Grundverständnis von Funktionen und Algorithmen erläutert. Im Rahmen der praktischen Umsetzung werden zunächst die technischen Umgebungen vorbereitet, anschließend Trainingsdaten beschafft und Modelle zur Umsetzung entwickelt. Im Fazit werden die Ergebnisse zusammengefasst und es wird ein Ausblick auf die praktische Umsetzbarkeit gegeben.

Die vorliegende Arbeit orientiert sich am Leitfaden zur Gestaltung wissenschaftlicher Arbeiten des Dekanats ING & IT Management der FOM Hochschule (Stand Februar 2022 ITM 1.3).

2 Theoretische Grundlagen

Technologien aus dem Bereich der KI werden zunehmend in unser Leben integriert. Unternehmen setzen verstärkt auf diese Algorithmen, um Prozesse zu vereinfachen und den Verbrauchererwartungen gerecht zu werden. Dabei werden die Begriffe KI, ML, Deep Learning (DL) und Neuronale Netze (NN) oft als Synonyme verwendet, obwohl es sich um verwandte Technologien handelt. Dies führt häufig zu Verwirrung über ihre Unterschiede. Die folgende Grafik in Abbildung 2 soll die Zusammenhänge vereinfachen.[10]

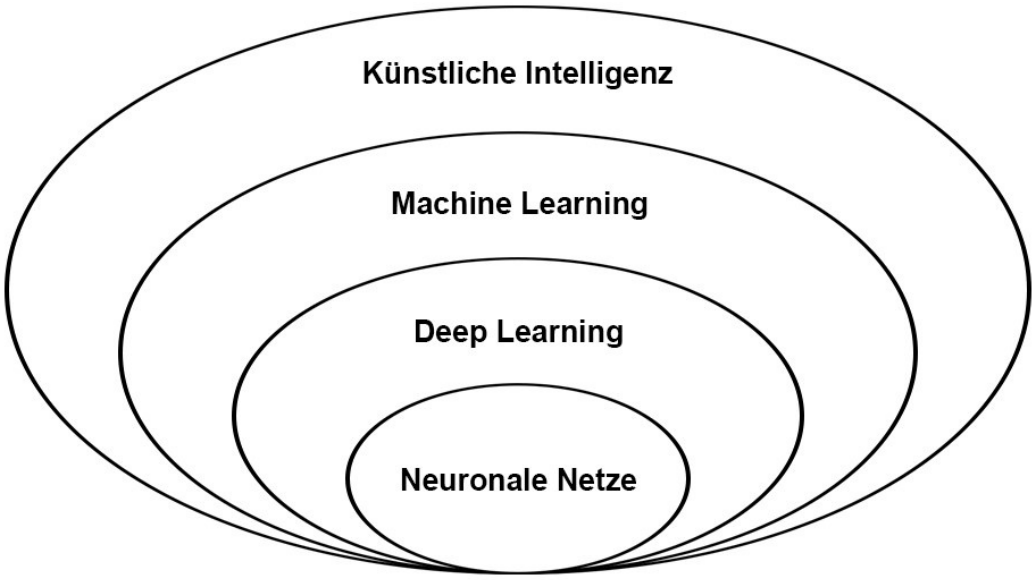


Abbildung 2: KI, ML, DL, NN Übersicht in Anlehnung an IBM Data und AI Team[11]

2.1 Einstieg in die Begrifflichkeiten

2.1.1 Künstliche Intelligenz

Bereits in den 1950er-Jahren wurde die erste KI entwickelt mit dem Ziel, einem Computer das Denken beizubringen.[12] Der berühmte ‚Turing-Test' ist ein Spiel, bei dem ein bewertender Mensch offene Fragen an zwei Teilnehmer stellt: einen anderen Menschen und einen Computer. Das Ziel ist herauszufinden, welcher der Teilnehmer

[10] vgl. *IBM Data and AI Team*, 2024.
[11] *IBM Data and AI Team*, 2024.
[12] vgl. *Chollet*, 2018, S. 22.

der Mensch ist. Wenn keine Entscheidung getroffen werden kann, wird davon ausgegangen, dass es sich um einen intelligenten Computer handelt.[13] Seit ihrem Beginn in den 1950er Jahren, über das goldene Zeitalter der KI von 1956 bis 1974, den KI-Winter bis etwa 1980, die Expertensysteme bis in die 1990er Jahre und bis hin zu den heutigen NN, DL und ML, begleitet uns KI ständig.[14] Die genaue Definition von KI ist aufgrund ihrer vielen Anwendungsbereiche und der Komplexität des Begriffs Intelligenz jedoch nicht einfach zu beantworten.[15] KI in ihrer einfachsten Form verknüpft robuste Datensätze mit dem Gebiet der Informatik, um Problemlösungen zu ermöglichen. Dieser Bereich umfasst auch Teilbereiche wie ML, DL und NL, die auf Grundlage von Eingabedaten Vorhersagen treffen können.[16]

2.1.2 Machine Learning

Computerprogramme so zu entwickeln, dass sie auf der Grundlage von Daten lernen können, ist die Kunst der Wissenschaft von ML.[17] Das Verständnis über Daten, sowie die angemessene Nutzung dieser, bilden das Herzstück von ML. Um Modelle mittels Lernalgorithmen zu trainieren, müssen Daten richtig gesammelt, bereinigt und verarbeitet werden, um Vorhersagen treffen zu können.[18] Die Qualität und der Umfang der Daten sind in allen Fällen ausschlaggebend für den Erfolg der getroffenen Vorhersagen.[19]

ML wird heute als der dominierende Ansatz der KI angesehen. Die Entwicklung selbst ist schwierig und kann als eine Art Kunst betrachtet werden. Bibliotheken von Anbietern wie scikit-learn[20] oder TensorFlow[21], sowie integrierte Werkzeuge wie RapidMiner[22] oder Knime[23], bieten Hunderte von verschiedenen ML-Ansätzen zur Auswahl.[24] ML lernt die Parameter des Modells direkt aus den Merkmalen der Trainingsbeispiele. Die meisten

---

[13] vgl. *Taulli*, 2022, S. 2.
[14] vgl. *Taulli*, 2022, 8-18.
[15] vgl. *Buxmann*, 2021, S. 6.
[16] vgl. *IBM*, 2024.
[17] vgl. *Géron*, 2023, S. 32.
[18] vgl. *Gollapudi*, 2016, S. 4.
[19] vgl. *Mohri/Rostamizadeh/Talwalkar*, 2018, S. 1.
[20] vgl. *scikit-learn*, 2024.
[21] vgl. *TensorFlow*, 2024.
[22] vgl. *RapidMiner*, 2024.
[23] vgl. *KNIME*, 2024.
[24] vgl. *Humm, Bernhard G., Bense, Hermann, Fuchs, Michael, Gernhardt, Benjamin, Hemmje, Matthias, Hoppe, Thomas, Kaupp, Lukas, Lothary, Sebastian, Schäfer, Kai-Uwe, Thull, Bernhard, Vogel, Tobias, Wenning, Rigo*, 2021, S. 109.

Algorithmen für ML sind solche Algorithmen. Ausnahmen bilden das DL, bei dem Algorithmen mit mehr als einer Schicht zwischen Eingabe und Ausgabe verwendet werden.[25]

2.1.3 Deep Learning

Eines der aktuellen Themen in der ML-Forschung ist das DL.[26] Es ermöglicht dem Computer, komplexe Konzepte aus einfacheren zu bilden.[27]

DL-Modelle sind Modelle, die aus vielen Schichten aufgebaut sind und immer größer werden. Sie sollen Algorithmen entwickeln, die komplexe Aufgaben wie ein Gehirn erledigen können.[28] DL zeichnet sich durch die Erkennung von Mustern in unstrukturierten Daten aus, die den meisten Menschen als Medien wie Bilder, Ton, Video und Text bekannt sind.[29]

2.1.4 Neuronale Netze

NN bilden die Grundlage für DL-Algorithmen. Sie ahmen Neuronen im Gehirn nach und werden daher als neuronal bezeichnet. Die Netze bestehen aus Schichten von Knoten, nämlich einer Eingabeschicht, einer oder mehreren verborgenen Schichten und einer Ausgabeschicht.[30]

Jede Einheit in der versteckten Schicht hat einen Wert, eine Gewichtung und einen Bias. Der Bias ist eine Konstante, die zur Berechnung der Funktion verwendet wird, deren Ergebnis die Ausgabe darstellt.[31]

2.2 Datengrundlage

Durch ML wird aus Daten Wissen generiert. Es ist wichtig, die verschiedenen Datenstrukturen zu erläutern.[32]

Insgesamt gibt es mehrere Möglichkeiten, Daten zu organisieren. Diese werden primär in strukturierte und unstrukturierte Daten unterteilt.

---

[25] vgl. *Burkov*, 2020, S. 13.
[26] vgl. *Raschka/Mirjalili*, 2021, S. 409.
[27] vgl. *Goodfellow/Courville/Bengio*, 2016, S. 5.
[28] vgl. *Bonaccorso*, 2018, S. 16.
[29] vgl. *Velayutham*, 2020, S. 383.
[30] vgl. *IBM Data and AI Team*, 2024.
[31] vgl. *Taulli*, 2022, S. 83.
[32] vgl. *Frochte*, 2021, S. 18.

Zunächst sind strukturierte Daten zu erwähnen. Diese Art von Daten ist in Bezug auf die Analyse unkompliziert und in der Regel einfacher zu verarbeiten.[33]

Strukturierte Daten können in Tabellenform dargestellt werden. Dabei enthalten die Spalten Merkmale und die Zeilen Datensätze, die mehrere Merkmale beinhalten können.[34] Unstrukturierte Daten sind unformatierte Daten wie Bilddateien, Musik, Videos und Textdateien. Sie haben keine vordefinierte Formatierung.[35]

Die Einordnung und Bewertung der Trainingsdaten ist von großer Bedeutung. Es gibt eine Vielzahl von schlechten Daten, wie zum Beispiel unzureichende oder nicht repräsentative Trainingsdaten, Daten minderer Qualität, irrelevante Merkmale sowie Overfitting und Underfitting.

Als Erstes die unzureichenden Trainingsdaten. Selbst einfache ML-Aufgaben erfordern normalerweise Tausende von Beispielen, um ein Modell zu generieren. Bei komplexeren Aufgaben wie der Bilderkennung oder Spracherkennung sind sogar Millionen von Beispielen erforderlich.[36]

Nicht-repräsentative Trainingsdaten können zu fälschlicher Modellbildung führen, da sie nicht verallgemeinerbare Situationen enthalten. Daher ist es wichtig, Trainingsdaten so auszuwählen, dass sie verallgemeinerbare Fälle abbilden und nicht nur einen Ausschnitt der Gesamtmenge darstellen. Für ein Beispiel zur Ziffernerkennung sollte das Modell alle Ziffern von 0 bis 9 trainiert und nicht nur einen prozentualen Anteil der Daten erhalten haben, um alle Zahlen mit einem Modell erkennen zu können.[37]

Bei minderwertigen Daten ist es wichtig zu beachten, dass diese keine Fehler, Ausreißer oder Rauschen enthalten. Es ist am sichersten, Merkmale (Spalten) oder Instanzen (Zeilen) mit fehlenden Werten komplett aus den Daten zu entfernen. Sollte das Entfernen aufgrund der geringen Menge an Trainingsdaten nicht umsetzbar sein, sollten fehlende Werte so gut wie möglich ergänzt werden. Hierzu können Interpolationsverfahren eingesetzt werden, um anhand anderer Trainingsdaten die Werte zu schätzen.[38]

---

[33] vgl. *Taulli*, 2022, S. 24.
[34] vgl. *Frochte*, 2021, S. 18.
[35] vgl. *Taulli*, 2022, S. 24.
[36] vgl. *Géron*, 2023, S. 55.
[37] vgl. *Chollet*, 2018, S. 138.
[38] vgl. *Raschka/Mirjalili*, 2021, S. 133–137.

Bei irrelevanten Merkmalen geht es um den Zusammenbau der Trainingsdaten. Es ist wichtig, dass die Trainingsdaten möglichst wenige irrelevante Merkmale enthalten, damit der Algorithmus nicht erst nach diesen Merkmalen suchen muss.[39]

Overfitting, auch Überanpassung genannt, beschreibt das Problem, dass ein Modell sich zu stark an die Trainingsdaten anpasst und dadurch bei neuen Daten schlechte Ergebnisse liefert. Es ist wichtig, dass das Modell nicht nur die Trainingsdaten, sondern auch neue Daten gut verarbeiten kann.

Underfitting, auch als Unteranpassung bezeichnet, beschreibt das Gegenteil von Overfitting. Es tritt auf, wenn ein Modell nicht genügend Informationen berücksichtigt, um die tatsächlichen Daten genau zu modellieren. Wenn Trainingsdatensätze zu klein sind, können keine optimalen Lösungen gefunden werden.[40]

2.3 Unterschiedliche Machine Learning-Systeme

Aufgrund der Vielzahl von Arten von ML-Systemen werden die Kriterien unterschiedlichen Kategorien zugeordnet.

In der ersten Kategorie geht es um die Form des Überwachens. Hier wird zwischen überwachtem, unüberwachtem, teilüberwachtem, selbstüberwachtem und verstärktem Lernen unterschieden.

Überwachtes Lernen, auch als Supervised Learning bezeichnet, ist die häufigste und erfolgreichste Art des ML. Es wird angewendet, wenn ein bestimmtes Ergebnis aus einer gegebenen Eingabe vorhergesagt werden soll. Die Trainingsdaten enthalten die gewünschten Lösungen, auch Labels genannt. Zu den bekanntesten Aufgaben zählen die Klassifikation, wie zum Beispiel der Spamfilter für E-Mails, und die Regression, wie zum Beispiel die Vorhersage von Zielgrößen.[41]

Unüberwachtes Lernen, auch als Unsupervised Learning bezeichnet, beinhaltet keine vorgegebenen Lösungen, auch Labels genannt, innerhalb der Trainingsdaten. Der Algorithmus muss selbstständig ein Modell der Daten erstellen, um unerwartete Muster zu erkennen. Bekannte Anwendungen sind die Anomalieerkennung und das Clustering.[42]

---

[39] vgl. *Géron*, 2023, S. 58 f.
[40] vgl. *Gollapudi*, 2016, S. 12.
[41] vgl. *Müller/Guido*, 2017, S. 25.
[42] vgl. *Shai, Shalev-Shwartz, Shai, Ben-David*, 2014, S. 22 f.

Das teilüberwachte Lernen ist eine Mischform aus beiden Lernmethoden. Es entstand aus der Notwendigkeit, dass das Labeling sehr zeitaufwändig und teuer ist. Daher gibt es bei den Trainingsdaten nur wenige gelabelte und sehr viele ungelabelte Daten. In der Regel werden Algorithmen aus einer Kombination von überwachtem und unüberwachtem Lernen verwendet.

Bei der Form des selbstüberwachten Lernens werden vollständig gelabelte Datensätze aus unvollständig gelabelten Datensätzen erstellt. Sobald der Datensatz vollständig gelabelt ist, kann jeder Algorithmus mit überwachtem Lernen die Daten weiterverarbeiten.

Als letzte Form des ML ist das verstärkte Lernen, auch Reinforcement Learning genannt, zu nennen. Hierbei werden Aktionen unter der Aufsicht eines menschlichen Agenten ausgeführt und entweder belohnt oder bestraft. Anhand dieses Belohnungssystems muss das System selbst die beste Strategie herausfinden, um die meisten Belohnungen zu erhalten.[43]

Als Kategorie zwei ist zu unterscheiden, ob das System inkrementell aus Datenströmen dazulernen kann oder nicht. Es wird zwischen Online-Learning und Batch-Learning Prozessen unterschieden. Auch ist zwischen inkrementellem Lernen aus Datenströmen und anderen Lernprozessen zu unterscheiden. Inkrementelles Lernen kann entweder online oder in Batches erfolgen.

Batch-Learning beschreibt den Lernprozess, bei dem das Programm zuerst auf einer großen Menge an Daten trainiert wird, bevor es die erworbene Expertise in der Echtdatenumgebung einsetzt. Das System lernt nicht inkrementell. Die Performance des Systems tendiert oft dazu, im Laufe der Zeit schlechter zu werden.[44]

Beim Online-Learning lernt das System inkrementell, wodurch Lernschritte schneller und kostengünstiger werden, da keine großen Datenmengen für einmalige Verarbeitungen benötigt werden. Es ist jedoch wichtig, auf die Lernrate zu achten. Wenn diese zu hoch ist, lernt das System zwar schnell Neues, vergisst aber auch schnell das Gelernte.[45]

---

[43] vgl. *Géron*, 2023, S. 42–46.
[44] vgl. *Shai, Shalev-Shwartz, Shai, Ben-David*, 2014, S. 24.
[45] vgl. *Géron*, 2023, S. 47.

2.4 Arten von Algorithmen

Im Folgenden werden, wie in Abbildung 3 dargestellt, einige Arten von Algorithmen zur Verwendung unter ML näher erläutert.

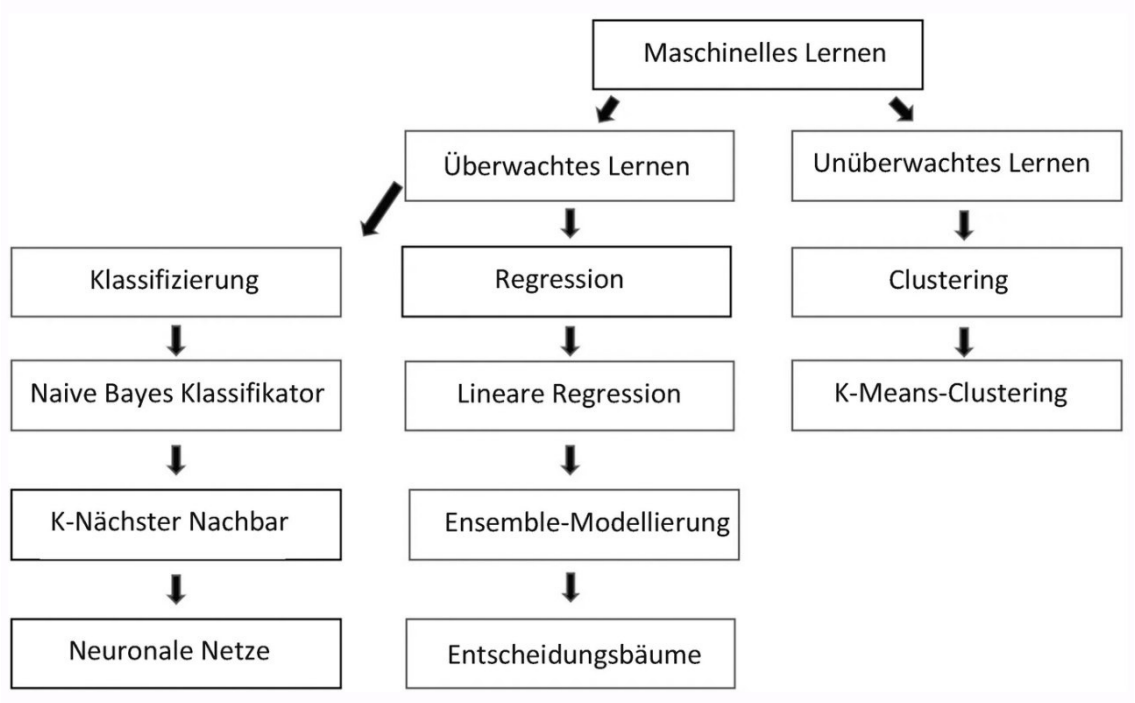


Abbildung 3: Allgemeiner Rahmen für Algorithmen des maschinellen Lernens[46]

2.4.1 Lineare Regression

Zu den einfachsten Algorithmen des ML gehört die lineare Regression. Sie geht davon aus, dass Beziehungen zwischen dem Zielvektor und den Merkmalen annähernd linear sind.[47] Vorhersagen werden allgemein getroffen, indem Eingabemerkmale zu einer gewichteten Summe addiert werden, zusammen mit der Konstante des Bias-Terms.[48]

[46] *Taulli*, 2022, S. 63.
[47] vgl. *Albon*, 2018, S. 223 f.
[48] vgl. *Géron*, 2023, S. 164.

Die Formel lautet wie folgt:

$$\hat{f}(x) = \omega_0 + \omega_1 a_1(x) + \cdots + \omega_n a_n(x)$$

Formel 1: Lineares Regressionsmodell[49]

In dieser Gleichung ist $\hat{f}(x)$ der vorhergesagte Wert, ω_j der Modellparameter mit Bias-Term ω_0 und der gewichteten Merkmale $\omega_1, \omega_2, \ldots, \omega_n$, a_i ist der i. Wert des Merkmals. n ist die Anzahl der Merkmale.[50]

Neben der linearen Regression gibt es weitere Abwandlungen, die vor allem im Bereich des Übertrainierens eines Algorithmus, des sogenannten Overfitting, zum Einsatz kommen. Hierzu zählen die Ridge-Regression, die Lasso-Regression und die Elastic-Net-Regression, auf die jedoch in dieser Arbeit nicht näher eingegangen wird.

2.4.2 Entscheidungsbäume

Entscheidungsbäume eignen sich aufgrund ihrer bildlichen Darstellung besonders gut, wenn die Interpretierbarkeit eines Modells von hoher Bedeutung ist. Das Modell erlernt durch eine Reihe von Fragen alle Klassenbezeichnungen der vorhandenen Objekte in den Trainingsdaten, auch ohne kategoriale Merkmale. Die Daten werden rekursiv auf der Grundlage der wichtigsten Attribute in kleinere Gruppen unterteilt, bis der Endknoten bestimmt werden kann.[51] Zur Analyse der Daten können die Fragen und ihre möglichen Antworten direkt in den Entscheidungsbaum eingebracht werden, wie in Abbildung 4 dargestellt.[52]

---

[49] vgl. *Mitchell*, 2010, S. 237.
[50] vgl. *Géron*, 2023, S. 165.
[51] vgl. *Mangal, Ekta, Divya, Shubham, Gussain, Radhika*, 2023, S. 3122.
[52] vgl. *Raschka/Mirjalili*, 2021, S. 115.

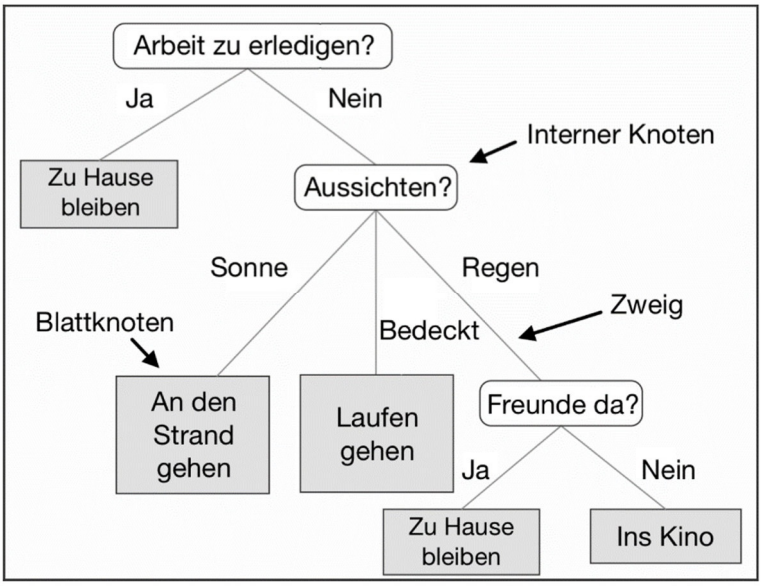


Abbildung 4: Beispiel eines Entscheidungsbaumes[53]

Diese Bäume bestehen dabei aus einer Wurzel *W*, die zu Beginn jeder Auswertung steht, vielen Entscheidungsknoten *t*, von denen weitere Entscheidungen getroffen werden, sowie den Endknoten *b*, auch als Blatt bezeichnet, von denen keine Entscheidungen mehr ausgehen.[54]

2.4.3 Ensemble-Modellierung, Random Forests

Ein häufiges Problem bei der Verwendung von Entscheidungsbäumen ist, dass die Modelle zu stark an den Trainingsdaten ausgerichtet sind. Dadurch erzielen sie zwar hohe Trefferquoten auf den Trainingsdaten, aber bei anderen Daten sind die Ergebnisse weniger erfolgreich.[55]

Die Ensemble-Modellierung verfolgt den Ansatz, mehrere Lerner wie z.B. Entscheidungsbäume in Kombination zu betreiben, um das Trainingsergebnis zu maximieren.[56] Hierzu stehen zwei Ansätze zur Verfügung: das Boosting und das Bagging. Beim Boosting werden mehrere schwache Lernmodelle hintereinandergeschaltet, um das Ergebnis des Gesamtmodells zu verbessern. Das Bagging hat den Kerngedanken, mehrere Versionen eines Modells zu erstellen, indem es wiederholt Bootstrap-Stichproben verwendet. Dabei handelt es sich um zufällige Stichproben mit Ersetzung. Die

---

[53] *Raschka/Mirjalili*, 2021, S. 115.
[54] vgl. *Frochte*, 2021, S. 129.
[55] vgl. *Albon*, 2018, S. 239.
[56] vgl. *Mangal, Ekta, Divya, Shubham, Gussain, Radhika*, 2023, S. 3123.

Ersetzung bedeutet, dass dieselben Datensätze mehrmals pro Stichprobe vorkommen können. Der bekannteste Ansatz des Bagging ist der Random Forest. Dieser unterscheidet sich von einer reinen Bagging-Methode durch eine größere Anzahl an Variationen von Bäumen.[57] Der Random Forest ist eine praxistaugliche und robuste Methode, um spezialisierte Entscheidungsbäume zu trainieren. Diese können in großer Anzahl zusammengefasst werden.[58]

2.4.4 Naive Bayes-Klassifikator

Hierbei handelt es sich um einen Klassifikator, der die Wahrscheinlichkeit eines Ergebnisses bestimmt, wenn eine Reihe von Bedingungen vorliegt. Hierbei wird das Bayes'sche Theorem verwendet und alle bedingten Wahrscheinlichkeiten werden invertiert, um messbare Größen zu erhalten.[59] Der Naive Bayes-Klassifikator wird häufig für Textklassifizierungen genutzt und war ursprünglich eine Lösung für die Spam-Erkennung von E-Mails. Dieses Modell ist besonders bei großen Datenmengen hilfreich und einfach zu erstellen. Es bietet eine Methode zur Berechnung der Posteriorwahrscheinlichkeit $P(c|x)$, wie in Abbildung 5 dargestellt.[60]

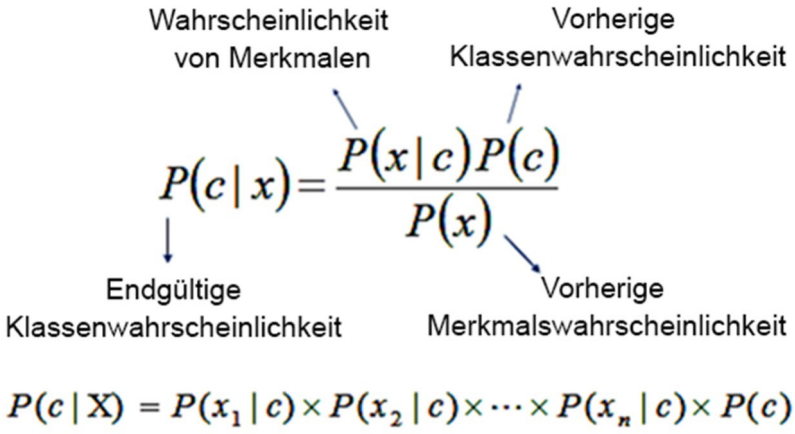


Abbildung 5: In Anlehnung an Naive Bayes-Theorem[61]

[57] vgl. *Frochte*, 2021, S. 329 f.
[58] vgl. *Chollet*, 2018, S. 39.
[59] vgl. *Bonaccorso*, 2018, S. 120.
[60] vgl. *Velayutham*, 2020, S. 164.
[61] *Velayutham*, 2020, S. 164.

2.4.5 K-Nächste Nachbarn

Diese Methode zählt zu den einfachsten und dennoch am häufigsten benutzten Klassifikatoren im Supervised Learning. Technisch gesehen wird hier kein Modell trainiert, sondern es wird durch eine Beobachtung die Klasse vorausgesagt, welche den größten Anteil der *K* nächstgelegenen Beobachtungen angehört.[62] Es handelt sich um einen nicht parametrischen Lernalgorithmus, der nicht auf eine feste Anzahl von Parametern beschränkt ist. Selbst bei großen Trainingsmengen können durch höhere Rechenkosten hohe Genauigkeiten erzielt werden.[63] In Abbildung 6 wird gezeigt, dass bei dieser Methode versucht wird, die nächsten Nachbarn unter Verwendung einer Abstandsmessung zu finden.[64]

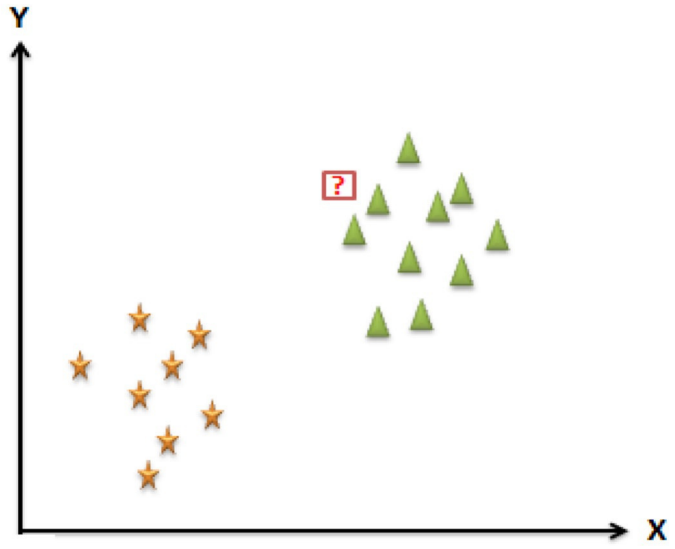


Abbildung 6: K-Nächste Nachbarn[65]

2.4.6 Neuronale Netze

Bei der Verwendung von NN wird versucht, das menschliche Gehirn zu simulieren. NN bestehen aus Knoten, welche die Neuronen darstellen, und Kanten, welche die Synapsen abbilden. Die Knoten werden in drei Kategorien aufgeteilt: Eingangsknoten, versteckte Knoten und Ausgangsknoten. Die Kanten zwischen den Knoten werden durch

---

[62] vgl. *Albon*, 2018, S. 251.
[63] vgl. *Goodfellow/Courville/Bengio*, 2016, S. 140.
[64] vgl. *Gollapudi*, 2016, S. 188.
[65] *Gollapudi*, 2016, S. 188.

das Gewicht *w* gewichtet, welches das Wissen des NN repräsentiert. Aufgrund der Trainingsdaten werden bei der Anwendung dieser Methode Veränderungen vorgenommen.[66] Wie in Abbildung 7 dargestellt, haben neuronale Netze die Fähigkeit, einfache Merkmale auf mehreren Ebenen zu kombinieren und zu gewichten, um komplexe Merkmale zu erzeugen.[67]

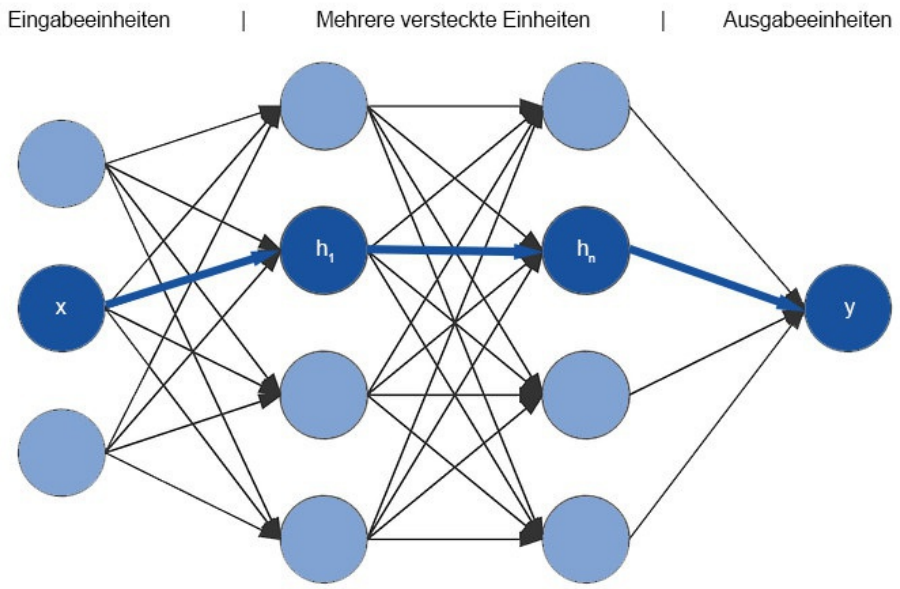


Abbildung 7: In Anlehnung an neuronale Netze[68]

NN lernen, indem sie den Prozess mehrmals wiederholen. Jeder Durchlauf aller Beobachtungen, die durch das NN geschickt wurden, wird als Epoche bezeichnet. Das Training besteht typischerweise aus mehreren Epochen, bei denen die Parameterwerte iterativ angepasst werden.[69]

2.4.7 K-Means-Clustering

Diese Methode ist ein einfacher Algorithmus, der Datensätze in sogenannte Cluster unterteilen kann. Hierbei wird in mehreren Iterationen versucht, den Mittelwert jedes Bereichs zu finden und anschließend jede einzelne Instanz dem nächstgelegenen Bereich zuzuordnen.[70] Es ist jedoch zu beachten, dass die Anzahl der Cluster vorab festgelegt

---

[66] vgl. *Buxmann*, 2021, S. 14–16.
[67] vgl. *Burkov*, 2020, S. 31.
[68] *Buxmann*, 2021, S. 15.
[69] vgl. *Albon*, 2018, S. 298.
[70] vgl. *Géron*, 2023, S. 297.

werden muss. Bei der Verwendung von höherdimensionalen Daten kann es schwierig bis unmöglich sein, die Anzahl der Cluster durch Visualisierung zu bestimmen.[71] Es ist auch zu beachten, dass selbst wenn die richtige Anzahl von Clustern bestimmt werden kann, nicht in jeder Form das Zentrum ermittelt werden kann. Dies gilt auch, wenn die Cluster sehr unterschiedliche Größen haben.[72] Das folgende Streudiagramm (Abbildung 8) zeigt einen K-Means-Clustering-Algorithmus mit drei Clustern, ihren ermittelten Zentren und den zugeordneten Instanzen.

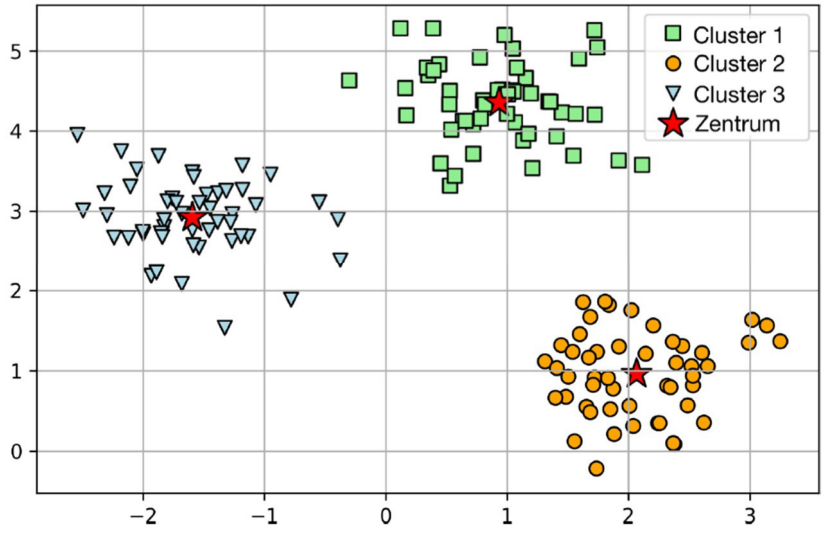


Abbildung 8: Beispiel Streudiagramm K-Means-Clustering[73]

2.5 Einordnung Fraud Detection

Die Fraud Detection, mit ihrem gelabelten Datensatz, die im Zuge dieser Arbeit verwendet wird, gliedert sich in die Anomalieerkennung ein und ist Teil der Klassifizierung.[74] Sie gehört somit zum Supervised Learning, dem überwachten Lernen.[75]

2.6 Modellbewertung

Um die Klassifikationsgüte eines Klassifizierungsmodells zu beurteilen, gibt es verschiedene Kriterien. Dazu gehören die Trefferquote (auch Recall genannt), die Genauigkeit (auch Precision genannt) sowie das aus beiden kombinierte F1-Maß (auch F1-

---

[71] vgl. *Raschka/Mirjalili*, 2021, S. 382.
[72] vgl. *Müller/Guido*, 2017, S. 173.
[73] *Raschka/Mirjalili*, 2021, S. 382.
[74] vgl. *Goodfellow/Courville/Bengio*, 2016, S. 99.
[75] vgl. *Velayutham*, 2020, S. 240.

Score genannt). Außerdem kann eine Verwechslungsmatrix (auch Confusion Matrix genannt) erstellt werden, um die Leistung eines Lernalgorithmus darzustellen.[76]

2.6.1 Confusion Matrix

Eine Confusion Matrix, wie in Abbildung 9 dargestellt, zeigt in Form einer Tabelle an, wie erfolgreich ein Klassifizierungsmodell ist, indem es die Vorhersagen in vier Klassen aufteilt.

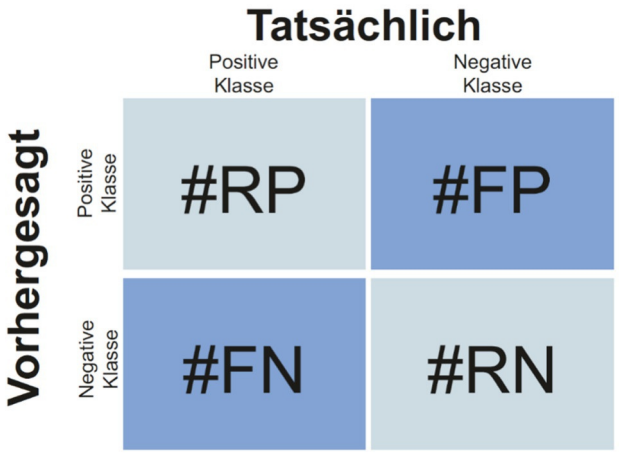


Abbildung 9: Confusion Matrix[77]

Richtig(True)-Positiv (RP/TP): Das Model sagt eine betrügerische Transaktion vorher, die tatsächlich betrügerisch ist.

Richtig(True)-Negativ (RN/TN): Das Model sagt eine legitime Transaktion vorher, die tatsächlich legitim ist.

Falsch(False)-Positiv (FP): Das Model sagt eine betrügerische Transaktion vorher, die aber in Wahrheit legitim ist.

Falsch(False)-Negativ (FN): Das Model sagt eine legitime Transaktion vorher, die aber in Wahrheit betrügerisch ist.[78]

2.6.2 Recall

Der Recall-Maßstab ist ein Leistungsmaßstab, der verwendet wird, um positive Stichproben zu identifizieren. Es ist wichtig, Falsch-Negativ-Ergebnisse zu vermeiden. Im Fall

---

[76] vgl. *Raschka/Mirjalili*, 2021, S. 234.
[77] *Buxmann*, 2021, S. 18.
[78] vgl. *Buxmann*, 2021, S. 17.

der Fraud Detection geht es darum, keine betrügerischen Transaktionen fälschlicherweise als legitim einzustufen.[79] Der Recall berechnet sich wie folgt:

$$Recall = \frac{Richtig - Positiv\ (RP/TP)}{Richtig - Positiv\ (RP/TP)\ +\ Falsch - Negativ\ (FN)}$$

Formel 2: Recall-Maßstab

2.6.3 Precision

Der Precision-Maßstab wird verwendet, um möglichst wenige falsch-positiven Vorhersagen zu erhalten. Im Fall der Fraud Detection geht es darum, keine legitimen Transaktionen als betrügerisch vorherzusagen. Dies ist besonders wichtig, da falsch-positiv identifizierte Transaktionen hohe Kosten verursachen können.[80] Die Precision wird wie folgt berechnet:

$$Precision = \frac{Richtig - Positiv\ (RP/TP)}{Richtig - Positiv\ (RP/TP)\ +\ Falsch(False) - Positiv\ (FP)}$$

Formel 3: Precision-Maßstab

2.6.4 F1-Score

Sowohl der Recall als auch der Precision gelten als wichtige Maßstäbe für die Klassifikationsgüte. Allerdings sind sie einzeln betrachtet nur ein Teil des Gesamtbildes. Erst in Kombination miteinander, dem sogenannten F1-Score, wird eine Gesamtgüte darstellbar.[81] Der F1-Score berechnet sich folgendermaßen:

$$F1 = 2 \times \frac{Precision\ \times\ Recall}{Precision\ +\ Recall}$$

Formel 4: F1-Score

---

[79] vgl. *Müller/Guido*, 2017, S. 283.
[80] vgl. *Burkov*, 2020, S. 142.
[81] vgl. *Müller/Guido*, 2017, S. 283 f.

3 Praktische Umsetzung

Diese Thesis beschäftigt sich mit der praktischen Umsetzung des Aufsetzens einer virtuellen Maschine (VM) unter Verwendung von VirtualBox sowie der Installation von Oracle Linux 9.3. Die erstellte Umgebung dient als Vorlage für das Projekt. Es wird darauf geachtet, dass die VM keine GPU zur Verfügung hat und TensorFlow als reine CPU-Anwendung installiert wird. Dies geschieht in Voraussicht auf eine praktische Anwendbarkeit im realen Arbeitskontext des Autors. Für die Untersuchung werden folgende Hardwarekomponenten verwendet: Ein AMD Ryzen 7 3700X 8-Core Prozessor mit 3,8 GHz, 64 GB DDR4 3600C16 Arbeitsspeicher, eine Lexar 2TB M.2 PCIe Gen4 NVMe NM790 Festplatte mit einer Lesegeschwindigkeit von bis zu 7400 MB/s und einer Schreibgeschwindigkeit von bis zu 6500 MB/s sowie eine NVIDIA GeForce RTX 3090 Ti mit 24GB GDDR6X Speicher. Als Betriebssystem wurde Windows 11 Pro 23H2 verwendet.

3.1 Vorbereitung der technischen Umgebung

3.1.1 Einrichtung einer VirtualBox Maschine

Für die Installation und Einrichtung der VM wird das Benutzerhandbuch von VirtualBox verwendet.[82] Zunächst muss die neueste Version von VirtualBox von der Website ‚https://www.virtualbox.org/wiki/Downloads' heruntergeladen werden. Zum Zeitpunkt der Installation ist dies die Version ‚VirtualBox-7.0.14-161095-Win.exe'. Bei der Installation sind keine weiteren Einstellungen erforderlich. Die automatische Installation kann durch mehrmaliges Klicken auf ‚Weiter' gestartet werden. Nach der Installation öffnet sich der Oracle VM VirtualBox Manager, wie in Abbildung 10 dargestellt.

[82] vgl. *Burkov*, 2020, S. 142.

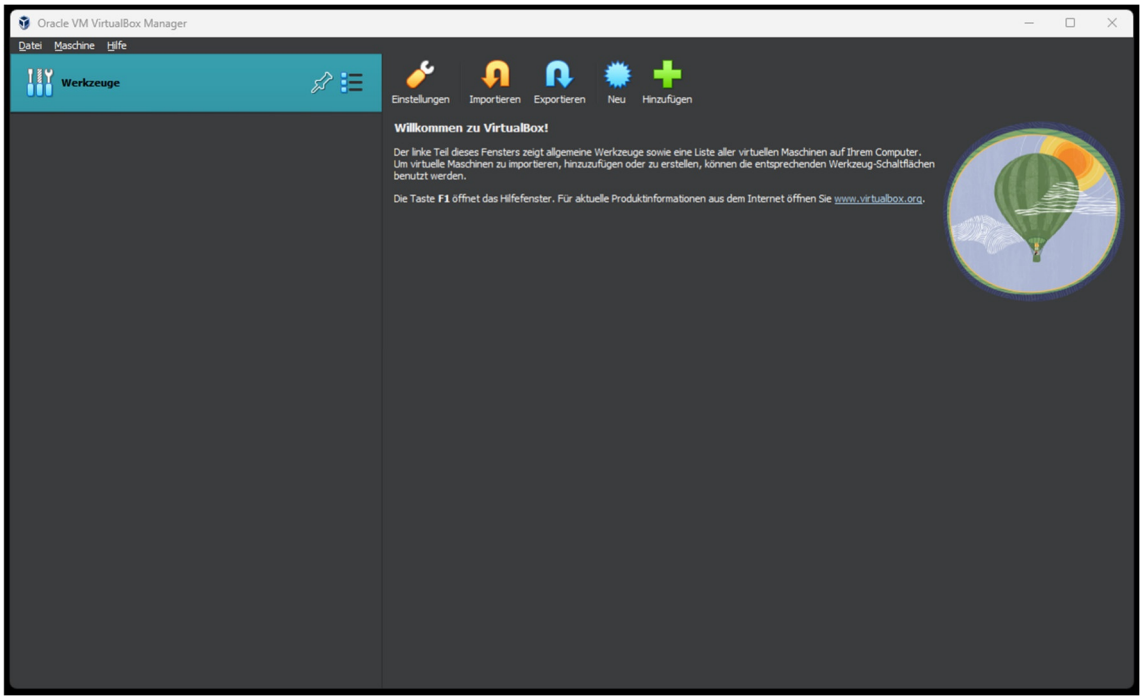


Abbildung 10: Oracle VM VirtualBox Manager

Als zweiten Schritt muss noch das VirtualBox Extension Pack von derselben Website heruntergeladen werden. Das Pack mit dem Namen ‚Oracle_VM_VirtualBox_Extension_Pack-7.0.14.vbox-extpack' ist die aktuelle Version. Die Installation selbst funktioniert mit nur zwei Klicks: Zuerst muss durch das Drücken von ‚Installieren' die Installation bestätigt und im Anschluss der Lizenz zugestimmt werden. Die zusätzlichen Werkzeuge des Extension-Packs werden hauptsächlich für das Drag-and-Drop sowie ‚STRG+C' oder ‚STRG+V' benötigt. Nach erfolgreicher Installation des Managers und Hinzufügen des Extension-Packs kann mit der Installation einer VM begonnen werden.

3.1.2 Installation einer Oracle Linux VM Maschine

Für den Installationsprozess und die Daten von Linux wird die Dokumentation von Oracle verwendet.[83] Bei der Installation einer neuen VM wird im Oracle VM VirtualBox Manager der Menüpunkt ‚Maschine' und dann ‚Neu' verwendet. Alternativ kann auch ‚STRG+N' gedrückt werden. Anschließend öffnet sich das Konfigurationsfenster für die neue Maschine, wie in Abbildung 11 dargestellt.

[83] vgl. *Oracle Help Center*, 2024.

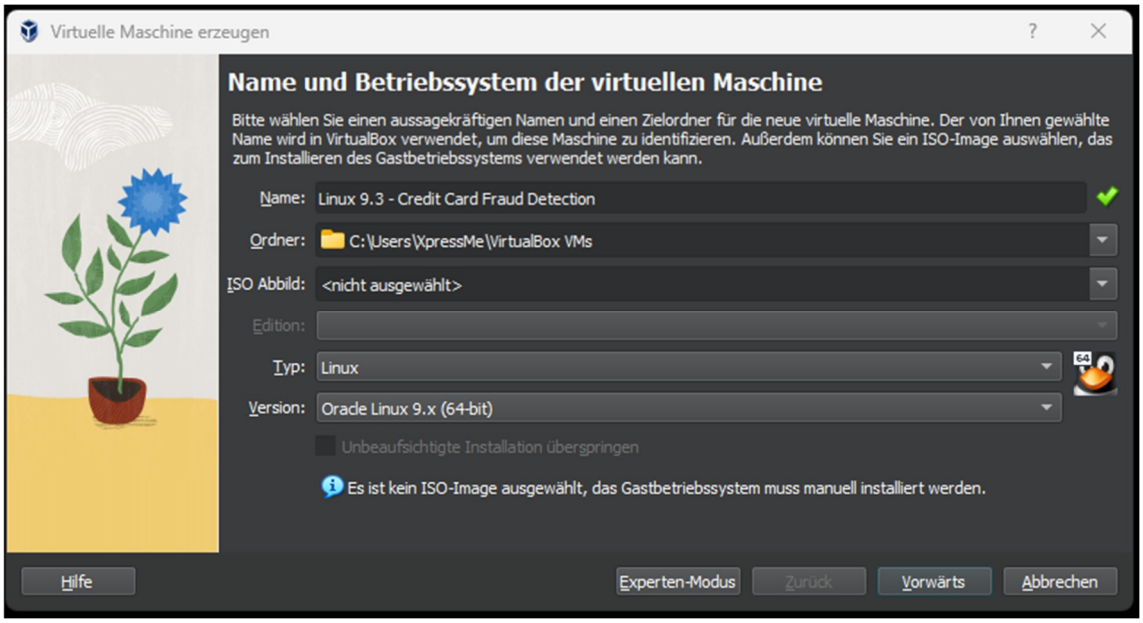


Abbildung 11: Einrichtungsfenster einer neuen virtuellen Maschine

Zuerst muss der Maschine ein Name gegeben werden. In diesem Projekt wird die Maschine ‚Linux 9.3-Credit Card Fraud Detection' genannt. Der Speicherort wird nicht geändert. Anschließend wird die neueste Version, Oracle Linux 9.3, als ISO-Image unter der Datei ‚OracleLinux-R9-U3-x86_64-dvd.iso' von der Webseite ‚https://yum.oracle.com/oracle-linux-isos.html' heruntergeladen. Diese Datei muss anschließend als CD-ISO abgelegt werden, um die Linux-Installation beim ersten Start zu starten. Die restlichen Punkte des Installationsfensters müssen auf Linux und Oracle Linux 9.3 (64 Bit) eingestellt werden. Nach dem Fortsetzen müssen weitere Einstellungen vorgenommen werden. Nun müssen der Maschine Ressourcen zugewiesen werden. Hierbei ist zu beachten, dass der VM nur die physikalisch vorhandenen Ressourcen zugewiesen werden können und die physikalische Maschine ebenfalls Ressourcen zum Betrieb benötigt. In diesem Projekt wurde der VM die Hälfte des Hauptspeichers, also 32.768 MB, sowie die Hälfte der Prozessoren, also 8 Stück, zugewiesen. Nochmaliges Bestätigen führt den Anwender zur Zuweisung einer virtuellen Festplatte. Für dieses Projekt wurde eine virtuelle Festplatte mit einer Größe von 30 GB definiert. Abschließend zeigt der Manager, wie in Abbildung 12 zu sehen, eine Zusammenfassung aller vorgenommenen Einstellungen, die mit ‚Fertigstellen' abgeschlossen werden können.

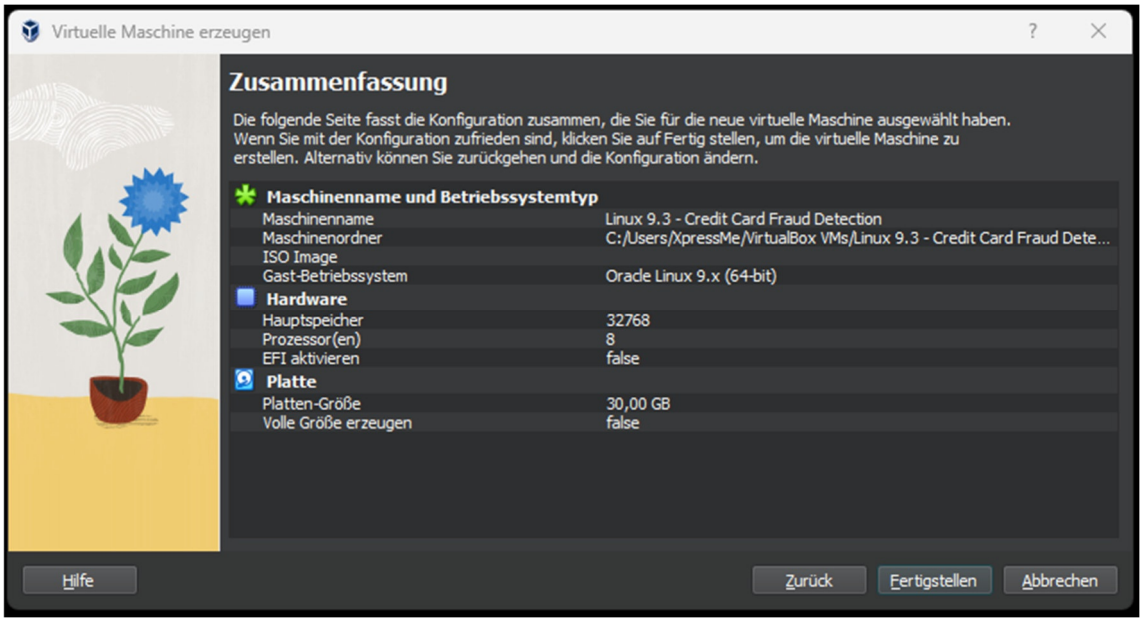


Abbildung 12: Zusammenfassung der neuen virtuellen Maschine

Nachdem die Konfiguration abgeschlossen wurde, erscheint die neue VM in der linken Auswahlliste. Diese muss nun durch einfaches Anklicken ausgewählt und über den Button ‚Ändern' noch angepasst werden. Wie in Abbildung 13 zu sehen ist, muss unter ‚Massenspeicher' unter ‚Controller: IDE' das optische Laufwerk mit einem ISO-Image geladen werden, indem auf die kleine blaue CD geklickt wird.

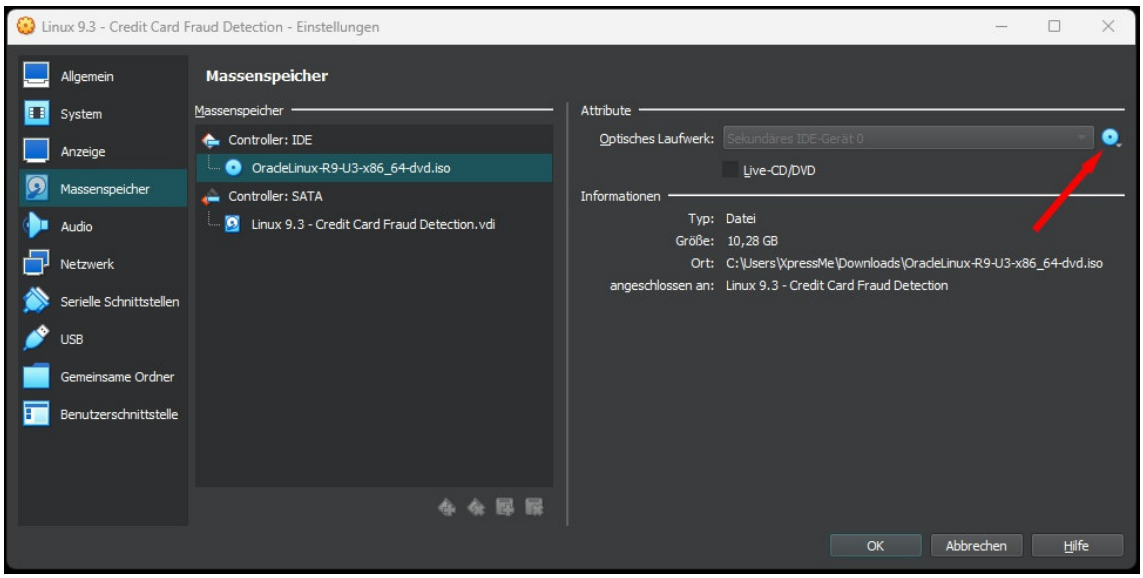


Abbildung 13: Einstellungen der virtuellen Maschine anpassen

Hier ist die zuvor heruntergeladene Datei abzulegen und das Fenster mit ‚OK' zu schließen.

Mit einem Klick auf ‚Start' startet die neue VM vollautomatisch in einem neuen Fenster mit dem Bootvorgang des Betriebssystems. Sollten während des Bootvorgangs Fehler auftreten, was zum Beispiel bei der CPU Virtualisierung öfters vorkommen kann, müssen diese individuell nach der jeweils verwendeten physikalischen Maschine untersucht werden. In unserem Projekt musste die AMD-V CPU Virtualisierung im Bios der physikalischen Maschine aktiviert werden, damit die VM fehlerfrei starten konnte.

Beim ersten Start der Linux VM erscheint die Linux Installationsanwendung, die weitere Einstellungen erfordert. Zunächst muss die gewünschte Installations- als auch die Eingabesprache ausgewählt werden. In diesem Projekt werden beide Einstellungen auf Deutsch gesetzt und mit der Installation fortgefahren. In der Installationsübersicht, wie in Abbildung 14 zu sehen, muss unter Software die Softwareauswahl angepasst werden. Dort muss ‚Arbeitsplatz' ausgewählt werden.

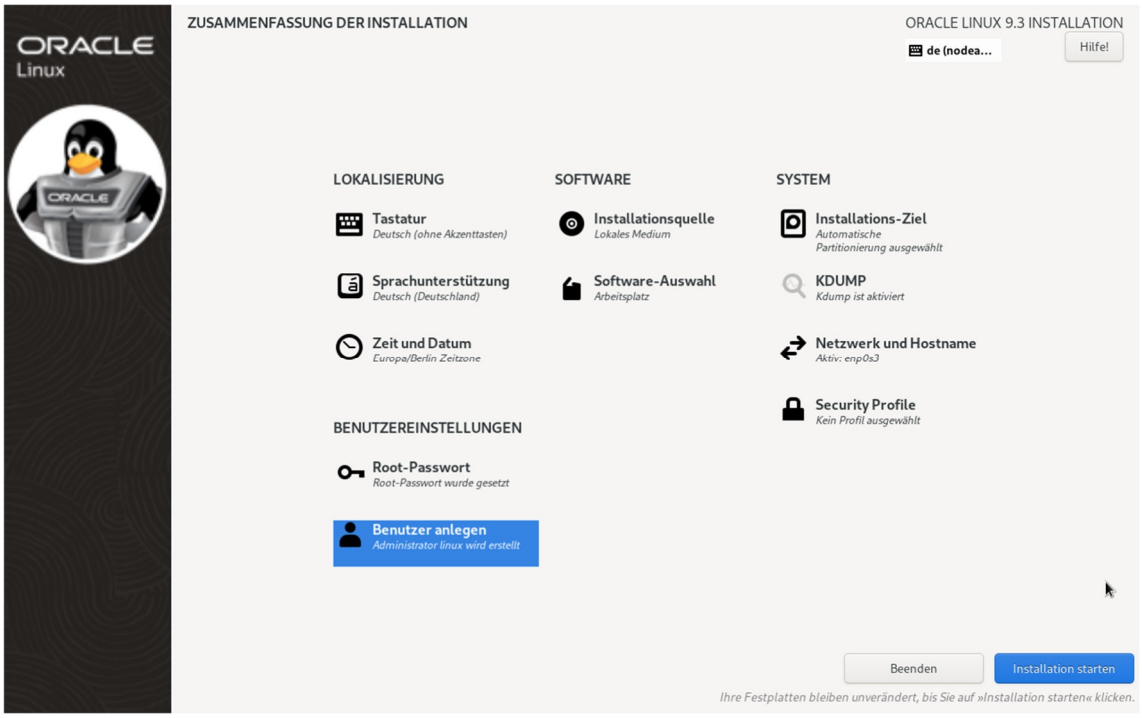


Abbildung 14: Zusammenfassung der Linux Installation

Anschließend muss dem Root-User, auch Administrator genannt, ein Passwort zugewiesen werden, welches sicher aufbewahrt werden sollte. Danach kann unter ‚Benutzer anlegen' der normale Linux-Benutzer angelegt werden. In diesem Projekt wurde der neu angelegte Benutzer direkt bei der Installation mit ‚Diesen Benutzer als Administrator festlegen' als Administrator gekennzeichnet, um bei den folgenden Installationen keine Rechteprobleme herbeizuführen. Abschließend kann die Installation über ‚Installation starten' gestartet werden. Die Installation selbst kann je nach physikalischer Maschine

einige Zeit in Anspruch nehmen. Wurde die Installation erfolgreich abgeschlossen, muss die Maschine mittels ‚Reboot' einem Neustart unterzogen werden. Anschließend erscheint nach dem Booten von Linux die Anmeldung des zuvor konfigurierten Benutzers. Damit sind die Anforderungen an die zu verwendende Umgebung für dieses Projekt abgeschlossen.

3.1.3 Installation und Konfiguration TensorFlow und Python

Für den Umgang mit Linux und der Kommandozeile sowie ihre Befehle wurden folgende drei Lektüren verwendet: ‚Linux pocket guide'[84], ‚The Linux command line'[85] und ‚How Linux works'[86]. Für die Installation und Einrichtung von TensorFlow wurde auf ‚https://www.tensorflow.org/install/pip'[87] zurückgegriffen. Für die Python Installation wurde die Python Dokumentation unter ‚https://docs.python.org/3'[88] verwendet.

Nach der erfolgreichen Anmeldung in der Linux-Umgebung wird zunächst über den ‚Activities'-Button ein Terminal geöffnet. In diesem Terminal müssen zunächst temporär erweiterte Rechte für den aktuellen Benutzer vergeben werden. Dies geschieht über den Befehl: ‚sudo su'. Nachdem dieser Befehl ausgeführt wurde, wechselt der angezeigte Benutzer in der Kommandozeile zu ‚[root@localhost linux]'. Bei vielen weiteren Befehlen wird der zusätzliche Parameter ‚-y' verwendet, um die Rückfragen automatisch mit ‚y = yes' zu bestätigen. Danach kann mit der Aktualisierung des Systems begonnen werden. Dazu wird der Befehl: ‚dnf update -y' verwendet. Das System überprüft nun alle installierten Pakete auf die neueste Version und aktualisiert diese gegebenenfalls. Danach kann mit der Installation von Python3 begonnen werden. Dies geschieht mit dem Befehl: ‚dnf install python3 -y'. Ist der Befehl erfolgreich abgeschlossen, kann die Python3-Version mit folgendem Befehl überprüft werden: ‚python3 --version'. Für die Installation von TensorFlow wird noch die Python3-Erweiterung pip benötigt. Diese kann mit: ‚dnf install python3-pip -y' installiert, mit: ‚pip3 --version' überprüft und mit: ‚pip3 install --upgrade pip' aktualisiert werden. Nachdem alle installierten Pakete aktualisiert, python3 inkl. pip installiert und geprüft wurden, kann TensorFlow installiert werden. Zuerst wird vom su-User zurück zum normalen User gewechselt mittels: ‚exit', da für die

---

[84] *Barrett*, 2016.
[85] *Shotts*, 2019.
[86] *Ward*, 2021.
[87] *TensorFlow*, 2024.
[88] *Python*, 2024.

Installation mittels pip keine su-Rechte mehr benötigt werden. Die Installation von TensorFlow erfolgt mit folgendem Befehl: ‚pip install tensorflow'. Nach erfolgreicher Installation von TensorFlow kann die Installation überprüft werden, indem ein Beispiel-Tensor erzeugt wird. Dies geschieht mit folgendem Befehl: ‚python3 -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))"'. Wenn hier ein Tensor zurückgegeben wird, wurde TensorFlow erfolgreich installiert.

Warnungen über fehlende cuda-Treiber können ignoriert werden, da in dieser praktischen Umsetzung auf die GPU-Unterstützung verzichtet und rein auf der CPU gerechnet werden soll. In der folgenden Tabelle 1 sind alle Befehle und ihre Beschreibungen noch einmal zusammengefasst.

Tabelle 1: Auflistung aller genutzten Linux Befehle

Befehl	**Beschreibung**
sudo su	Gibt dem aktuellen User vorübergehend Superuser Zugriff.
Befehlsparameter -y	Bestätigt automatische Rückfragen mit y für yes.
dnf update -y	Alle installierten Pakete auf die neuesten Versionen aktualisieren.
dnf install python3 -y	Installation des neusten Python3 Paketes.
python3 --version	Zeigt die aktuelle Version von Python3 an.
dnf install python3-pip -y	Installiert die Erweiterung pip, welche zur Installation von Python Modulen benötigt wird.
pip3 --version	Zeigt die aktuelle Version von pip3 an.
pip3 install --upgrade pip	Updatet zur neusten pip3 Version.
pip install tensorflow	Installiert TensorFlow mittels pip.
python3 -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))"	Überprüft die CPU Installation von TensorFlow.

3.1.4 Installation Jupyter Notebook im Offline-Modus

In Hinblick auf die praktische Anwendbarkeit im realen Arbeitskontext des Verfassers dieser Arbeit, wird von einer Maschine ausgegangen, die nach der Installation aller notwendigen Komponenten, keinen Zugriff mehr auf das Internet hat, sobald die Daten dorthin geladen wurden. Im Zuge dessen muss auch die verwendete Oberfläche zur Entwicklung mit Python und zur Ausgabe der Plots als Offline-Version installiert werden. Dazu wird zunächst eine Kommandozeile geöffnet und mittels ‚pip3 install --upgrade pip' die aktuelle Version von PIP ermittelt und ggf. aktualisiert. Anschließend kann mittels ‚pip3 install jupyter' die neueste Version von Jupyter Notebook installiert werden. Nach erfolgreicher Installation kann der Jupyter Notebook Dienst mit dem Befehl ‚jupyter notebook' gestartet werden. Zu beachten ist, dass die Kommandozeile danach geöffnet bleiben muss, um den Dienst aktiv zu halten. Wurde der Dienst erfolgreich gestartet, öffnet sich der Standardbrowser mit der Startseite von Jupyter Notebook, wie in Abbildung 15 dargestellt.

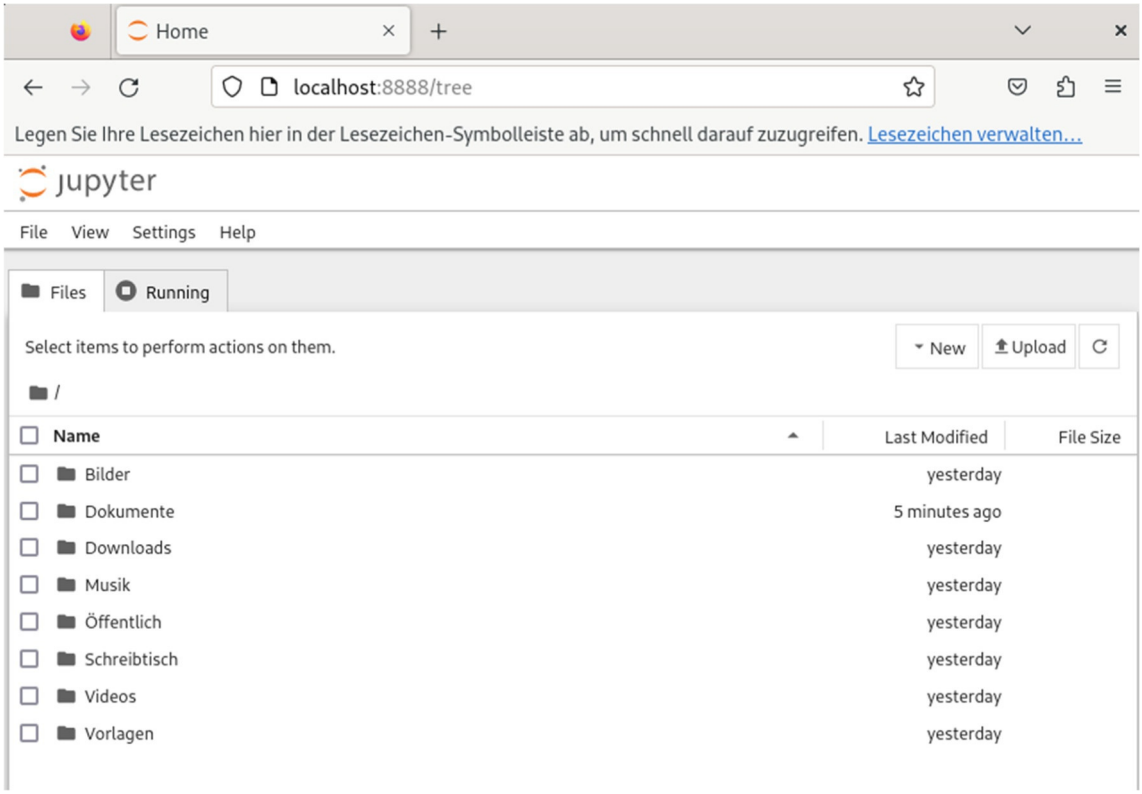


Abbildung 15: Oberfläche von Jupyter Notebook

Dieser kann bereits im Offline-Modus verwendet werden, was auch in der Adresszeile an der verwendeten Adresse: ‚localhost' zu sehen ist. Soll der Dienst gestoppt werden, kann dies mit ‚CTRL+C' in der noch geöffneten Kommandozeile geschehen. Nun kann

Jupyter Notebook in vollem Umfang als Offline-Version genutzt werden. Alle verwendeten Befehle sind noch einmal in Tabelle 2 zusammengefasst.

Tabelle 2: Auflistung aller Befehle zur Installation von Jupyter Notebook

Befehl	**Beschreibung**
pip3 install --upgrade pip	Updatet zur neusten pip3 Version.
pip3 install jupyter	Installiert Jupyter Notebook.
jupyter notebook	Startet den Jupyter Notebook Dienst und öffnen Jupyter Notebook im Browser.
STRG+C	In der Kommandozeile nutzen, um den Dienst zu stoppen.

Es ist jedoch zu beachten, dass alle Module, die in Jupyter Notebook zur Verfügung stehen sollen, auch einmal über die Kommandozeile installiert werden müssen. Ein Beispiel wäre hier, dass für die Verwendung von ‚import pandas as pd' in Jupyter Notebook zuerst ‚pip3 install pandas' auf der Kommandozeile ausgeführt werden muss. In diesem Projekt wurden folgende Module über ‚pip3 install ...' installiert: pandas, seaborn, matplotlib, imblearn, scikit-learn.

3.2 Datenbeschaffung

Für das Training von KI-Modellen gibt es eine Vielzahl von bereits aufbereiteten Daten im Internet. In diesem Bereich ist vor allem Kaggle eine beliebte Plattform, auf der öffentlich zugängliche Datensätze und Codeprojekte von Teilnehmern zu verschiedensten Themen aus dem Bereich des ML zu finden sind.[89] Darüber hinaus bietet Kaggle auch regelmäßig Wettbewerbe an, bei denen die Teilnehmer ihre Modelle auf eine gestellte Aufgabe trainieren müssen, um Preisgelder zu gewinnen. Einer der bekanntesten Wettbewerbe war der Netflix-Award, bei dem die Teilnehmer mit ihrem Modell ein Preisgeld von einer Million Dollar gewinnen konnten. Auch für dieses Projekt wurde ein Beispieldatensatz von Kaggle verwendet. Es handelt sich um den Datensatz ‚Credit Card Fraud Detection', der unter ‚https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud' zu

[89] vgl. *Taulli*, 2022, S. 36–38.

finden ist. Die Datei wurde in der VM im persönlichen Ordner unter Dokumente entpackt abgelegt.

3.3 Datenanalyse

Bevor mit der Modellauswahl respektive der Modellentwicklung begonnen werden kann, sollten die verwendeten Daten nach Möglichkeit genauer betrachtet und analysiert werden. Die Analyse der hier verwendeten Daten ergab bei einer Auswertung als ‚Excel-Import.csv' sowie in der Datenbeschreibung folgende Ergebnisse:

- Der Datensatz enthält Kreditkartentransaktionen, die im September 2013, binnen zwei Tagen, getätigt wurden.
- Es sind 284.807 Transaktionen, von denen nur 492 Betrugsfälle sind.
- Die positive Klasse (Betrugsfälle) beträgt dadurch nur 0,17 % des Datensatzes aus.
- Die 28 der 30 Inhaltsspalten beinhalten nur numerische Werte.
- Nur die Werte Zeit und Betrag sind keine anonymen Werte.
- Das Merkmal Zeit stellt nur eine fortlaufende Zeit ab Beginn der Datei dar.
- Das Merkmal Betrag stellt den Betrag der Transaktion dar.
- Das Merkmal Klasse ist die Betrugsvorhersage, wobei 0 eine legitime Transaktion und 1 eine betrügerische Transaktion darstellt.

3.4 Modellauswahl

Auf der Grundlage der verwendeten Daten, bei denen es sich um einen gelabelten Datensatz handelt, stehen mehrere Modelle zur Auswahl, um die Daten zu trainieren. Daher werden die folgenden Modelle näher betrachtet. NN, logistische Regression, Random Forest Klassifikator. Diese Modelle sind Teil des überwachten Lernens und eignen sich hervorragend, wenn das gewünschte Ergebnis der Anomalieerkennung bereits bekannt ist. Darüber hinaus ist zu überlegen, ob SMOTE (Synthetic Minority Over-sampling Technique), eine Methode zur Beseitigung von Ungleichgewichten, eingesetzt werden soll, da die Datensätze einen starken Mangel an Betrugsdatensätzen aufweisen und die Anwendung von SMOTE die Modelle und ihre Ergebnisse verbessern könnte.

3.5 Datenaufbereitung

Der in diesem Projekt verwendete Datensatz ist ein bereits aufbereiteter Datensatz. Hinzu kommt, dass 28 der 30 Inhaltsspalten keine Beschreibung enthalten, sondern anonym sind, so dass keine Rangfolge oder Gewichtung möglich ist. In diesem Projekt wird daher nur auf fehlende Werte, sogenannte Nullwerte, und auf Dubletten geprüft.

Wäre dies jedoch nicht der Fall, da es in der Realität in der Regel keine aufbereiteten Datensätze gibt, müssten folgende Datenaufbereitungsschritte berücksichtigt werden: Ersetzen fehlender Werte, Entfernen oder Korrigieren von Ausreißern, Korrigieren von Datenfehlern, Einhaltung einer Standardisierung der Datenformate, Deduplizierung, d.h. Löschen doppelter Datensätze, Validierung der Datenintegrität anhand bekannter Regeln, ggf. Anreicherung der Daten zur Verbesserung der Datenqualität und des Kontextes, ggf. Klassifikation. Darüber hinaus ist zu beachten, dass jede Manipulation der Daten nachgewiesen werden muss, um den Zusammenhang zwischen den Originaldaten und den letztlich verwendeten Trainingsdaten nicht zu verlieren. Für all diese Schritte gibt es auch Automatisierungswerkzeuge, auf die hier nicht näher eingegangen wird.[90]

3.6 Entwicklung des Fraud Detection-Modells

Für die praktische Umsetzung des Projektes werden neben den Dokumentationen von TensorFlow[91] und Python[92] auch die Dokumentationen von Jupyter[93] und Keras[94] verwendet. Hinzu kommen alle Dokumentationen der in Python verwendeten Module wie z.B. ‚panda'. Um mit der Entwicklung eines Modells zu beginnen, muss zunächst sichergestellt werden, dass eine Kommandozeile geöffnet ist und der benötigte Dienst mittels ‚jupyter notebook' gestartet wurde (siehe Abbildung 15 für einen erfolgreich gestarteten Dienst). In Jupyter Notebook wird zunächst per Doppelklick in den Unterordner ‚Documents' gewechselt. Hier kann dann über ‚File', ‚New', ‚Notebook' ein neues Notebook angelegt werden. Unter ‚File', ‚Rename... ' kann diesem Notebook ein passender Name gegeben werden. Im Rahmen dieser Arbeit wurde das Notebook ‚Credit Card Fraud Detection' genannt. Für das Datenhandling der .csv-Datei, das sogenannte I/O-Handling, wird zuerst, wie in Abbildung 16 zu sehen, das Panda-Modul importiert und

---

[90] vgl. *Taulli*, 2022, S. 36–38.
[91] *TensorFlow*, 2024.
[92] *Python*, 2024.
[93] *Jupyter*, 2024.
[94] *Keras*, 2024.

anschließend die bereits heruntergeladene Datei unter dem Alias ‚df' eingebunden. Anschließend
können durch Aufruf der Funktion ‚df.head()' die ersten fünf Zeilen ausgegeben werden, um das Einbinden der .csv-Datei zu testen.

```
import pandas as pd # Datenverarbeitung, CSV-Datei I/O (z.B. pd.read_csv)
```

```
df = pd.read_csv("creditcard.csv",header = 0)
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267

Abbildung 16: Datenhandling der .csv Datei sowie Test der Einbindung

Danach kann mit der Analyse der Datei und der Ausgabe der wichtigsten Informationen fortgefahren werden. Dazu wird zunächst der Befehl ‚df.describe()' verwendet, um eine Auflistung der wichtigsten Informationen über die Daten zu erhalten. Wie in Abbildung 17 zu sehen ist, kann bei der Analyse der Spalte ‚Time' festgestellt werden, dass es sich, wie unter ‚count' zu sehen ist, um 284.807 Datensätze mit einer ‚max' Laufzeit von 172.792 Sekunden handelt, was gerundet 50 Stunden oder knapp 2 Tagen entspricht.

```
df.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+0
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+0
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+(
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-(
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-(
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-(
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+(

Abbildung 17: Auslesen der Dateninformationen

Mit dem Befehl ‚df['Class'].value_counts()' kann angezeigt werden, dass es sich bei der Summe der Datensätze um 284.315 legitime und nur 492 betrügerische Datensätze handelt. Nach der Datenanalyse folgt die Datenaufbereitung. Wie bereits unter dem Punkt Datenaufbereitung erwähnt, muss zunächst auf fehlende Werte, sogenannte Nullwerte, geprüft werden. Für die Prüfung auf fehlende Werte wird die Funktion ‚df.isnull().sum()' verwendet, die die Anzahl der leeren Zellen pro Spalte zurückgibt. In dem hier verwendeten Datensatz ergibt sich unter ‚Amount' eine Summe von 0, was

darauf hinweist, dass es in keiner Spalte der Daten fehlende Werte gibt. Anschließend kann mit der Funktion ‚df.duplicated()' nach Duplikaten gesucht werden. Da hier jedoch eine Visualisierung schwierig ist, da jede Zeile einzeln ausgegeben wird, muss die Funktion etwas modifiziert werden. Wie in Abbildung 18 zu sehen ist, gibt die umgebaute Funktion nur noch doppelte Zeilen aus und falls keine Duplikate vorhanden sind, wird eine entsprechende Meldung ausgegeben.

```
duplicated = df[df.duplicated()]
if len(duplicated):
    print(duplicated)
else:
    print("keine Duplikate vorhanden")
```

```
            Time        V1        V2        V3        V4        V5        V6  \
33          26.0 -0.529912  0.873892  1.347247  0.145457  0.414209  0.100223
35          26.0 -0.535388  0.865268  1.351076  0.147575  0.433680  0.086983
113         74.0  1.038370  0.127486  0.184456  1.109950  0.441699  0.945283
114         74.0  1.038370  0.127486  0.184456  1.109950  0.441699  0.945283
115         74.0  1.038370  0.127486  0.184456  1.109950  0.441699  0.945283
...          ...       ...       ...       ...       ...       ...       ...
```

Abbildung 18: Prüfung auf Duplikate in den Daten

Insgesamt wurden 1.081 Zeilen als Duplikate erkannt und nach manueller Kontrolle in der .csv-Datei (z.B. Datensatz 33, 113, 115) bestätigt. Mit dem Befehl ‚df.drop_duplicates(inplace=True)' werden die Daten direkt im vorhandenen DataFrame gelöscht. Eine erneute Prüfung auf Duplikate ergibt nun keine weiteren Treffer. Nachdem das Einlesen, Auswerten und Aufbereiten der Daten abgeschlossen ist, kann mit dem Training der einzelnen Modelle begonnen werden. Dazu werden, wie in Abbildung 19 dargestellt, drei Kopien der folgenden Datei aufbereitet ‚df_logreg', ‚df_nn' und ‚df_ranfk'.

```
# Kopie von df erstellen für Logistische Regression

df_logreg = df
```

```
# Kopie von df erstellen für Neuronale Netze

df_nn = df
```

```
# Kopie von df erstellen für Random-Forest-Klassifikator

df_ranfk = df
```

Abbildung 19: Erstellen von Kopien für die Modelle

Bei allen Modellen wurde explizit auf eine Aufteilung nach dem Pareto-Prinzip (80 zu 20) verzichtet, da sonst möglicherweise zu wenige betrügerische Fälle in der Testdatei verbleiben würden. Zudem würde SMOTE nicht angewendet werden, falls es zur Verwendung kommt. Es ist außerdem wichtig zu beachten, dass bei allen Modellen die richtigen Vorhersagen bei 99,83 % und die falschen bei 0,17 % liegen würden, wenn jeder einzelne Datensatz als legitim vorhergesagt würde.

3.6.1 Logistische Regression

Für die Durchführung der logistischen Regression wird kein TensorFlow verwendet, da alle benötigten Modelle bereits im Modul ‚sklearn' vorhanden sind. Stattdessen kann hier SMOTE zur Beseitigung von Ungleichgewichten eingesetzt werden. Zunächst werden die Eingabe- von den Ausgabemerkmalen getrennt, wie in Abbildung 20 zu sehen ist. Anschließend werden die Eingabemerkmale skaliert und testweise ausgegeben.

```
# Trennen Eingabe- und Ausgabemerkmale, Skalieren Eingabe und Testausgabe

X=df_logreg.drop(columns=["Class"])
y=df_logreg["Class"]

names=X.columns
from sklearn import preprocessing
scaled_df = preprocessing.scale(X)
scaled_df = pd.DataFrame(scaled_df,columns=names)

scaled_df.head()
```

Abbildung 20: Trennen und Bearbeiten von Eingabe- und Ausgabemerkmalen

Nachdem die Merkmale erfolgreich getrennt und aufbereitet wurden, kann mit der Aufteilung in Trainings- und Testdaten begonnen werden. Für die Testdaten werden, wie in Abbildung 21 dargestellt, 30 % aller Daten verwendet. Anschließend werden sowohl die Anzahl der getrennten Daten als auch die Anzahl der legitimen und betrügerischen Fälle pro Trennung ausgegeben.

```
# Trennen nach Trainings- und Testdaten sowie Ausgabe

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(scaled_df, y, test_size = 0.30, random_state = 0, shuffle = True, stratify = y)
X_train.shape, X_test.shape
```

```
((198608, 30), (85118, 30))
```

```
# Ausgabe Train VALUE counts

y_train.value_counts()
```

```
Class
0    198277
1       331
Name: count, dtype: int64
```

```
# Ausgabe Test VALUE counts

y_test.value_counts()
```

```
Class
0    84976
1      142
Name: count, dtype: int64
```

Abbildung 21: Trennen nach Trainings- und Testdaten

Nach erfolgreicher Trennung kann mit der Aufbereitung durch SMOTE begonnen werden. Diese Methode erzeugt synthetische Daten in der Minderheitsklasse, um Ungleichgewichte zu beseitigen, anstatt einfach nur Daten der Minderheitsklasse zu duplizieren. SMOTE sollte hierbei nur auf die Trainingsdaten angewandt werden, da die Testdaten zur Validierung unverändert bleiben sollen. Wie in Abbildung 22 dargestellt, wurden die Datensätze von legitimen und betrügerischen Transaktionen durch die Verwendung der SMOTE-Methode angeglichen.

```
# Ungleichgewichte der Trainingsdaten beseitigen

from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 33)
X_train_new, y_train_new = sm.fit_resample(X_train, y_train.to_numpy())

import matplotlib.pyplot as plt
pd.Series(y_train_new).value_counts().plot(kind="bar")
```

```
<Axes: >
```

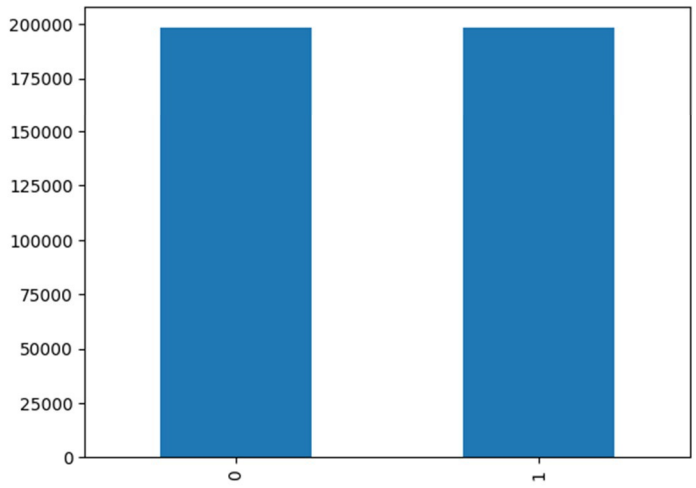


Abbildung 22: Ungleichgewicht der Trainingsdaten beseitigen

Anschließend kann mit der Umsetzung der logistischen Regression begonnen werden. Hierfür wird der ‚lbfgs'-Solver der LogisticRegression aus dem sklearn-Modul verwendet. Zunächst werden die Trainingsdaten zum Lernen zugewiesen, wie in Abbildung 23 dargestellt, um anschließend Vorhersagen für sowohl die Trainings- als auch die Testdaten zu erhalten.

```
# Trainieren der Daten

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, precision_score, recall_score, f1_score

clf = LogisticRegression(solver = 'lbfgs')
clf.fit(X_train_new, y_train_new)
train_pred = clf.predict(X_train_new)
test_pred = clf.predict(X_test)
```

Abbildung 23: Trainieren der Daten mittels logistischer Regression

Für eine detaillierte Auswertung der Vorhersagen der logistischen Regression eignet sich am besten eine Confusion Matrix über die Testdaten. Wie in Abbildung 24 zu sehen ist, wurden bei dem Modell 82.744 legitime sowie 129 betrügerische Fälle richtig vorhergesagt. Lediglich 2.232 legitime und 13 betrügerische Fälle wurden falsch vorhergesagt.

```
# Ausgabe eines Plots einer Confusion Matrix

cm=confusion_matrix(y_test, test_pred)
import seaborn as sns
plt.figure(figsize=(5,4))
sns.set(font_scale=1.2)
sns.heatmap(cm, annot=True, fmt = 'g', cmap="Reds", cbar = False)
plt.xlabel("Predicted Label", size = 15)
plt.ylabel("True Label", size = 15)
```

```
Text(25.75, 0.5, 'True Label')
```

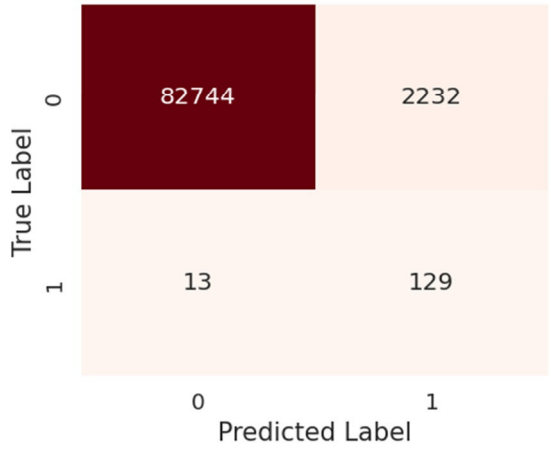


Abbildung 24: Confusion Matrix der logistischen Regression

Bei der Berechnung von 2.232 / 84.976 ergibt sich eine falsche Vorhersage von nur 2,63 % bei den legitimen und bei 13 / 142 eine falsche Vorhersage von 9,15 %.

3.6.2 Random-Forest-Klassifikator

Für die Durchführung des Random-Forest-Klassifikators wird ebenfalls kein TensorFlow verwendet, da alle benötigten Modelle bereits im Modul sklearn vorhanden sind. Außerdem wird keine SMOTE-Methode zur Beseitigung von Ungleichgewichten verwendet, da dies im Random-Forest-Modell nicht zwingend erforderlich ist.

Auch im Random-Forest-Klassifikator-Modell wird zunächst, wie in Abbildung 25 zu sehen, mit der Trennung der Merkmale nach Eingabe- und Ausgabemerkmale und der Aufteilung der Eingabemerkmale in Trainings- und Testdaten begonnen.

```
# Trennen Eingabe- und Ausgabemerkmale sowie Aufteilung in Trainings- und Testdaten

target = 'Class'
predictors = ['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11', 'V12', 'V13', 'V14', 'V15',\
              'V16', 'V17', 'V18', 'V19', 'V20', 'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount']

train_df_ranfk, test_df_ranfk = train_test_split(df_ranfk, test_size = 0.30, random_state = 0, shuffle = True,\
                                stratify = df_ranfk["Class"])
```

Abbildung 25: Bearbeiten der Daten des Random-Forest-Klassifikators

Für die Funktion des RandomForestClassifier, die von sklearn bereitgestellt wird, werden folgende Einstellungen festgelegt:

- Anzahl paralleler Aufträge = 5,
- Anzahl der Sortierdurchläufe vor der Aufteilung = 1000,
- verwendete Metrik = ‚gini',
- Anzahl der Schätzer = 100.

Die hier verwendete Metrik ‚gini' ist ein Maß für die Unreinheit oder Vielfalt der Elemente innerhalb eines Datensatzes. Mit diesen Einstellungen kann nun mit dem Training des Random-Forest-Klassifikators begonnen werden, wie in Abbildung 26 dargestellt.

```
# Trainieren des Random-Forest-Klassifikators

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_jobs = 5, random_state = 1000, criterion = 'gini', n_estimators = 100, verbose = False)

clf.fit(train_df_ranfk[predictors], train_df_ranfk[target].values)
```

```
RandomForestClassifier
RandomForestClassifier(n_jobs=5, random_state=1000, verbose=False)
```

Abbildung 26: Trainieren des Random-Forest-Klassifikators

Es ist zu beachten, dass das Training hier mehrere Minuten in Anspruch nehmen kann. Nach Abschluss des Trainings kann mit der Vorhersage der Testdaten fortgefahren werden. Die Ergebnisse können ebenfalls direkt ausgewertet werden. Dazu wird, wie in

Abbildung 27 zu sehen ist, nach der Vorhersage zuerst eine Anzeige der wichtigsten ermittelten Merkmale ausgegeben.

```
# Vorhersage der Testdaten Ergebnisse

df_ranfk_predictions = clf.predict(test_df_ranfk[predictors])
```

```
# Anzeige der wichtigsten Merkmale

tmp = pd.DataFrame({'Feature': predictors, 'Feature importance': clf.feature_importances_})
tmp = tmp.sort_values(by='Feature importance',ascending=False)
plt.figure(figsize = (10,4))
s = sns.barplot(x='Feature',y='Feature importance',data=tmp, palette = 'hsv', hue = predictors, legend=False)
s.set_xticks(predictors)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
plt.show()
```

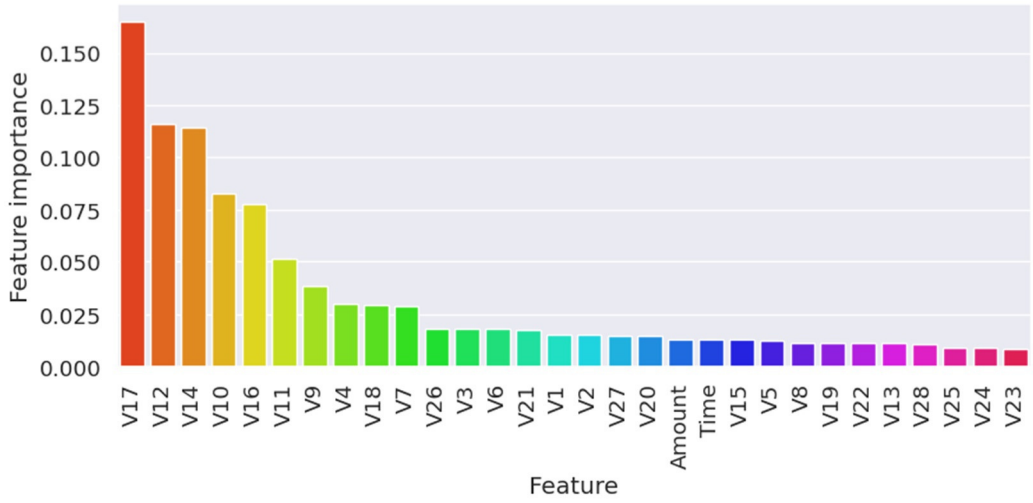


Abbildung 27: Vorhersage der Testdaten sowie Auswertung der Merkmale

Für die Auswertung des Random-Forest-Klassifikators eignet sich auch eine Confusion Matrix über die Testdaten. Wie in Abbildung 28 zu sehen ist, wurden 84.974 legitime Fälle sowie 114 betrügerische Fälle richtig vorhergesagt. Lediglich 2 legitime und 28 betrügerische Fälle wurden falsch vorhergesagt.

```
cm_ranfk = confusion_matrix(test_df_ranfk[target].values, df_ranfk_predictions)
plt.figure(figsize=(5,4))
sns.set(font_scale=1.2)
sns.heatmap(cm_ranfk, annot=True, fmt = 'g', cmap="Reds", cbar = False)
plt.xlabel("Predicted Label", size = 15)
plt.ylabel("True Label", size = 15)
```

```
Text(25.75, 0.5, 'True Label')
```

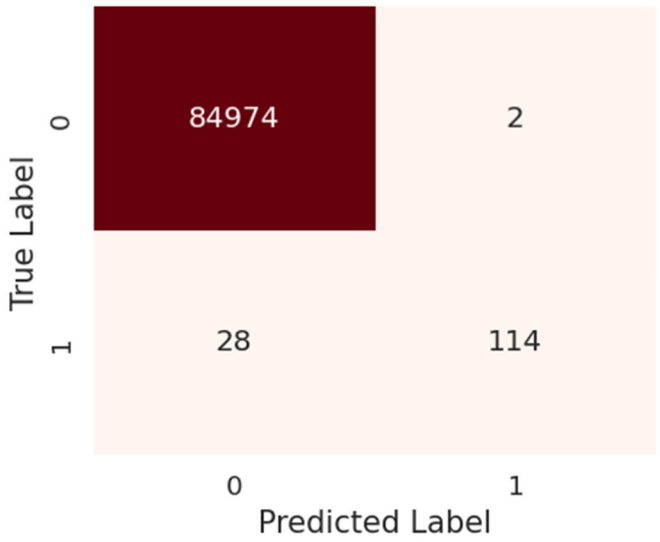


Abbildung 28: Confusion Matrix des Random-Forest-Klassifikators

Bei einer Berechnung mit 2 / 84.976 ergibt sich eine falsche Vorhersage von nur 0,00 % bei den legitimen und mit 28 / 142 eine falsche Vorhersage von 19,71 %.

3.6.3 Neuronale Netze

Für den Einsatz des NN werden sowohl TensorFlow als auch Keras genutzt. Dabei wird vor allem das Sequential-Modell verwendet, welches eine lineare Stapelung von Schichten darstellt, um ein NN zu bilden. Zudem werden die Trainingsdaten vorher durch SMOTE aufbereitet. Wie in Abbildung 29 dargestellt, wurden die Daten zunächst nach Ein- und Ausgabemerkmale getrennt, skaliert und anschließend in Trainings- und Testdaten aufgeteilt. Abschließend wurden die Trainingsdaten durch SMOTE aufbereitet.

```
# Trennen nach Eingabe- und Ausgabemerkmale sowie skalieren der Daten

df_nn_x = df_nn.drop(columns=["Class"])
df_nn_y = df_nn["Class"]

df_nn_names = df_nn_x.columns
from sklearn import preprocessing
df_nn_scaled_df = preprocessing.scale(df_nn_x)
df_nn_scaled_df = pd.DataFrame(df_nn_scaled_df,columns = df_nn_names)
```

```
# Trennen nach Trainings- und Testdaten

from sklearn.model_selection import train_test_split
df_nn_x_train, df_nn_x_test, df_nn_y_train, df_nn_y_test = train_test_split(df_nn_scaled_df, df_nn_y, test_size = 0.30,\
                                                           random_state = 0, shuffle = True, stratify = df_nn_y)
```

```
# Ungleichgewichte der Trainingsdaten beseitigen

from imblearn.over_sampling import SMOTE
df_nn_sm = SMOTE(random_state = 33)
df_nn_x_train_new, df_nn_y_train_new = df_nn_sm.fit_resample(df_nn_x_train, df_nn_y_train.to_numpy())
```

Abbildung 29: Aufbereitung der Daten für neuronale Netze

Nach erfolgreicher Datenverarbeitung kann mit der Entwicklung des NN-Modells fortgefahren werden. Verwendet wird das Sequential-Modell, welches eine schichtweise Strukturierung des NN-Modells ermöglicht, wie in Abbildung 30 dargestellt. Als Aktivierungsfunktion wird zuerst ‚relu' eingesetzt, was für ‚Rectified Linear Unit' steht. Diese Funktion gibt den Wert ‚0' für eine negative Eingabe aus und gibt eine positive Eingabe unverändert weiter. Danach werden zwei versteckte Schichten mit jeweils 64 Einheiten definiert. Als Aktivierungsfunktion für den Output wird die Funktion ‚Sigmoid' verwendet, welche Eingabewerte zwischen 0 und 1 abbildet. Dies ist besonders geeignet für die Ausgabe von Klassifizierungen.

```
# Festlegen der Einstellungen für das NN Model

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout, BatchNormalization

model = Sequential()
model.add(Dense(df_nn_x_train_new.shape[1], activation = 'relu', input_dim = df_nn_x_train_new.shape[1]))
model.add(BatchNormalization())

model.add(Dense(64, activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(64, activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(1, activation = 'sigmoid'))
```

Abbildung 30: Festlegen der Einstellungen für das NN-Model

Anschließend wird das NN-Modell kompiliert. Dabei ist es wichtig, wie in Abbildung 31 dargestellt, einer Überanpassung des Modells vorzubeugen. Hierfür wird die EarlyStopping-Funktion verwendet, welche die Anzahl der Epochen stoppt, wenn sich die Bewertungsmetriken nicht weiter verbessern. Für die Ausführung wurden maximal 150 Epochen festgelegt, um zu überprüfen, ob die EarlyStopping-Funktion ausgelöst wird, wenn die Lernkurve zu flach wird. Als Lernrate wurde der Wert ‚0.0001' festgelegt, um Überanpassungen zu minimieren und dem Modell nur vorsichtige Anpassungen zu ermöglichen. Dies erfordert jedoch eine höhere Anzahl an Epochen, um eine optimale Leistung zu erreichen.

```
# NN Model kompilieren

optimizer = keras.optimizers.Adam(learning_rate = 0.0001)
model.compile(optimizer = optimizer, loss = 'binary_crossentropy')
```

```
# Festlegen der EarlyStopping Funktion

from tensorflow.keras.callbacks import EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience = 10)
```

```
# Auswerten des Models mit 150 Epochen zum Test der EarlyStopping Funktion

history = model.fit(x = df_nn_x_train_new, y = df_nn_y_train_new, batch_size = 256, epochs=150,
          validation_data=(df_nn_x_test, df_nn_y_test), verbose=1,
          callbacks=[early_stop])
```

```
Epoch 1/150
1550/1550 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.4890 - val_loss: 0.0914
Epoch 2/150
1550/1550 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.1530 - val_loss: 0.0712
Epoch 3/150
1550/1550 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.1076 - val_loss: 0.0533
```

Abbildung 31: NN-Model kompilieren und EarlyStopping Funktion

Nach dem Training über alle Epochen wurde nach 63 Epochen die EarlyStopping-Funktion ausgelöst, wie in Abbildung 32 zu sehen ist.

```
1550/1550 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.0040 - val_loss: 0.0068
Epoch 63/150
1550/1550 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.0038 - val_loss: 0.0068
Epoch 63: early stopping
```

Abbildung 32: Auslösung der EarlyStopping Funktion

Anschließend kann eine Verlustfunktion dargestellt werden, wie in Abbildung 33 zu sehen ist. Diese Funktion misst, wie gut das Modell während des Trainings lernt und beobachtet die Lernkurve zwischen den Vorhersagen und den tatsächlichen Werten.

```
# Darstellung Verlustfunktion für Trainings- und Testdaten

evaluation_metrics=pd.DataFrame(model.history.history)
evaluation_metrics.plot(figsize=(10,5))
plt.title("Verlustfunktion für Trainings- und Testdaten", size = 20)
```

```
Text(0.5, 1.0, 'Verlustfunktion für Trainings- und Testdaten')
```

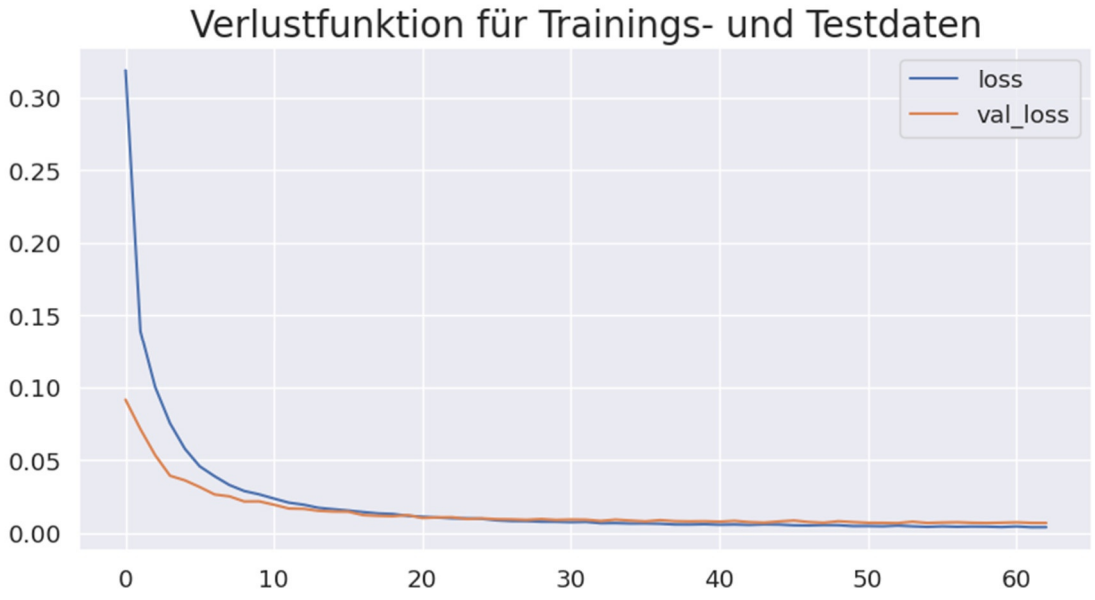


Abbildung 33: Verlustfunktion des NN-Models

Für die Auswertung des NN wird wiederholt eine Confusion Matrix über die Testdaten verwendet. Wie in Abbildung 34 zu sehen ist, wurden 84.903 legitime Fälle sowie 119 betrügerische Fälle richtig vorhergesagt. Lediglich 73 legitime und 23 betrügerische Fälle wurden falsch vorhergesagt. Bei der Ausgabe ist es besonders wichtig, bei einer binären Klassifizierung die Prediction > 0.5 abzufangen, da sonst falsche Werte in der Confusion Matrix angezeigt werden. Lediglich bei einer Multi-Klassen-Klassifizierung kann der Code ‚numpy.argmax(model.prdict(df_nn_x_test),axis=1)' verwendet werden.

```
# Ausgabe eines Plots einer Confusion Matrix
# WICHTIG: Hier gilt es bei der Ausgabe besonders darauf zu achten, dass bei einer binären Klassifizierung die
# Prediction > 0.5 abgefangen werden muss, da sonst völlig falsche Werte in der Confusion Matrix angezeigt werden.
# Lediglich eine Multi-Klassen-Klassifizierung könnte mittels des Codes:
# „numpy.argmax(model.predict(df_nn_x_test),axis=1)“
# ausgegeben werden.

df_nn_y_pred = (model.predict(df_nn_x_test) > 0.5).astype("int32")
cm_df_nn=confusion_matrix(df_nn_y_test, df_nn_y_pred)

plt.figure(figsize=(5,4))
sns.set(font_scale=1.2)
sns.heatmap(cm_df_nn, annot=True, fmt = 'g', cmap="Reds", cbar = False)
plt.xlabel("Predicted Label", size = 15)
plt.ylabel("True Label", size = 15)
```

```
2660/2660 ━━━━━━━━━━━━━━━━━━━━ 2s 601us/step
Text(25.75, 0.5, 'True Label')
```

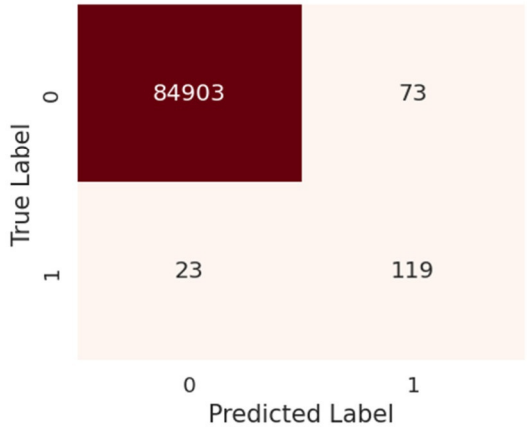


Abbildung 34: Confusion Matrix des NN-Models

3.7 Auswertung

Beim Vergleich der drei Modelle in Abbildung 35 ergeben sich unterschiedliche Ergebnisse. Es ist zu beachten, dass selbst geringfügige Veränderungen der Modelle zu unterschiedlichen Ergebnissen führen können. Selbst wenn das gleiche Projekt wiederholt ausgeführt wird, kann dies bereits zu unterschiedlichen Ergebnissen führen. Dies liegt daran, dass bei der Datenaufteilung oft Zufälligkeiten genutzt werden und viele Modelle zum Trainieren Gewichte, stochastische Trainingsprozesse, Regularisierungstechniken oder Konvergenzen nutzen. Dadurch entstehen unterschiedliche Ergebnisse in der Confusion Matrix.

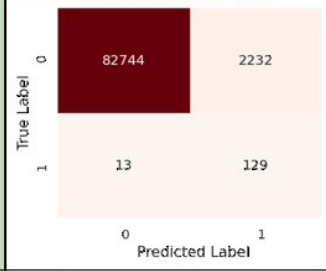
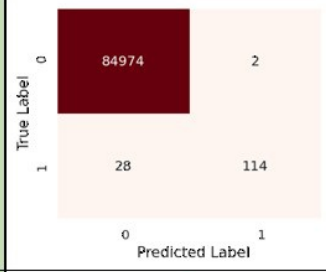
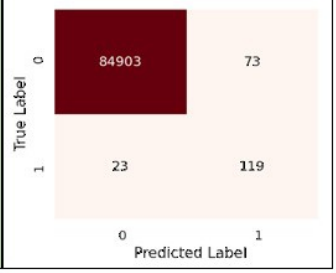
Model	TensorFlow	SMOTE	TP	TN	FN	FP	Grafik Confusion Matrix
Logistische Regression (kein TensorFlow, dafür SMOTE)	x	✓	82.744	2.232	13	129	
Random-Forest-Klassifikator (kein TensorFlow, kein SMOTE)	x	x	84.974	2	28	114	
Neuronale Netze (mit TensorFlow und SMOTE)	✓	✓	84.896	80	25	117	

Abbildung 35: Auswertung aller Modelle

Um eine Auswertung oder einen Vergleich der einzelnen Modelle durchzuführen, muss zunächst die Bedeutung der einzelnen Ergebnisse bestimmt werden. Es kann beispielsweise eine Priorität sein, möglichst viele betrügerische Fälle zu identifizieren. Allerdings kann dies dazu führen, dass sehr viele legitime Datensätze als betrügerisch markiert werden, was wiederum eine große Menge an Zeit und Geld für Kontrollen erfordern würde. Wenn die Modelle jedoch zu sehr darauf ausgerichtet sind, legitime Fälle korrekt zu markieren, könnten große Summen an betrügerischen Transaktionen übersehen werden. Darüber hinaus könnte durch eine Vielzahl von erfolgreichen betrügerischen Aktivitäten die Anzahl dieser weiter ansteigen. In der praktischen Umsetzung sowie im Kontext von Banken und Banksystemen sind die Prioritäten und die Wichtigkeit der Ergebnisse nicht bekannt, weshalb keine Schlussbewertung und kein Vergleich der Modelle zu einem besten Modell führen kann.

4 Fazit

4.1 Diskussion der Ergebnisse

Im Einleitungskapitel wurde die Datenerhebung des Bundeskriminalamts der in Deutschland polizeilich erfassten Fälle von EC-Kartenbetrug zwischen 2011 und 2022 auf der Plattform Statista vorgestellt. Es wird darauf eingegangen, dass im digitalen Zeitalter die Möglichkeiten, an Kreditkartendaten zu gelangen, nahezu endlos sind. Als einer der beiden Mechanismen zur Bekämpfung von Kreditkartenbetrug wird sich in dieser Arbeit auf das Erkennen des Betrugs spezialisiert. Mit der Umsetzung wurde begonnen, indem die technische Umgebung vorbereitet wurde. Da eine praktische Anwendbarkeit im realen Arbeitskontext angestrebt wurde, waren die Parameter der technischen Umsetzung bereits gegeben. Die praktische Umsetzung dieser Parameter, einschließlich der Einrichtung der technischen Umgebung, verlief problemlos dank zahlreicher Fachlektüre zu Linux. Die Installation des Linux-Betriebssystems sowie die Hinzufügung von Python, TensorFlow und Jupyter Notebook verliefen ohne Verzögerungen oder Komplikationen. Die Verwendung von Jupyter Notebook wurde durch zahlreiche Wikis, Dokumentationen und andere Projekte von Kaggle erleichtert. Zur Beschaffung der Daten wurde auf eine bereits aufbereitete Datei zurückgegriffen. Die Datenaufbereitung wurde ausführlich erläutert und es wurden Datenüberprüfungen und -bereinigungen für dieses Projekt durchgeführt. Nach erfolgreicher Datenanalyse konnten drei Modelle identifiziert und praktisch umgesetzt werden.

Zur Umsetzung der Modelle wurden zunächst die Daten bereinigt und aufgeteilt, damit jedes Modell eigene Daten verwenden kann und es bei den Ergebnissen nicht zu Verwechslungen kommt. Die drei unterschiedlichen Modelle ermöglichten auch unterschiedliche Datenaufbereitungen. Bei der logistischen Regression wurde ohne TensorFlow gearbeitet, dafür mit der SMOTE-Methode, welche synthetische Daten in der Minderheitsklasse erzeugt. Auch ohne TensorFlow und SMOTE konnte der Random-Forest-Klassifikator aufgrund der Anzahl der Parameter und verwendeten Einstellungen durch viele Durchläufe ein sehr gutes Ergebnis erzielen. Für die NN wurde, wie in dieser Arbeit geplant, TensorFlow verwendet. Die Verwendung von TensorFlow bietet eine Vielzahl von Einstellungen und Möglichkeiten, um ein Modell zu trainieren. Gemäß Kapitel 3.6.3 wurde ein Sequential-Modell gewählt, bei dem Schicht für Schicht zusammengestellt wird. Das NN wurde mit einer ‚relu'- und einer ‚sigmoid'-Funktion sowie zwei versteckten Schichten mit jeweils 64 Einheiten implementiert. Darüber hinaus wurde ein Optimierer

mit einer vorgegebenen Lernrate für langsames Lernen mithilfe von TensorFlow verwendet. Zudem konnte eine Methode namens ‚EarlyStopping', die ebenfalls zu TensorFlow gehört, angewandt werden. Diese Methode verhindert unnötiges Durchlaufen aller Epochen, auch wenn kein Mehrwert mehr entsteht, was erheblich Zeit spart, falls zu viele Epochen gewählt sind und die Lernkurve zu schnell abflacht. Die Ergebnisse der Modelle wurden mittels Confusion Matrix dargestellt, nachdem sie auf den zuvor separierten Testdaten angewendet wurden. Für eine Auswertung oder einen Vergleich der einzelnen Modelle fehlen in diesem Projekt die Wichtigkeiten und Prioritäten der einzelnen Parameter der Confusion Matrix.

Abschließend ist festzuhalten, dass alle drei Modelle erfolgreich mit Python und TensorFlow analysiert wurden.

4.2 Limitationen

Bei der Erkennung von Kreditkartenbetrug müssen verschiedene Limitationen bei der Modellierung und Datenanalyse berücksichtigt werden. Eine dieser Limitationen ist die Auswahl und Implementierung passender Modelle für das Trainieren und Auswerten der Daten. Aufgrund von Ressourcenbeschränkungen wie Zeit und Rechenkapazität sowie der großen Anzahl möglicher Modelle können nicht alle potenziell relevanten Modelle berücksichtigt werden. Die spezifischen Algorithmen in den verschiedenen Modellen, wie in Kapitel 2.4 vorgestellt, könnten dazu beitragen, eine höhere Genauigkeit oder Effizienz bei der Betrugserkennung zu erreichen. Es ist jedoch zu beachten, dass die Komplexität und Vielzahl der Einstellmöglichkeiten jedes Modells zu unterschiedlichen Ergebnissen führen können.

Eine weitere Limitation dieser Arbeit betrifft die Qualität und Auswertbarkeit der verwendeten Datensätze. Für ein effektives Training ist eine große, aktuelle und repräsentative Datenmenge erforderlich, die idealerweise ein ausgewogenes Verhältnis zwischen legitimen und betrügerischen Datensätzen aufweist. Hierbei müssen die fehlenden Informationen zu den Inhalten der Spalten V1 bis V28 genannt werden. Diese Informationen sind notwendig, um eine effiziente Auswertung und einen Vergleich der Confusion Matrix der verschiedenen Modelle zu ermöglichen und das am besten geeignete Modell zu bestimmen. Ein genaueres Verständnis der Daten und der Interpretation der Modelle kann durch das Wissen über den Inhalt der Spalten V1 bis V28 erheblich verbessert werden. Darüber hinaus ist es aufgrund der starken Unausgewogenheit der bereitgestellten Daten nicht möglich, eine prozentuale Bewertung der Genauigkeit der Modellvorhersagen vorzunehmen. Selbst wenn jeder Datensatz als legitim eingestuft würde,

würde eine Trefferquote von 99,83 % erreicht werden. Möglicherweise wäre es jedoch sinnvoll, weitere Daten zu sammeln, um eine aussagekräftigere Bewertung vornehmen zu können.

4.3 Zusammenfassung

Das Hauptziel dieser Arbeit war es, Ansätze zur Erkennung von Kreditkartenbetrug mittels Python und TensorFlow zu finden. Der Datensatz wies ein starkes Ungleichgewicht von nur 0,17 % an betrügerischen Fällen auf, im Gegensatz zu gleichmäßig gewichteten Datensätzen. Es ist daher schwierig zu beurteilen, ob ein Trainingsmodell gute Ergebnisse erzielen kann. Basierend auf den theoretischen Grundlagen, die in Kapitel 2 beschrieben wurden, konnten drei Modelle identifiziert und analysiert werden. In Kapitel 3 wurden zuerst die technischen Voraussetzungen der Maschinen behandelt, um anschließend mit der Installation und Konfiguration des Betriebssystems, sowie den verwendeten Python Modulen fortzufahren. Anschließend wurde, nach der Beschaffung und Aufbereitung der Daten, mit der Umsetzung der drei zu analysierenden Modelle begonnen. Die Analyse der Daten mittels Python und TensorFlow war sowohl mittels logistischer Regression, einem Random-Forest-Klassifikator als auch mithilfe NN möglich, weshalb sich die Ansätze zur Analyse mit Python und TensorFlow durchaus zur Erkennung von Kreditkartenbetrug eignen.

Die Leitfrage dieser Arbeit: ‚Wie trägt der Einsatz von ML-Technologien, insbesondere unter Verwendung von Python und TensorFlow, zur Aufdeckung und Prävention von Kreditkartenbetrug im Finanzsektor bei?' kann abschließend wie folgt beantwortet werden: Selbst bei stark ungleich gewichteten Kreditkartendaten aus dem Finanzsektor kann mithilfe von Python und TensorFlow eine signifikante Verbesserung bei der Erkennung und Prävention von Kreditkartenbetrug erreicht werden. Hierbei können verschiedene Modelle wie die logistische Regression, der Random-Forest-Klassifikator oder NN eingesetzt werden.

4.4 Ausblick

Diese Arbeit zeigt mögliche Herangehensweisen auf, wie mithilfe von Python und TensorFlow und dem Einsatz von ML eine Fraud Detection auch bei ungleich gewichteten Daten durchgeführt werden kann. Es ergibt sich die logische Schlussfolgerung, dass der praktische Einsatz im Arbeitsumfeld eine Vielzahl von Anwendungsfeldern in diesem Gebiet bietet. Somit könnten in Zukunft auch in anderen Branchen Technologien zur Betrugserkennung eingesetzt werden, beispielsweise im Gesundheitswesen, im E-

Commerce, in der Gesundheitsbranche oder in der Steuerverwaltung, in denen Betrug ein Problem darstellt. Die kontinuierliche Weiterentwicklung von ML-Methoden bietet zudem immer neue Möglichkeiten, eine Fraud Detection präziser und adaptiver zu gestalten. In Fortführung der hier entwickelten Thesis wird folgendes neues Projekt umgesetzt: Das Ziel ist eine praktische Anwendbarkeit im realen Arbeitskontext. Die Entwicklung einer KI-basierten Entscheidungsunterstützung [redacted]

„Dieser Abschnitt wurde aufgrund von Geheimhaltungspflichten geschwärzt."

Anhang

Anhang 1: Quellcode

Der gesamte Quellcode wurde in Jupyter Notebooks geschrieben und dokumentiert. Die Datei ‚Credit Card Fraud Detection.ipynb' befindet sich im elektronischen Repository unter ‚Daten'.

Anhang 2: Projektdateien

Komplettes Arbeitsumfeld, inklusive VM, Datendatei, Quellen, Citavi Export und vielem mehr.

Der Link zu allen benötigten Materialien, einschließlich des Arbeitsumfelds als VM im ‚.ova-Format', der Datendatei ‚creditcard.csv.zip' von Kaggle, aller verwendeten Quellen, des Citavi-Exports, des genutzten Leitfadens und vielem mehr, ist hier zu finden:

‚https://nextcloud.[redacted]

Das Passwort für den Lesezugriff lautet: [redacted] „Dieser Abschnitt wurde aufgrund von Geheimhaltungspflichten geschwärzt."

Die Projektdateien sind temporär bis zur Erteilung der Bachelor-Urkunde abrufbar.

Literaturverzeichnis

Aisha Abdallah, Mohd Aizaini Maarof, Anazida Zainal (2016): Fraud detection system: A survey, in: Journal of Network and Computer Applications 68 (2016), S. 90-113, https://doi.org/10.1016/j.jnca.2016.04.007

Albon, Chris (2018): Machine learning with Python cookbook: Practical solutions from preprocessing to deep learning, First edition, Beijing u. a.: O'Reilly, 2018

Barrett, Daniel J. (2016): Linux pocket guide, Third edition, Beijing, China: O'Reilly Media, 2016

Bonaccorso, Giuseppe (2018): Machine Learning Algorithms: Popular Algorithms for Data Science and Machine Learning, Second Edition, Birmingham/Mumbai: Packt, 2018

Burkov, Andriy (2020): Machine learning engineering, Quebec City, Canada: True Positive Inc, 2020

Buxmann, Peter (2021): Künstliche Intelligenz: Mit Algorithmen Zum Wirtschaftlichen Erfolg, 2nd ed., Berlin, Heidelberg: Springer Berlin / Heidelberg, 2021

Chollet, François (2018): Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek, Frechen: mitp, 2018

Frochte, Jörg (2021): Maschinelles Lernen: Grundlagen und Algorithmen in Python, 3., überarbeitete und erweiterte Auflage, München: Hanser, 2021

Géron, Aurélien (2023): Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme, übers. von *Kristian Rother*/*Thomas Demmig*, 3., aktualisierte und erweiterte Auflage, Heidelberg: O'Reilly, 2023

Gollapudi, Sunila (2016): Practical machine learning: Tackle the real-world complexities of modern machine learning with innovative and cutting-edge techniques, Birmingham, UK: Packt Publishing, 2016

Goodfellow, Ian/Courville, Aaron/Bengio, Yoshua (2016): Deep learning, Cambridge, Massachusetts: The MIT Press, 2016

Humm, Bernhard G., Bense, Hermann, Fuchs, Michael, Gernhardt, Benjamin, Hemmje, Matthias, Hoppe, Thomas, Kaupp, Lukas, Lothary, Sebastian, Schäfer, Kai-Uwe,

Thull, Bernhard, Vogel, Tobias, Wenning, Rigo (2021): Machine intelligence today: applications, methodology, and technology, in: Informatik Spektrum 44 (2021), S. 104-114

Mangal, Ekta, Divya, Shubham, Gussain, Radhika (2023): Credit Card Fraud Detection using Python & Machine Learning Algorithms, in: International Journal for Research in Applied Science and Engineering Technology Volume 11 Issue V (2023), S. 3120–3128

Mitchell, Tom M. (2010): Machine learning, International ed., [Reprint.], New York, NY: McGraw-Hill, 2010

Mohri, Mehryar/Rostamizadeh, Afshin/Talwalkar, Ameet (2018): Foundations of machine learning, Second edition, Cambridge, Massachusetts/London, England: The MIT Press, 2018

Müller, Andreas C./Guido, Sarah (2017): Introduction to machine learning with Python: A guide for data scientists, First edition, Sebastopol, CA: O'Reilly Media, 2017

Raschka, Sebastian/Mirjalili, Vahid (2021): Machine Learning mit Python und Keras, TensorFlow 2 und Scikit-learn: Das umfassende Praxis-Handbuch für Data Science, Deep Learning und Predictive Analytics, übers. von *Knut Lorenzen*, 3., aktualisierte und erweiterte Auflage, Frechen: mitp, 2021

Shai, Shalev-Shwartz, Shai, Ben-David (2014): Understanding machine learning: From theory to algorithms, Cambridge u. a.: Cambrige University Press, 2014

Shotts, William E. (2019): The Linux command line: A complete introduction, 2nd edition, San Francisco: No Starch Press, 2019

Taulli, Tom (2022): Grundlagen der Künstlichen Intelligenz: Eine nichttechnische Einführung, Berlin, Heidelberg: Springer Berlin Heidelberg, 2022

Velayutham, Sathiyamoorthi (2020): Handbook of research on applications and implementations of machine learning techniques, Hershey PA, USA: IGI Global, 2020

Ward, Brian (2021): How Linux works: What every superuser should know, 3rd edition, San Francisco: No Starch Press, 2021

Internetquellen

BKA - Bundeskriminalamt (2024): Zahlungskartenkriminalität, <https://www.bka.de/DE/UnsereAufgaben/Deliktsbereiche/Zahlungskartenkriminalitaet/zahlungskartenkriminalitaet_node.html> (2024-01-08) [Zugriff 2024-01-11]

Deutsche Bank Research (2018): Kartenbetrug in Deutschland: Geringer Anteil, aber hohe Kosten, <https://www.dbresearch.de/PROD/RPS_DE-PROD/PROD0000000000484136/Kartenbetrug_in_Deutschland:_Geringer_Anteil%2C_aber.xhtml> (2024-01-08) [Zugriff 2024-01-11]

Expert.ai (2022): What Is Machine Learning? A Definition, in: expert.ai v. 14.03.2022, <https://www.expert.ai/blog/machine-learning-definition/> [Zugriff 2024-01-11]

IBM (2024): What is Artificial Intelligence (AI) ?, <https://www.ibm.com/topics/artificial-intelligence> (2024-01-08) [Zugriff 2024-01-10]

IBM Data and AI Team (2024): AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the difference?, <https://www.ibm.com/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks/> (2024-01-08) [Zugriff 2024-01-11]

Juniper Research (2024): Online Payment Fraud Losses to Exceed $206 Billion Over the Next Five Years, <https://www.juniperresearch.com/press/online-payment-fraud-losses-to-exceed-206-billion/> (2024-01-08) [Zugriff 2024-01-11]

Jupyter (2024): Jupyter Documentation, <https://docs.jupyter.org/en/latest/start/index.html> (2024-01-08) [Zugriff 2024-01-11]

Kaggle (2024): Your Machine Learning and Data Science Community, <https://www.kaggle.com/> (2024-01-08) [Zugriff 2024-01-10]

Keras (2024): GitHub Keras 2.0 release notes, <https://github.com/keras-team/keras/wiki/Keras-2.0-release-notes> (2024-01-08) [Zugriff 2024-01-11]

KNIME (2024): Open for Innovation, <https://www.knime.com/> (2024-01-11) [Zugriff 2024-01-15]

Oracle Help Center (2024): Oracle Linux 9 - Documentation, <https://docs.oracle.com/en/operating-systems/oracle-linux/9/> (2024-01-07) [Zugriff 2024-01-09]

Oracle® (2024): VM VirtualBox®, <https://www.virtualbox.org/manual/UserManual.html> (2024-01-05) [Zugriff 2024-01-07]

Python (2024): 3.12.1 Documentation, <https://docs.python.org/3/> (2024-01-08) [Zugriff 2024-01-11]

RapidMiner (2024): Amplify the Impact of Your People, Expertise & Data, <https://rapidminer.com/> (2024-01-11) [Zugriff 2024-01-15]

scikit-learn (2024): machine learning in Python, <https://scikit-learn.org> (2024-01-11) [Zugriff 2024-01-15]

Shift Credit Card Processing (2021): Credit Card Fraud Statistics 2021, <https://shiftprocessing.com/credit-card-fraud-statistics/> (2021-09-29) [Zugriff 2024-01-11]

Statista für Bundeskriminalamt (2024): Anzahl der polizeilich erfassten Fälle von EC-Kartenbetrug in Deutschland von 2011 bis 2022, <https://de.statista.com/statistik/daten/studie/5635/umfrage/anzahl-erfasster-faelle-von-ec-kartenbetrug-in-deutschland/> (2024-01-08) [Zugriff 2024-01-11]

TensorFlow (2024): TensorFlow 2, <https://www.tensorflow.org> (2024-01-08) [Zugriff 2024-01-11]

Wikipedia (2024): Kreditkartenbetrug, <https://de.wikipedia.org/w/index.php?title=Kreditkartenbetrug&oldid=241016326> (2024-01-09) [Zugriff 2024-01-11]

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist. Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde/Prüfungsstelle vorgelegen hat. Ich erkläre mich damit einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Fellbach, 04.05.2024 M. Kroll

(Ort, Datum) (Eigenhändige Unterschrift Matthias Kroll)