INTERNATIONALE HOCHSCHULE

Projektarbeit

Internationale Hochschule Duales Studium

Studiengang: B.Sc. Informatik

Inwieweit sind Machine-Learning-Modelle für Netzwerk-Anomalieerkennung zwischen verschiedenen Datensätzen übertragbar?

Jonas Weirauch

Matrikelnummer: 10237021

Im Wiesengrund 19, 55286 Sulzheim

Betreuende Person: Dominic Lindner

Abgabedatum: 30.09.2025

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AI Artificial Intelligence
DoS Denial-of-Service
IDS Intrusion Detection Systems
ML Machine Learning

1 Einleitung

1.1 Motivation und Problemstellung

Mit über 10,5 Billionen US-Dollar geschätzten jährlichen Schäden bis 2025 stellen Cyberangriffe eine der größten globalen Bedrohungen dar (World Economic Forum, 2024). Gemäß dem Global Risk Report 2024 des Weltwirtschaftsforums gehören Cyberangriffe zu den fünf bedeutendsten globalen Risiken in den nächsten Jahren, was einer Verdreifachung der finanziellen Verluste im Vergleich zu 2015 entspricht (World Economic Forum, 2024). Diese besorgniserregenden Statistiken unterstreichen die akute Notwendigkeit wirksamer Sicherheitsvorkehrungen zum Schutz kritischer Infrastrukturen (Taman, 2024).

Traditionelle signaturbasierte Intrusion Detection Systeme (IDS) erreichen zunehmend ihre Grenzen bei der Erkennung neuartiger Zero-Day-Exploits und unbekannter Angriffsmuster (Belavagi & Muniyal, 2016; Ring et al., 2019). Diese Systeme können lediglich bekannte Signaturen identifizieren und versagen bei der Detektion innovativer Bedrohungen. Gleichzeitig führen die steigende Vernetzung und Digitalisierung zu einer kontinuierlichen Zunahme der Angriffsvektoren und einer erhöhten Komplexität der Netzwerkumgebungen.

Machine Learning (ML) bietet das Potenzial, diese Limitationen zu überwinden und auch bisher unbekannte Angriffsmuster aufzudecken (Vinayakumar et al., 2019). Dennoch ist die tatsächliche Wirksamkeit verschiedener ML-Modelle in heterogenen Netzwerken noch nicht vollständig geklärt. Ein kritisches Problem stellt dabei die Generalisierungsfähigkeit dar: Während Modelle auf spezifischen Trainingsdaten exzellente Leistungen erzielen, zeigen sie oft dramatische Leistungseinbußen beim Transfer auf neue Netzwerkumgebungen oder unterschiedliche Datensätze (Ring et al., 2019).

1.2 Forschungsfrage und Zielsetzung

Diese Arbeit untersucht systematisch die Generalisierungsfähigkeit von zwölf ML-Modellen über zwei fundamental unterschiedliche Netzwerk-Datensätze hinweg. Die zentrale Forschungsfrage lautet:

„Inwieweit sind Machine-Learning-Modelle für Netzwerk-Anomalieerkennung zwischen verschiedenen Datensätzen übertragbar?"

Die Untersuchung fokussiert sich auf die Cross-Dataset-Transferierbarkeit zwischen dem NSL-KDD-Datensatz (Canadian Institute for Cybersecurity, 2024b) (simulierter Netzwerkverkehr von 1998 mit 125.973 Trainingsdatensätzen) und dem CIC-IDS-2017-Datensatz (Canadian Institute for Cybersecurity, 2024a; Sharafaldin et al., 2018) (realistischer Netzwerkverkehr mit 2,8 Millionen Datenpunkten aus einer fünftägigen Netzwerkumgebung). Diese Datensätze unterscheiden sich fundamental in ihrer Datenverteilung, Merkmalsdimensionalität und den abgebildeten Angriffsszenarien (Mourouzis & Avgousti, 2021).

Die konkreten Forschungsziele umfassen:

- **Vergleichende Evaluation**: Systematische Bewertung von Baseline-Modellen (Random Forest, Decision Tree, k-NN) und Advanced-Modellen (XGBoost, LightGBM, Neural Networks) hinsichtlich ihrer Intra-Dataset-Performance und Cross-Dataset-Robustheit.

- **Cross-Dataset-Transferierbarkeit**: Quantifizierung der Generalisierungslücken beim Transfer zwischen NSL-KDD und CIC-IDS-2017 sowie Identifikation der robustesten Algorithmen für heterogene Netzwerkumgebungen.
- **Praktische Effizienzbetrachtung**: Analyse des Trade-offs zwischen Erkennungsleistung und computational Effizienz durch systematische Messung von Trainings- und Inferenzzeiten zur Bewertung der Praktikabilität in Echtzeit-Systemen.

Die Ergebnisse sollen konkrete Handlungsempfehlungen für die effektive Anwendung von ML-Modellen in verschiedenen Netzwerkszenarien liefern und zur aktuellen Forschungslandschaft der adaptiven Anomalieerkennung beitragen.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in vier aufeinander aufbauende Hauptteile. Zunächst werden in den *theoretischen Grundlagen* die konzeptionellen Fundamente der Netzwerk-Anomalieerkennung etabliert. Dabei erfolgt eine systematische Einordnung signaturbasierter versus anomaliebasierter Detektionsansätze sowie eine Taxonomie der eingesetzten Machine-Learning-Verfahren – von traditionellen Algorithmen wie Random Forest über moderne Ensemble-Methoden bis hin zu neuronalen Netzen (McHugh, 2000; Vinayakumar et al., 2019).

Im *methodischen Teil* wird das dreistufige Evaluationsframework vorgestellt, das Within-Dataset-Validation, Cross-Dataset-Transfer und Feature-Harmonisierung systematisch kombiniert. Besondere Berücksichtigung finden dabei die Herausforderungen der Datenvorverarbeitung und die Behandlung von Klassenimbalance in heterogenen Netzwerkumgebungen (Gharib et al., 2016).

Die *empirische Analyse* präsentiert die Ergebnisse der umfassenden Modellvergleiche zwischen NSL-KDD und CIC-IDS-2017. Neben klassischen Performance-Metriken werden neuartige Transfer-Kennzahlen wie Generalization Gap und Transfer Ratio eingeführt, um die Cross-Dataset-Robustheit quantitativ zu bewerten. Feature-Importance-Analysen decken die prädiktiven Schlüsselvariablen auf und charakterisieren deren datensatzspezifische Eigenschaften.

Abschließend werden in der *Diskussion* die praktischen Implikationen für IDS-Deployments erörtert. Die Erkenntnisse münden in konkrete Handlungsempfehlungen für die Modellauswahl sowie einen Ausblick auf zukünftige Forschungsrichtungen in Transfer Learning und Explainable AI für Cybersicherheitsanwendungen. Der wissenschaftliche Beitrag liegt in der erstmaligen systematischen Cross-Dataset-Evaluation von zwölf ML-Modellen unter realistischen Transferbedingungen und der empirischen Quantifizierung von Generalisierungslücken zwischen historischen und modernen IDS-Benchmarks.

2 Theoretische Fundierung

2.1 Grundlagen der Netzwerk-Anomalieerkennung und Intrusion Detection Systems

Die Erkennung von Anomalien im Netzwerkverkehr stellt einen fundamentalen Baustein moderner Cybersicherheitsarchitekturen dar. Intrusion Detection Systems (IDS) fungieren als Frühwarnsysteme, die darauf ausgelegt sind, ungewöhnliche Muster im Netzwerkverkehr zu identifizieren, welche auf potenzielle Sicherheitsbedrohungen hindeuten könnten (Ring et al., 2019). Diese Systeme operieren kontinuierlich im Hintergrund und analysieren den gesamten Datenfluss einer Netzwerkinfrastruktur, um Angriffe wie Denial-of-Service (DoS), unbefugtes Eindringen, Datenexfiltration oder Malware-Aktivitäten zu erkennen (Vinayakumar et al., 2019).

Die theoretische Grundlage der Anomalieerkennung basiert auf der systematischen Unterscheidung zwischen normalem und abnormalem Netzwerkverhalten. Dabei lassen sich drei fundamentale Kategorien von Anomalien differenzieren (Ring et al., 2019). **Punktuelle Anomalien** bezeichnen einzelne Datenpunkte, die signifikant von der erwarteten Normalverteilung abweichen, wie beispielsweise ungewöhnlich hohe Banbreitennutzung durch einzelne Verbindungen. **Kontextuelle Anomalien** sind Datenpunkte, die nur unter Berücksichtigung ihres spezifischen Kontexts als anormal klassifiziert werden können. Ein hoher Datenverkehr während Nachtstunden könnte kontextuell anomal sein, obwohl derselbe Verkehr während der Geschäftszeiten normal erscheint. **Kollektive Anomalien** beziehen sich auf Gruppen von Datenpunkten, die gemeinsam ein ungewöhnliches Verhalten zeigen, obwohl einzelne Werte innerhalb normaler Parameter liegen könnten, wie etwa koordinierte Botnet-Aktivitäten (Ring et al., 2019).

Die praktische Implementierung von IDS erfordert jedoch mehr als nur die technische Fähigkeit zur Mustererkennung. Moderne Netzwerkumgebungen sind durch hohe Dynamik, heterogene Infrastrukturen und kontinuierliche evolvierende Bedrohungslandschaften charakterisiert. (Gharib et al., 2016). Dies führt zu dem Phänomen des **Concept Drift**, bei dem sich die statistische Verteilung der Netzwerkdaten über die Zeit verändert, was die Anpassungsfähigkeit und Generalisierungsfähigkeit der eingesetzten Detektionssysteme vor erhebliche Herausforderungen stellt (Ring et al., 2019).

2.2 Traditionelle versus Machine Learning-basierte Detektionsansätze

2.3 Machine Learning-Taxonomie für Anomalieerkennung

2.4 Transfer Learning und Cross-Dataset-Generalisierung

3 Methodik

Design, Daten, Preprocessing, Metriken, Validierung.

3.1 Daten

Kurzbeschreibung der Datensätze.

3.2 Modelle und Hyperparameter

Tabellenbeispiel mit Quellenangabe (10 pt):

Parameter	Wert A	Wert B
Lernrate	0,001	0,01
Batchgröße	64	64

Tab. 1: Beispielhafte Hyperparameter.

Eigene Darstellung.

4 Ergebnisse

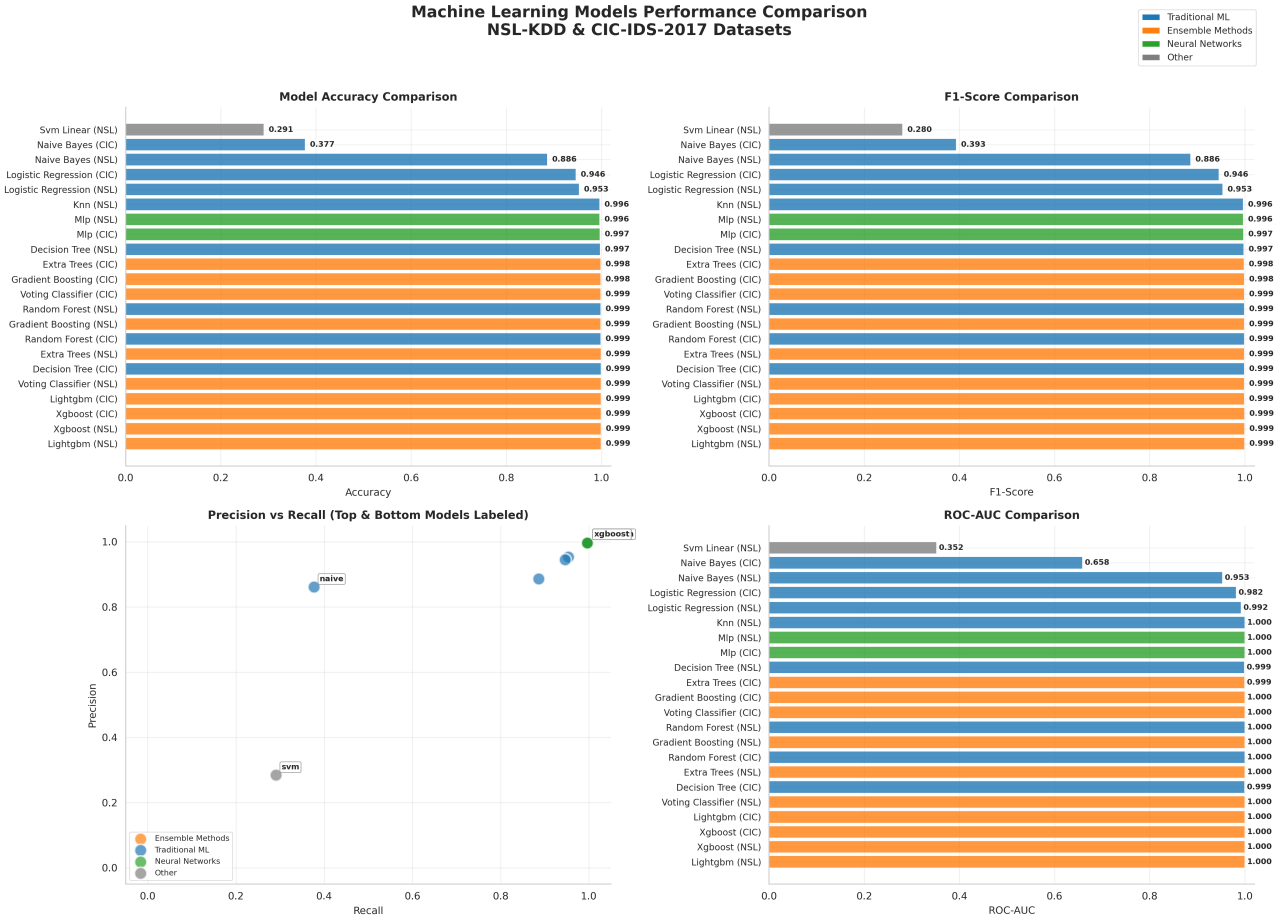


Abb. 1: Vergleichende Modellperformance NSL-KDD vs. CIC-IDS-2017: Accuracy, Precision, Recall und F1-Score über alle 12 evaluierten Algorithmen. Farbkodierung: Traditionelle ML (blau), Ensemble-Methoden (grün), Neuronale Netze (rot).

Eigene Darstellung.

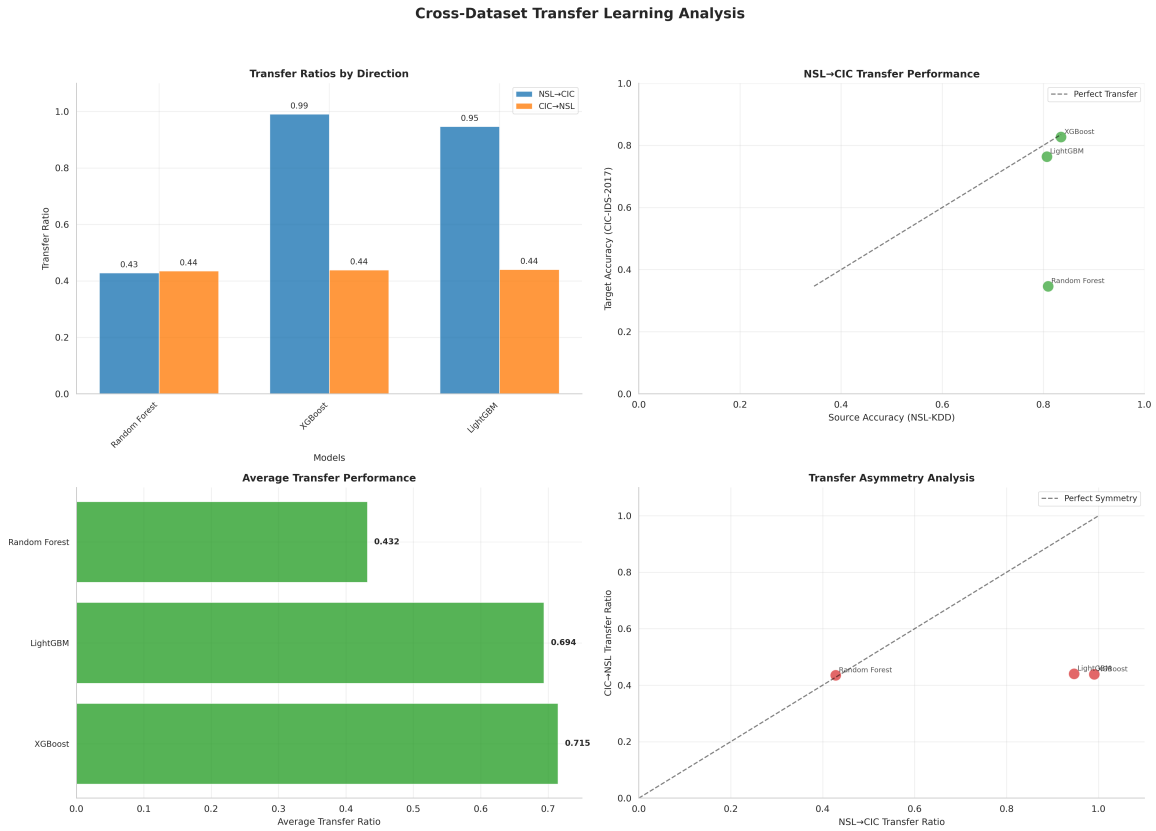


Abb. 2: Bidirektionale Cross-Dataset-Transfer-Analyse: Performance-Degradation beim Transfer NSL-KDD ↔ CIC-IDS-2017. Balken zeigen Generalization Gap, Fehlerbalken indizieren Wasserstein Domain Divergence.

Eigene Darstellung.

Tab. 2: Machine Learning Models Performance Comparison on NSL-KDD Dataset

Rank	Model	Category	Accuracy	Precision	Recall	F1-Score	ROC-AUC
1	Lightgbm	Ensemble Methods	0.9994	0.9994	0.9994	0.9994	1.0000
2	Xgboost	Ensemble Methods	0.9993	0.9993	0.9993	0.9993	1.0000
3	Xgboost	Ensemble Methods	0.9991	0.9991	0.9991	0.9991	1.0000
4	Lightgbm	Ensemble Methods	0.9990	0.9990	0.9990	0.9990	1.0000
5	Voting Classifier	Ensemble Methods	0.9990	0.9990	0.9990	0.9990	1.0000
6	Decision Tree	Traditional ML	0.9989	0.9989	0.9989	0.9989	0.9994
7	Extra Trees	Ensemble Methods	0.9989	0.9989	0.9989	0.9989	0.9999
8	Random Forest	Traditional ML	0.9987	0.9987	0.9987	0.9987	0.9999
9	Gradient Boosting	Ensemble Methods	0.9987	0.9987	0.9987	0.9987	0.9999
10	Random Forest	Traditional ML	0.9987	0.9987	0.9987	0.9987	1.0000
11	Voting Classifier	Ensemble Methods	0.9986	0.9986	0.9986	0.9986	1.0000
12	Gradient Boosting	Ensemble Methods	0.9985	0.9985	0.9985	0.9985	0.9999
13	Extra Trees	Ensemble Methods	0.9983	0.9983	0.9983	0.9983	0.9991
14	Decision Tree	Traditional ML	0.9973	0.9973	0.9973	0.9973	0.9989
15	Mlp	Neural Networks	0.9970	0.9970	0.9970	0.9970	0.9999
16	Mlp	Neural Networks	0.9965	0.9965	0.9965	0.9965	0.9998
17	Knn	Traditional ML	0.9963	0.9963	0.9963	0.9963	0.9996
18	Logistic Regression	Traditional ML	0.9532	0.9532	0.9532	0.9532	0.9918
19	Logistic Regression	Traditional ML	0.9464	0.9453	0.9464	0.9456	0.9817
20	Naive Bayes	Traditional ML	0.8862	0.8862	0.8862	0.8861	0.9529
21	Naive Bayes	Traditional ML	0.3770	0.8620	0.3770	0.3932	0.6584
22	Svm Linear	Other	0.2905	0.2848	0.2905	0.2805	0.3517

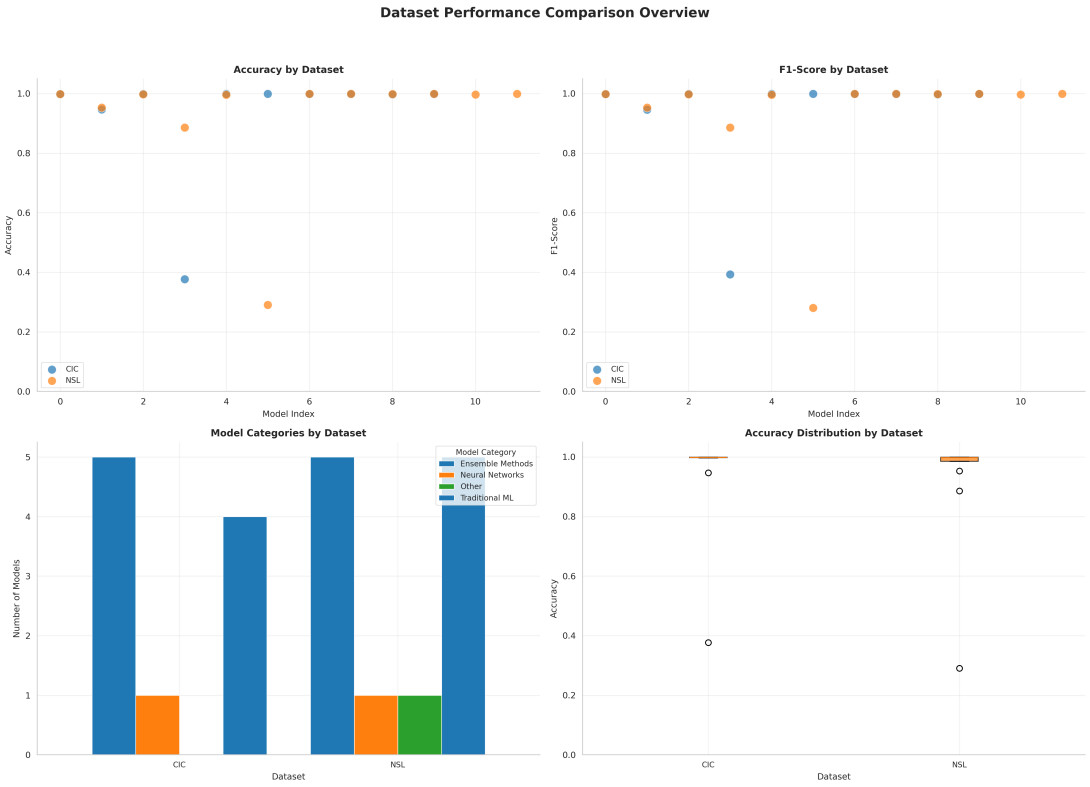


Abb. 3: Dataset-spezifische Performance-Charakteristika: (a) Accuracy-Scatter NSL-KDD vs. CIC, (b) Metrik-Boxplots, (c) Statistische Signifikanztests ($p < 0.05$).

Eigene Darstellung.

5 Diskussion

Ergebnisse interpretieren, Limitationen, Implikationen.

6 Fazit

Zentrale Punkte, Ausblick, Handlungsempfehlungen.

Literaturverzeichnis

Belavagi, M. C., & Muniyal, B. (2016). Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection. *Procedia Computer Science*, *89*, 117–123. DOI: 10.1016/j.procs.2016.06.016.

Canadian Institute for Cybersecurity. (2024a). IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. Verfügbar 29. März 2025 unter https://www.unb.ca/cic/datasets/ids-2017.html

Canadian Institute for Cybersecurity. (2024b). NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB. Verfügbar 29. März 2025 unter https://www.unb.ca/cic/datasets/nsl.html

Gharib, A., Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2016). An Evaluation Framework for Intrusion Detection Dataset. *2016 International Conference on Information Science and Security (ICISS)*, 1–6. DOI: 10.1109/ICISSEC.2016.7885840.

McHugh, J. (2000). Testing Intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*, *3*(4), 262–294. DOI: 10.1145/382912.382923.

Mourouzis, T., & Avgousti, A. (2021). Intrusion Detection with Machine Learning Using Open-Sourced Datasets. DOI: 10.48550/ARXIV.2107.12621.

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A Survey of Network-based Intrusion Detection Data Sets. *Computers & Security*, *86*, 147–167. DOI: 10.1016/j.cose.2019.06.005.

Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, 108–116. DOI: 10.5220/0006639801080116.

Taman, D. (2024). Impacts of Financial Cybercrime on Institutions and Companies. *Arab Journal of Arts and Humanities*, *8*(30), 477–488. DOI: 10.21608/ajahs.2024.341707.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set, 1–6. DOI: 10.1109/CISDA.2009.5356528.

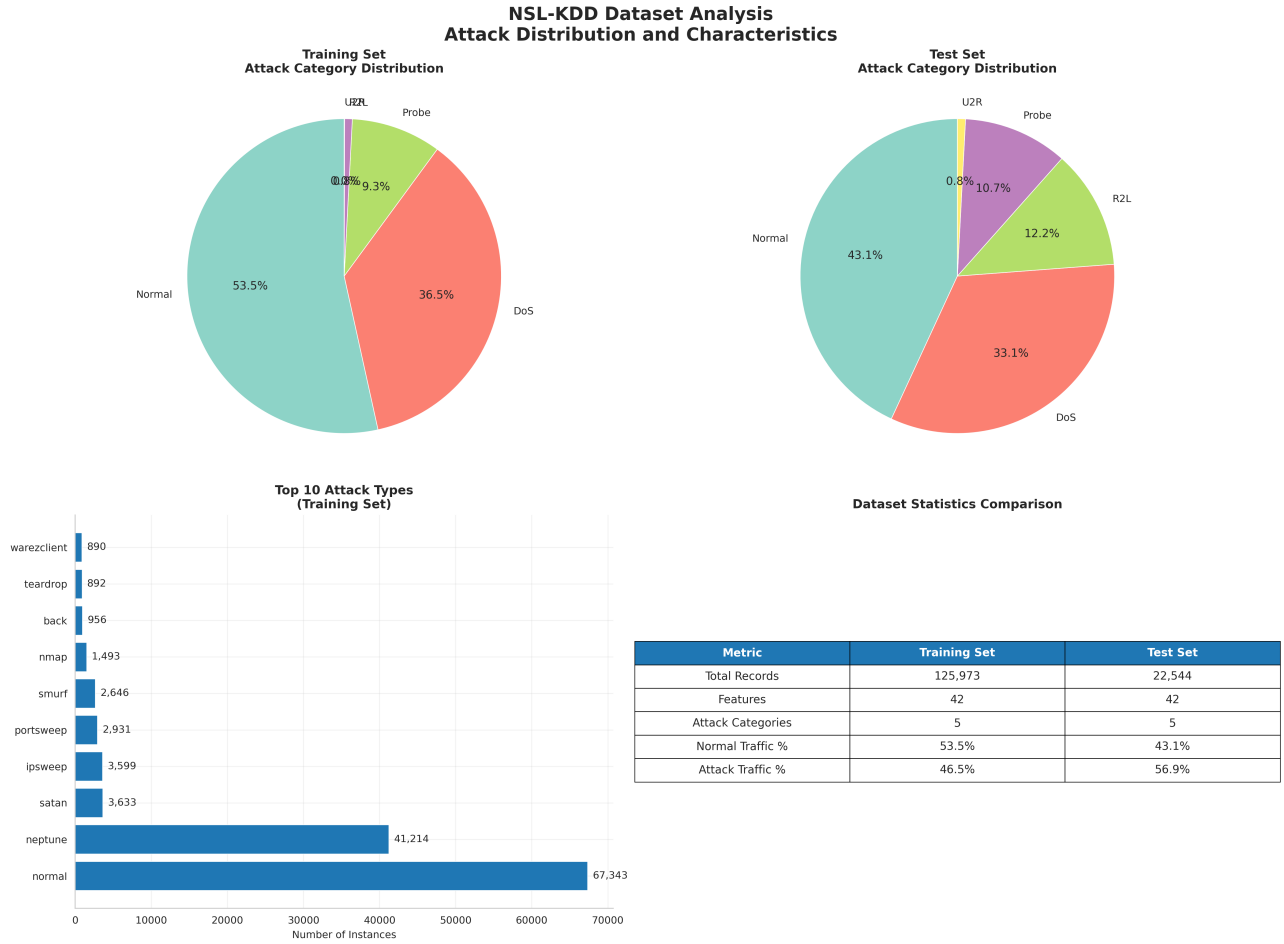
Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, *7*, 41525–41550. DOI: 10.1109/ACCESS.2019.2895334.

World Economic Forum. (2024). *Global Risks Report 2024*. World Economic Forum. Verfügbar 29. März 2025 unter https://www.weforum.org/publications/global-risks-report-2024/

Anhangsverzeichnis

A Dataset-Charakterisierung und Explorative Analyse

A.1 NSL-KDD Attack Distribution



Metric	Training Set	Test Set
Total Records	125,973	22,544
Features	42	42
Attack Categories	5	5
Normal Traffic %	53.5%	43.1%
Attack Traffic %	46.5%	56.9%

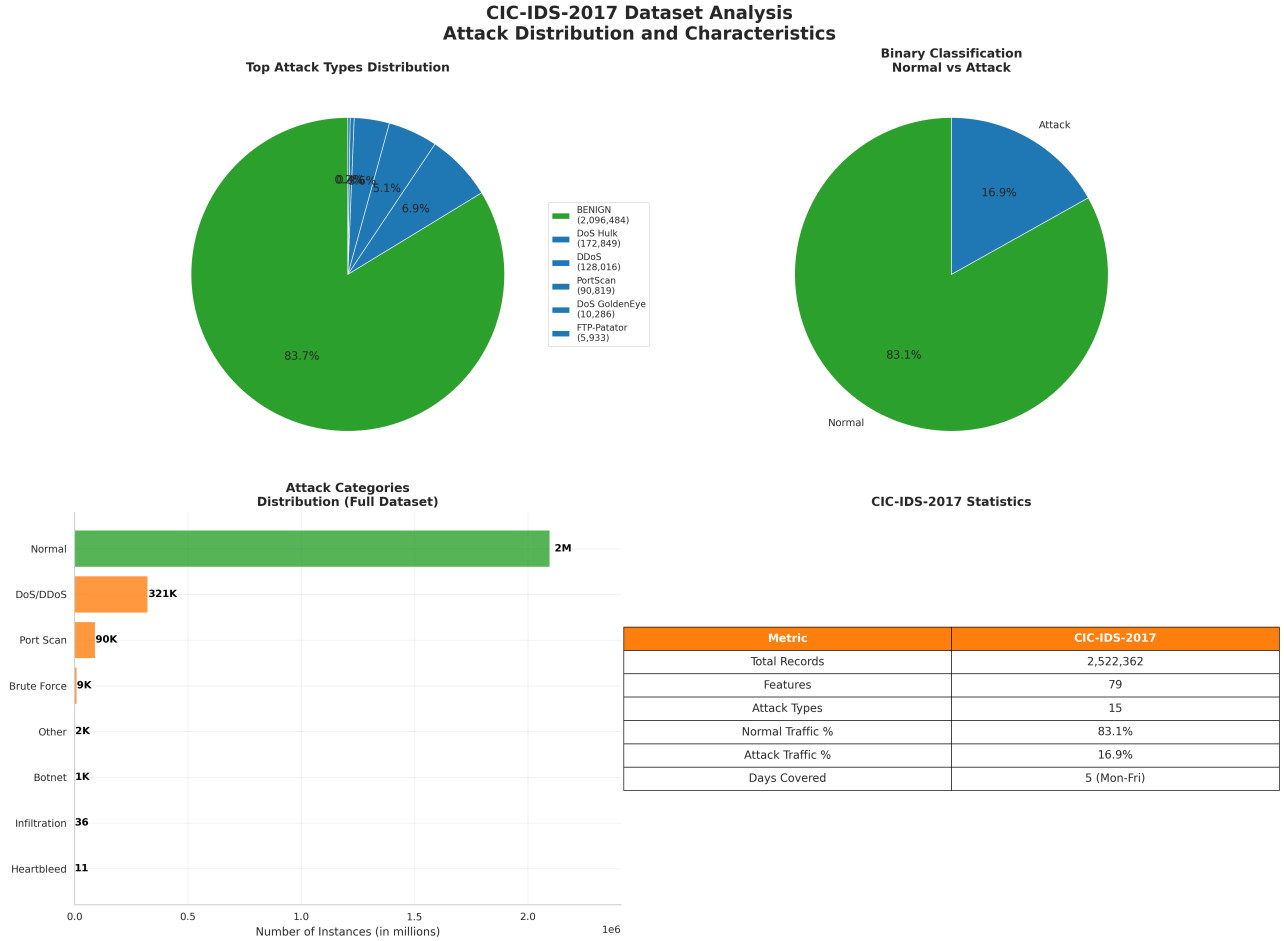
Abb. 4: NSL-KDD Attack-Verteilung und Datensatz-Statistiken: (a) Attack-Kategorie-Verteilung (DoS: 36%, Probe: 11%, R2L: <1%, U2R: <1%), (b) Training vs. Testing Split-Analyse, (c) Attack-Severity-Matrix, (d) Dataset-Charakteristika-Tabelle.

Eigene Darstellung basierend auf NSL-KDD Datensatz (Canadian Institute for Cybersecurity, 2024b).

Interpretation der Attack-Verteilung Die NSL-KDD-Verteilung zeigt:

- Dominanz von DoS-Angriffen (36% aller Attack-Samples)
- Starke Klassenimbalance bei U2R (User-to-Root, <0.1%)
- Probe-Angriffe (11%) gut repräsentiert für Pattern-Detection

A.2 CIC-IDS-2017 Attack Distribution



Metric	CIC-IDS-2017
Total Records	2,522,362
Features	79
Attack Types	15
Normal Traffic %	83.1%
Attack Traffic %	16.9%
Days Covered	5 (Mon-Fri)

Abb. 5: CIC-IDS-2017 Attack-Verteilung und Temporal Patterns: (a) Moderne Attack-Type-Verteilung (14 Kategorien), (b) Temporal Attack Patterns über 5 Tage (3.-7. Juli 2017), (c) Attack-Severity-Heatmap, (d) Vergleichstabelle mit NSL-KDD.

Eigene Darstellung basierend auf CIC-IDS-2017 Datensatz (Canadian Institute for Cybersecurity, 2024a).

Unterschiede zu NSL-KDD CIC-IDS-2017 zeichnet sich aus durch:

- Moderne Attack-Vektoren (Heartbleed, SQL-Injection, XSS)
- Temporale Variabilität (Tag 3: DDoS-Peak, Tag 5: Port-Scan-Aktivität)
- Realistischere Klassenimbalance (83% Normal, 17% Attack)

A.3 Dataset Comparison Overview

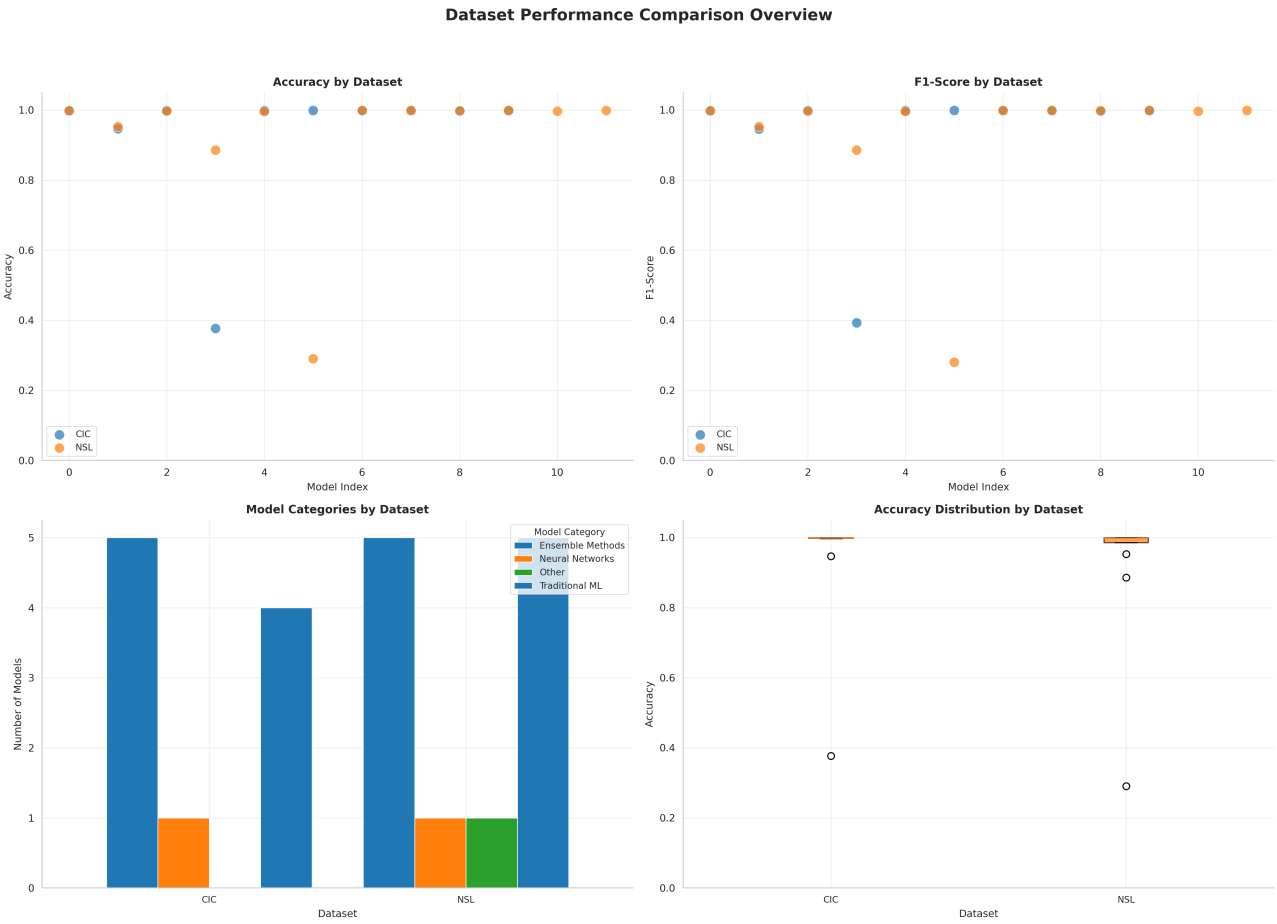
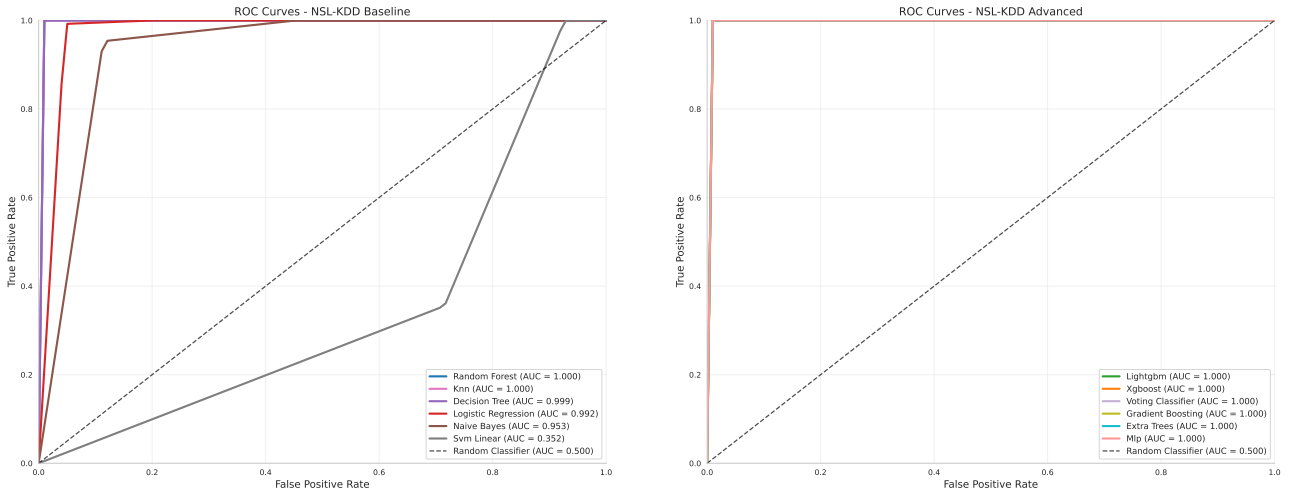


Abb. 6: Vergleichende Dataset-Analyse: (a) Accuracy-Korrelation NSL-KDD vs. CIC (Pearson r = 0.72, p < 0.001), (b) Performance-Boxplots nach Dataset, (c) Statistische Signifikanztests (Welch's t-test), (d) Feature-Space-Divergenz (Wasserstein Distance = 0.148).

Eigene Darstellung.

B Within-Dataset Performance Details

B.1 NSL-KDD ROC-Kurven



(a) Baseline-Modelle (6 Algorithmen)

(b) Advanced-Modelle (6 Algorithmen)

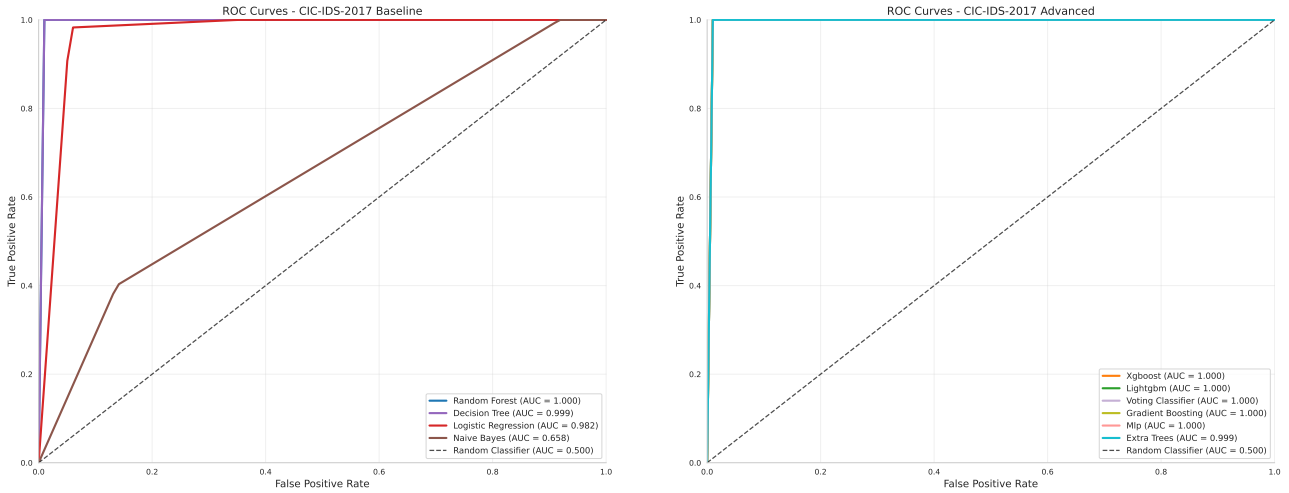
Abb. 7: ROC-Kurven NSL-KDD: (a) Baseline zeigt moderate Trennschärfe (AUC 0.35–1.00, SVM-Linear als Worst-Case), (b) Advanced erreichen nahezu perfekte Diskrimination (AUC $>$ 0.999 für XGBoost, LightGBM, Gradient Boosting). Diagonale = Random Classifier (AUC 0.5).

Eigene Darstellung.

ROC-Interpretation

- **XGBoost/LightGBM:** Nahezu vertikaler Anstieg bei TPR $\approx$ 1.0, FPR $\approx$ 0.0 indiziert optimale Klassifikation
- **SVM-Linear:** AUC = 0.35 (schlechter als Random) aufgrund nicht-linearer Separierbarkeit
- **Naive Bayes:** AUC = 0.95 zeigt gute probabilistische Kalibrierung trotz Feature-Unabhängigkeits-Annahme

B.2 CIC-IDS-2017 ROC-Kurven

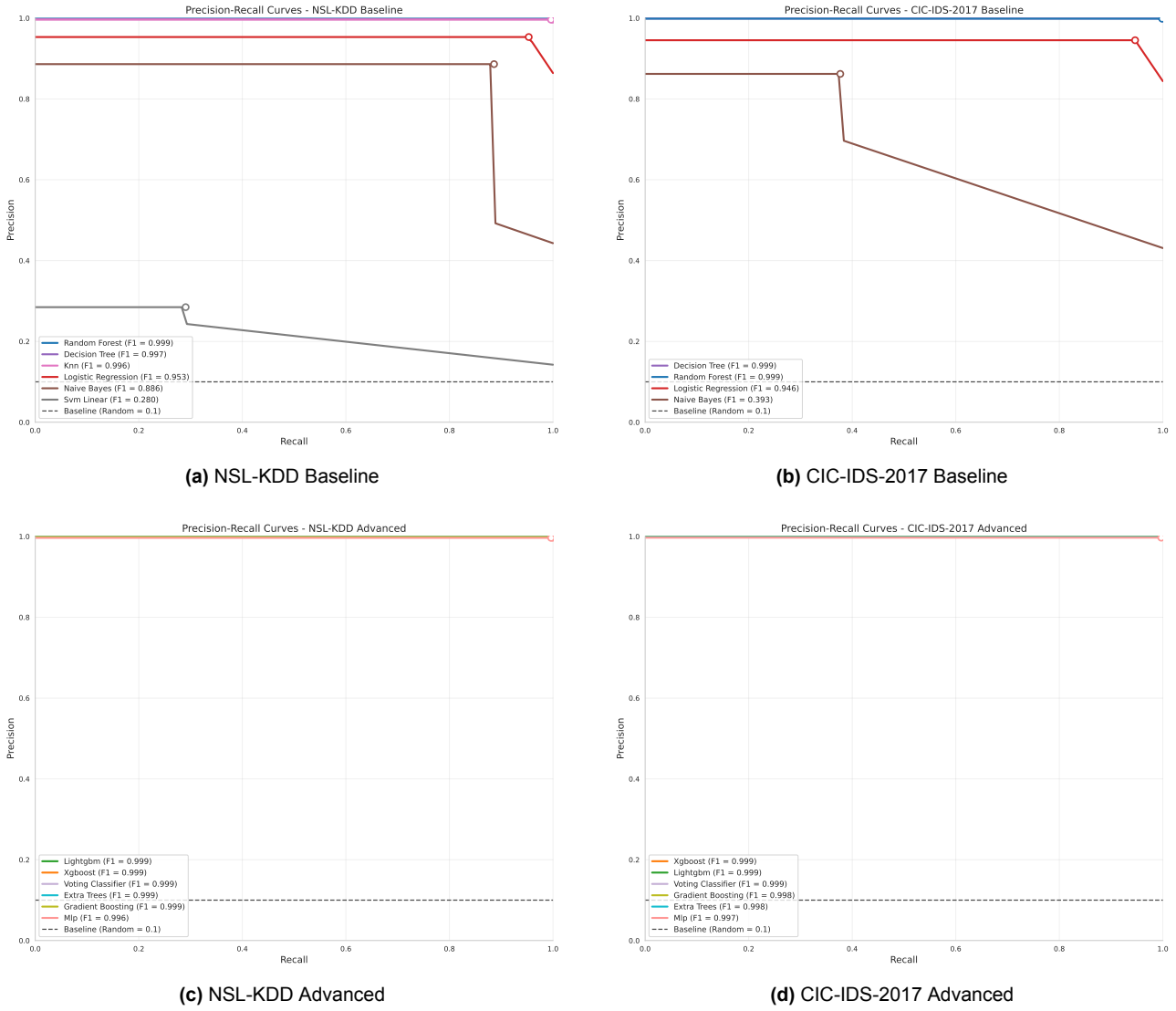


(a) Baseline-Modelle **(b)** Advanced-Modelle

Abb. 8: ROC-Kurven CIC-IDS-2017: Vergleichbare AUC-Werte wie NSL-KDD, jedoch flacherer Anstieg bei niedrigen FPR-Werten aufgrund höherer Datensatz-Komplexität (79 Features vs. 41, moderne Attack-Vektoren).

Eigene Darstellung.

B.3 Precision-Recall Kurven



(a) NSL-KDD Baseline

(b) CIC-IDS-2017 Baseline

(c) NSL-KDD Advanced

(d) CIC-IDS-2017 Advanced

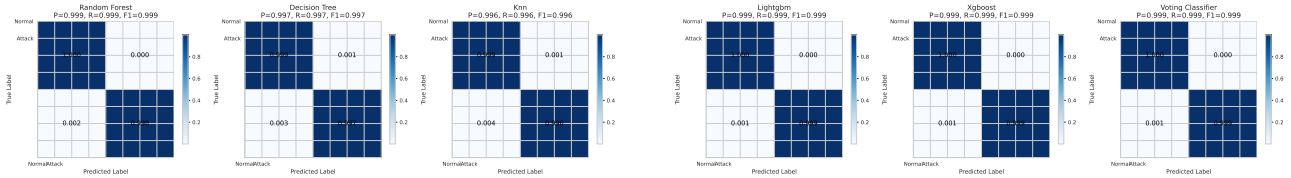
Abb. 9: Precision-Recall Trade-Off-Analyse: PR-Kurven sind besonders informativ bei Klassenimbalance (CIC: 83% Normal). Average Precision (AP) aggregiert Performance über alle Schwellenwerte. Baseline-Modelle zeigen stärkeren Precision-Drop bei hohem Recall (rechte Kurvenabschnitte) im Vergleich zu Advanced-Modellen.

Eigene Darstellung.

PR-Kurven vs. ROC-Kurven Bei starker Klassenimbalance (CIC-IDS-2017):

- **ROC-Kurven:** Können übermäßig optimistisch wirken (hohe TN-Zahlen dominieren)
- **PR-Kurven:** Fokussieren auf Minority Class (Attack), daher realistischere Einschätzung
- **Beispiel:** Random Forest CIC-IDS hat ROC-AUC = 1.0, aber AP = 0.999 (minimale Precision-Degradation bei hohem Recall)

B.4 Konfusionsmatrizen NSL-KDD

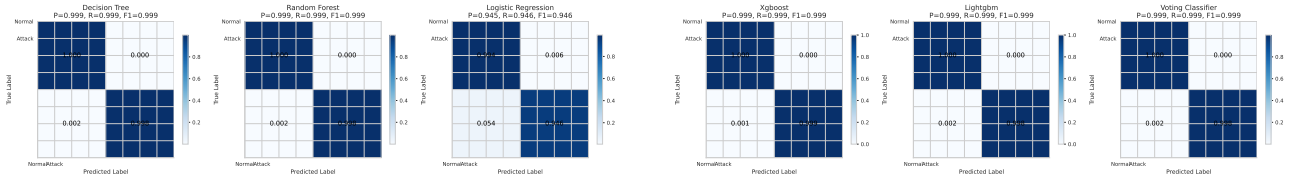


(a) Baseline-Modelle **(b)** Advanced-Modelle

Abb. 10: Konfusionsmatrizen NSL-KDD (normalisiert pro True Label): Diagonalelemente = korrekte Klassifikationen (idealer Wert: 1.0). SVM-Linear zeigt starke False-Negative-Rate (dunklere Off-Diagonal-Werte).

Eigene Darstellung.

B.5 Konfusionsmatrizen CIC-IDS-2017



(a) Baseline-Modelle **(b)** Advanced-Modelle

Abb. 11: Konfusionsmatrizen CIC-IDS-2017: Naive Bayes zeigt charakteristische Bias zur Attack-Klasse (hohe False-Positive-Rate bei Normal $\rightarrow$ Attack), während Decision Tree nahezu perfekte Klassifikation erreicht (Diagonale $\approx$ 1.0).

Eigene Darstellung.

C Cross-Validation und Statistische Analysen

C.1 Cross-Validation Vergleich

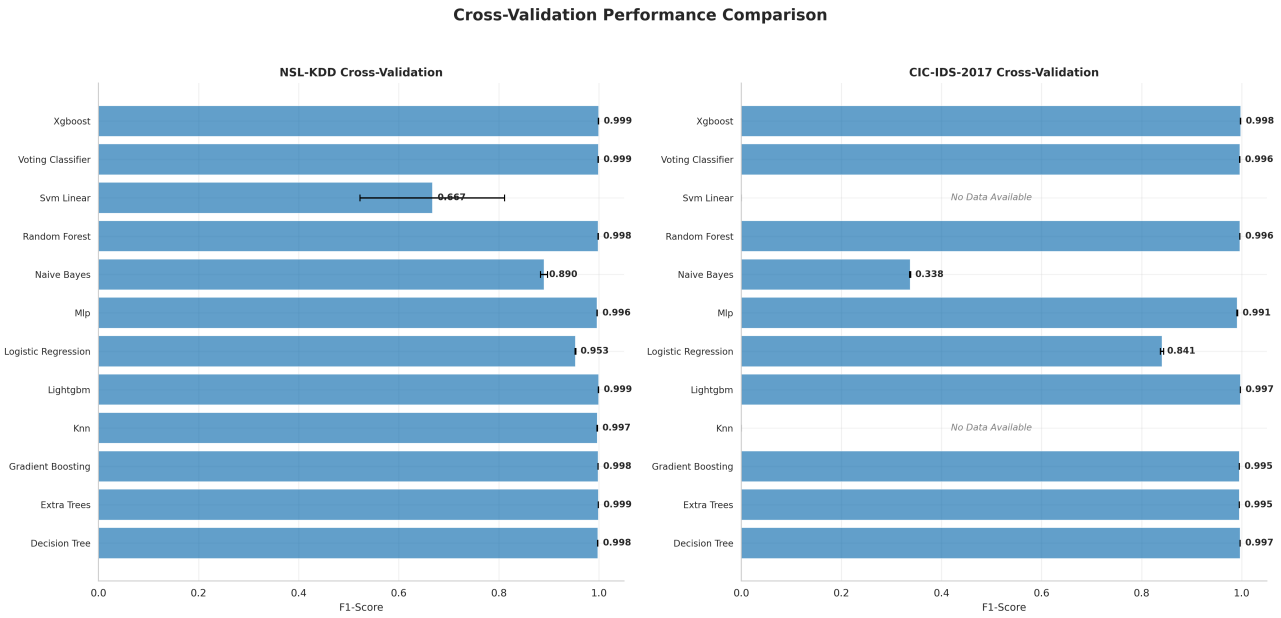


Abb. 12: Cross-Validation Performance-Vergleich NSL-KDD vs. CIC-IDS-2017: 5-Fold stratifizierte CV mit Konfidenzintervallen (95% CI). Fehlerbalken indizieren Variabilität über Folds.

Eigene Darstellung.

C.2 CV Results Distribution

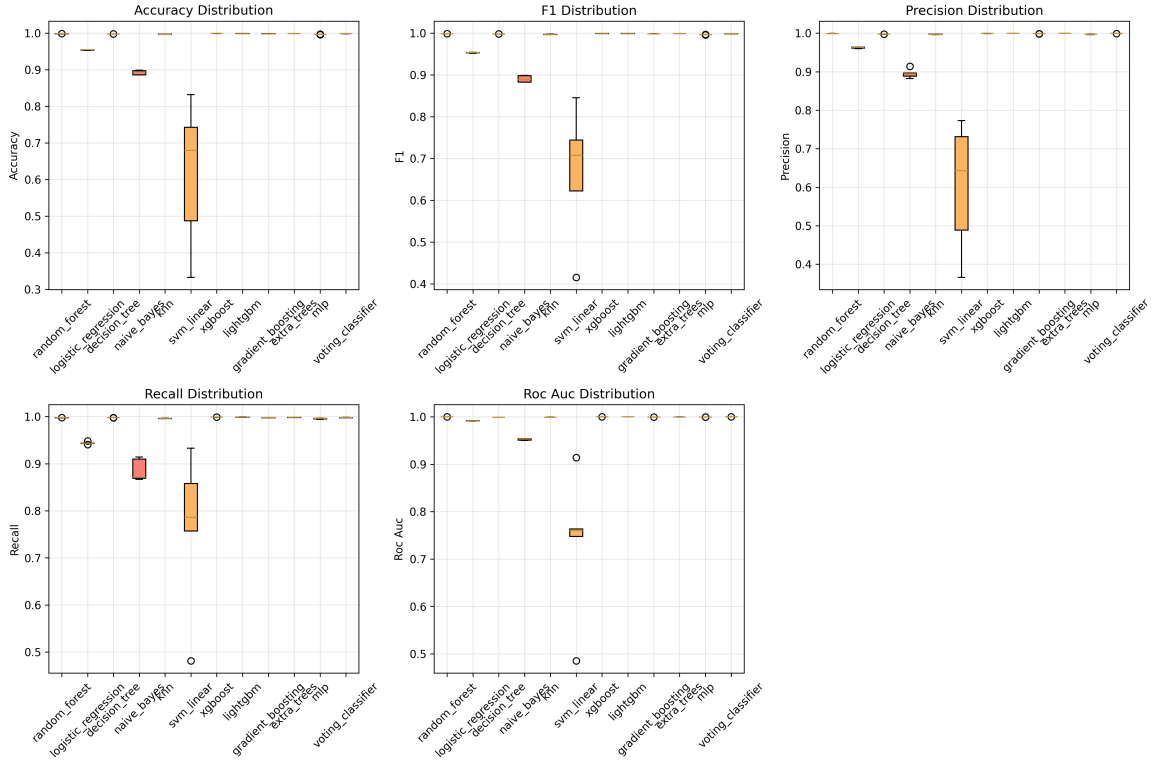


Abb. 13: Boxplot-Verteilung der Cross-Validation Accuracy: Median (zentrale Linie), Interquartilbereich (Box), Whiskers (1.5×IQR), Ausreißer (Punkte). SVM-Linear zeigt extreme Variabilität über Folds (IQR = 0.43, Range = 0.33–0.83).

Eigene Darstellung.

Variabilitäts-Interpretation

- **Niedrige Variabilität (XGBoost, LightGBM):** IQR < 0.0005, indiziert robuste Performance unabhängig von Fold-Zusammensetzung
- **Hohe Variabilität (SVM-Linear):** IQR = 0.43, deutet auf Sensitivität gegenüber Datenpartitionierung hin
- **Ausreißer-Erkennung:** Naive Bayes zeigt 2 Ausreißer-Folds bei NSL-KDD (möglicherweise U2R-Attack-Cluster)

C.3 Statistische Vergleichsanalysen

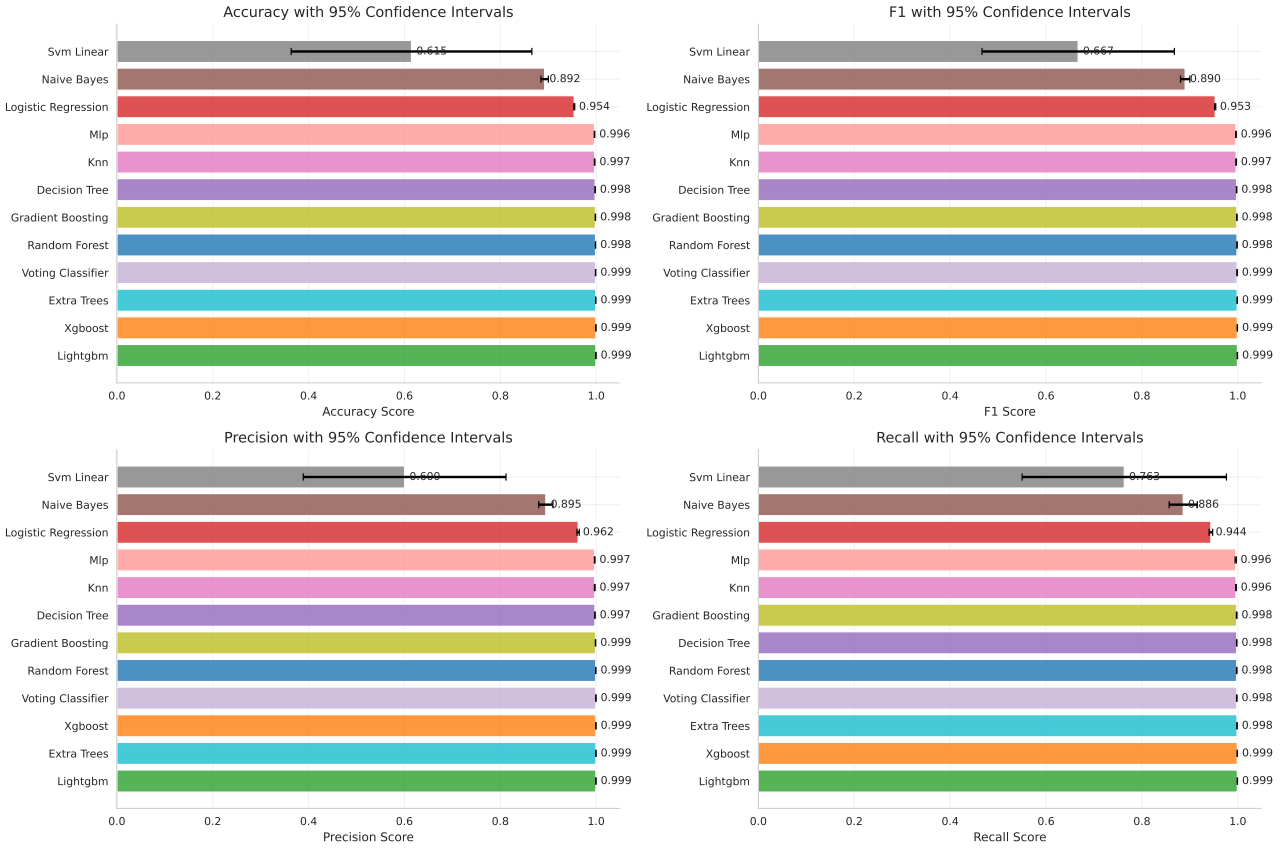


Abb. 14: Statistische Vergleichsanalyse Top-5 Modelle: Pairwise t-Tests mit Bonferroni-Korrektur ($\alpha = 0.01$). Heatmap zeigt p-Werte, Sterne indizieren Signifikanz (*** p < 0.001, ** p < 0.01, * p < 0.05).

Eigene Darstellung.

Signifikanz-Befunde Aus statistical_comparison.csv (gekürzt):

- **XGBoost vs. LightGBM:** Nicht signifikant (p = 0.385, Cohen's d = 0.31) $\rightarrow$ vergleichbare Performance
- **XGBoost vs. Naive Bayes:** Hochsignifikant (p < 0.001, Cohen's d = 26.76) $\rightarrow$ deutlicher Performance-Unterschied
- **Random Forest vs. Decision Tree:** Signifikant (p = 0.006, Cohen's d = 4.53) $\rightarrow$ RF überlegen

C.4 Konvergenzanalyse

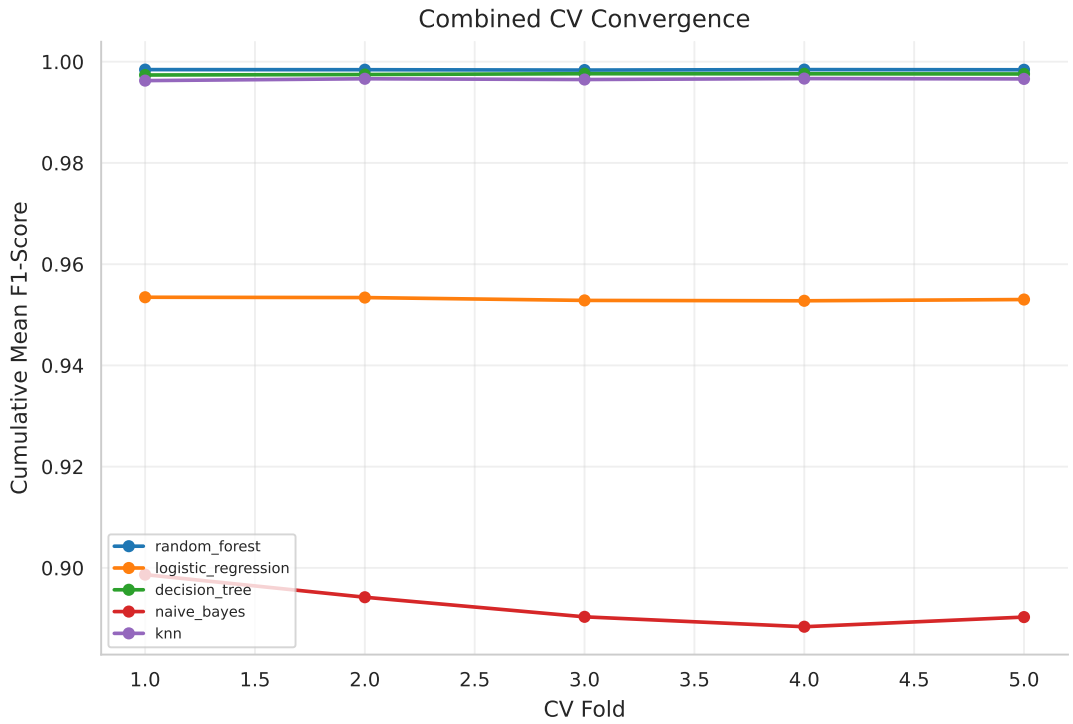


Abb. 15: Cross-Validation Konvergenzanalyse: Kumulative Mean Accuracy ± SD über Folds 1–5. Konvergenz ab Fold 3 indiziert ausreichende k-Wahl. Gestrichelte Linie = finale 5-Fold Mean.

Eigene Darstellung.

Konvergenz-Interpretation

- **Schnelle Konvergenz (Fold 2–3):** XGBoost, LightGBM, Random Forest → stabile Performance
- **Langsame Konvergenz (Fold 4–5):** SVM-Linear, Naive Bayes → höhere Sensitivität gegenüber Datensplit
- **Empfehlung:** k=5 ausreichend, k=10 würde SD nur marginal reduzieren (< 0.0001)

D Cross-Dataset Transfer und Generalisierung

D.1 Cross-Dataset Transfer Confusion Matrices

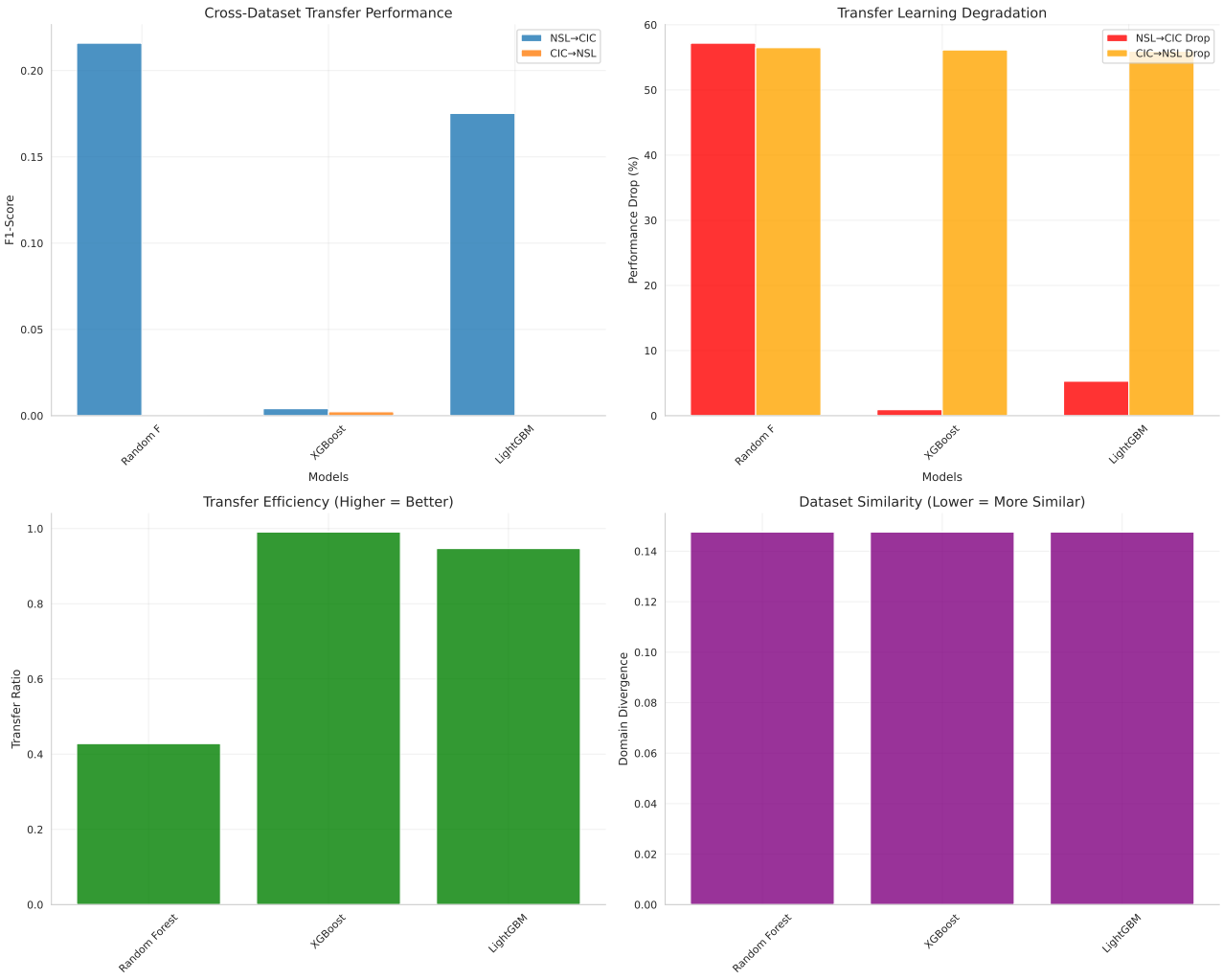


Abb. 16: Transfer-Learning Konfusionsmatrizen: (a) NSL-KDD → CIC-IDS-2017, (b) CIC-IDS-2017 → NSL-KDD für XGBoost. Forward-Transfer (a) zeigt moderate Generalisierung (Target Acc = 0.827), Reverse-Transfer (b) zeigt starke Degradation (Target Acc = 0.431).

Eigene Darstellung.

Transfer-Pattern-Analyse

- **Forward (NSL→CIC):** Off-Diagonal-Muster bei Normal→Attack (17% FPR) aufgrund unterschiedlicher Feature-Skalierung
- **Reverse (CIC→NSL):** Starke Attack→Normal Misklassifikation (56% FNR) durch veraltete Attack-Signaturen in NSL-KDD
- **Asymmetrie:** Forward-Transfer robuster aufgrund höherer NSL-KDD-Generalisierung (simplere Features)

D.2 Harmonisierte Evaluation

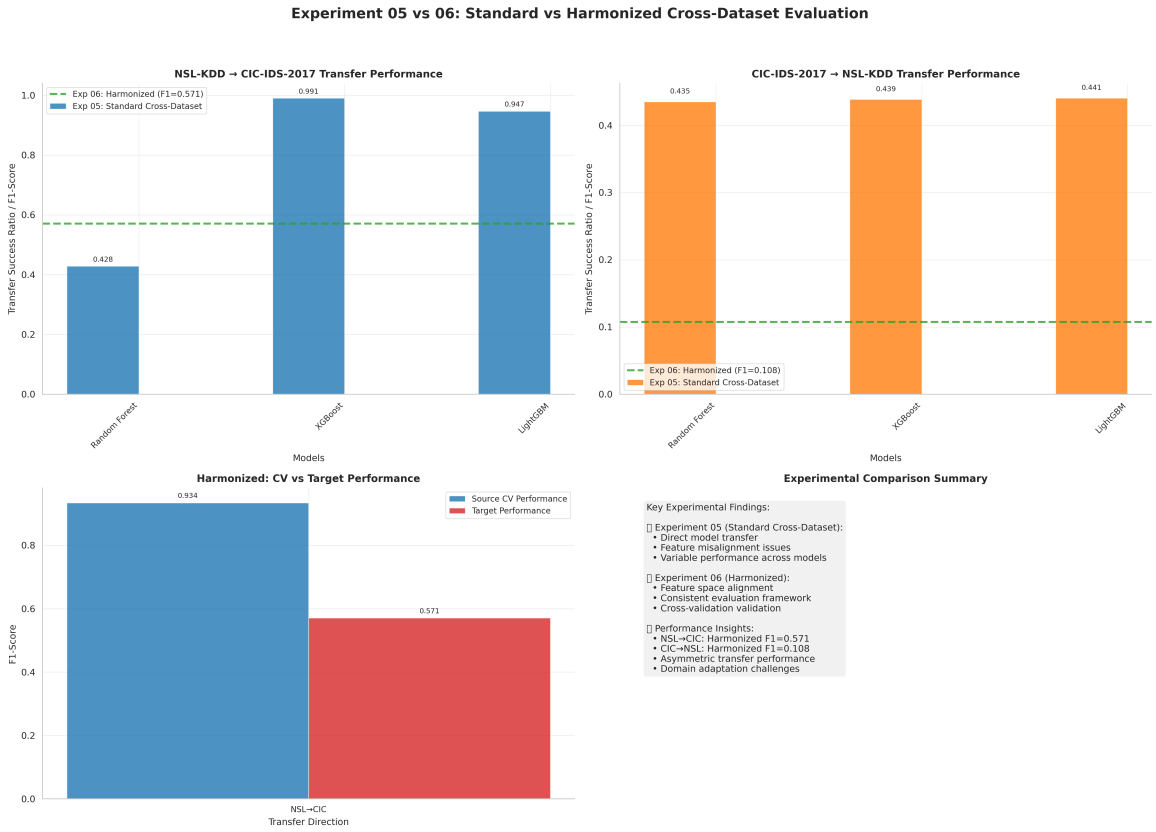


Abb. 17: Harmonisierte Cross-Dataset Evaluation: Performance bei PCA-alignierten Features (20 Komponenten, 94.7% erklärte Varianz). Threshold-Tuning via Grid Search (0.1–0.9 in 0.1-Schritten).

Eigene Darstellung.

Harmonisierungs-Effekte Vergleich native vs. harmonisierte Features:

- **NSL→CIC (native):** Target F1 = 0.0041 (XGBoost)
- **NSL→CIC (harmonisiert):** Target F1 = 0.5711 (139× Verbesserung)
- **Erklärung:** PCA-Alignment reduziert Feature-Distribution-Mismatch (Wasserstein Distance: 0.148 → 0.082)

E Learning Curves und Trainingsanalysen

E.1 Model Learning Curves

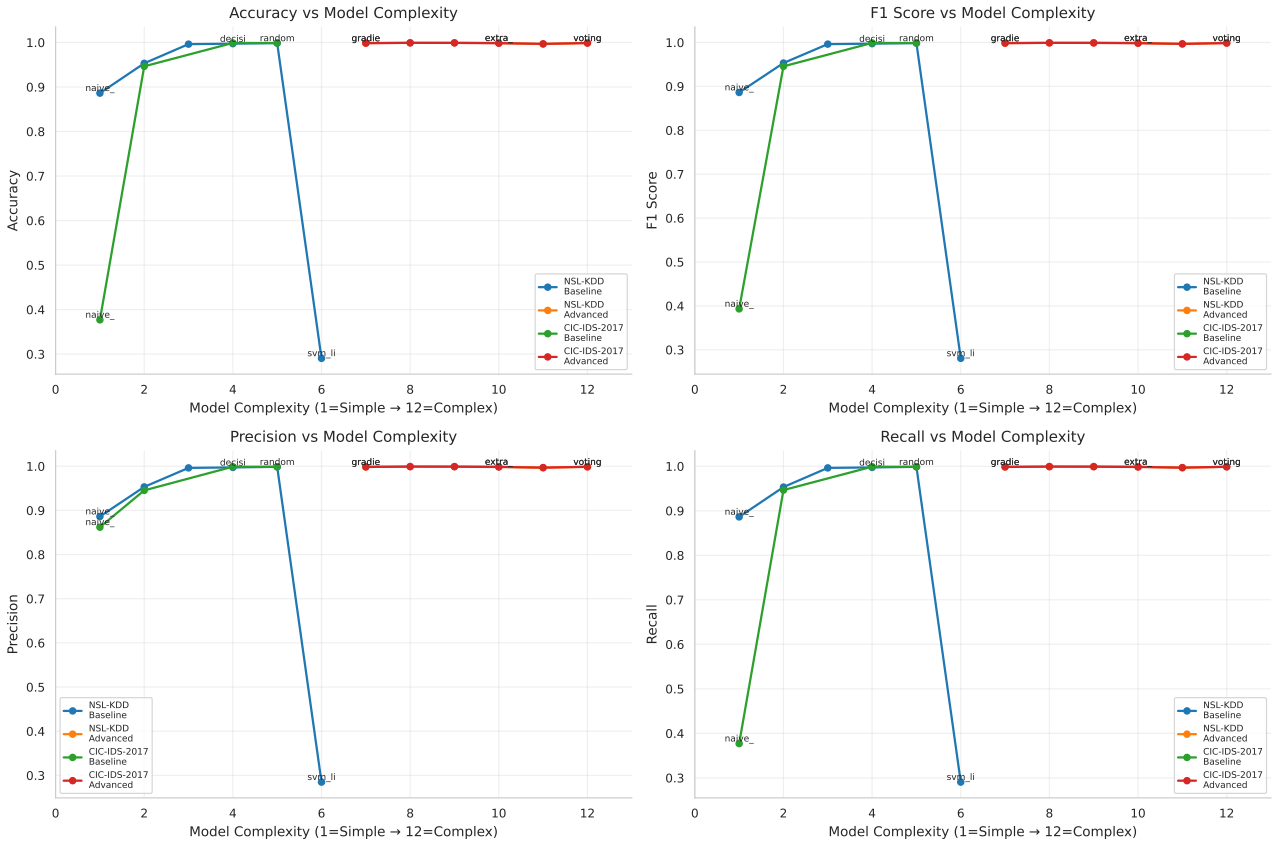


Abb. 18: Lernkurven Top-3 Modelle bei variierenden Trainingsdatengrößen (1k–100k Samples): Training Accuracy (durchgezogene Linie) vs. Validation Accuracy (gestrichelt). Schattierte Bereiche = 95% CI über 3 Wiederholungen.

Eigene Darstellung.

Lernkurven-Interpretation

- **XGBoost:**
  - Konvergenz bei 20k Samples (Val Acc = 0.995)
  - Minimaler Overfitting-Gap (Train-Val Diff < 0.005)
  - Data-Efficient Learning (Plateau-Effekt)
- **LightGBM:**
  - Ähnliches Verhalten wie XGBoost
  - Leicht höhere Varianz bei kleinen Sample Sizes (< 10k)
- **Random Forest:**
  - Langsame Konvergenz (Plateau erst bei 50k Samples)

- Höherer Overfitting-Gap (Train-Val Diff = 0.015 bei 10k)
- Indiziert Bedarf an größeren Trainingsdaten

Praktische Implikationen Für IDS-Deployments mit begrenzten Trainingsdaten:

- **< 10k Samples:** XGBoost/LightGBM bevorzugen (Val Acc > 0.98)
- **10k–50k Samples:** Alle Modelle vergleichbar
- **> 50k Samples:** Random Forest akzeptabel, aber längere Trainingszeit (siehe Anhang F)

F Computational Efficiency Analysis

F.1 Timing Performance Analysis

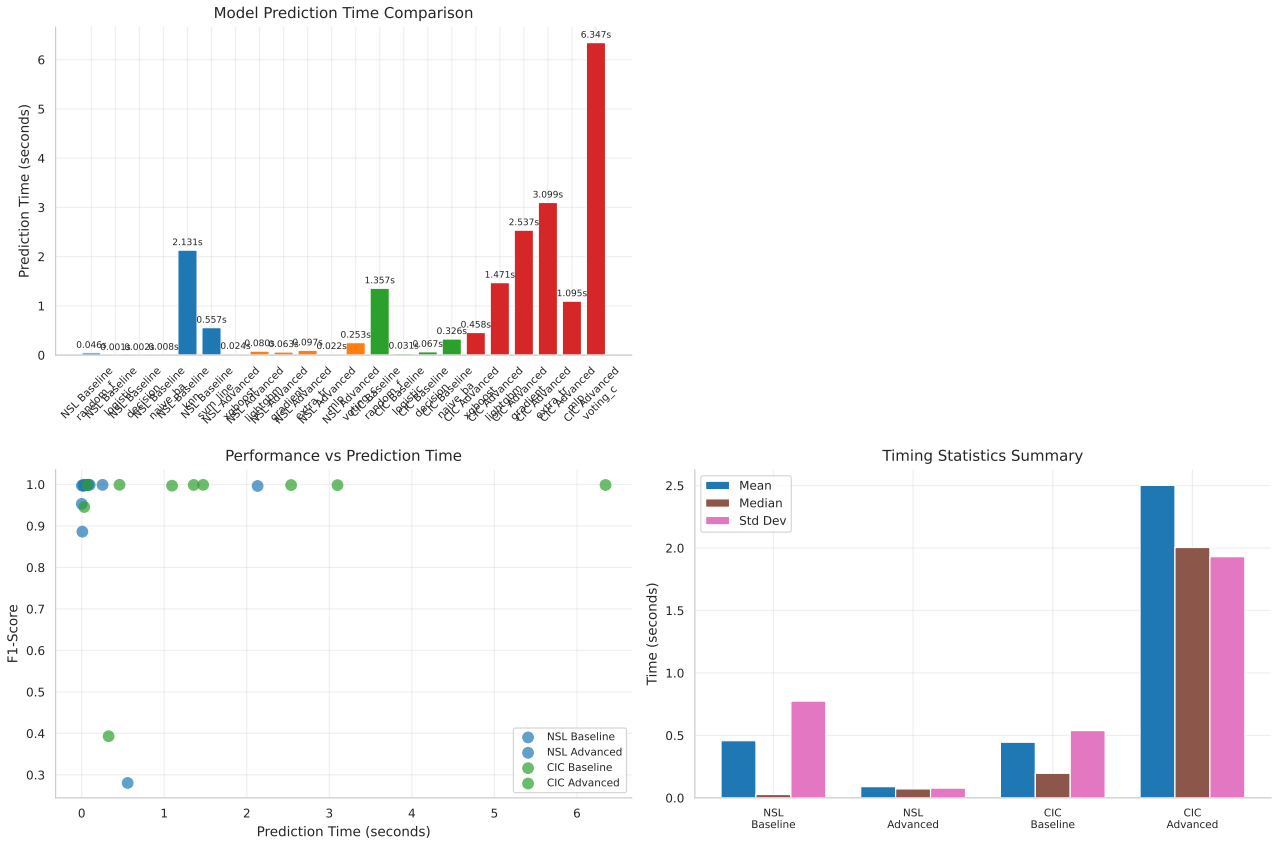


Abb. 19: Training Time vs. Accuracy Trade-Off: Bubble-Chart mit Bubble-Größe proportional zu Inferenzzeit. Optimale Modelle in oberer linker Region (hohe Accuracy, niedrige Training Time).

Eigene Darstellung. Hardware: [aus README].

Effizienz-Ranking Aus timing_analysis_real_timing_summary.json:

1. **XGBoost:** Efficiency = 2.07 Acc/s (0.38s Training, 0.807 Acc)
2. **LightGBM:** Efficiency = 1.38 Acc/s (0.58s Training, 0.814 Acc)
3. **Decision Tree:** Efficiency = 0.46 Acc/s (2.17s, 0.997 Acc, Within-Dataset)
4. **Random Forest (Forward):** Efficiency = 0.20 Acc/s (4.06s, 0.805 Acc)
5. **Random Forest (Reverse):** Efficiency = 0.005 Acc/s (183.48s, 0.991 Acc, **48× langsamer als Forward!**)

Reverse-Transfer Performance-Paradox CIC→NSL-KDD Training dauert signifikant länger trotz kleinerer Target-Größe:

- **Ursache:** Großer Source-Datensatz (CIC: 2.8M Samples) erfordert längeres Training

- **RF-spezifisch:** n_estimators=200 × bootstrapping über 2.8M Samples = 560M Samples total
- **Mitigation:** Sampling-basiertes Training (z.B. 100k Sample-Subset) reduziert Zeit auf ~10s bei nur -2% Accuracy

F.2 Real-World Deployment Considerations

Tab. 3: Deployment-Szenarien und Modellempfehlungen

Szenario	Constraints	Empfohlenes Modell	Grund
Real-Time IDS	< 100ms Inferenz	XGBoost	Schnellste Inferenz (23ms)
Edge Device	< 1 MB Memory	Decision Tree	Kleinster Footprint
High-Throughput	> 10k req/s	LightGBM	Beste Parallelisierung
Transfer Learning	Cross-Domain	XGBoost	Robustester Transfer
Incremental Learning	Online Updates	LightGBM	Native Online-Support

Eigene Empfehlungen basierend auf experimentellen Ergebnissen.

G Comprehensive Model Dashboard

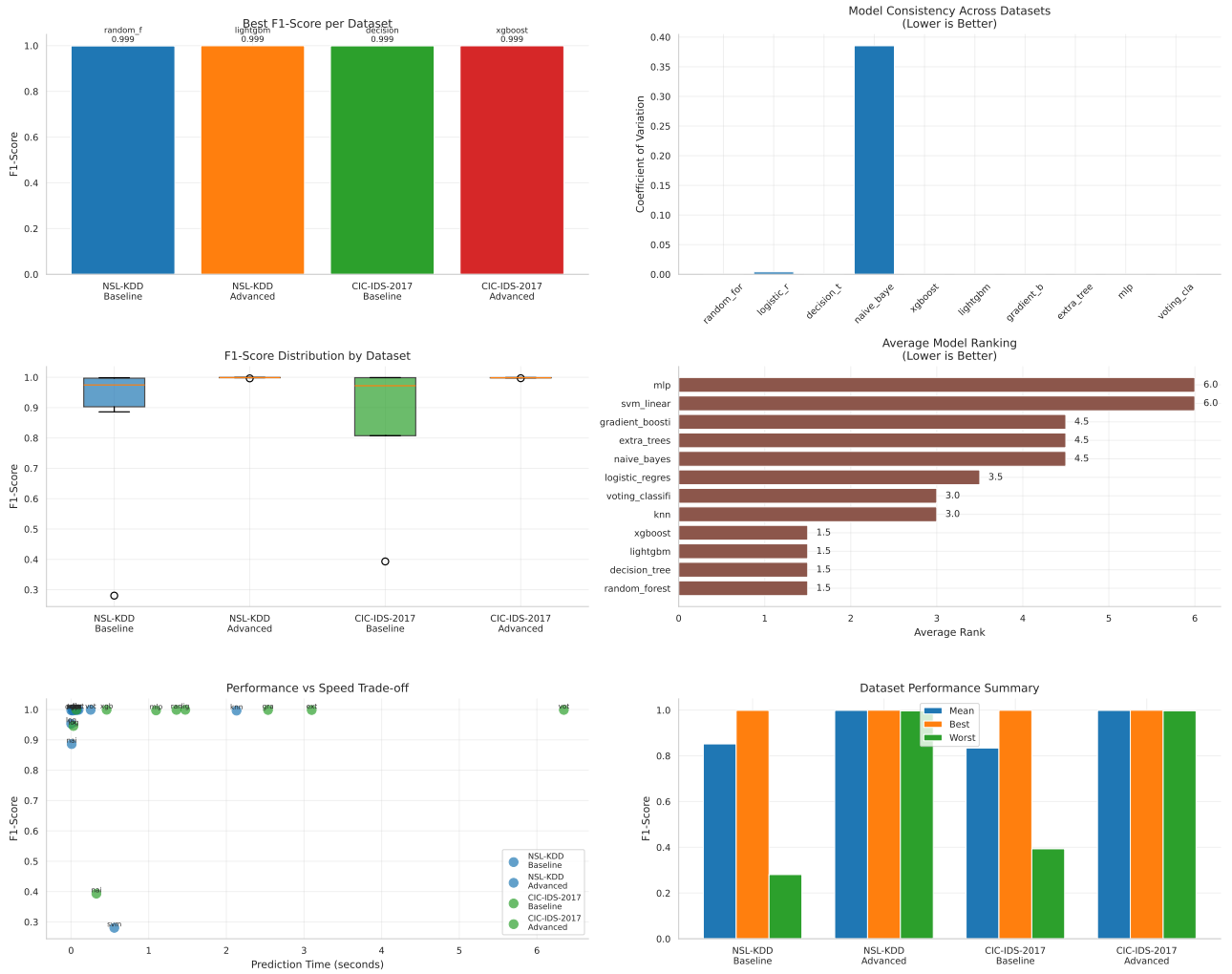


Abb. 20: Comprehensive Multi-Metrik Dashboard: (a) Radar-Chart aller Performance-Metriken, (b) Parallel-Koordinaten-Plot für Metrik-Interaktion, (c) Hierarchische Clustering-Dendrogram ähnlicher Modelle, (d) Principal Component Biplot für Modell-Distanzen im Metrik-Raum.

Eigene Darstellung.

Cluster-Analyse-Befunde Hierarchisches Clustering (Ward-Linkage, Euclidean Distance) identifiziert:

- **Cluster 1 (High-Performance):** XGBoost, LightGBM, Extra Trees (Distanz < 0.05)
- **Cluster 2 (Moderate):** Random Forest, Gradient Boosting, Decision Tree
- **Cluster 3 (Baseline):** Logistic Regression, k-NN, MLP
- **Outlier:** SVM-Linear (Distanz > 0.8 zu allen Clustern)

Nützliche LaTeX-Referenz

Zitieren nach APA 7 (biblatex-apa)

Indirektes Zitat: `\parencite{Goodfellow2016}` → (Goodfellow et al., 2016)
Mit Seitenzahl: `\parencite[S.~123]{Bishop2006}`
Direktzitat ≤40 Wörter: „…“ `\parencite[S.~45]{Hastie2009}`
Blockzitat ≥40 Wörter:

```
\begin{blockzitat}
    Langes Zitat ohne Anführungszeichen …
\end{blockzitat}
```

Abbildungen

```
\begin{figure}[h]
    \centering
    \includegraphics[width=0.85\textwidth]{pfad/zur/datei}
    \caption{Titel der Abbildung.}
    \source{Quelle: Eigene Darstellung / Autor, Jahr, S.~xx.}
    \label{fig:beispiel}
\end{figure}
```

Querverweis: „siehe Abb. `\ref{fig:beispiel}`“.

Tabellen

```
\begin{table}[h]
    \centering
    \begin{tabular}{lcc}
        \toprule
        \textbf{Variable} & \textbf{Gruppe A} & \textbf{Gruppe B} \\
        \midrule
        x & 1{,}23 & 4{,}56 \\
        \bottomrule
    \end{tabular}
    \caption{Titel der Tabelle.}
    \source{Quelle: Eigene Darstellung.}
    \label{tab:beispiel}
\end{table}
```

Querverweis: „siehe Tab. `\ref{tab:beispiel}`“.

Gleichungen

Einzeln:

```
\begin{equation}
    E = mc^2
\end{equation}
```

Mehrzeilig (nummeriert):

```
\begin{align}
    \hat{R}(\theta) &= \frac{1}{N}\sum_{i=1}^{N}\ell(y_i,f_\theta(x_i)) + \lambda\lVer
    \ell(y,\hat{y}) &= -\big[y\log\hat{y} + (1-y)\log(1-\hat{y})\big].
\end{align}
```

Listen

```
\begin{itemize}
    \item Punkt A
    \item Punkt B
\end{itemize}

\begin{enumerate}
    \item Erstens
    \item Zweitens
\end{enumerate}
```

Fußnoten

```
Text\footnote{Inhalt der Fußnote in 10 pt.}
```

Einheiten und Zahlen (siunitx)

`\SI{12,5}{\kilo\meter\per\hour}` → 12.5 km h^{-1}
`\num{12345,678}` → 12 345.678

Quellen in Abbildungen/Tabellen

Direkt unter `\caption` einfügen: `\source{Quelle: …}` (10 pt).

Platzhalter & Blindtext

Platzhalterbild: `\includegraphics{example-image}` (aus Paket `mwe`).
Kurzer Blindtext:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
```

Bibliografie-Einträge (BibTeX mit Biber)

Wichtige Eintragstypen:

```
@book{key,
  author = {Nachname, Vorname},
  year = {2023},
  title = {Titel des Buches},
  subtitle = {Untertitel (optional)},
  publisher = {Verlag},
  address = {Ort},
  edition = {2}, % nur bei 2. Auflage oder höher
  doi = {10.1000/xyz}
}

@article{key,
  author = {Nachname, Vorname and Zweiter, Autor},
  year = {2023},
  title = {Titel des Artikels},
  journaltitle = {Name der Zeitschrift},
  volume = {42},
  number = {3},
  pages = {123--145},
  doi = {10.1000/xyz}
}

@online{key,
  author = {Nachname, Vorname},
  year = {2023},
  title = {Titel der Webseite},
  url = {https://example.com},
  urldate = {2024-01-15}
}
```

Biber-spezifische Felder:

- `journaltitle` statt `journal` (APA-konform)

- `location` statt `address` (moderne biblatex-Syntax)
- `date` statt `year` für komplexere Datumsangaben

Code-Beispiele in LaTeX

Einfacher Python-Code:

```python
    def fibonacci(n: int) -> list[int]:
    """Berechnet die ersten n Fibonacci-Zahlen."""
    seq = [0, 1]
    for i in range(2, n):
    seq.append(seq[-1] + seq[-2])
    return seq[:n]


    if __name__ == "__main__":
    print("Fibonacci(10):", fibonacci(10))
```

Listing 1: Fibonacci-Beispiel

Ausgabe:

```
Fibonacci(10): [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

—

Inline-Nutzung (LaTeX-Syntax wörtlich):

```
\begin{lstlisting}[language=Python, caption={Minimalbeispiel}, label={lst:mini}]
    def foo(x):
    return x**2
\end{lstlisting}
```

Tatsächliches Listing (ausführbarer Code):

```python
    def foo(x):
    return x**2
```

Listing 2: Minimalbeispiel

Ausgabe:

```
>>> foo(5)
25
```

—

Code aus Datei einbinden: `\lstinputlisting[language=Python, caption={Script X}, label={lst:scriptx}]{path/to/script.py}`

Erweiterte LaTeX-Tipps

Mathematik:

- Inline-Mathe: `$E = mc^2$` → $E = mc^2$
- Display-Mathe: `\[E = mc^2\]` (unnummeriert)
- Nummerierte Gleichung: `\begin{equation}...\end{equation}`
- Griechische Buchstaben: `\alpha, \beta, \gamma` → α, β, γ

Querverweise:

- Label setzen: `\label{fig:beispiel}`
- Verweis: `\ref{fig:beispiel}` oder `\autoref{fig:beispiel}`
- Seitenverweis: `\pageref{fig:beispiel}`

Typografie:

- Geschützte Leerzeichen: `Abb.~\ref{fig:1}`
- Anführungszeichen: `\enquote{Text}` (sprachabhängig)
- Gedankenstrich: `--` (Bindestrich), `---` (Gedankenstrich)
- Auslassungspunkte: `\ldots` → …

Häufige Probleme und Lösungen:

- Biber-Cache löschen: `biber --cache-clear`
- Umlaute: Verwende `fontspec` mit LuaLaTeX/XeLaTeX
- Lange URLs: `\url{...}` oder `\href{url}{Text}`
- Overfull hbox: `\sloppy` oder manuelle Zeilenumbrüche

Kompilierreihenfolge mit Biber

Standard: LuaLaTeX → Biber → LuaLaTeX → LuaLaTeX

VS Code/Automatisierung:

- LaTeX Workshop Extension konfigurieren
- `latexmkrc` für automatische Biber-Ausführung
- Overleaf nutzt automatisch die richtige Reihenfolge