ChatGPT

Roadmap Monat 4–12: AI-Entwicklung mit integriertem Security-Basiswissen

Nach den ersten 3 Monaten Grundlagen (Python, Mathematik, AI-Konzepte) beginnt ab **Woche 13 (Monat 4)** die **Stage 2** der Ausbildung [1]. In den Monaten 4 bis 12 wird Martin seine Fähigkeiten in **angewandter KI** gemäß der ursprünglichen Roadmap ausbauen – mit Fokus auf Machine Learning, Deep Learning und ersten Projekten – **und parallel wöchentlich ~3 Stunden Security-Grundlagen** erlernen. Der folgende Plan ist nach Monaten strukturiert. Jeder Monatsabschnitt enthält die **AI-Schwerpunkte** (Hauptfokus), **Security-Schwerpunkte** (3 h/Woche Nebenfokus), **Lernressourcen** sowie **Projektideen** zur Verbindung beider Themen. Die Security-Lernziele sind **breit gefächert** (Web/App-Security, Pentesting, Privacy, DevSecOps), ohne Überlastung und ohne Zertifikatsprüfungen im ersten Jahr. Wichtig ist, dass der AI-Fortschritt im Vordergrund bleibt, während nach und nach ein Security-Bewusstsein für spätere Spezialisierungen wächst.

Monat 4: Machine-Learning-Grundlagen & Web-Security-Basics

- **AI-Schwerpunkt:** Start in die Kern-ML-Ausbildung. Einschreibung in einen renommierten **Machine-Learning-Kurs** (z. B. Andrew Ng's *Machine Learning* auf Coursera) als strukturierten Leitfaden [2] [3]. Ziel in Monat 4 ist es, die **Grundlagen von ML-Algorithmen** (Lineare Regression, Logistische Regression, einfache Klassifikatoren) zu verstehen und erste Übungen umzusetzen. Martin plant ~7 Stunden pro Woche hierfür ein, was gut zu Kursen wie Andrew Ng passt (ca. 2 Stunden Videomaterial plus Übungen pro Woche [4]). Nebenbei festigt er seine Python-Kenntnisse durch kleine Skripte (z. B. mit **pandas** Daten laden und einfache Analysen durchführen). *Meilenstein:* Ende Monat 4 sollte er ~50 % des ML-Kurses durchgearbeitet haben und Begriffe wie *Training vs. Test*, *Accuracy*, *Overfitting* verstehen. Dies legt das Fundament für kommende Projekte.

- **Security-Schwerpunkt (3 h/Woche): Einstieg in Web- und Application Security.** Zunächst verschafft sich Martin einen Überblick über **häufige Web-Schwachstellen**. Er studiert die **OWASP Top 10** – eine weithin anerkannte Liste der kritischsten Sicherheitsrisiken für Webanwendungen [5]. Dieses Dokument dient Entwicklern als erster Leitfaden für sicherere Programmierung. Er lernt dabei grundlegende Angriffsarten kennen (z. B. SQL-Injection, XSS, unsichere Authentifizierung) und versteht, warum Eingabefilterung, Zugriffskontrollen etc. so wichtig sind. Parallel dazu absolviert er erste interaktive Übungen: **PortSwigger's Web Security Academy** bietet kostenfreie Labs, in denen man typische Schwachstellen gefahrlos ausprobieren kann [6]. Zum Beispiel löst er ein Einsteiger-Lab zu **SQL-Injection**, um praktisch nachzuvollziehen, wie ein einfacher Injection-Angriff funktioniert und wie man ihn verhindert. *Meilenstein:* Am Monatsende kennt Martin die Top-10-Risiken und hat exemplarisch eine Handvoll typischer Webangriffe gesehen. Er baut ein Sicherheitsgrundbewusstsein auf, das ihn künftig bei der Entwicklung sicherer AI-Anwendungen leiten wird.

- **Lernressourcen:** *AI:* Coursera – Andrew Ng's **Machine Learning** Kurs (oder alternativ **fast.ai** Practical Deep Learning, falls er einen code-orientierteren Ansatz bevorzugt) [3]. Online-Community (z. B. Foren, Discord) zur Unterstützung bei Übungen. *Security:* **OWASP Top 10 Dokumentation** (deutschsprachige Zusammenfassungen sind online verfügbar), YouTube-Kanal "OWASP Top 10 einfach erklärt" für visuelle Erläuterungen. **PortSwigger Web Security Academy**

(kostenloses Trainingsportal mit Texten und Labs) [6] für praktische Übungen; insbesondere die Labs zu grundlegenden Webangriffen wie SQLi und XSS. Evtl. Einführungsvideos zu Web Security (z. B. „Web Security Basics“ von freeCodeCamp auf YouTube).

- **Projektidee (Mini-Projekt): “Hello World” Data Analysis.** Parallel zu Kurs und Security-Intro kann Martin ein kleines Python-Datenprojekt starten, um Gelerntes anzuwenden. Z. B. ein **Mini-Analyseprojekt**: Er nimmt einen öffentlichen Datensatz (vielleicht etwas Einfaches von Kaggle, z. B. Iris-Datensatz) und schreibt ein Jupyter-Notebook, das Daten einlädt, bereinigt und einfache Statistiken oder Visualisierungen ausgibt. Dieses Projekt (Code auf GitHub) dient als Grundstock seines Portfolios und demonstriert grundlegende Datenkompetenz. Security-Aspekt in diesem Mini-Projekt: Er richtet gleich ein **GitHub-Repository** ein und übt den sicheren Umgang mit Code (z. B. keine sensiblen Daten ins Repo laden, README mit Hinweisen). Dies ist zwar noch kein richtiges Security-Feature, aber fördert früh die Gewohnheit, Code und Dokumentation sauber zu halten.

Monat 5: Vertiefung ML-Konzepte & Sichere Webentwicklung

- **AI-Schwerpunkt:** Fortführung und **Abschluss des ML-Grundlagenkurses**. In Monat 5 schließt Martin Andrew Ng’s Machine-Learning-Kurs ab – inklusive Wochenaufgaben zu Themen wie Decision Trees und Naive Bayes. Er wendet Gelerntes direkt praktisch an: Nachdem er nun Kernalgorithmen kennt, probiert er ein kleines eigenes ML-Experiment. Beispielsweise lädt er einen **Kaggle-Datensatz** (z. B. Titanic Survival Data) und baut einen einfachen Klassifikator mit **scikit-learn**. Dadurch übt er das gesamte Pipeline: Datenvorbereitung, Training, Evaluation (Accuracy, Confusion Matrix) – wichtige Fähigkeiten laut 12-Monats-Milestones [7] [8]. Der Fokus liegt darauf, **Verständnis vor Perfektion** zu stellen – er muss nicht jedes mathematische Detail ableiten können, aber sollte wissen, *wie* man ein Modell trainiert und validiert. *Meilenstein:* Ende Monat 5 hat Martin mindestens **ein eigenes kleines ML-Modell** trainiert (z. B. Prädiktion, ob ein Patient einen Termin verpasst, basierend auf einfachen Faktoren – oder ein anderes Mini-Projekt aus seinem Umfeld). Damit erreicht er den Status **“ML Beginner”**, der es ihm erlaubt, im nächsten Schritt ins Deep Learning einzusteigen.

- **Security-Schwerpunkt (3 h/Woche): Sichere Webentwicklung & Gegenmaßnahmen.** Aufbauend auf dem OWASP-Wissen vertieft Martin im Monat 5 konkrete **Abwehrmaßnahmen** gegen die häufigsten Web-Attacken. Er lernt z. B., wie man **SQL-Injections** durch Prepared Statements/ORM vermeidet, wie **XSS** durch Output-Encoding und CSP gemindert wird, und wie man Passwörter richtig hasht (Bcrypt, Salt). Praktisch übt er dies mit einem Mini-Webprojekt: Er schreibt eine sehr einfache **Flask-Webanwendung** mit einem Eingabeformular (z. B. ein Mini-Gästebuch) und implementiert absichtlich eine Schwachstelle (z. B. ungefilterte Eingabe, die XSS ermöglicht). Dann versucht er, diesen Prototyp selbst anzugreifen (z. B. einen `<script>`-Tag einschleusen) und **behebt** die Schwachstelle durch entsprechende Filter oder Framework-Funktionen. Durch dieses *Break-and-Fix*-Mini-Projekt versteht er die Entwicklerperspektive: Wie sieht unsicherer Code aus und wie wandelt man ihn in sicheren Code um? Ergänzend experimentiert er mit Tools: Er testet z. B. **OWASP ZAP** (Zed Attack Proxy) oder die Burp Suite Community Edition, um seine kleine Webapp automatisch auf Lücken scannen zu lassen – ein erster Einblick in Security-Testing-Werkzeuge. *Meilenstein:* Martin kann am Monatsende grundlegende Sicherheitsmaßnahmen benennen und hat ein Gefühl dafür, wie man eine einfache Webanwendung “härtet”. Er versteht nun Aussagen wie *“Eingaben validieren!”* oder *“Prinzip der minimalen Rechte”* und kann sie bei eigenen Projekten berücksichtigen.

- **Lernressourcen:** *AI:* Abschluss von Andrew Ng's ML (Coursera) – ggf. zusätzlich **Übung auf Kaggle**: die *Titanic* oder *Iris* Challenge als bekannte Einsteigerprojekte. Kurze Tutorials zu scikit-learn (z. B. YouTube: „Train your first model in 10 minutes"). *Security:* **Secure Coding Guides** – z. B. das OWASP Cheat Sheet „Secure Coding Practices" als Referenz. Online-Kursmaterial „Secure Web Development" (es gibt z. B. auf Pluralsight oder LinkedIn Learning kurze Kurse zu Web App Security, die man mit Testphase nutzen kann). **OWASP Juice Shop** (ein bewusst verwundbares Webapp-Projekt) könnte er lokal aufsetzen, um zu sehen, ob er bekannte Schwachstellen darin findet; die Dokumentation gibt Tipps zum Auffinden der OWASP Top 10 in der Anwendung. Für Praxis: weiter PortSwigger Academy – dort die *"Authentication"*-Sektion (um zu verstehen, wie man Authentifizierung knackt und schützt) und *"Security Misconfigurations"*-Labs.

- **Projektidee (Mini-Projekt): "Gästebuch 2.0" – sichere Mini-Webapp mit ML-Funktion.** Als Integration beider Bereiche kann Martin zum Monatsende ein kleines **Web-ML-Projekt** wagen: z. B. ein schlichtes **Flask**-Webformular, wo ein User einen Text eingibt und ein ML-Modell eine Vorhersage trifft (etwa Stimmung analysieren – positiv/negativ – mithilfe eines vorher trainierten einfachen NLP-Modells). Dieses Projekt verbindet Webentwicklung und ML. Den **Security-Aspekt** betont er, indem er folgende Features einbaut: (1) **Eingabefelder validieren** (keine zu langen Texte, Filterung potenziell gefährlicher Tags), (2) ein einfaches **Auth-System** (Registrierung/Login mit Passwort-Hashing, z. B. Werkzeug-Bibliothek in Flask verwenden), und (3) sicherer Umgang mit dem ML-Modell-Backend (Requests nur serverseitig, keine vertraulichen Modellinfos im Client). Das Projekt bleibt klein (z. B. Stimmungsklassifikation mit ein paar Beispielsätzen), aber demonstriert bereits einen **"Full-Stack"-Workflow**: vom Browser-Input über ein ML-Modell zum Output, abgesichert durch grundlegende Security-Maßnahmen. Martin stellt den Code auf GitHub bereit – ein weiteres Portfolio-Stück, das zeigt, dass er früh an Sicherheit denkt.

Monat 6: Einstieg Deep Learning & Pentesting-Grundlagen

- **AI-Schwerpunkt: Deep Learning Grundlagen**. Nach dem klassischen ML-Kurs startet Martin nun mit **Neuronalen Netzen**. Er beginnt z. B. Andrew Ng's **Deep Learning Specialization** (5-teilige Coursera-Serie) oder alternativ den **fast.ai**-Kurs, der praxisnah mit PyTorch arbeitet [9] [10]. In Monat 6 konzentriert er sich auf die ersten Themen: Verständnis eines Perzeptrons/Neuronaler Netzwerke, Forward und Backpropagation (intuitiv, nicht im vollen Detail) und erste Implementierungen in **TensorFlow/Keras** oder **PyTorch**. Er schreibt sein erstes einfaches **Neuronales Netz** – z. B. ein 3-Layer-Netz für MNIST-Ziffernerkennung, wie es in vielen Kursen demonstriert wird. Wichtig ist, dass er lernt, *wie* man ein Netz trainiert (Epochen, Loss-Funktion, Optimizer) und was Konzepte wie Overfitting nun im DL-Kontext bedeuten. *Meilenstein:* Am Monatsende hat Martin mindestens ein **kleines DL-Modell** gebaut und trainiert (z. B. MNIST-Bilder klassifiziert). Er versteht grundlegende Begriffe: **Layer**, **Aktivierungsfunktion**, **Gradient Descent**, **Epoch** usw., und kann ein Training interpretieren (Loss Curve). Dies bereitet ihn auf komplexere DL-Aufgaben in kommenden Monaten vor.

- **Security-Schwerpunkt (3 h/Woche): Pentesting 101 – Angriffssicht kennenlernen.** Im sechsten Monat wechselt Martin die Perspektive: vom Verteidiger zum **Angreifer (Ethical Hacker)**. Er taucht in die **Pentesting-Grundlagen** ein, um zu verstehen, wie man Systeme und Apps gezielt auf Schwachstellen prüft. Zuerst klärt er die Begriffe: Ein *Penetrationstest* ist ein simulierter Cyberangriff auf ein System, um Schwachstellen aufzudecken, bevor echte Angreifer diese ausnutzen [11]. Er lernt die **Phasen eines Pentests** kennen (Reconnaissance, Scanning, Exploitation, Post-Exploitation, Reporting) [12] [13]. Praktisch startet er mit **einsteigerfreundlichen Hacking-Labs**: Plattformen wie **TryHackMe** bieten guided Challenges für Anfänger – z. B. den „Pre Security"- oder „Junior Penetration Tester"-Pfad. Hier lernt er Schritt für Schritt den Gebrauch typischer Tools: **Nmap** (Port-Scanner) zum Auffinden offener Ports und

Dienste, **Burp/ZAP** für Web-Pentests (hat er schon angerissen), einfache **Bruteforce-Tools** (wie `hydra` für SSH/FTP), etc. Er könnte z. B. das TryHackMe-**Modul "OWASP Top 10"** bearbeiten, um die in Monat 4/5 kennengelernten Weblücken praktisch zu exploiten (z. B. mittels SQLi einen Dummy-Datenbank-Login umgehen). Zusätzlich schaut er sich **Netzwerksicherheit** an: Grundlagen von TCP/IP, was ein DNS-Poisoning ist, etc., aber nur soweit es in den Pentest-Übungen vorkommt – tiefe Netzwerktheorie ist nicht Schwerpunkt, sondern praxisnahes *Learning by Doing*. *Meilenstein:* Martin hat bis Ende des Monats **einige CTF-artige Aufgaben** gelöst – z. B. erste Räume auf TryHackMe oder HackTheBox (Starting Point) abgeschlossen. Er konnte dabei mindestens **eine einfache Schwachstelle ausnutzen** (etwa eine bekannte Default-Credential, oder eine Datei mit Klartext-Passwörtern finden) und hat gesehen, wie ein Angreifer vorgeht. Dieses Wissen stärkt wiederum sein Entwicklerdenken: Er weiß nun, wie wichtig Patchen, sichere Konfiguration und Zugriffsbeschränkungen sind, weil er gesehen hat, wie leicht unsichere Systeme sonst zu kompromittieren sind.

- **Lernressourcen:** *AI:* Coursera – **Deep Learning Specialization (Andrew Ng/Deeplearning.ai)** Kurs 1 (Neural Networks & Deep Learning). Alternativ: **fast.ai – Practical Deep Learning for Coders**, Kapitel 1–2 (bildet schon ein ResNet-Modell für Bilderkennung, sehr praxisorientiert). YouTube-Videos (3Blue1Brown's Neural Networks series) zur anschaulichen Erklärung von Backpropagation. *Security:* **TryHackMe** (Online-Plattform) – interaktive Labs für Einsteiger (kostenlose Grundmodule) [14]. Empfehlenswert sind dort "*Intro to Cyber Security*" oder "*OWASP Top 10*" Lernpfade. **HackTheBox** (Starting Point labs) als Alternative, um praktische Erfahrung zu sammeln. Zusätzlich das Buch **"The Web Application Hacker's Handbook"** (Dafydd Stuttard) als Nachschlagewerk – Martin kann die ersten Kapitel lesen, um ein Gefühl für systematisches Pentesten zu bekommen (dieses Buch ist fortgeschritten, aber in Teilen hilfreich). Für schnelles Lernen: evtl. die **Tibee YouTube-Serie "Pentesting for Beginners"**. Tools: **Nmap** Tutorial (overthewire Bandit Level [optional]) und **Burp Suite** Einsteiger-Guide (PortSwigger hat Video-Tutorials dafür).

- **Projektidee (Mini-Projekt): Pentest-Report zu eigener Webapp.** Martin kann diesen Monat das im Vormonat gebaute „Gästebuch 2.0" (oder eine ähnliche Test-Webapp) aus der Angreiferperspektive überprüfen. Er erstellt einen kleinen **Pentest-Bericht**: führt einen selbst initiierten Sicherheitstest auf seine Anwendung durch (manuell und mit Tools) und dokumentiert die gefundenen Schwachstellen sowie die Behebung. Beispielsweise checkt er: Sind Eingaben ausreichend validiert? Funktioniert das Login-Bruteforce-Schutz? Welche Infos gibt der Server in Fehlerseiten preis? Er verwendet **OWASP ZAP** im automatisierten Scan-Modus auf seine App und notiert die Ergebnisse (ZAP findet evtl. fehlende Sicherheitsheader etc.). Den *Report* schreibt er in Markdown und fügt ihn dem GitHub-Repo bei. Dieses „Projekt" ist zwar kein klassisches Software-Projekt, aber es demonstriert Martins Fähigkeit, Security-Analysen durchzuführen und sicherheitsbewusst zu denken. Außerdem lernt er dabei, **Schwachstellen sauber zu dokumentieren** – eine wichtige Fähigkeit für zukünftige Teamarbeit oder Bug-Bounty-Aktivitäten.

Monat 7: Deep-Learning-Praxis & Datenschutz/Privacy-Grundlagen

- **AI-Schwerpunkt: Deep Learning vertiefen – CNNs und/oder NLP.** Im siebten Monat erweitert Martin sein Deep-Learning-Wissen auf gängige Architekturen. Im Andrew Ng Kurs wären das **Convolutional Neural Networks (CNN)** und evtl. **Recurrent Neural Networks (RNN/LSTM)** in den Folgekursen. Er lernt, wie CNNs für Bilddaten funktionieren (Faltung, Pooling) und trainiert ein Beispielmodell – etwa ein vorgefertigtes **CNN auf einem Bilddatensatz** (z. B. Klassifikation

von Bilder-Kategorien). Alternativ, falls sein Interesse eher zu NLP geht, kann er einen Teil auf **Word Embeddings und Sequenzmodelle** verwenden (Kurs 4 der Spezialisierung oder entsprechende fast.ai Lektionen). Er probiert zum Beispiel ein einfaches **Sentiment-Analysetool** mit einem kleinen LSTM-Modell auf Textdaten. Wichtig ist wieder praxisnahes Arbeiten: Er nutzt **Keras**-Beispielcode oder fast.ai Library, um schnell Ergebnisse zu sehen. Gegen Ende des Monats plant er sein erstes größeres **DL-Portfolio-Projekt** vor: Basierend auf seinen Interessen wählt er ein Thema – z. B. **Bildklassifikation** in seiner Domain (als Ex-Dentist vielleicht Zahnröntgenbilder erkennen, falls Datensätze auffindbar sind) oder ein **Textklassifikationsprojekt**. *Meilenstein:* Martin hat bis Monatsende einen **Projektplan** für ein Deep-Learning-Vorhaben erstellt und die Grundlagen dafür gelegt (Modelle trainiert, Datensätze recherchiert). Außerdem hat er sein Verständnis gefestigt: Er kann nun in groben Zügen erklären, wie ein CNN funktioniert und hat es praktisch angewendet.

- **Security-Schwerpunkt (3 h/Woche): Datenschutz & Privacy-by-Design.** Da Martin nun mit immer mehr Daten hantiert (Bilder, Texte etc.), widmet er Monat 7 dem Thema **Datenschutz**. Er verschafft sich ein **breites Grundlagenwissen über Privacy** in IT-Systemen, besonders im Kontext von KI. Zunächst arbeitet er die wichtigsten Punkte der **DSGVO (EU-Datenschutz-Grundverordnung)** durch, die für Entwickler relevant sind: Was gilt als *personenbezogene Daten*? Wann braucht man Einwilligungen? Welche Prinzipien (Datenminimierung, Zweckbindung) müssen beachtet werden? Er liest dazu einen Entwickler-Leitfaden oder Blog („KI und Datenschutz – was Entwickler beachten müssen"). Ihm wird klar, dass unsauberer Umgang mit sensiblen Daten zu Datenschutzverletzungen führt, mit Risiken von Datenlecks bis hin zu Rechtsfolgen [15]. Technisch schaut er sich **Privacy-by-Design-Praktiken** an: z. B. **Anonymisierung vs. Pseudonymisierung** – er übt an einem kleinen Datensatz, persönliche Identifikatoren (Namen, IDs) zu entfernen oder zu verschlüsseln, bevor er damit arbeitet. Auch interessant: **Differential Privacy** als fortgeschrittenes Konzept – er findet vielleicht ein einfaches Tutorial, das zeigt, wie man durch Hinzufügen von Rauschen statistische Ergebnisse ziehen kann, ohne individuelle Daten preiszugeben. Für den Anfang reicht ein qualitatives Verständnis: Er weiß danach, dass es Techniken gibt, um ML-Modelle zu trainieren und dennoch individuelle Datenpunkte zu schützen. *Meilenstein:* Am Ende von Monat 7 hat Martin ein erstes Bewusstsein für **Datenethik** entwickelt. Er kann die Begriffe **Privacy**, **DSGVO-Konformität**, **Datenanonymisierung** erläutern. Er hat praktisch z. B. ein kleines **Script zur Maskierung sensibler Daten** geschrieben (etwa alle Namen/Adressen in einem Datensatz durch Platzhalter ersetzt), um das Konzept zu üben. Dieses Script kann er ebenfalls in sein Portfolio aufnehmen (als Utility-Funktion in Python). Insgesamt ist er nun gewappnet, in seinen ML-Projekten auf den Schutz etwaiger Benutzerdaten zu achten.

- **Lernressourcen:** *AI:* Coursera Deep Learning Kurs 2 (CNNs) und Kurs 3 (Sequences) – je nach Interesse. **fast.ai** Kapitel zu CNN (Pet-Bilderkennung) für hands-on Übung. Keras-Beispiele (auf keras.io gibt es Tutorials für CNN und Text-LSTM, die er nacharbeiten kann). Pretrained-Modelle: ggf. Experiment mit einem **Transfer Learning** – z. B. ein vortrainiertes ResNet auf eigenen Bilddaten feinjustieren (Keras Tutorial). *Security:* **GDPR für Entwickler** – z. B. der kostenlose Kurs "Udacity – Intro to GDPR" oder Artikel vom BSI. Webseiten der Datenschutzbehörden (oft FAQ für Technik). **Blogbeiträge** über *Privacy in Machine Learning* (OpenMined hat Ressourcen, oder Googles Developers-Blog zu Differential Privacy). Falls verfügbar: Coursera-Kurs "Privacy in Data Science" (es gibt z. B. von University of Michigan einen Kurs zu Datenethik). Ein **YouTube-Video über Differential Privacy** (z. B. von Google I/O oder ähnlichen Konferenzen). **OpenMined's PySyft** Bibliothek – er muss sie nicht meistern, aber kann reinschauen, wie federated learning und DP in Python ausschauen würden.

- **Projektidee (Mini-Projekt): Privacy-Check im ML-Projekt.** Martin kann sein geplantes DL-Projekt (vom AI-Teil dieses Monats) gleich unter Privacy-Aspekten betrachten. Angenommen, er will einen Datensatz mit Patientenbildern nutzen – er formuliert einen **Privacy Plan**: welche persönlichen Informationen sind enthalten und wie werden sie geschützt? Als **Mini-Projekt** könnte er einen **Datensatz vorbereiten**, der privacy-freundlich ist: z. B. findet er einen öffentlichen Datensatz ohne direkte Personenbezüge, oder er entfernt Metadaten aus Bilddateien (die oft Kameradaten oder Standort enthalten). Alternativ implementiert er ein **einfaches Differential-Privacy-Beispiel**: nimmt eine kleine Tabelle mit Personendaten und zeigt, wie man mittels Rauschen einen aggregierten Wert berechnen kann, der Individuen nicht preisgibt. Dieses Experiment (z. B. Durchschnittsalter mit und ohne DP vergleichen) dokumentiert er. Das Resultat ist ein kurzer **Bericht im Wiki seines GitHub-Repos**: "Privacy Considerations for Project X". Darin listet er Datenschutzrisiken seines Projekts und wie er sie mindert (ähnlich einer Datenschutz-Folgenabschätzung in klein). Damit demonstriert er, dass er als AI-Entwickler früh die **Verantwortung für den Schutz von Daten** mitdenkt – ein Pluspunkt, da Arbeitgeber zunehmend Wert auf AI Ethics legen [16].

Monat 8: Abschluss Deep-Learning-Kurse & DevSecOps-Einführung

- **AI-Schwerpunkt: Deep Learning Kurse abschließen & erstes größeres Projekt starten.** Im achten Monat schließt Martin die formalen DL-Kurse ab. Er beendet z. B. die letzten Kurse der Deep Learning Spezialisierung (CNN/RNN und evtl. einen Kurs zu Deployment oder spezialisierten topics). Damit hat er eine **strukturierte Übersicht über ML/DL** erlangt – von Regression über CNNs bis hin zu einfachen **Sequenzmodellen** – was dem Jahresziel entspricht, ein breites Fundament aufzubauen [1]. Nun legt er den Fokus voll auf sein **Praxisprojekt**: Er beginnt mit der Umsetzung des gewählten **Deep-Learning-Projekts**. Angenommen, er hat sich entschieden, einen **Bildklassifikator für Zahnmedizin-Bilder** zu bauen: In Monat 8 sammelt/organisiert er die Daten (z. B. nutzt er einen offenen Datensatz oder generiert synthetische Daten), definiert die Projektstruktur und startet mit ersten Experimenten (trainiert vielleicht ein Baseline-Modell und evaluiert es grob). Alternativ, falls Text-Projekt: er bereitet einen Korpus auf, baut ein erstes Modell. Ein wichtiger Schritt jetzt ist auch, das Projekt **öffentlich zu machen** – er erstellt ein **GitHub-Repo** speziell dafür und beginnt, seinen Code und Fortschritt dort zu versionieren. *Meilenstein:* Ende Monat 8 sind die Deep-Learning-Kurse abgeschlossen (ggf. erhält er ein Abschlusszertifikat, aber wichtiger ist das Wissen). Sein **Hauptprojekt** hat konkrete Gestalt angenommen: Er kann jemandem erklären, woran er arbeitet (z. B. "Ein CNN, das Kategorien von Röntgenbildern unterscheidet, aktuell ~70 % Genauigkeit, Ziel >85 %"). Diese Grundlage bereitet ihn auf den Feinschliff in den nächsten Monaten vor.

- **Security-Schwerpunkt (3 h/Woche): DevSecOps – Sicherheit in den Entwicklungsprozess integrieren.** Jetzt, da Martin an einem größeren Softwareprojekt arbeitet, richtet er den Security-Blick auf den **Software-Lifecycle**. Er lernt, was **DevSecOps** bedeutet: nämlich Sicherheits-Praktiken in jede Phase der Softwareentwicklung zu integrieren [17]. Er liest einen Einstiegsartikel („Was ist DevSecOps?" von AWS oder RedHat) und versteht die Kernaussage: **Sicherheit als gemeinsame Verantwortung** von Entwicklung, Betrieb und Security-Team, und „Shift-Left" – also frühzeitiges Einbauen von Sicherheitschecks. Praktisch wendet er das direkt auf sein Projekt an. Er richtet z. B. eine **automatisierte Code-Prüfung** ein: GitHub bietet *Dependabot* und *CodeQL* Scans – er aktiviert diese fürs Projektrepo, um Abhängigkeits-Schwachstellen oder Code-Smells aufzudecken. Zudem schreibt er für sein Projekt einfache **Unit-Tests**, die sicherheitsrelevant sind (z. B. ein Test, der prüft, ob nur erwartete Dateitypen geladen werden, um *Phishing* über Dateinamen zu vermeiden). Auch das Thema **CI/CD** schaut er sich an:

Falls er z. B. GitHub Actions verwendet, integriert er einen **Security-Linter** (etwa *Bandit* für Python, der Code auf bekannte Fehler wie `assert False` oder unsichere Funktionen prüft). So erlebt er DevSecOps im Kleinen: jedes Mal, wenn er Code pusht, laufen Tests und Scans, die sofort Feedback zu potenziellen Sicherheitsproblemen geben. Zusätzlich liest er über **Container-Security**: sollte er seine Anwendung containerisieren (Docker), achtet er auf ein schlankes Base-Image und scanned es mit einem Tool wie *Trivy* auf Vulnerabilities. *Meilenstein:* Ende des Monats hat Martin grundlegende DevSecOps-Praktiken bei sich etabliert. Sein Projekt-Workflow beinhaltet jetzt **automatisierte Sicherheitsüberprüfungen**, und er versteht, dass Security nicht erst am Ende (Deployment) angehängt wird, sondern von Anfang an Teil des Entwicklungsprozesses ist. Dieses Mindset und die einfachen CI-Setups, die er gebaut hat, kann er in Zukunft für professionellere Pipelines ausbauen.

- **Lernressourcen:** *AI:* Abschluss von Deep Learning Specialization Kursen 4–5 (z. B. „Sequences & NLP“ und ein Kurs zu Deployment oder TensorFlow in Practice). **fast.ai** ggf. letztes Kapitel (Deployment on Render etc.). Buch *“Hands-On Machine Learning mit Scikit-Learn, Keras & TensorFlow”* (Géron) kapitelweise als Nachschlagewerk beim Coden [18]. *Security:* **Blogposts/ Artikel über DevSecOps** (z. B. „DevSecOps 101“ auf Medium, AWS-Online-Doku “What is DevSecOps?”) [17]. Atlassian DevSecOps Tutorial, falls verfügbar. **GitHub Docs**: wie man Dependabot und Actions Security scans einschaltet. **OWASP Dependency-Check** (Tool) – vielleicht probiert er es lokal aus. **Docker-Best-Practices** (Docker Docs Kapitel “Improve Container Security”). Evtl. ein Kurs auf Pluralsight: „DevSecOps: Integrating Security into CI/ CD“ (sofern Zeit, aber da er nur 3h/Woche hat, eher fokussiert auf seine eigenen Pipeline-Setups).

- **Projektidee (Integration): “Secure AI Pipeline” – DevSecOps im ML-Projekt.** Das gesamte Monat 8 steht im Zeichen, Martins Hauptprojekt robust und sicher aufzusetzen. Er kann dies als **Teilprojekt** dokumentieren: Im GitHub-Repo richtet er z. B. einen **CI-Workflow** ein (YAML für GitHub Actions), der bei jedem Commit das Projekt baut/testet. Er integriert einen **Security-Scan-Schritt** (z. B. `pip install bandit && bandit -r .` um Python-Code zu scannen). Auch ein **konfiguriertes Linting** (Flake8/Pylint) mit Regeln, die Security betreffen (z. B. Warnung bei Verwendung von eval()). Als Ergebnis erstellt er eine **README-Sektion „Security“**, wo er aufführt: *“Dieses Projekt verwendet Dependabot für Dependency Security und führt automatisierte Code-Scans durch. Alle Secrets (API-Schlüssel etc.) werden in GitHub Actions als verschlüsselte Secrets verwaltet, nicht im Code.”* Dies demonstriert, dass er bereits als Solo-Entwickler **professionelle DevSecOps-Praktiken** übernimmt. Im Kontext von KI heißt das etwa: Er stellt sicher, dass z. B. ein API-Schlüssel von OpenAI für ein späteres LLM-Modul nicht im Klartext im Code landet, sondern sicher injected wird – solche Details kann er dort erwähnen. Dieses “Secure Pipeline” Unterfangen ist zwar kein sichtbares Produkt, aber für Reviewer eines GitHub-Repos ein starkes Signal von Kompetenz.

Monat 9: Erste DL-Projekt-Fertigestellung & Security-Review

- **AI-Schwerpunkt: Deep-Learning-Projekt abschließen (Version 1) & Beginn LLM-Exploration.** Im neunten Monat möchte Martin sein erstes größeres **Deep-Learning-Projekt** zu einem vorzeigbaren Ergebnis bringen. Er verbringt viel Zeit damit, das Modell zu **trainieren und zu optimieren**: experimentiert mit Hyperparametern, probiert ggf. **Transfer Learning** (fine-tuned ein vortrainiertes Modell, um die Performance zu heben) [19] [20]. Bis etwa Mitte des Monats sollte er einen **funktionierenden Prototyp** haben – z. B. sein Bildklassifikator erreicht eine akzeptable Genauigkeit auf Testdaten. Er erstellt **Visualisierungen** (Konfusionsmatrix, Beispieloutputs) und integriert diese in sein Projekt-Repo. Damit erfüllt er einen wichtigen

Meilenstein der Roadmap: er hat ein komplettes ML/DL-Projekt end-to-end umgesetzt (Daten -> Modell -> Evaluation) [8]. Nachdem Version 1 steht, beginnt er, **neue Zukunftsthemen** anzuschneiden, vor allem **Large Language Models (LLMs)**, da diese in Stage 3 Schwerpunkt werden [21] [22]. Gegen Ende Monat 9 fängt Martin klein an: Er macht den kostenlosen kurzen **Kurs "ChatGPT Prompt Engineering for Developers"** (von OpenAI/DeepLearning.AI) und lernt, wie man mit LLM-APIs arbeitet [4]. Dann baut er etwas Spaßiges: Mit dem neu erworbenen Prompt-Wissen ruft er über die OpenAI-API **GPT-4** auf und schreibt ein Skript für einen simplen **Q&A-Chatbot**. Thema am besten vertraut – vielleicht ein „Dental Advice Assistant", der Fragen zu Zahnpflege beantwortet (so kann er sogar Domänenwissen einfließen lassen, was seiner Hintergrundbranche entspricht). Das ist weitgehend *low-code* (eine API-Anfrage mit Python), aber der Lerneffekt liegt im **Prompt-Experimentieren** – Martin probiert z. B. verschiedene Prompts aus, um formatierte Antworten zu erhalten [23]. *Meilenstein:* Ende Monat 9 kann Martin stolz sagen, dass er **2–3 ML-Projekte im Portfolio** hat (Datenanalyse vom Anfang, das große DL-Projekt, evtl. noch ein kleines Kaggle oder Web-ML-Demo) [24]. Außerdem hat er **erste LLM-Erfahrung** gesammelt: Er hat GPT-4 per API genutzt und verstanden, wie man mit Prompts Ergebnisse steuert [25]. Dieses Vorwissen bereitet ihn auf intensiveres LLM-Arbeiten in den folgenden Monate vor.

- **Security-Schwerpunkt (3 h/Woche): Security-Review & Auffrischung.** Nach dem breiten Themensprint der letzten Monate nutzt Martin Monat 9, um sein Security-Wissen zu konsolidieren **und auf sein Projekt anzuwenden**. Einerseits wiederholt er kurz alle bisherigen Domains: Er prüft, ob er die OWASP Top 10 aus dem Stegreif aufzählen kann, was ein Pentest beinhaltet, welche Privacy-Prinzipien es gibt und was DevSecOps bedeutet. Was er merkt: die stückweisen 3h/Woche haben ihm zwar Basiswissen gegeben, aber es tut gut, das **Gelernte zu rekapitulieren**. Das tut er, indem er versucht, **sein eigenes Projekt einem Security-Audit zu unterziehen**. Er nimmt die Rolle eines *Security Engineers* und schaut systematisch: Enthält mein DL-Projekt potentielle Schwachstellen? Er checkt z. B., ob in seinem Code irgendwo **geheime Zugangsschlüssel** fälschlich hardcoded sind (sollte nicht, dank DevSecOps-Vorkehrungen). Er untersucht, wie er **Benutzereingaben** handhabt, falls sein Modell Teil einer Webanwendung wird. Falls sein Projekt eine Web-Demo hat (z. B. via Streamlit oder Gradio), testet er diese auf die früher gelernten Dinge (XSS, etc.). Eventuell bittet er einen Bekannten, das Tool auszuprobieren und Feedback zu Lücken zu geben – ein inoffizieller *Peer Review*. Nebenbei hält er sich über **News** auf dem Laufenden: Er liest z. B. Artikel über aktuelle KI-bezogene Security-Vorfälle (wie *Prompt Injection* bei LLMs oder Datenlecks bei AI APIs) um zu sehen, wie seine Kenntnisse einzuordnen sind. *Meilenstein:* Nach diesem Review hat Martin seine Projekte soweit möglich **abgesichert**: insbesondere das große DL-Projekt hat eine Security-Checkliste durchlaufen. Er schreibt ggf. eine kurze **Release-Doku** (im GitHub Wiki oder README) dazu: "Known limitations and security considerations". Darin notiert er ehrlich, welche Sicherheitsthemen noch offen sind (z. B. "Modell nicht robust gegen adversarial examples") und welche er adressiert hat. Diese Reflexion zeigt potentiellen Mentoren/Arbeitgebern, dass Martin ein **ganzheitliches Verständnis** entwickelt – er liefert nicht nur funktionierende KI, sondern denkt an Robustheit und Sicherheit.

- **Lernressourcen:** *AI:* Eigene Projekt-Daten/Github. Dokumentation von **OpenAI API** (für Python). Kurzer **Prompt Engineering Kurs** (DeepLearning.AI, free, ~1 Stunde) [4]. **Hugging Face Transformers** Kurs (Beginnt eventuell damit, da dieser in Stage 3 tieferkommt – aber er kann reinschnuppern, wie man open-source LLMs lädt) [26]. *Security:* **Checklisten** – z. B. OWASP ASVS (Application Security Verification Standard) im Abschnitt Level 1 als Orientierung, welche Punkte eine App sicher machen (z. B. Eingabevalidierung überall, sichere Passwortspeicherung, etc.). Blog **"AI Security"** – Artikel über Prompt Injection, Adversarial Examples (um eine Vorschau auf KI-spezifische Security-Themen zu erhalten). **OWASP Top 10 für LLMs** (gerade 2024 erschienen,

listet spezielle Gefahren für LLM-Anwendungen). Diese Ressourcen zeigen ihm, dass Security ein kontinuierliches Feld ist – Lernen hört nie auf.

- **Projektidee (Abschluss Phase 1 Projekt): Finalisierung & Präsentation**. Im September schließt Martin Phase 1 seines DL-Hauptprojekts ab. Er plant eine **Projektvorstellung** – als würde er sich auf einen Junior-AI-Dev-Posten bewerben. Dazu gehört ein **Projekt-README** mit allen wesentlichen Infos, einer **Sektion zu Security/Privacy** (aus dem Vormonat) und evtl. einer **Live-Demo**. Falls möglich, setzt er sein Modell als **Web-App online** (z. B. mittels Streamlit Sharing oder Hugging Face Spaces) [27]. Diese Web-Demo versieht er mit grundlegender Absicherung: vielleicht einem einfachen **Login**, damit nicht jeder fremde Nutzer Unfug treibt, und Hinweisen, was mit eingegebenen Daten geschieht (Privacy Notice). So verbindet er noch einmal AI und Security: ein ML-Demo-Produkt, das nicht nur technisch funktioniert, sondern auch sicher bereitgestellt wird. Er übt die Präsentation (vielleicht schreibt er einen **Blogpost** auf Medium oder LinkedIn über sein Projekt, inkl. der Security Aspekte). Damit legt er den Grundstein für **Stage 3 (Jahr 2)**, in der er sich gezielt spezialisieren und ein „Standout"-Projekt entwickeln will.

Monat 10: LLM-Integration & Sichere AI-Anwendungen

- **AI-Schwerpunkt: Applied LLMs – erstes eigenes LLM-App-Projekt.** Nun hat Martin ein solides erstes KI-Portfolio aufgebaut. Im zehnten Monat vertieft er sich in die Welt der **Large Language Models** intensiver, da diese als zukunftsträchtiger Bereich identifiziert wurden [28] [29]. Er nimmt sich vor, eine **kleine Anwendung mit einem LLM** umzusetzen. Ein naheliegendes Projekt: einen **Chatbot** oder Q&A-Assistenten, der auf einem *Knowledge Base* arbeitet. Dazu lernt er Grundlagen von **Retrieval-Augmented Generation (RAG)** kennen – also wie man ein Sprachmodell mit eigenen Daten verbindet [29] [30]. In kleinem Rahmen versucht er Folgendes: Er sammelt z. B. 20 häufige Fragen und Antworten aus seinem Fachgebiet (Zahnmedizin oder einem anderen Thema, das ihn interessiert) und baut einen Prototyp, der **auf Nutzerfrage die relevanteste Antwort** zurückgibt. Evtl. nutzt er dafür OpenAI GPT-4 mit **LangChain** Bibliothek, um Fragen zu parsen und Antworten zu formieren. Das ist schon recht fortgeschritten, aber auch wenn es nicht perfekt klappt, lernt Martin viel über **LLM-Workflow**: Embeddings erzeugen, einen **Vektor-Index** (z. B. FAISS oder Chroma) aufbauen, relevante Dokumente finden und dem LLM als Kontext geben – das Prinzip von RAG [31] [32]. *Alternativ:* Falls das zu komplex ist, baut er zumindest einen **einfachen GPT-4 Chatbot mit GUI** (z. B. via Gradio/Streamlit) der generische Gespräche führt, um die Integration eines LLM in eine Weboberfläche zu üben. *Meilenstein:* Ende Monat 10 hat Martin zumindest **ein LLM-basiertes Tool** zum Laufen gebracht – sei es ein Fragebeantworter mit beschränktem Wissen oder ein netter Chatbot. Wichtig ist, dass er dabei die Programmierschnittstellen von LLMs versteht und erste Hürden (z. B. Rate Limits, Formatierung von Prompts) gemeistert hat. Zudem hat er LangChain und evtl. HuggingFace *in action* gesehen, was ihn für Stage 3 (wo LLMs, Agenten, RAG Hauptthema sind) vorbereitet.

- **Security-Schwerpunkt (3 h/Woche): Sicherheit in LLM-Anwendungen & Authentifizierung.** Da Martin jetzt mit **API-Schlüsseln** und potentiell **Benutzereingaben an ein LLM** arbeitet, stehen neue Security-Fragen im Raum. Er lernt das Konzept **"Prompt Injection"** kennen – ein spezifisches LLM-Problem, wo ein Angreifer das Modell verleiten kann, geschützte Infos preiszugeben oder falsche Aktionen zu machen. Er liest die **OWASP Top 10 für LLMs** (gerade veröffentlicht) oder Blogposts dazu, um Beispiele zu sehen (z. B. der berühmte "Ignore previous instructions…" Exploit) [33]. Das sensibilisiert ihn, dass auch KI-Systeme Angriffspunkte bieten. Konkrete Maßnahmen: Er implementiert bei seinem Chatbot eine **Eingabevalidierung** für Nutzerfragen (um z. B. sehr lange Eingaben oder bestimmte Keywords zu filtern) – ein noch rudimentärer Schutz gegen Prompt Injection. Außerdem kümmert er sich um **Zugangsbeschränkung**: Wenn er seine LLM-App eventuell online stellt, soll sie nicht

missbraucht werden. Er fügt also zumindest eine **einfache Authentifizierung** hinzu oder einen **API Key** Mechanismus für Nutzer. Das kann so simpel sein wie ein Login mit vordefiniertem Passwort oder ein Token, den er manuell ausgibt. Damit testet er gleich wieder Web-Security-Wissen: Er muss die Auth mechanik sicher implementieren (z. B. Sessions oder JWT korrekt handhaben, Passwörter hashen – vieles hat er im Monat 5 gelernt). *Meilenstein:* Am Monatsende läuft Martins LLM-App nicht nur funktional, sondern ist **zugriffsbeschränkt** (nicht offen für jeden im Internet) und hat gewisse **Sicherheitsfilter**. Er hat verstanden, dass AI-Anwendungen neue Angriffsflächen mit sich bringen (LLM-spezifische Probleme) und dass klassische Security wie Authentifizierung weiterhin essenziell ist. Er dokumentiert diese Überlegungen in seinem Projekt. Sollte er z. B. einen *Readme* für den Chatbot haben, schreibt er einen Abschnitt "Security Considerations", wo steht: *"Only authorized users can query the bot; user inputs are trimmed to max length and certain patterns are sanitized to prevent prompt manipulation."* – Das Niveau ist ein Einsteigeransatz, aber es zeigt, dass Martin Security als integralen Bestandteil seiner AI-Projekte sieht.

- **Lernressourcen:** *AI:* **LangChain-Dokumentation** (insbesondere Tutorials für Q&A-Bots). **OpenAI Cookbook** (GitHub Repo mit Beispielcode für API-Nutzung, OpenAI hat dort Codebeispiele für Chatbots). HuggingFace **Datasets/Embeddings** Tutorials – falls er RAG probiert, die HF-Docs für *sentence-transformers* Embeddings. *Security:* **OWASP Top 10 for LLMs (2024)** – sofern verfügbar, oder der OWASP Wiki-Artikel zu "LLM Application Security". Blog „**Prompt Injection 101**" (z. B. von Weaviate oder anderen LLM devs). **Auth0 Blog** zum Thema "Implementing Auth in a Flask app" (um ein bewährtes Muster für Login zu nutzen). **OAuth/OpenID** Grundlagen – er muss es nicht implementieren, aber eine Idee kriegen, wie professionelle Auth funktioniert (damit er nicht unsicheres eigenes Design baut). Kurzer Crash-Kurs **"JWT für Anfänger"** (YouTube), falls er Token-basierte Auth ausprobieren will.

- **Projektidee: "Secure Chatbot" – AI meets Security.** Der Chatbot, den Martin baut, *ist* bereits das Projekt. Um den Security-Aspekt zu verstärken, könnte er sich eine **Spezialfunktion** überlegen: z. B. ein Bot, der **Security-Tipps gibt**. Das hätte doppelten Lerneffekt – er müsste dem Bot einige Security-Fakten als Kontext geben (auffrischen seines Wissens) und der Bot beantwortet Fragen wie "Wie schütze ich mich vor Phishing?". Alternativ implementiert er einen **zweistufigen Bot**: erst prüft ein kleiner NLP-Filter (könnte ein einfaches Regelwerk oder ein kleines Toxicity-Model sein), ob die Nutzereingabe "sicher" ist (keine bösen Intents, keine persönlichen Daten), bevor sie an GPT-4 geht. Dieser vorgeschaltete Security-Filter ist eine Mini-Version von dem, was große Anbieter machen (Inhaltssicherheit). Natürlich kann Martin in 3h/Woche keinen vollwertigen Filter bauen, aber allein die Architektur zu durchdenken ist lehrreich. Er kann darüber im Projektbericht schreiben: *"I implemented a simple pre-filter for user input to the LLM to reject queries containing blacklisted terms (like 'system:' which could indicate a prompt injection attempt)."* – Das zeigt fortgeschrittenes Verständnis und kombiniert AI-Development mit Security Mindset unmittelbar.

Monat 11: LLM-Fine-Tuning & Privacy in ML-Anwendungen

- **AI-Schwerpunkt: LLM-Fortgeschritten: Fine-Tuning und Custom Models.** Jetzt wagt sich Martin an etwas ambitioniertere LLM-Arbeit. Er experimentiert mit **Open-Source-LLMs** und dem **Fine-Tuning** von Modellen [26]. Zum Beispiel lädt er via HuggingFace ein kleines Modell wie **GPT-2** oder **Llama-2-7B** (wenn Hardware/Colab es zulässt) und versucht, es mit eigenen Daten **nachzutrainieren**. Das könnten z. B. FAQs aus seinem Unternehmen oder ein Satz an Beispielkonversationen sein. Er nutzt dabei Tools wie HuggingFace's **Trainer API** oder das PEFT (Parameter-Efficient Fine-Tuning), um im Rahmen seiner Ressourcen zu bleiben. Selbst wenn er nur einen Teil des Fine-Tuning-Prozesses schafft, lernt er die Schritte: Datensatz formatieren,

Tokenizer anpassen, Training starten und überwachen [34] . Alternativ, falls das zu aufwendig ist, macht er einen kleineren **Embedding-Finetune** – z. B. fine-tuned ein **BERT-Modell** für ein Textklassifizierungsproblem (es gibt viele Tutorials dafür). So oder so, Martin **demystifiziert die Verwendung von LLMs** abseits der reinen API-Nutzung. Neben dem Training beschäftigt er sich mit dem **Deployen** solcher Modelle: er probiert etwa aus, ein feingetuntes Modell via **HuggingFace Hub** oder auf einer VM bereitzustellen, und merkt die Herausforderungen (Modelle sind groß, brauchen viel RAM etc.). *Meilenstein:* Ende Monat 11 hat Martin zumindest **ein kleines Sprachmodell angepasst** – z. B. sein GPT-2 liefert jetzt spezialisierte Antworten auf Zahnmedizin-Fragen, oder ein BERT klassifiziert Support-Tickets nach Themen. Er versteht nun viel besser, was *unter der Haube* passiert, wenn man ein LLM personalisiert, was ihn in Richtung Year 2 (Spezialisierung) vorbereitet [35] [36] .

- **Security-Schwerpunkt (3 h/Woche): Sichere Datenhaltung & ML-Modelle absichern.** In Monat 11 kehrt Martin noch einmal zum Thema **Datenschutz und Sicherheit in ML-Pipelines** zurück – diesmal praktischer. Er hat mittlerweile umfangreiche Daten und Modelle im Einsatz. Er lernt, wie man **sichere Infrastruktur** für ML gestaltet: z. B. wenn er ein Modell fine-tuned, achtet er darauf, dass sensible Trainingsdaten **nur lokal oder in vertrauenswürdiger Cloud** verarbeitet werden (Stichwort Datenverschlüsselung, kein Upload auf unsichere Dienste). Er schaut sich **Model Card** Best Practices an – also wie man einen Modellsteckbrief erstellt, der auch Bias und Privacy erwähnt (gehört zu AI Ethics). Außerdem interessiert ihn die Frage: **Können ML-Modelle selbst zum Privacy-Problem werden?** Er liest über **Model Inversion & Membership Inference** Attacken (Angriffe, bei denen man aus einem trainierten Modell Rückschlüsse auf die Trainingsdaten ziehen kann). Das öffnet ihm die Augen, dass sogar ein gut gemeintes veröffentlichtes Modell Trainingsgeheimnisse leaken könnte. Er versucht, einfache Gegenmaßnahmen zu verstehen: z. B. **Differential Privacy Training** – es gibt Bibliotheken (wie TensorFlow Privacy oder PyTorch Opacus) dafür. Vielleicht wagt er ein Mini-Experiment: Er trainiert ein einfaches Modell *mit* Differential Privacy (es gibt Tutorials dazu) und schaut, wie sich Accuracy vs. Privacy-Budget verhalten. Er muss die Mathematik dahinter nicht komplett erfassen, aber das praktische Ausprobieren festigt sein Verständnis von Privacy in ML. *Meilenstein:* Bis Monatsende hat Martin konkret umgesetzt, dass all seine Projektdaten **geordnet und sicher abgelegt** sind (z. B. keine Roh-Personendaten im Git, sensiblere Daten ggf. anonymisiert). Er hat zudem eine **Model Card** für sein DL-Hauptprojekt geschrieben, wo er offenlegt, welche Daten genutzt wurden und welche Vorsichtsmaßnahmen er traf (z. B. "This model was trained on public dataset X and does not contain personal data; the model is not intended to be used on sensitive personal information."). Damit zeigt er, dass er als AI-Entwickler die **Verantwortung für vertrauenswürdige KI** ernst nimmt.

- **Lernressourcen:** *AI:* **Hugging Face Course** (Kapitel zu Fine-Tuning Transformers) [26] – ideal, es führt durch ein GPT-2 Fine-Tuning mit Code. Google Colab oder Kaggle Notebooks for free GPU – er nutzt diese, um das Fine-Tuning praktisch durchzuführen. **LoRA/Peft Blog** (wie man große Modelle mit wenig Ressourcen finetuned). *Security:* **TensorFlow Privacy** oder **PyTorch Opacus** Tutorials – zeigen, wie DP in Training eingebaut wird. Artikel "**Privacy in ML**" auf arXiv (falls motiviert für Theorie). **Model Cards** (z. B. Blog von HuggingFace oder Google Model Card Toolkit docs) als Guideline, was man dokumentieren sollte. Eventuell **Kaggle's "ML & Privacy"** Diskussionen – Kaggle hatte mal Wettbewerbe zu DP, die Forenbeiträge dort sind lehrreich. Ein Podcast oder Talk zum Thema "Securing ML Models" für breiteren Kontext (z. B. von DEFCON oder BlackHat Konferenz gibt es Talks über Adversarial ML, die spannend sein könnten).

- **Projektidee: Erweiterung Hauptprojekt – Privacy Layer.** Monat 11 kann genutzt werden, um Martins großes ML-Projekt aus Monat 8/9 um eine **Privacy-Funktion** zu erweitern. Angenommen, sein Projekt verarbeitet irgendeine Form von Benutzerdaten – er könnte eine

Option einbauen, die diese Daten gleich nach Inferenz löscht oder nur aggregiert speichert (Privacy by Design Prinzip). Falls sein Projekt z. B. Bilder klassifiziert, kann er einen **Blurrer einbauen**, der Gesichter auf Bildern unkenntlich macht, bevor die Bilder ins Modell gehen (sofern anwendbar). Oder, falls Textdaten: einen **Regex-Filter**, der E-Mails/Telefonnummern aus Texten entfernt, bevor sie im Modell landen. Diese Erweiterung mag die Performance minimal beeinflussen, aber erhöht die Privacy. Er dokumentiert dies als **"Privacy Mode"** seines Tools. Zusätzlich kann er überlegen, eine abgespeckte **Demo-Version des Modells** zu veröffentlichen: z. B. trainiert er sein Modell mit absichtlich etwas Rauschen (DP) und stellt es als *privacy-preserving model* online, während das *volle Modell* privat bleibt. So kann er zeigen, dass er verstanden hat, wie man Abwägungen trifft zwischen Nutzen und Datenschutz. Insgesamt wird sein Projekt dadurch komplexer und runder – ein super Abschluss für Jahr 1.

Monat 12: Abschluss Roadmap Jahr 1 – Integration & Ausblick

- **AI-Schwerpunkt: Refactoring, Dokumentation & Spezialisierungsplan.** Im zwölften Monat fasst Martin alle Ergebnisse seines ersten Jahres zusammen. Er hat nun diverse **Portfolio-Stücke**: klassische ML-Projekte, ein Deep-Learning-Projekt, LLM-Demos. Er nimmt sich Zeit, diese Projekte zu **polieren**: Er überprüft den Code-Stil, ergänzt **Dokumentation** (Jupyter Notebooks mit Erläuterungen, READMEs mit Nutzungshinweisen) und stellt sicher, dass **alle Projekte auf GitHub lauffähig** sind (inkl. eventuell notwendiger Umgebungsdateien). Außerdem sammelt er **Lernerfolge**: Er erstellt eine persönliche Checkliste und stellt fest, dass er die Meilensteine erreicht hat – er kann ML-Algorithmen implementieren und evaluieren, hat 2-3 eigene Projekte und erste LLM-Erfahrung [7] [24]. Nun wagt er einen Blick nach vorn: In Stage 3 (Jahr 2) will er sich ja spezialisieren (LLMs, Agents, RAG) [21]. Er plant also, **Wissenslücken** gezielt zu füllen. Beispielsweise merkt er, dass er zwar LLM-Anwendungen gebaut hat, aber die **Agenten-Thematik** (AutoGPT etc.) noch kaum angerührt. Oder dass er zwar von Adversarial ML gehört hat, aber nicht praktisch probiert. Er priorisiert, was im nächsten Jahr fokussiert werden soll. Eventuell wählt er schon jetzt einen **Spezialbereich**, der ihn am meisten reizt – vielleicht merkt er, dass ihm **AI Security** selbst Spaß macht (eine mögliche Nische), oder er bleibt bei **Healthcare AI** mit Security als Zusatzkompetenz. Diese Planung hilft ihm, im kommenden Jahr gerichteter zu lernen. *Meilenstein:* Ende Jahr 1 hat Martin offiziell den **Übergang vom Quereinsteiger zum AI-Developer** geschafft. Er verfügt über ein breites Fundament in AI/ML sowie grundlegendes Security-Know-how, das er parallel aufgebaut hat, ohne den Fokus auf AI zu verlieren. Er kann eigenständig Projekte entwickeln und hat bewiesen, dass er Best Practices (von sauberem Code bis Security) mitverfolgt. Er ist bereit für fortgeschrittene Projekte oder auch erste **Zusammenarbeiten/Open-Source-Beiträge** im nächsten Jahr.

- **Security-Schwerpunkt: Ganzheitliche Wiederholung & Spezialisierungsentscheidung.** In diesem Monat schaut Martin auf das Security-Wissensspektrum, das er abgedeckt hat, und evaluiert, **wohin es ihn am meisten zieht**. Er hat Grundlagen in Web Security, Pentesting, Privacy, DevSecOps – nun könnte er entscheiden, ob er ab Jahr 2 eine dieser Sparten intensiver verfolgen will (z. B. Schwerpunkt AppSec oder MLOps/DevSecOps). Er informiert sich über fortgeschrittene Möglichkeiten: z. B. **Zertifizierungen ab Jahr 2** (OSCP für Pentesting? CISSP für generelle Security? Oder spezialisierte Kurse für „AI Security"?). Er wird im Jahr 1 bewusst keine gemacht haben, aber jetzt hat er die Grundlagen, um solche Ziele einschätzen zu können. Zudem verbindet er seine Security- und AI-Kenntnisse und reflektiert, wo Überschneidungen Zukunftspotenzial haben – etwa **"Secure AI"** als Feld (Modellabsicherung, AI im Security-Bereich einsetzen, etc.). Vielleicht beschließt er, im nächsten Jahr ein Projekt zu machen, das **AI für Security** nutzt (z. B. einen ML-basierten Anomalie-Detector für Logs) oder **Security für AI** (z. B. ein Tool zum Prüfen von ML-Modellen auf Bias/Sicherheit). Für den Moment hält er diese Ideen fest. *Meilenstein:* Martin schließt den Monat (und das Jahr) mit einem **klaren Bild seiner**

Fortschritte ab und hat einen **groben Fahrplan für Jahr 2**. Er bleibt beim wöchentlichen Habit: ~7–10h AI-Entwicklung plus ~3h Security, weil es für ihn gut funktioniert hat. Mit dem erreichten Portfolio kann er sich nun bereits selbstbewusst als **"AI Developer (mit Security Awareness)"** präsentieren – ein Profil, das ihn von manch anderen Junior-Entwicklern abheben kann.

- **Lernressourcen:** *AI:* Übersichtskurs oder Buchkapitel zur Wiederholung (z. B. **Datacamp's "AI Developer Roadmap" Blog** als Check [37] [38], oder Andrew Ng's **AI Glossary** zum Prüfen von Begriffen). *Security:* ggf. **CompTIA Security+ Study Guide** (nicht für Prüfung jetzt, aber zum Schließen etwaiger Lücken in Netzwerktechnik, Kryptografie-Grundlagen usw., die noch offen sind). **Konferenz-Videos**: z. B. DEF CON AI Village Talks (um Cutting-Edge an AI Security zu sehen), OWASP Talks. *Karriere:* Networking: Martin kann online Communities beitreten (Reddit r/MachineLearning, r/cybersecurity, Discords) um Gleichgesinnte zu finden. Mentorensuche: Plattformen wie MentorCruise – vielleicht findet er jemanden im Bereich AI Security, der ihn ab und zu berät. Das Jahr 2 kann kommen!

- **Projektidee: Portfolio & Blog**. Zum Abschluss fasst Martin alles in einem **Online-Portfolio** oder Blog zusammen. Er erstellt z. B. eine einfache **Website** (kann GitHub Pages nutzen) mit einer **Timeline seiner Projekte**, jeweils mit Beschreibung, Highlights und Link zum Code. Darin betont er auch immer die Security-Aspekte die er berücksichtigt hat – so signalisiert er potentiellen Auftrag- oder Arbeitgebern sein duales Können. Als besonderes Element kann er einen **Blogartikel** verfassen à la *"Vom Quereinsteiger zum AI-Entwickler mit Security-Fokus – Jahresrückblick"*. Darin erzählt er von seinem Lernprozess, den Projekten und warum Security sich auszahlt, selbst wenn man primär AI macht. Dieses schriftliche Projekt krönt seine Roadmap, stärkt sein eigenes Verständnis (durch Reflektion) und dient als **Selbstmarketing**. Es ist zugleich der Übergang zum nächsten Kapitel: Mit solidem Fundament in AI und Security startet Martin nun in Jahr 2, wo er diese beiden Pfade noch wirkungsvoller zusammenbringen und sich zum High-Impact AI Developer mit einzigartigem Security Profil weiterentwickeln wird. [1] [8]

---

[1] [2] [3] [4] [7] [8] [9] [10] [16] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37]
[38] Roadmap to Transition from Dentist to High-Impact AI Developer_Founder.pdf
file://file-LFRae73wZG7VaneaXE3pAC

[5] [15] Was ist OWASP? Die 10 größten Schwachstellen und Risiken von OWASP | F5
https://www.f5.com/de_de/glossary/owasp

[6] Web Security Academy: Free Online Training from PortSwigger
https://portswigger.net/web-security

[11] Your Guide to Simulated Cyberattacks: What is Penetration Testing? | Varonis
https://www.varonis.com/de/blog/penetration-testing

[12] [13] Was ist Penetration Testing? | DataGuard
https://www.dataguard.de/blog/was-ist-penetration-testing/

[14] Learn Cyber Security | TryHackMe Cyber Training
https://tryhackme.com/

[17] Was ist DevSecOps? – Sicherheitsoperationen für Entwickler erklärt – AWS
https://aws.amazon.com/de/what-is/devsecops/