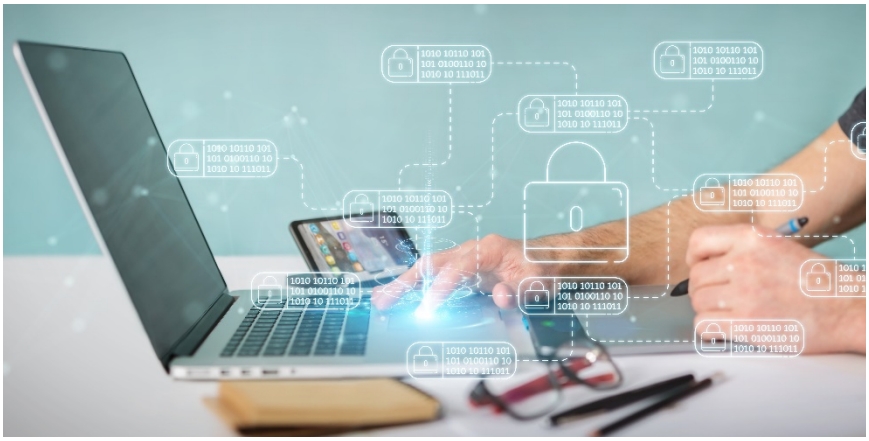
A-SIT

Zentrum für sichere Informationstechnologie - Austria

Theoretische Analyse des Vorgehens bei der Entwicklung KI-basierter Dienste: Low-/No vs. High Code



Theoretische Analyse des Vorgehens bei der Entwicklung KI-basierter Dienste

Autor:
Bianca Danczul
Mail: bianca.danczul@iaik.tugraz.at
Datum: 25.04.2023

Abstract/Zusammenfassung:
Künstliche Intelligenz wird sowohl im öffentlichen als auch im privaten Sektor mit steigender Tendenz laufend eingesetzt. Dies wird insbesondere durch die Verfügbarkeit von sogenannten Low-/No Code Entwicklungsframeworks weiter angetrieben. Doch durch einen steigenden Einsatz kombiniert mit teils wenig Verständnis aufgrund des Black-Box-Charakters ergeben sich erhöhte Anforderungen, insbesondere an Sicherheit und Datenschutz. Maßnahmen, die die zuvor genannten Anforderungen behandeln und zur Entwicklung sicherer Software beitragen, sind in der klassischen Softwareentwicklung bereits gut etabliert, nicht jedoch in der KI-Entwicklung.

Erschwerend kommt dazu, dass Entwicklungsmöglichkeiten wie No-/Low und High Code wiederum ganz unterschiedliche Sicherheitsanforderungen bergen. Ziel ist es, auf die unterschiedlichen Entwicklungsvarianten einzugehen und besondere Sicherheitsanforderungen hervorzuheben, sowie deren Unterschiede zu diskutieren.

Inhaltsverzeichnis

1. Einleitung - 2 -
2. No und Low Code - 2 -
2.1. Anwendungsbereiche und Überblick - 2 -
2.2. Vor- und Nachteile - 3 -
2.3. Entwicklungsprozess für Low Code / No Code - 4 -
2.4. Sicherheitsimplikationen - 5 -
3. High Code - 6 -
4. Fazit und Diskussion - 7 -
Literatur - 8 -

1. Einleitung

Die zunehmende Nutzung von künstlicher Intelligenz in verschiedenen Branchen und Sektoren stellt eine wachsende Herausforderung für die Sicherheit und den Datenschutz dar. Folglich ist es von großer Bedeutung, Maßnahmen zu ergreifen, um eine sichere KI-Entwicklung zu gewährleisten. Während es in der traditionellen Softwareentwicklung bereits etablierte Maßnahmen zur Gewährleistung von Sicherheit und Datenschutz gibt, sind diese in der KI-Entwicklung jedoch noch nicht so weit verbreitet. Erschwerend kommt hinzu, dass es in der KI-Entwicklung unterschiedliche Ansätze wie *No Code*, *Low Code* und *High Code* gibt, die jeweils unterschiedliche Anforderungen an die Sicherheit mit sich bringen [1].

No Code bezeichnet hierbei einen benutzerfreundlichen Ansatz in der KI-Entwicklung, der keine Programmierkenntnisse oder technisches Fachwissen erfordert, und den Benutzer*innen somit eine einfache Anwendung von vorgefertigten KI-Modellen ermöglicht, dadurch aber auch starke Risiken in Bezug auf das Verständnis birgt, was zu Sicherheitsimplikationen führen kann. *Low Code* hat eine höhere Komplexität als *No Code* und erfordert den Einsatz einfacher Skriptsprachen, allerdings nach wie vor wenig technisches Know-How und Hintergrundwissen zu den verwendeten Modellen. *High Code* schließlich ist die klassische Variante der KI-Entwicklung und bietet das höchste Maß an Flexibilität und Anpassungsfähigkeit, erfordert jedoch ein hohes Maß an technischem Fachwissen.

Aufgrund der großen Unterschiede zwischen den verschiedenen KI-Entwicklungsvarianten ist es wichtig, auf die spezifischen Sicherheitsanforderungen einzugehen. Deshalb werden in diesem Artikel die verschiedenen Ansätze zur KI-Entwicklung erläutert und die spezifischen Sicherheitsanforderungen hervorgehoben sowie die Ansätze in einer Diskussion verglichen.

2. No und Low Code

No-/Low Code KI Entwicklung ist ein Ansatz zur Erstellung von KI-Modellen und Anwendungen, der weder Programmierkenntnisse noch ein hohes technisches Fachwissen erfordert [2] und wird in der Literatur oft gemeinsam verwendet. Dieser Ansatz soll es Laien ermöglichen, KI-Modelle schnell und einfach zu erstellen, ohne komplexe Programmiersprachen lernen oder ein Expert*innenteam engagieren zu müssen. In weiterer Folge wird die grobe Vorgehensweise bei der *Low-/No* Code Entwicklung kurz beschrieben sowie auf Vorteile, Nachteile und Sicherheitsrisiken eingegangen.

2.1. Anwendungsbereiche und Überblick

Low/No-Code-KI-Entwicklungstools bieten Drag-and-Drop-Schnittstellen, vorgefertigte Module und automatisierte Workflows, mit denen Benutzer*innen KI-Modelle erstellen können, ohne Code schreiben zu müssen [3]. Diese Tools bieten in der Regel eine Reihe von KI-Funktionen, wie z.B. die Verarbeitung natürlicher Sprache, Bilderkennung und prädiktive Analytik. Zu den beliebtesten *Low-/No*-Code-KI-Entwicklungstools gehören Google AutoML[1], IBM Watson Studio[2] und Microsoft Azure Machine Learning[3]. Konkret gibt es u.a. folgende Anwendungsbereiche:

- **Chatbots und virtuelle Assistenten**: *Low-No* Code KI-Entwicklungstools wie Chatfuel[4] basierend auf GPT-4 oder ManyChat[5] sowie ChatGPT[6] bieten die Möglichkeit, Chatbots und virtuelle Assistenten ohne Programmierkenntnisse zu erstellen. Diese Tools verwenden vorgefertigte Module für die Verarbeitung

[1] https://cloud.google.com/automl
[2] https://www.ibm.com/cloud/watson-studio
[3] https://azure.microsoft.com/en-us/products/machine-learning/
[4] https://chatfuel.com/
[5] https://manychat.com/
[6] https://chat.openai.com/

natürlicher Sprache und das Gesprächsmanagement und können in beliebte Messaging-Plattformen oder Webseiten integriert werden.

- **Business Intelligence und Datenanalyse**: Microsoft Power BI[7] und Tableau[8] ermöglichen es Benutzer*innen, Datenvisualisierungen und Dashboards zu erstellen, ohne dass Programmierkenntnisse erforderlich sind. Diese Tools verwenden vorgefertigte Konnektoren und Datenmodelle und ermöglichen die Erstellung von Datenfeldern per Drag-and-Drop, um interaktive Visualisierungen zu erstellen.
- **E-Commerce**: Tools wie Zyro[9] und Wix[10] ermöglichen die Erstellung von Webseiten und Online-Shops mit KI-gestützten Produktempfehlungen, personalisierten Inhalten und automatisierter Bestellabwicklung. Membrace[11] ermöglicht KI-gestützte Contentmoderierung. Diese Tools verwenden vorgefertigte Module für die Verwaltung von Produktkatalogen, die Zahlungsabwicklung und die Versandlogistik.
- **Bilderkennung und Objekterkennung**: *Low-No-Code*-KI-Entwicklungstools wie Google Cloud Vision[12] und Clarifai[13] ermöglichen es den Anwender*innen, ohne Programmierkenntnisse Modelle zur Bilderkennung und Objekterkennung zu erstellen. Diese Tools verwenden vorgefertigte Modelle und APIs für die Bildanalyse und können in verschiedene Plattformen und Anwendungen integriert werden.
- **Sprachübersetzung**: Übersetzungsdienste wie Google Translate[14] oder DeepL[15] bieten die Möglichkeit, KI-gestützte Sprachübersetzung in Anspruch zu nehmen. Diese Tools verwenden vorgefertigte Modelle und APIs für die Verarbeitung natürlicher Sprache und können in verschiedene Plattformen und Anwendungen integriert werden.

Wie gezeigt wurde, gibt es viele Anwendungsbereiche von *Low-/No* Code. Dies spiegelt sich auch darin wider, dass es bereits in verschiedenen Domänen eingesetzt wird, wie bspw. von Rokis und Kirikova [2] gezeigt. Dazu gehören neben den bereits erwähnten E-Commerce und Business Intelligence Bereichen auch Kundenbeziehungsmanagement, Unterhaltung, Content-Management-Systeme, Automatisierung von Roboterprozessen oder Medizin. Weitere Anwendungen sind Geschäftsprozessautomatisierung in der Fertigung, Anwendungen im Bereich des öffentlichen Gesundheitswesens, Empfehlungssysteme oder Überwachung von Modellen des maschinellen Lernen selbst [2]. Auch in Smart Cities werden sie bereits eingesetzt [4, 5], sowie im Bereich von Function Block Diagram Programmen [6] oder bei Supply Chains [7].

2.2. Vor- und Nachteile

Im Folgenden werden kurz einige allgemeine Vor- und Nachteile der *Low-No Code* KI-Entwicklung angeführt, bevor auf den generischen Entwicklungsprozess und die Sicherheitsimplikationen eingegangen wird.

Die Vorteile sind [8]:

- **Benutzerfreundlichkeit:** Auch technisch nicht versierte Benutzer*innen können mit Hilfe von *Low-No-Code* KI-Entwicklungsframeworks einfach und ohne Fachkenntnisse KI-Modelle und -Anwendungen erstellen. Dadurch wird KI einem breiteren Nutzerkreis zugänglich und ermöglicht es Unternehmen, KI-Technologie zu nutzen, ohne Fachkräfte einstellen zu müssen.

[7] https://powerbi.microsoft.com/en-us/
[8] https://www.tableau.com/
[9] https://zyro.com/de/ki
[10] https://de.wix.com/
[11] https://membrace.ai/
[12] https://cloud.google.com/vision?hl=de
[13] https://www.clarifai.com/
[14] https://translate.google.com/?hl=de
[15] https://www.deepl.com/de/translator

- **Geschwindigkeit und Effizienz**: *Low-No-Code*-KI-Entwicklungstools bieten vorgefertigte Module und automatisierte Workflows, mit denen Anwender*innen schnell und effizient KI-Modelle erstellen können.
- **Skalierbarkeit**: Die *Low-No* Code KI-Entwicklungstools sind für die Verarbeitung großer Datenmengen ausgelegt und können je nach Projektbedarf leicht nach oben oder unten skaliert werden. Dies erleichtert die Erstellung von KI-Modellen und -Anwendungen, die auf verschiedenen Plattformen und Geräten verwendet werden können, ohne dass zusätzliche Entwicklungsarbeit erforderlich ist.
- **Kosteneffizient**: *Low-/No*-Code-KI-Entwicklungstools können kosteneffizienter sein als die Einstellung eines Entwicklerteams, da sie den Bedarf an spezialisierten technischen Fachkräften eliminieren und den Zeit- und Ressourcenaufwand für die Entwicklungsarbeit reduzieren.

Natürlich haben *Low-/No* Code KI-Entwicklungsframeworks nicht nur Vor- sondern auch Nachteile, welche untenstehend kurz angeführt werden:

- **Begrenzte Anpassungsmöglichkeiten [2, 3]**: Solche Tools stützen sich in hohem Maße auf vorgefertigte Module und automatisierte Workflows, was den Grad der Kontrolle und Anpassung durch die Anwender*innen stark einschränkt. Dies kann die Erstellung komplexer KI-Modelle und -Anwendungen erschweren oder unmöglich machen.
- **Eingeschränkte Funktionalität [3]**: *Low-/No-Code*-KI-Entwicklungstools weisen im Vergleich zu Low-Code- oder High-Code-KI-Entwicklungstools nur eine stark eingeschränkte Funktionalität auf, was ebenfalls die Erstellung komplexer KI-Modelle und -Anwendungen erschweren kann.
- **Sicherheitsaspekte [3]**: *Low-/No-Cod*e-KI-Entwicklungstools können u.a. Sicherheitsschwachstellen aufweisen, die von den unerfahrenen Benutzer*innen aufgrund des fehlenden Hintergrundwissens nicht erkannt werden können, und zu vielen Risiken führen, die später separat angeführt werden.
- **Abhängigkeit von Tool-Anbietern**: Oft werden *Low-/No-Code*-KI-Entwicklungstools von Drittanbietern zur Verfügung gestellt, was bedeutet, dass die Anwender*innen eine große Abhängigkeit in Bezug auf laufenden Support und Updates haben. Dies wiederum kann zu Unsicherheiten und Risiken führen, wenn der Anbieter beispielsweise den Support für das Tool einstellt.
- **Fehlendes Verständnis [2]**: Das Verständnis der Datenbasis und der Modelle ist eine Grundvoraussetzung für ein erfolgreiches Machine Learning Modell. Aufgrund des Black-Box-Wesens von *Low-/No* Code Frameworks ist für die Benutzer*innen das Ergebnis jedoch oft nicht nachvollziehbar, was zu falschen Ergebnissen und Problemen in der Fehlersuche führen kann.
- **Fehlende Testmöglichkeiten [2]:** Derzeit gibt es fehlende Möglichkeiten, *Low-/No* code Frameworks tatsächlich umfassend zu testen, was zu unerwünschten Nebeneffekten führen kann.

Wie soeben gezeigt wurde, bieten *Low-/No*-Code-KI-Entwicklungstools Zugänglichkeit, Geschwindigkeit, Effizienz, Skalierbarkeit und Kosteneffizienz, haben aber auch Einschränkungen in Bezug auf Anpassung, Sicherheitsbedenken, Abhängigkeit von Tool-Anbietern und begrenzte Funktionalität.

2.3. Entwicklungsprozess für Low Code / No Code

Gemaß Rokis und Kirikova [2] umfasst der Entwicklungsprozess von Low-Code/No-Code-Anwendungen mehrere Phasen, die sie im Zuge einer Literaturrecherche erhoben haben:

- **Anforderungsanalyse**: Dies ist der erste und kontinuierliche Schritt während des gesamten Entwicklungsprozesses, und essenziell für den Erfolg der Anwendung, beeinflusst dieser Schritt doch die Plattformauswahl, Validierungsmethoden oder Benutzer*innenanforderungen. In dieser Phase ist man sehr stark von den Angeboten der Tool-Anbieter abhängig, was eine starke Herausforderung sein kann. Zudem ist die spätere Änderung von Anforderungen schwierig, da die Tools evtl. die neuen Anforderungen nicht erfüllen. Positiv anzumerken ist, dass die Erstellung eines ersten Prototyps mit *Low-*

/No Code Entwicklungsframeworks oft schnell von statten geht und somit eine enge Abstimmung mit den Kund*innen vereinfacht.

- **Planung:** In diese Phase fällt die Machbarkeitsstudie samt der Auswahl einer geeigneten Plattform. Aufgrund der großen Anzahl von verfügbaren Plattformen und Entwicklungsframeworks gestaltet sich diese Phase schwierig, weshalb verschiedene Vergleichsmerkmale in Betracht gezogen werden können, wie graphische Benutzeroberfläche, Sicherheit, Interoperabilität, Wiederverwendbarkeit, Skalierbarkeit, Anwendungsunterstützung oder Einsatzunterstützung. Weitere angeführte Merkmale sind die Spezifikation von Rollen und Benutzern, Unterstützung der Anforderungsspezifikation und -überprüfung, Unterstützung von Tests und Validierung sowie Verfügbarkeit von Komponenten künstlicher Intelligenz. In dieser Phase ist auch essenziell zu entscheiden, ob die Abhängigkeit des Plattformanbieters in Kauf genommen werden möchte.

- **Anwendungsdesign**: In dieser Phase wird das Design festgelegt, dazu gehören beispielsweise die Konfiguration des Datenschemas für die Anwendung, z. B. die Erstellung von Entitäten, die Definition von Beziehungen, Einschränkungen und Abhängigkeiten. Weitere Inhalte sind die Definition der Benutzeroberfläche sowie die Definition des Benutzerzugriffs und der Sicherheitsverwaltung, also grob gesagt die Architektur, Skalierbarkeit und Modularität. Herausforderungen ergeben sich insbesondere in der Erweiterbarkeit, was schwer bis unmöglich sein kann, sowie in der Interoperabilität aufgrund des Fehlens von Standards. Doch auch die Skalierbarkeit kann zu Problemen führen.

- **Entwicklung**: Diese Phase umfasst die Implementierung verschiedener Operationen für Schnittstellenelemente und Implementierung von Geschäftslogikregeln und Workflows. Hierzu gehört auch die Integration externer Dienste, was die direkte Nutzung externer Dienste in der *Low-/No*-Code-Softwareentwicklung unter Verwendung von Anwendungsprogrammierschnittstellen (APIs), die von Dritten bereitgestellt werden, ermöglicht. Probleme treten hier insbesondere durch unvollständige oder falsche Dokumentationen, das Fehlen von Lernressourcen oder nicht intuitive Schnittstellen auf. Zudem sind in manchen Bereichen früher oder später doch Kenntnisse der Softwareentwicklung notwendig, was technisch wenig versierte Anwender*innen überfordern kann. Nicht zuletzt stellt die begrenzte Flexibilität der Funktionalitäten und Benutzer*innenoberflächen die Benutzer*innen vor weitere Herausforderungen, neben einer begrenzten Debugging Möglichkeit.

- **Testen**: In diese Phase fällt das Durchführen verschiedener Tests. Herausforderungen treten in dieser Phase insbesondere aufgrund von fehlender Dokumentation auf. Viele Entwicklungsplattformen bieten zwar Testtools, diese sind jedoch oft in deren Funktionalität beschränkt und bieten wenig Möglichkeiten, nicht funktionale Tests durchzuführen.

- **Einsatz**: In diese Phase fällt die Bereitstellung der Anwendung. In der Praxis treten insbesondere Probleme bei der Konfiguration und Zugänglichkeit der Anwendung auf, die durch unvollständige Plattformdokumentationen noch verstärkt werden. Auch die Versionskontrolle wird als Problem dargestellt.

- **Wartung**: Die letzte, kontinuierliche Phase ist die Wartung. Auch wenn *No-/Low* Code Plattformen prinzipiell als leicht wartbar angesehen werden, kann das Debugging neben anderer Funktionen, wie Überwachung oder Anomalieerkennung zu Herausforderungen führen.

2.4. Sicherheitsimplikationen

Low-/No Code AI-Entwicklungstools haben spezifische Sicherheitsimplikationen, die bei der Verwendung dieser Tools berücksichtigt werden sollten. In Folge sind die wichtigsten angeführt:

- **Anfälligkeit für Cyber-Angriffe [1, 8]**: *Low-/No* Code AI Development Tools können Sicherheitslücken aufweisen, die von Cyber-Angreifern ausgenutzt werden können. Diese Schwachstellen können in den vorgefertigten Modulen und Workflows, die zur Erstellung von KI-Modellen und -Anwendungen verwendet werden, oder im Tool selbst vorhanden sein. Technisch nicht versierte Benutzer*innen sind

sich dieser Schwachstellen möglicherweise nicht bewusst oder verfügen nicht über die nötigen Kenntnisse, um sie zu erkennen und zu beheben.

- **Datenschutzbedenken [8]**: *Low-/No*-Code-KI-Entwicklungstools erfordern möglicherweise den Zugriff auf sensible Daten wie persönliche Informationen, Finanzdaten oder vertrauliche Geschäftsinformationen, um KI-Modelle und -Anwendungen zu erstellen. Wenn das Tool nicht ordnungsgemäß gesichert ist, besteht die Gefahr, dass diese Daten an Unbefugte weitergegeben werden.
- **Bias [1, 8]**: Low-/No-Code Anwendungen können Bias und Desinformation begünstigen.
- **Compliance-Risiken**: *Low-/No*-Code-KI-Entwicklungstools entsprechen möglicherweise nicht den einschlägigen Datenschutzgesetzen und -vorschriften. Für Unternehmen, die diese Tools verwenden, besteht das Risiko der Nichteinhaltung und es können rechtliche und finanzielle Strafen drohen.
- **Mangel an Transparenz und Kontrolle [8]**: *Low-/No*-Code-KI-Entwicklungstools bieten möglicherweise keine Transparenz darüber, wie KI-Modelle und -Anwendungen erstellt werden, was die Identifizierung und Behebung von Sicherheits- und Datenschutzrisiken erschweren kann. Nicht versierte Benutzer*innen haben möglicherweise auch keine Kontrolle über die erstellten KI-Modelle und -Anwendungen, was es schwierig machen kann, sicherzustellen, dass diese Modelle und Anwendungen sicher sind und den einschlägigen Vorschriften entsprechen.
- **Abhängigkeit von Tool-Anbietern [8]**: *Low-/No*-Code-KI-Entwicklungstools werden häufig von Drittanbietern zur Verfügung gestellt, was bedeutet, dass die Benutzer*innen bei Sicherheitsupdates und Patches vom Tool-Anbieter abhängig sind. Wenn der Tool-Anbieter keine regelmäßigen Updates oder Support anbietet, sind die mit dem Tool erstellten KI-Modelle und Anwendungen möglicherweise durch Sicherheitslücken gefährdet.

Zusammenfassend lässt sich sagen, dass *Low-/No*-Code-KI-Entwicklungstools Auswirkungen auf die Sicherheit haben, die bei der Verwendung dieser Tools berücksichtigt werden sollten. Dazu gehören die Anfälligkeit für Cyberangriffe, Datenschutzbedenken, Compliance-Risiken, mangelnde Transparenz und Kontrolle sowie die Abhängigkeit von Tool-Anbietern. Es ist wichtig, sicherzustellen, dass *Low-/No*-Code-KI-Entwicklungstools ordnungsgemäß gesichert sind und den einschlägigen Vorschriften entsprechen, um diese Risiken zu mindern, was aber in der Praxis aufgrund des fehlenden Verständnisses ein Problem darstellen kann. Bei einem ordnungsgemäßen Einsatz können *Low-/No*-Code Entwicklungsframeworks jedoch sogar dabei helfen, Sicherheitsrisiken zu minimieren [8].

3. High Code

High Code bezeichnet die Entwicklung von KI-Applikationen mit klassischen Machine Learning (ML) Applikationen. Da dieses Thema bereits in mehreren Studien der Autorin behandelt wurde (u.a. [9]), wird in Folge nur ein kurzer Überblick gegeben, um der nachfolgenden Diskussion folgen zu können.

In der KI-basierten Softwareentwicklung sollte ein genereller Softwareentwicklungszyklus befolgt werden, der in der Regel Phasen wie Anforderungserfassung, Entwurf, Implementierung, Tests und Wartung umfasst, aber auch die Phasen Initiierung, Konzeptentwicklung, Anforderungsanalyse, Technisches Design, Entwicklung und Tests, Implementierung, Bereitstellung und Optimierung sind üblich. Erwähnenswert dabei ist, dass die in der Entwicklung die Phasen Training, Modellierung, Anwendung und Ableitung essenziell sind und ein tiefes Verständnis der statistischen Modellierung, der Datenanalyse und der Konzepte des maschinellen Lernens erfordern. Risiken und Sicherheitsbedenken treten insbesondere im Zuge der Sammlung, Analyse und Verarbeitung von Daten auf, sowie aufgrund der KI-Entscheidungen. Eliminieren lassen sich die Risiken durch umfassende Tests sowie dem Einsatz von speziellen Techniken, wie Data Augmentation oder Modellregularisierung.

4. Fazit und Diskussion

Bei der Entwicklung von KI-Systemen geht es in erster Linie darum, Modelle zu erstellen und zu trainieren, um auf der Grundlage von Eingabedaten Vorhersagen oder Entscheidungen zu treffen. Traditionell war dieser Prozess sehr technisch und erforderte spezielle Kenntnisse in Programmierung und Datenwissenschaft. Mit dem Aufkommen der Unterteilung in *Low/No*-Code- und *High*-Code-Entwicklung wurde dieser Prozess jedoch auch für nichttechnische Benutzer*innen zugänglicher.

Low/No-Code-Entwicklung für maschinelles Lernen bezieht sich auf die Verwendung von Plattformen und Tools, die wenig oder gar keine Programmierung für die Erstellung von ML-Modellen erfordern. Diese Plattformen verwenden oft Drag-and-Drop-Schnittstellen und vorgefertigte Komponenten, um den Benutzern die Erstellung von Modellen zu erleichtern, ohne dass sie selbst Code schreiben müssen. Einige beliebte Beispiele für ML-Entwicklungsplattformen mit geringem/keinem Code sind Google AutoML oder IBM Watson Studio.

Die *High-Code*-Entwicklung für maschinelles Lernen bezieht sich dagegen auf den traditionellen Prozess der Erstellung von ML-Modellen mit Programmiersprachen wie Python oder R. Bei diesem Ansatz wird Code geschrieben, um Algorithmen zu implementieren, Daten zu verarbeiten und Modelle fein abzustimmen. Bei der ML-Entwicklung mit hohem Codeanteil haben die Entwickler mehr Kontrolle über den Prozess der Modellerstellung und können ihn besser anpassen, was jedoch auch mehr technisches Fachwissen erfordert.

Generell lässt sich sagen, dass die *Low/No*-Code-Entwicklung für maschinelles Lernen oft schneller und leichter zugänglich ist als die *High*-Code-Entwicklung. Sie eignet sich daher ideal für technisch nicht versierte Benutzer*innen oder Personen mit begrenzten Ressourcen. Diese Plattformen enthalten außerdem häufig integrierte Funktionen für die Datenvorverarbeitung, Modellauswahl und -auswertung, so dass es einfacher ist, genaue Modelle ohne umfassende Kenntnisse der Datenwissenschaft zu erstellen.

Low-/No-Code-Plattformen haben jedoch auch ihre Grenzen. Sie bieten möglicherweise nicht dasselbe Maß an Flexibilität und Anpassung sowie Sicherheit wie die *High*-Code-Entwicklung, und haben starke Einschränkungen im Funktionsumfang, der Datenmenge oder der Art der Modelle. Zudem ist das Verständnis der Modelle durch ihren Black-Box-Charakter sehr schwer, was zu zusätzlichen Sicherheitsrisiken und wenig Testmöglichkeiten führen kann.

Die High-Code-Entwicklung für maschinelles Lernen bietet Entwickler*innen dagegen mehr Kontrolle und Flexibilität bei der Modellerstellung. Entwickler*innen können spezifischen Code schreiben, um fortschrittliche Algorithmen zu implementieren oder Daten auf eine Weise vorzuverarbeiten, die speziell auf die Bedürfnisse ihrer Organisation zugeschnitten ist. Dieser Grad der Anpassung ermöglicht genauere Modelle, erfordert jedoch ein höheres Maß an technischem Know-how und kann naturgemäß zeitaufwändiger und ressourcenintensiver sein.

Zusammenfassend lässt sich sagen, dass sowohl die *Low/No*-Code- als auch die *High*-Code-Entwicklung für maschinelles Lernen ihre Vor- und Nachteile haben, und dass die Entscheidung für eine der beiden Methoden von Faktoren wie dem technischen Know-how der Benutzer*innen, den verfügbaren Ressourcen und dem spezifischen Anwendungsfall abhängt. Für einfache Projekte oder solche mit begrenzten Ressourcen kann die *Low/No*-Code-Entwicklung die beste Option sein, während die *High*-Code-Entwicklung für komplexere Projekte oder solche, die fortgeschrittene Algorithmen oder Anpassungen erfordern, notwendig sein kann. Die Sicherheit darf jedoch nie vernachlässigt werden.

Literatur

[1] G. Hurlburt, „Low-Code, No-Code, What's Under the Hood?“, *IT Prof.*, Jg. 23, Nr. 6, S. 4–7, 2021, doi: 10.1109/MITP.2021.3123415.

[2] K. Rokis und M. Kirikova, „Challenges of Low-Code/No-Code Software Development: A Literature Review“ in *Lecture Notes in Business Information Processing, PERSPECTIVES IN BUSINESS INFORMATICS RESEARCH: 21st international conference*, Ē. Nazaruka, K. Sandkuhl und U. Seigerroth, Hg., [S.l.]: SPRINGER INTERNATIONAL PU, 2022, S. 3–17, doi: 10.1007/978-3-031-16947-2_1.

[3] M. Woo, „The Rise of No/Low Code Software Development-No Experience Needed?“ (eng), *Engineering (Beijing, China)*, Jg. 6, Nr. 9, S. 960–961, 2020, doi: 10.1016/j.eng.2020.07.007.

[4] Fitsilis und P. Fitsilis, Hg., *Building on Smart Cities Skills and Competences: Human Factors Affecting Smart Cities Development*, 1. Aufl. Cham: Springer, 2022.

[5] A. ElBatanony und G. Succi, „No-Code for Smart Cities“ in *Internet of Things, Building on Smart Cities Skills and Competences: Human Factors Affecting Smart Cities Development*, Fitsilis und P. Fitsilis, Hg., 1 Aufl., Cham: Springer, 2022, S. 247–256, doi: 10.1007/978-3-030-97818-1_15.

[6] O. Ogundare, G. Q. Araya und Y. Qamsane, „No Code AI: Automatic Generation of Function Block Diagrams from Documentation and Associated Heuristic for Context-Aware ML Algorithm Training“ in *2022 7th International Conference on Mechanical Engineering and Robotics Research (ICMERR)*, Krakow, Poland, 2022, S. 191–195, doi: 10.1109/ICMERR56497.2022.10097820.

[7] S. K. Jauhar, S. M. Jani, S. S. Kamble, S. Pratap, A. Belhadi und S. Gupta, „How to use no-code artificial intelligence to predict and minimize the inventory distortions for resilient supply chains“, *International Journal of Production Research*, S. 1–25, 2023, doi: 10.1080/00207543.2023.2166139.

[8] Z. Yan, „The Impacts of Low/No-Code Development on Digital Transformation and Software Development“, 28. Dez. 2021. [Online]. Verfügbar unter: https://arxiv.org/pdf/2112.14073.

[9] Bianca Danczul, *Sichere Entwicklung von KI-Diensten.* [Online]. Verfügbar unter: https://technology.a-sit.at/sichere-entwicklung-von-ki-diensten/.