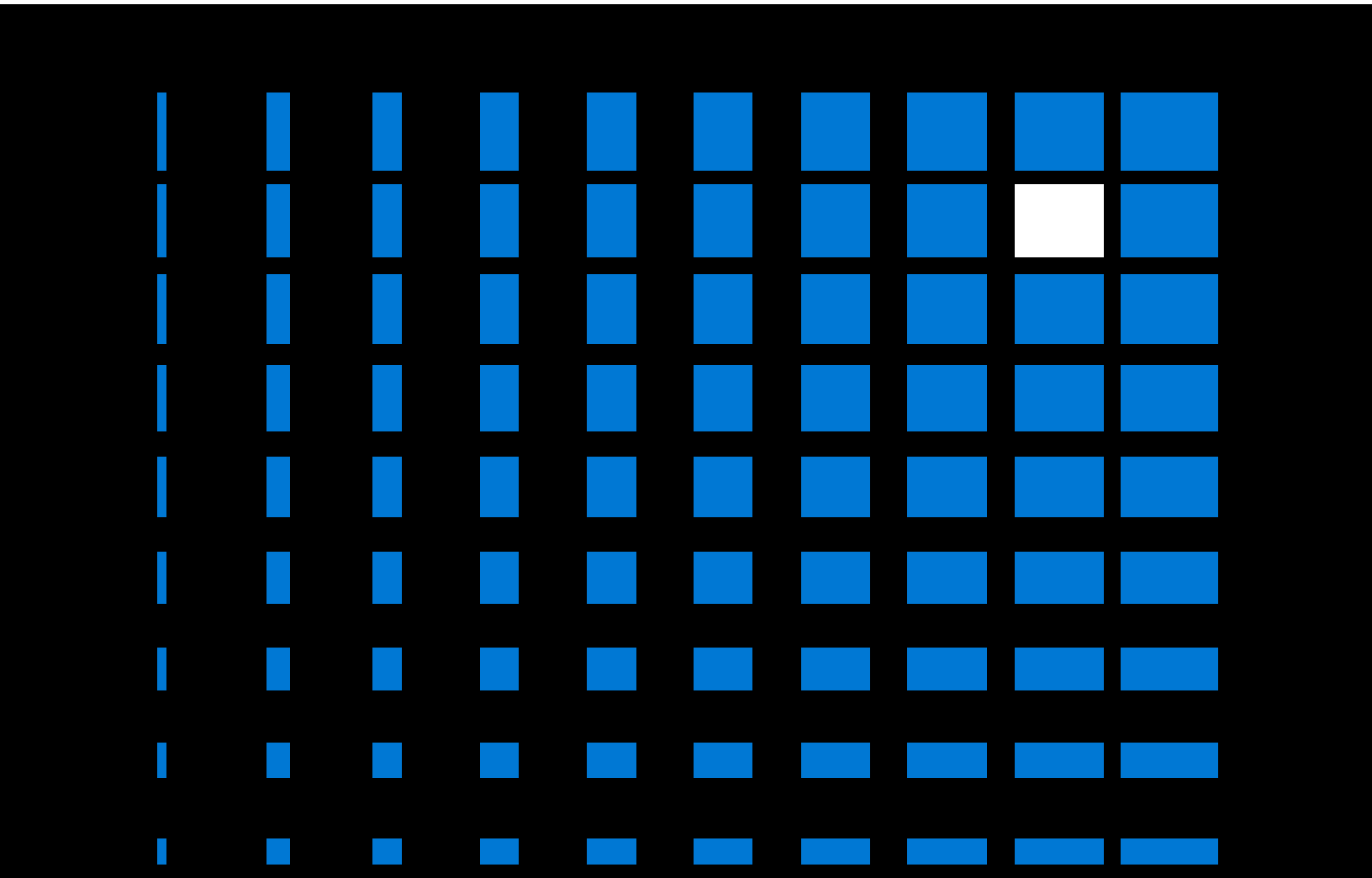
Whitepaper

Microsoft Azure

Höhere Effizienz und Produktivität mit Machine Learning-Prozessen

Implementieren Sie DevOps für Ihre Machine-Learning-Pipeline



Wenn Sie Data Scientist oder Machine Learning (ML) Engineer sind, ist Azure Machine Learning eine optimale Option für Sie. Integrierte MLOps erleichtern die Implementierung von DevOps für Machine Learning durch umfassenden Support aller Phasen des ML-Lebenszyklus.

Dank der Funktionalität von Azure Machine Learning sind Sie nicht länger zu ineffizientem Arbeiten gezwungen und auch nicht mehr mit banalen Aufgaben überlastet. Sie haben die nötigen Tools zur Hand, um den End-to-End-ML-Lebenszyklus zu optimieren, zu automatisieren und in vorhandene DevOps-Prozesse einzubinden, und können so mit den App-Entwicklern kooperieren und bei der Entwicklung ML-gestützter Apps in demselben Rhythmus arbeiten.

Zielgruppe dieses Whitepapers
Dieses Whitepaper wurde für ML-Experten geschrieben, die nach einer besseren Arbeitsweise suchen – mit Zugriff auf dieselben Arten von komplexen Tools und optimierten Engineering-Verfahren, die für Entwickler in einer modernen DevOps-Umgebung selbstverständlich sind. Zunächst wollen wir vorhandene ML-Workflows untersuchen und dabei auch auf einige große Lücken in Hinblick auf Tools und Automatisierung eingehen. Anschließend werden wir MLOps und die Voraussetzungen hierfür vorstellen. Danach werden wir erörtern, wie die Features und Funktionen von Azure Machine Learning in die MLOps- und DevOps-Lebenszyklen integriert werden können. Am Ende des Whitepapers finden Sie einen kurzen Leitfaden für den Einstieg in Azure Machine Learning. Hier erfahren Sie auch, wie Sie alle Möglichkeiten zur Produktivitätssteigerung erkunden können.

Starten Sie noch heute mit Azure Machine Learning

Für die Verwendung von Azure Machine Learning benötigen Sie ein Azure-Konto. Wenn Sie noch nicht über ein Azure-Abonnement verfügen, können Sie kostenlos ein Konto erstellen und erhalten eine Gutschrift über 200 USD, die Sie in den ersten 30 Tagen einsetzen können.

1. Besuchen Sie die Seite für das kostenlose Konto für Azure Machine Learning, und wählen Sie **Jetzt kostenlos einsteigen** aus.
2. Melden Sie sich mit Ihrem Microsoft-Konto (oder Ihrem GitHub-Konto) an. Wenn Sie noch nicht über ein Microsoft-Konto verfügen, können Sie eines erstellen.
3. Geben Sie Ihren Namen, Ihre Telefonnummer und Ihre E-Mail-Adresse ein.
4. Bestätigen Sie Ihre Identität mit einem Textcode und Ihren Kreditkarteninformationen. (Ihre Karte wird erst bei Abschluss der Testversion belastet, wenn Sie bereit sind, sich für einen Abonnementplan zu entscheiden.)

Nach Auswahl von **Weiter** können Sie direkt mit der Verwendung von Azure Machine Learning beginnen. Informieren Sie uns, wenn Sie Fragen haben.

Inhalt

Inhalt

Einführung

Wenn Sie in irgendeiner Form mit Code arbeiten, haben Sie wahrscheinlich schon von DevOps gehört. Dieser Ansatz der Lebenszyklusverwaltung für Anwendungen nutzt eine (idealerweise vollständig automatisierte) Pipeline für Continuous Integration/Continuous Deployment (CI/CD), um den Prozess des Entwickelns, Testens und Bereitstellens von neuem Code in einer Produktionsumgebung zu optimieren. Dies bietet mehrere Vorteile – angefangen bei einer einfacheren Arbeitsweise für Anwendungsentwickler bis hin zur schnelleren und effizienteren Operationalisierung von neuem Code –, die alle dafür sorgen, dass schneller ein geschäftlicher Nutzen erzielt werden kann.

Zwar haben sich die Ziele von DevOps in den letzten Jahren nicht geändert, die Möglichkeiten zur Erreichung dieser Ziele haben sich jedoch weiterentwickelt. Es werden jetzt neue Technologien unterstützt. Cloudbasierte Apps und Containerisierung sind zwei gute Beispiele hierfür. Doch wie sieht es mit Machine Learning (ML) aus, das in modernen Anwendungen eine immer größere Rolle spielt? Empfehlungssysteme, Bildklassifizierungen, Gesichtserkennungsscanner, Betrugserkennungs-Pipelines für Werbeklicks und Spracherkennungsdienste sind gute Beispiele für ML-gestützte Szenarien. Ermöglicht werden solche Szenarien von den Data Scientists, die ML-Modelle entwickeln und validieren, sowie von den ML Engineers, die diese Modelle verpacken, bereitstellen und ausführen. Diese Personen haben jedoch selten Zugriff auf die komplexen Tools und optimierten, standardisierten Engineering-Verfahren, die für Entwickler in einer modernen DevOps-Umgebung selbstverständlich sind.

Dies hat zur Folge, dass Data Scientists und ML Engineers oft ineffizient arbeiten und mit manuellen Aufgaben überlastet sind, wie beispielsweise der Übergabe an die Entwickler, in deren Apps ihre ML-Modelle letztendlich ausgeführt werden. Dadurch ist es schwer möglich, die ML-Modelle in demselben Rhythmus – und in denselben DevOps-Prozessen – wie die Anwendung zu verwenden. Letztendlich verlangsamt dies die Bereitstellung ML-fähiger Anwendungen und verringert die Rentabilität von Investitionen in solche Lösungen.

Im weiteren Verlauf dieses Whitepapers werden wir:

- vorhandene ML-Workflows genauer untersuchen und dabei auch auf einige große Lücken in Hinblick auf Tools und Automatisierung eingehen.
- *MLOps* – ein Konstrukt, das Data Scientists und ML Engineers die moderne Welt der DevOps erschließt – und die Voraussetzungen hierfür vorstellen.
- Erörtern, wie Azure Machine Learning diese MLOps-Vision ermöglicht, und dabei auch die wichtigsten Funktionen und die Möglichkeiten ihrer Integration in die kompletten ML- und DevOps-Lebenszyklen erläutern.

01 /

Festlegung der Herausforderung

DevOps ist die Standardmethode zur Verwaltung von Anwendungslebenszyklen über eine Pipeline von Tools zur Versionskontrolle, für Tests, zum Entwickeln und Bereitstellen. Idealerweise bedeutet dies eine vollständig automatisierte CI-/CD-Pipeline – vom Softwareentwickler, der den Code zur zentralen Versionskontrolle (meistens ein Git-Repository) übermittelt, bis hin zum Entwickeln, Testen und Bereitstellen in einer Produktionsumgebung.

Wie könnte eine ähnliche CI-/CD-Lösung für ML aussehen? Hierfür wären Pipelines für die Automatisierung, die Validierung der Funktionalität und Leistung von Modellen sowie den Support für die Bereitstellung in der für Rückschlüsse verwendeten Infrastruktur notwendig. Dies stellt eine besondere Herausforderung dar, wenn sich Daten im Laufe der Zeit ändern und Modelle regelmäßig neu trainiert werden müssen, wie dies bei vielen großen, ML-gestützten Systemen der Fall ist. Die Komplexität nimmt weiter zu, wenn die Modelle in einer hybriden Umgebung mit Intelligent Edge plus intelligenter Cloud bereitgestellt werden.

Da das Gebiet des Machine Learning jedoch im Vergleich zur herkömmlichen Softwareentwicklung noch jung ist, müssen sich Best Practices und Lösungen rund um das ML-Lebenszyklusmanagement erst noch etablieren. Die Modellentwicklung erfolgt oft am Laptop des Data Scientist, vielleicht in einem Jupyter-Notebook oder einem anderen Tool, und die Orchestrierung wird häufig manuell oder ad hoc unter Verwendung von benutzerdefiniertem Code und Skripten durchgeführt – wie dies auch bei der herkömmlichen App-Entwicklung vor dem Aufkommen von DevOps-Best Practices der Fall war.

Heutige Arbeitsweise

DevOps-Teams sind bestrebt, alle ihre Workflows zu automatisieren, und dies ist ihnen heute größtenteils gelungen. Data Scientists befinden sich jedoch in der Regel in einer isolierten Position abseits der Automatisierung. Dadurch ist es schwer, die ML-Modelle in demselben Rhythmus – und in denselben DevOps-Prozessen – wie die Anwendungsentwickler zu verwenden, die über bewährte Methoden für Entwicklung, Tests und Auslieferung von Apps verfügen.

Für Entwickler beginnt der Prozess mit dem Schreiben von Code, der in das Front-End der DevOps-Pipeline eingecheckt wird. Andererseits neigen Data Scientists dazu, viel mehr im Vorfeld zu experimentieren. Sie verbringen viel Zeit damit, die Daten zu strukturieren, herauszufinden, welche Funktionen für ihr Szenario am wichtigsten sind und welcher Algorithmus zur Lösung ihres Problems am besten geeignet ist, und die zugehörigen Hyperparameter zu optimieren. Erst nachdem sie das richtige Rezept für all das gefunden haben, sind sie für das Training ihres ML-Modells bereit – den ersten Schritt in der von uns definierten ML-Pipeline. (Es ist zwar nicht Thema dieses Whitepapers, doch schon vorab fällt eine Menge Arbeit für die Dateningenieure an, die in der Regel die Daten bereinigen, standardisieren und anderweitig aufbereiten, bevor sie sie den Data Scientists übergeben.)

Nachdem die Data Scientists ein praktikables ML-Modell erstellt haben, ist gewöhnlich der Zeitpunkt gekommen, an dem sie anfangen, mit den Entwicklern zu interagieren, um dieses Modell in eine App zu integrieren. Gewöhnlich erfolgt dies durch manuelle Einbettung oder die Freigabe von einfachen Benutzeroberflächen. Die Integrationskosten können steigen, wenn Code für die Featurebereitstellung neu geschrieben werden muss, damit er mit dem Rückflussstapel verwendet werden kann, wenn die beim Training verfügbaren Daten nicht mit den für die Rückschlüsse verfügbaren Daten übereinstimmen, usw.

Was fehlt?

Heute funktioniert der Prozess ziemlich genau so. Wir wollen nun untersuchen, wo erste Schwachstellen zu finden sind.

Zunächst benötigen Sie eine **Quellcodeverwaltung**, um neben den Entwicklern auch mit anderen Data Scientists zusammenarbeiten zu können. Um die Reproduzierbarkeit zu gewährleisten, ist es entscheidend, dass alle Artefakte, die zum Generieren von ML-Modellen verwendet werden (auch als Trainingscode und Eingabedaten), über die Quellcodeverwaltung erfasst werden. Dies ist wichtig, da Modelle in vielen Situationen neu trainiert und bereitgestellt werden müssen, etwa wenn das Modell von Zeitreihendaten oder einer bestimmten Reihe von Eingabesignalen abhängig ist, die sich im Laufe der Zeit ändern können.

Als Nächstes benötigen Sie eine **reproduzierbare Trainingspipeline**, um die Modellentwicklung durch Automatisierung wiederkehrender Aufgaben zu optimieren. Durch Automatisierung können neue Modelle wiederum schneller generiert werden. Dadurch werden eine **Speicherung und Versionskontrolle der Modelle** notwendig. Dies sollte in einer Form erfolgen, die eine einfache Auffindbarkeit, Freigabe und Zusammenarbeit ermöglicht. Auch Zugriffskontrolle und Rückverfolgbarkeit sind wichtig, damit Sie kontrollieren können, wer worauf Zugriff hat (und wissen, wer was verwendet hat).

Nachdem Sie über das gewünschte Modell verfügen, müssen Sie sich um die **Modellverpackung** kümmern. In diesem Zusammenhang müssen die Abhängigkeiten erfasst werden, die für die Ausführung des Modells in der gewünschten Rückschlussumgebung erforderlich sind. Containerisierung ist die naheliegende Lösung. Container sind heute die faktische Ausführungseinheit sowohl in der Cloud als auch am Intelligent Edge. Sie sollten auch Modellformate in Betracht ziehen, die vom Trainings- und Bereitstellungs-Fabric unabhängig sind. Hier können wiederverwendbare Formate wie Open Neural Network Exchange (ONNX) nützlich sein.

Als Nächstes müssen Sie sich um die **Modellvalidierung** kümmern. Diese Validierung kann aus einfachen, grundlegenden Komponententests für den Trainingscode, einem A/B-Vergleich mit einer früheren Version eines Modells oder einer umfassenden Reihe von Funktions- und Leistungstests bestehen.

Wenn Modelle innerhalb einer automatisierten Pipeline reproduzierbar, Speicherung und Versionsverwaltung für die Modelle etabliert und Validierungsmethoden eingerichtet sind, verfügen Sie damit über eine End-to-End-Pipeline, die die **Bereitstellung** neuer Modelle auf einer Vielzahl von Plattformen in allen Szenarien unterstützen kann.

Schließlich müssen Sie die Möglichkeit haben, Ihre Modelle in einer Produktionsumgebung zu **überwachen**, um zu verstehen, welche Daten an Ihr Modell gesendet werden und welche Prognosen dieses zurückgibt. Sie sollten auch den Datendrift zwischen Ihrem Trainings-Dataset und den Rückschlussdaten überwachen, damit Sie wissen, ob und wann Ihr Modell **neu trainiert** und wieder in Ihre automatisierte Pipeline eingespeist werden muss.

Der gewünschte Zustand

Der zuvor beschriebene End-to-End-ML-Lebenszyklus sieht ungefähr wie in Abbildung 1 dargestellt aus. In größeren Unternehmen könnte ein Data Scientist für die ersten Schritte und ein ML Engineer für den Rest verantwortlich sein. In kleineren Unternehmen könnte ein Data Scientist für alle Schritte zuständig sein. Aus Gründen der Einfachheit ordnen wir alle diese Schritte im weiteren Verlauf dieses Whitepapers dem Data Scientist zu.

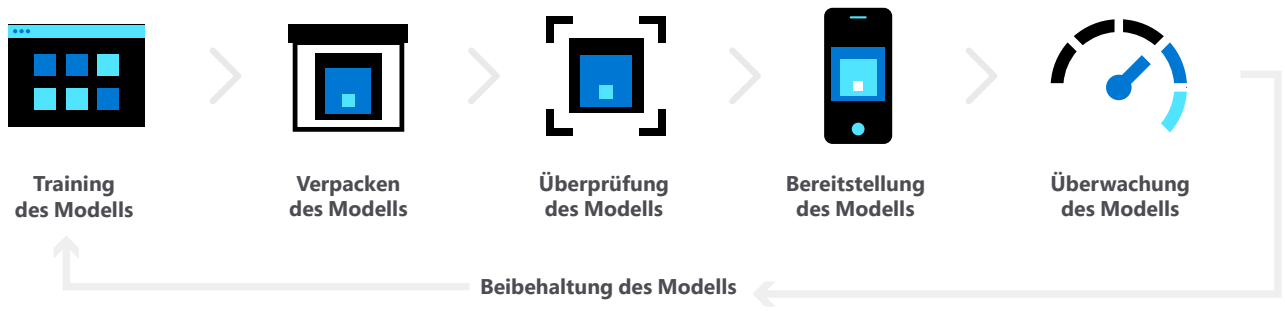


Abbildung 1. Der End-to-End-ML-Lebenszyklus.

02 /

MLOps – ein Lebenszyklus-konzept

MLOps, ein relativ neuer Trend in der Branche, soll Data Scientists und ML Engineers die Möglichkeiten bieten, die Entwicklern mit DevOps zur Verfügung stehen. Es handelt sich nicht um ein Produkt oder einen Dienst, sondern um ein Konzept oder eine Arbeitsweise. MLOps ließe sich als Ansatz zur Optimierung des End-to-End-ML-Lebenszyklus beschreiben, der in bestehende DevOps-Prozesse und -Tools eingebunden werden kann, damit Data Scientists, ML Engineers und App-Entwickler effektiv zusammenarbeiten und im gleichen Rhythmus auf die Bereitstellung ML-gestützter Apps hinarbeiten können.

Eine ziemlich genaue Beschreibung, aber auch ein hochgestecktes Ziel. Lassen Sie uns also weiter ins Detail gehen und prüfen, was genau ein MLOps-Ansatz bieten muss. Welche Funktionalität wird benötigt? Können vorhandene DevOps-Prozesse und -Tools dies bieten? Wenn ja, in welchem Umfang? Und welche Lücken bleiben bestehen?

Um diese Fragen zu beantworten, wollen wir zunächst die wichtigsten Anforderungen analysieren, die im Abschnitt Was fehlt? in Fettdruck angegeben sind. Zur Erinnerung. Es geht um folgende Anforderungen:

- Quellcodeverwaltung und Reproduzierbarkeit des Modells – Prozesse und Verfahren, um die Modelle reproduzierbar zu machen (von der Quellcodeverwaltung bis hin zu Datenaufbewahrungsrichtlinien).
- Versionskontrolle und Speicherung der Modelle – Ermöglichen einer einfachen Freigabe, Zusammenarbeit und Wiederverwendung von Modellen in einer Umgebung mit umfassender Zugriffskontrolle und Rückverfolgbarkeit.
- Modellverpackung – beispielsweise Containerisierung zum Erfassen aller Abhängigkeiten und Verwendung von ONNX für die Modellinteroperabilität und Wiederverwendung in einer Vielzahl von Rückschlussstapeln.
- Modellvalidierung – Komponenten-, Funktions- und Leistungstests (sowohl isoliert als auch in eine Anwendung eingebettet).
- Modellbereitstellung – ein effizienter Prozess zum Erstellen eines Modells in einer Anwendung oder einem Dienst, den Anforderungen der Situation des Anwenders entsprechend.
- Kontinuierliches erneutes Training – als Möglichkeit für den Umgang mit den vielen Situationen, in denen Modelle auf der Grundlage der Verfügbarkeit neuer Daten, sich ständig weiterentwickelnder Daten oder anderer Signale neu trainiert werden müssen.

Auf den folgenden Seiten werden wir uns diese wichtigen Anforderungen etwas genauer ansehen. Natürlich müssen Sie auch grundlegende Elemente, wie z. B. Unterstützung gängiger Sprachen und Frameworks, berücksichtigen.

Quellcodeverwaltung und Reproduzierbarkeit

Auch wenn viele Data Scientists die Modellentwicklung in einem Notebook oder einer anderen Umgebung beginnen, die keine Quellcodeverwaltung erfordert, kann dies in frühen Phasen dennoch für die Wiederverwendbarkeit und Zusammenarbeit nützlich sein. Wenn überhaupt keine Quellcodeverwaltung erfolgt, können mehrere Probleme auftreten:

- Modelle können nicht reproduziert werden, und es ist nicht rückverfolgbar, wie sie ursprünglich erstellt wurden.
- Governance- und Compliance-Anforderungen sind schwerer zu erfüllen.
- Teams können nicht modellübergreifend zusammenarbeiten.
- Der Quellcode ist für jedes Modell anders strukturiert.
- Data Scientists können Modelle nicht klonen und trainieren.
- Für App-Entwickler ist es schwieriger, herauszufinden, wie ein Modell funktioniert und wie es zu verwenden ist.

Dies sind nur einige Gründe, warum Quellcodeverwaltung ein notwendiger erster Schritt ist – eine Voraussetzung für die Reproduzierbarkeit des Modells, die Zusammenarbeit, Automatisierung, Rückverfolgbarkeit und mehr. Wenn Sie den Trainingscode *und* die Eingabedaten, die zum Generieren eines Modells verwendet wurden, nicht erfassen, können Sie nie genau verfolgen, welche Eingaben zur Ausgabe eines Modells geführt haben, das Sie in der Produktion verwenden. Natürlich gilt es hier, ein empfindliches Gleichgewicht zu wahren, insbesondere wenn es um Compliance-Anforderungen wie die EU-Datenschutz-Grundverordnung geht: Zwar soll durch lange Aufbewahrung von Daten die Reproduzierbarkeit sichergestellt werden, Sie müssen den Kunden jedoch die Kontrolle über ihre Daten geben, einschließlich der Möglichkeit, sie zu löschen.

Versionsverwaltung und Speicherung

Mit zunehmender Reife des Lebenszyklus der Modellentwicklung, wenn eine immer größere Zahl von Data Scientists immer mehr Modelle erstellt, steigt die Notwendigkeit einer effektiven Versionsverwaltung und Speicherung der Modelle. Ohne dies:

- **werden Freigabe, Zusammenarbeit und Wiederverwendung schwieriger.** Ohne ein zentrales System zum Protokollieren aller Modelle, mit dem die Modelle durchsuchbar werden und von den Engineers als Microservices bereitgestellt werden können, stellen Unternehmen in der Regel fest, dass sich der Arbeitsaufwand für manche Teams verdoppelt, während andere Teams ein Modell benötigen, aber nicht über die notwendigen Ressourcen für die Entwicklung verfügen.
- **können Sie nicht kontrollieren, wer worauf Zugriff hat (und wissen nicht, wer was verwendet hat).** Das geistige Eigentum eines Unternehmens für jeden in dem Unternehmen frei zugänglich zu machen, ist schlechte IT-Governance. Daher müssen Lösungen für die Modellspeicherung Zugriffskontrolle beinhalten. In diesem Zusammenhang sollte es Möglichkeiten geben, zu verwalten, wer Modelle bereitstellen und wer darauf zugreifen kann. Außerdem sollte vollständige Rückverfolgbarkeit gewährleistet sein, damit Sie jederzeit feststellen können, wer was bereitgestellt oder wer worauf zugegriffen hat.

Es gibt verschiedene Strategien für die Versionsverwaltung von Modellen. Je nach Größe (z. B. beim Vergleich traditioneller ML-Klassifizierungen mit Deep-Learning-Netzwerken) können Modelle direkt in die Quellcodeverwaltung eingebettet oder in einem Paket-Feed mit Versionsverwaltung gespeichert werden. Unabhängig davon, welche Methode Sie verwenden, sollten Sie berücksichtigen, dass erfahrene Teams überprüfte, wiederverwendbare Modelle wünschen. Um dies zu unterstützen, muss die Versionsverwaltung der Modelle in irgendeiner Form den Experimentverlauf einbeziehen, unabhängig davon, ob dieser in einem Notebook, der Quellcodeverwaltung oder einer zentralen Datenbank erfasst wird. Die Experimente können aus einer Vielzahl von Quellspeicherorten stammen, wie z. B. einem SDK, einer IDE oder einem CI-Auftrag – alle können nachverfolgt werden. Idealerweise sollten Sie sowohl Informationen über das Modell selbst als auch über seine relative Genauigkeit für ein bestimmtes Szenario erfassen.

Darüber hinaus sollten Sie die Modellinteroperabilität als eine Möglichkeit berücksichtigen, den Data Scientists das Experimentieren mit einer Vielzahl flexibler Tools zu erleichtern – und letztendlich neue Ideen schneller in die Produktion zu bringen. Eine Framework-übergreifende Zwischensprache zum Speichern von Modellen (z. B. ONNX) kann die Wiederverwendung von Modellen in einer Vielzahl von Rückschlussstapeln einfach machen – und den Wechsel von Frameworks im Trainingsprozess ohne massives Neuschreiben von Code auf der Rückschlussseite ermöglichen.

Verpackung

Bevor Sie ein Modell einsetzen können, müssen Sie es für Ihre Ziel-Rückschlussumgebung verpacken. Container sind hier die logische Wahl, da es sich dabei um die standardmäßige Ausführungseinheit sowohl in modernen Cloudumgebungen als auch am Intelligent Edge handelt. Container bieten auch eine einfache Möglichkeit, Laufzeitabhängigkeiten – darunter Conda-Umgebungen, Python-Bibliotheken mit Versionsverwaltung und andere Bibliotheken, auf die das Modell verweisen könnte – zu erfassen, die alle für die ordnungsgemäße Ausführung des Modells erforderlich sind.

Beim Verpacken Ihres Modells sollten Sie auch offene Formate wie Open Neural Network Exchange (ONNX) in Betracht ziehen, die zur Optimierung der Leistung beim Rückschluss beitragen können. Mit ONNX haben Sie die Möglichkeit, ein Modell mit Ihrem bevorzugten Toolsatz in einer Umgebung zu trainieren und es dann für Rückschlüsse und Prognosen in einer anderen Umgebung bereitzustellen. PyTorch, MXNet, Caffe2 und einige weitere gängige Frameworks bieten native Unterstützung für ONNX, und für andere Frameworks wie TensorFlow, Core ML und scikit-learn gibt es Konverter. ONNX ist auch mit mehreren Laufzeiten, Compilern und Schnellansichten kompatibel. (Eine vollständige Liste aller Tools, die ONNX unterstützen, finden Sie hier.)

Validierung

Es wird eine effiziente Möglichkeit der Modellvalidierung benötigt, um die Modelle kontinuierlich zu trainieren, die Qualität zu gewährleisten und letztlich zu verhindern, dass schlechte Modelle in die Produktion gelangen. Ein Modell muss nicht nur die gewünschte Funktionalität bieten und den Leistungsanforderungen genügen, sondern darf auch nicht abstürzen oder Fehler verursachen, wenn es geladen wird oder ungültige oder unerwartete Eingaben erhält. Schließlich darf es nicht zu viele Systemressourcen beanspruchen.

Idealerweise besteht die Modellvalidierung aus zwei Teilen: Komponenten- und Integrationstests des Modells selbst sowie Funktions- und Leistungstests des in eine App oder einen Dienst eingebetteten Modells. Wenn Sie beispielsweise ein Modell mit Eingabedaten in einem anderen Format trainieren, als für den Rückschlussdienst verfügbar, könnte es sein, dass das Modell zwar während des Trainings gut funktioniert, nicht jedoch in der Produktion.

- **Testen des Modells selbst.** Herkömmliche Komponenten- und Integrationstests, die an einer kleinen Anzahl von Eingaben ausgeführt werden, sollten stabile Ergebnisse liefern. Zu beachten ist, dass es sich anders als bei einer herkömmlichen (Nicht-ML-)App um statistische Ergebnisse handelt. Es gibt also eine Reihe akzeptabler Werte (ein erwartetes Ziel und seine Entwicklung) gegenüber einem begrenzten Abnahmekriterium. Dies kann auch Tests der zum Erstellen des Modells verwendeten Daten beinhalten, um sicherzustellen, dass diese in Bezug auf Schema und Funktionen den Daten entsprechen, die im Bewertungsszenario verfügbar sein werden.
- **Gemeinsames Testen der App und des Modells.** Sie sollten sicherstellen, dass sich Ihr Modell im Kontext Ihrer größeren App korrekt verhält. Zu diesem Zweck können Sie eine vorhandene Version Ihrer App (oder einer Stub-Variante hiervon) verwenden, um relevante Teile der Testsuite der Host-App auszuführen. Mithilfe solcher Tests kann auch sichergestellt werden, dass die Datenschemas (Eingabe/Ausgabe) und Verhaltensweisen für alle Basisfälle in einer Anwendung ausreichend abgedeckt sind. Mit zunehmender Entwicklung erstellen die meisten Unternehmen einen benutzerdefinierten Stack für diese Ebene der Modellvalidierung.

Bereitstellung

In Bezug auf die Zusammenarbeit zwischen Data Scientists und App-Entwicklern stellt der Prozess der Modellbereitstellung einen der größten Schwachpunkte dar. Typische Probleme sind die Schwierigkeit, das Modell in eine App oder einen Dienst einzubinden, fehlende Mechanismen, um die Auslieferung einer funktionalen Regression oder Leistungsregression zu vermeiden, und der große zeitliche (und manuelle) Aufwand, der mit Staging- und Freigabeprozessen verbunden ist.

Durch eine Optimierung der Modellbereitstellung bleibt den Data Scientists mehr Zeit, sich auf die Modellierung von Aufgaben zu konzentrieren, und die Entwickler können sich verstärkt mit anderen Aspekten ihrer Apps befassen. Hierfür benötigen Sie Best Practices, die eine einfache – hochwertige und sichere – Modellbereitstellung für eine Vielzahl von Rückschlusszielen ermöglichen.

Wir wollen dies etwas genauer analysieren, beginnend mit dem Ziel für ein Modell, das wie folgt lauten könnte:

- Bereitstellung als eigenständiger, Container-Rückschlussdienst
- Verwendung als Teil einer Batchverarbeitungs-Pipeline
- Einbettung in eine vorhandene Anwendung oder einen vorhandenen Dienst

Auch wenn die Bereitstellungseinheit variieren kann, bleibt das Ziel unverändert: die möglichst einfache, schnelle und effiziente Erstellung und Freigabe dieser Bereitstellungseinheit. Um dies zu erreichen, sollte eine optimale Lösung für die Modellbereitstellung einfache Benutzeroberflächen beinhalten, über die Engineers die Modelle problemlos mit minimaler zusätzlicher Konfiguration bereitstellen und überwachen können. Außerdem sollten Benutzer mit unterschiedlicher ML-Erfahrung mithilfe dieser Lösung in der Lage sein, ihre Daten und Modelle zu verstehen und zu analysieren.

Einige Faktoren, die die Modellbereitstellung vereinfachen können:

- Bereitstellung eines benutzerfreundlichen Modellformats, wie z. B. PNNL oder ONNX.
- Vereinfachung des Prozesses zur Interaktion mit dem Modell – z. B. durch Codegenerierung, über API-Spezifikationen oder andere Methoden.
- Unterstützung einer Vielzahl von Rückschlusszielen – einschließlich Cloud, App, Edge, spezieller Hardware wie FPGAs und dedizierter Frameworks wie Core ML und WinML.
- Einbindung der Verwaltung von Geheimnissen und Serviceendpunkten zur Vereinfachung der Konfiguration.

Selbst mit allen oben genannten Faktoren kann die Verwaltung des Freigabe- und Bereitstellungsprozesses eine Herausforderung darstellen. Daher benötigen Sie auch umfassende Zugriffskontrollen, manuelle und automatisierte Überprüfungen sowie eine durchgängige Überprüfbarkeit – insbesondere, wenn es um Kompatibilitätsprobleme und/oder Kundendaten geht.

Kontinuierliches erneutes Training

Schließlich sollten Sie eine Schleife für erneutes Training einrichten. Die meisten Trainingspipelines werden als Workflows oder Abhängigkeitsdiagramme eingerichtet, die bestimmte Vorgänge oder Aufträge in einer definierten Reihenfolge ausführen: Wenn ein Team mit neuen Daten trainieren muss, kann derselbe Workflow oder dasselbe Diagramm erneut ausgeführt werden. Viele Anwendungsfälle in der Praxis erfordern Folgendes:

- Erneutes Training bei Verfügbarkeit neuer Daten (wobei der Trainingscode selbst statisch bleibt).
- Erneutes Training bei sich weiterentwickelnden Daten (z. B. ein bewegliches Fenster über die letzten *n* Tage eines Protokolldatenstroms).
- Erneutes Training auf der Grundlage anderer Signale (z. B. Datendrift).

03 /

Azure Machine Learning ist MLOps-fähig

Azure Machine Learning, ein Clouddienst, verfügt über integrierte MLOps. Der Dienst unterstützt alle Phasen des Lebenszyklus des ML-Modells – mit vollständigem Support für gängige Open-Source-Python-Pakete wie z. B. scikit-learn, TensorFlow, PyTorch und MXNet.

Azure Machine Learning bietet Ihnen folgende Möglichkeiten:

- **Verwandlung Ihres Trainingsprozesses in eine reproduzierbare Pipeline.** Verwenden Sie ML-Pipelines, um alle Schritte Ihres Modelltrainingsprozesses von der Datenaufbereitung über die Feature-Extraktion bis hin zur Optimierung von Hyperparametern und Modellauswertung zu vereinigen.
- **Registrieren und Nachverfolgen der ML-Modelle.** Verwenden Sie die Modellregistrierung, um Ihre trainierten Modelle in Ihrem eigenen Arbeitsbereich in der Azure-Cloud zu speichern und zu versionieren, sodass sie leicht nachverfolgt und organisiert werden können.
- **Verpacken und Debugging von Modellen.** Modelle werden vor der Bereitstellung in ein Docker-Image gepackt. Dies kann automatisch im Hintergrund erfolgen, oder Sie können das Image manuell angeben. Bei Problemen ist eine lokale Bereitstellung für Problembehandlung und Debugging möglich.
- **Modellvalidierung und Profilerstellung.** Azure Machine Learning kann ein Profil für Ihr Modell erstellen, um die idealen CPU- und Speichereinstellungen für die Bereitstellung zu ermitteln. Die Modellvalidierung erfolgt im Rahmen dieses Prozesses. Dabei werden Daten verwendet, die Sie für die Profilerstellung bereitstellen.
- **Konvertieren und Optimieren von Modellen.** Durch die Konvertierung Ihres Modells in Open-Neural-Network-Exchange (ONNX)-Format lässt sich durchschnittlich eine Verdopplung der Leistung erzielen. Mithilfe von Azure Machine Learning können Sie ein neues ONNX-Modell erstellen oder ein vorhandenes Modell in ONNX konvertieren.
- **Bereitstellung fast überall.** Mit Azure Machine Learning können Sie Ihre ML-Modelle als Webdienste in der Cloud, in Ihrer lokalen Entwicklungsumgebung oder auf Azure IoT Edge-Geräten bereitstellen. Die Bereitstellungen können CPU, GPU oder Field Programmable Gate Arrays (FPGA) für Rückschlüsse verwenden. Sie haben sogar die Möglichkeit, ML- Modelle in Analytics-Apps einzubetten, die auf Microsoft Power BI basieren.
- **Erfassen eines End-to-End-Überwachungspfads.** Azure Machine Learning kann mit Git integriert werden, um nachzuverfolgen, woher Ihr Code stammt, und mit Azure ML-Datasets, um Ihre Daten nachzuverfolgen und zu versionieren. Der Ausführungsverlauf speichert eine Momentaufnahme des Codes, der Daten und Computeressourcen, mit denen ein Modell trainiert wurde. Die Modellregistrierung erfasst alle Metadaten, die Ihrem Modell zugeordnet sind.
- **Automatisieren des End-to-End-ML-Lebenszyklus.** Mit der Azure Machine Learning-Erweiterung können Sie Azure Pipelines verwenden, um Continuous Integration zu ermöglichen, indem Sie automatisch eine Trainingsausführung starten, wenn Sie eine Änderung in ein Git-Repository einchecken. Sie können auch Releasepipelines erstellen, die ausgelöst werden, wenn neue Modelle in einer Trainingspipeline erstellt werden.

Auf den folgenden Seiten werden wir eingehender erläutern, wie Azure Machine Learning MLOps in jeder Phase des ML-Lebenszyklus unterstützt. Sie können auch direkt mit dem Abschnitt Einstieg in Azure Machine Learning fortfahren, wenn Sie den Dienst lieber selbst erkunden möchten.

04 /

Die einzelnen Schritte im Überblick

Bevor wir uns mit den spezifischen Funktionen von Azure Machine Learning befassen und erörtern, wie diese MLOps möglich machen, wollen wir Azure Machine Learning zunächst definieren: es handelt sich hierbei um eine verwaltete Sammlung von Clouddiensten für Machine Learning, die in Form eines Arbeitsbereichs und eines SDK angeboten werden. Azure Machine Learning soll die Produktivität der Data Scientists verbessern, die im großen Maßstab Machine-Learning-Modelle entwickeln, trainieren und bereitstellen, und auch den ML Engineers, die Machine-Learning-Pipelines verwalten, nachverfolgen und automatisieren, eine höhere Produktivität ermöglichen.

Etwas genauer betrachtet, umfasst Azure Machine Learning folgende Komponenten:

- Ein SDK, das mit allen Python-basierten IDEs, Notebooks oder CLIs verbunden werden kann
- Eine Compute-Umgebung, die Workloads jeder Größe und Komplexität unterstützen kann, einschließlich der Möglichkeit einer zentralen oder horizontalen Hochskalierung, einer integrierten automatischen Skalierung und der flexiblen Option, eine CPU-basierte oder GPU-basierte Infrastruktur für das Training zu verwenden.
- Eine zentralisierte Modellregistrierung zur Verfolgung von Modellen und Experimenten, unabhängig davon, wo und wie diese erstellt wurden.
- Integrierter Support für Azure Container Instances, Azure Kubernetes Service und Azure IoT Hub – für die containerbasierte Bereitstellung von Modellen in der Cloud und am Intelligent Edge.
- Einen Überwachungsdienst, der Metriken für Modelle erfasst und nachverfolgt, die mit Azure Machine Learning registriert und bereitgestellt werden.

Wir wollen nun herausfinden, welche Möglichkeiten Azure Machine Learning in jeder Phase des End-to-End-ML-Lebenszyklus mit sich bringt – und wie sich die Funktionalität in den traditionellen DevOps-Lebenszyklus einfügt, wie in Abbildung 2 dargestellt.

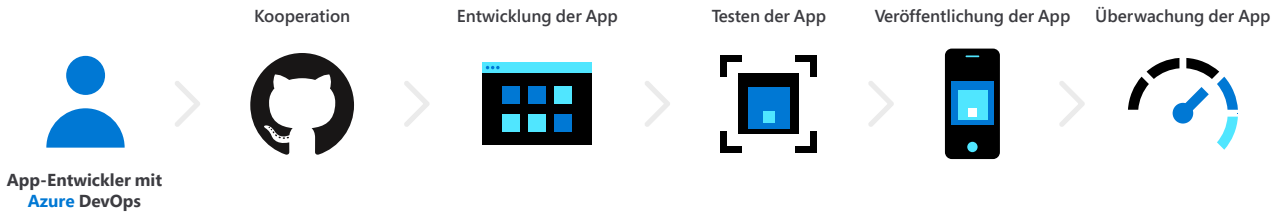


Abbildung 2. Typischer DevOps-Workflow und Prozesse.

Auf den folgenden Seiten verwenden wir für die Features und Funktionen die Namen aus Azure Machine Learning, damit Sie besser damit vertraut werden. Wenn unsere umfassende Onlinedokumentation mehr Informationen zu einem Thema enthält, als dieses Whitepaper bieten kann, finden Sie einen Link zu dem betreffenden Artikel.

Training eines reproduzierbaren Modells

Mit Azure Machine Learning haben Sie die nötigen Tools an der Hand, um schnell und effizient ein reproduzierbares Modell zu entwickeln und zu trainieren.

Arbeitsbereiche

Der Arbeitsbereich, die Ressource der obersten Ebene für Azure Machine Learning, ist ein zentraler Ort zum Arbeiten mit allen Artefakten, die Sie bei Verwendung des Diensts erstellen. Innerhalb Ihres Arbeitsbereichs können Sie Benutzerrollen definieren, um eine gemeinsame Nutzung mit anderen Benutzern, Teams oder Projekten zu ermöglichen.

Ein Arbeitsbereich kann Notebook-VMs enthalten, die mit der für die Ausführung von Azure Machine Learning benötigten Python-Umgebung konfiguriert sind. Zum Erstellen und Ausführen von Experimenten (den Trainingsausführungen, mit denen Sie Ihre Modelle entwickeln) können Sie das Azure Machine Learning SDK für Python, automatisierte Machine-Learning-Experimente (Vorschau) oder Azure Machine Learning-Designer (Vorschau) verwenden. Während der Iteration bewahrt der Arbeitsbereich den Verlauf aller Trainingsausführungen auf (einschließlich Log-Dateien, Metriken, der Ausgabe und einer Momentaufnahme Ihrer Skripte), damit Sie leichter herausfinden können, bei welcher Ausführung das beste Modell entsteht. In Ihrem Arbeitsbereich definieren Sie zudem die Computeziele für die Ausführung Ihrer Experimente. Auch die Datasets für das Modelltraining und die Pipeline-Erstellung definieren und verwalten Sie hier.

Code-, Dataset- und Umgebungsverwaltung

Während des Arbeitens müssen Sie mehrere Dinge verwalten und nachverfolgen, angefangen mit Ihrem Code. Dieser befindet sich höchstwahrscheinlich in einem Git-Repository. Azure Machine Learning umfasst eine Git Repository-Nachverfolgung: Wenn Sie Codeartefakte an den Dienst senden, können Sie einen Git-Repository-Verweis angeben. Dies geschieht automatisch, wenn die Ausführung über eine CI-/CD-Lösung wie Azure Pipelines erfolgt.

Sie müssen auch die Daten definieren und verwalten, die Sie für das Modelltraining verwenden möchten. Azure Machine Learning-Datasets bieten eine Möglichkeit der Versionsverwaltung, Profilerstellung und Momentaufnahme Ihrer Daten. Da Sie auf diese Weise Zugriff auf dieselben Daten haben, können Sie somit Ihren Trainingsprozess reproduzieren. Durch einen Vergleich der Dataset-Profile können Sie zudem herausfinden, wie stark sich Ihre Daten geändert haben oder ob Sie Ihr Modell neu trainieren müssen.

Darüber hinaus müssen Sie die Computeziele konfigurieren und verwalten, die für die Ausführung Ihrer Experimente verwendet werden. Hier können ML-Umgebungen, die von Azure Machine Learning verwaltet werden, hilfreich sein. Sie können für alle Aktivitäten mit Azure Machine Learning verwendet werden. Folglich können Sie damit die Übergabe vom Training zum Rückschluss vereinfachen oder eine Trainingsumgebung lokal reproduzieren. Die Umgebungen bieten automatische Verwaltung und Zwischenspeicherung von Docker-Images sowie Nachverfolgung, um die Reproduzierbarkeit zu gewährleisten.

ML-Pipelines

Mithilfe von Azure Machine Learning-Pipelines können Sie alle Schritte in Ihrem Modellentwicklungsprozess vereinigen und optimieren – von der Datenaufbereitung und Feature- Extraktion bis hin zur Hyperparameter-Optimierung, der Modellauswertung und letztendlich der Bereitstellung. Durch die Nutzung von Pipelines können Sie Ihre Workflows in puncto Geschwindigkeit, Portabilität und Wiederverwendung optimieren und haben damit mehr Zeit, um Ihr ML-Fachkenntnisse anzuwenden.

Pipelines werden von mehreren Schritten ausgehend erstellt, bei denen es sich um verschiedene Recheneinheiten handelt. Jeder Schritt kann unabhängig ausgeführt werden und isolierte Computeressourcen verwenden. Somit können mehrere Data Scientists gleichzeitig an derselben Pipeline arbeiten, ohne dass die Computeressourcen überlastet werden. Außerdem ist es auf diese Weise leicht möglich, für die einzelnen Schritte verschiedene Compute-Umgebungen zu verwenden.

Nachdem Sie eine Pipeline entworfen haben, werden Sie wahrscheinlich noch die enthaltene Trainingsschleife optimieren müssen. Bei der erneuten Ausführung einer Pipeline wird direkt zu den Schritten übergegangen, die erneut ausgeführt werden müssen (z. B. aktualisiertes Trainingsskript), und die unveränderten Schritte werden übersprungen. Dasselbe gilt für unveränderte Skripte, die für die Ausführung des Schritts verwendet werden. All das kann helfen, eine unnötige erneute Ausführung kostspieliger und zeitintensiver Schritte (z. B. Datenerfassung und -transformation) zu vermeiden, wenn sich die zugrunde liegenden Daten nicht geändert haben.

Nachverfolgen und Überwachen von Experimenten
Beim Iterieren zur Ermittlung des besten Modells sollten Sie Ihre Experimente nachverfolgen und überwachen. Das Azure Machine Learning SDK für Python und die Azure Machine Learning-CLI bieten verschiedene Methoden zum Überwachen, Organisieren und Verwalten Ihrer Ausführungen für Training und Experimentierung. Sie können auch Metriken für Trainingsausführungen protokollieren, indem Sie Ihrem Trainingsskript Protokollierungscode hinzufügen, eine Experimentausführung übermitteln, die Ausführung überwachen und die Ergebnisse überprüfen. Wenn Sie MLflow, eine Open-Source-Bibliothek für die Verwaltung von ML-Experimenten, verwenden, können Sie MLflow-Tracking mit Azure Machine Learning nutzen. Darüber hinaus können Sie mit TensorBoard die Struktur und das Ergebnis des Experiments prüfen und nachvollziehen.

Modellregistrierung
Wenn Sie über ein Modell verfügen, das Sie für gut befinden, können Sie es in der Modellregistrierung in Ihrem Arbeitsbereich registrieren. Dies macht eine einfache Speicherung, Versionsverwaltung, Organisation und Nachverfolgung aller trainierten Modelle möglich. Bei jedem registrierten Modell handelt es sich um einen einzelnen logischen Container für eine oder mehrere Dateien. Wenn Ihr Modell in mehreren Dateien gespeichert ist, können Sie diese in Ihrem Azure Machine Learning-Arbeitsbereich folglich als einzelnes Modell registrieren. Wenn Sie das registrierte Modell herunterladen oder bereitstellen, werden alle registrierten Dateien einbezogen.

Registrierte Modelle werden anhand ihres Namens und ihrer Version identifiziert. Wenn Sie ein Modell mit demselben Namen wie ein bereits vorhandenes Modell registrieren, erhöht die Registrierung die Version. Sie können auch zusätzliche Metadaten angeben, die bei der Suche nach Modellen verwendet werden sollen. Sie können Modelle registrieren, die außerhalb des Azure Machine Learning-Diensts trainiert wurden. Ein registriertes Modell, das in einer aktiven Bereitstellung verwendet wird, kann nicht gelöscht werden.

Die Modellregistrierung bildet in vielerlei Hinsicht die Grundlage des ML-Lebenszyklus-Verwaltungsprozesses. Sie ermöglicht die Versionskontrolle von Modellen, das Speichern von Modellmetriken, eine Ein-Klick-Bereitstellung und die Nachverfolgung aller Bereitstellungen Ihrer Modelle. So können Sie Maßnahmen ergreifen, wenn ein Modell veraltet ist oder eine nicht mehr akzeptable Wirksamkeit aufweist. Die Modellregistrierung spielt auch eine Rolle bei der Auslösung anderer Aktivitäten im ML-Lebenszyklus, beispielsweise wenn neue Änderungen auftreten oder Metriken einen Schwellenwert überschreiten.

Automatisiertes ML und Hyperparameter-Optimierung
Verschiedene Algorithmen und Hyperparameter-Kombinationen auszuprobieren, bis ein akzeptables Modell gefunden wird, kann für Data Scientists eine eintönige Aufgabe darstellen. Zwar lässt sich durch eine solche Iteration die Effizienz des Modells möglicherweise erheblich verbessern, sie erfordert jedoch Zeit und Ressourcen.

Das automatisierte ML in Azure Machine Learning verwendet Konzepte aus dem Forschungsbericht zur probabilistischen Matrixfaktorisierung, um eine automatisierte, parallelisierte Pipeline zu implementieren, die verschiedene Algorithmen und Hyperparameter-Einstellungen auf der Grundlage von Datenheuristiken durchführt und dann eine Reihe von Modellen präsentiert, die für das gegebene Problem und Dataset vermutlich am besten geeignet sind.

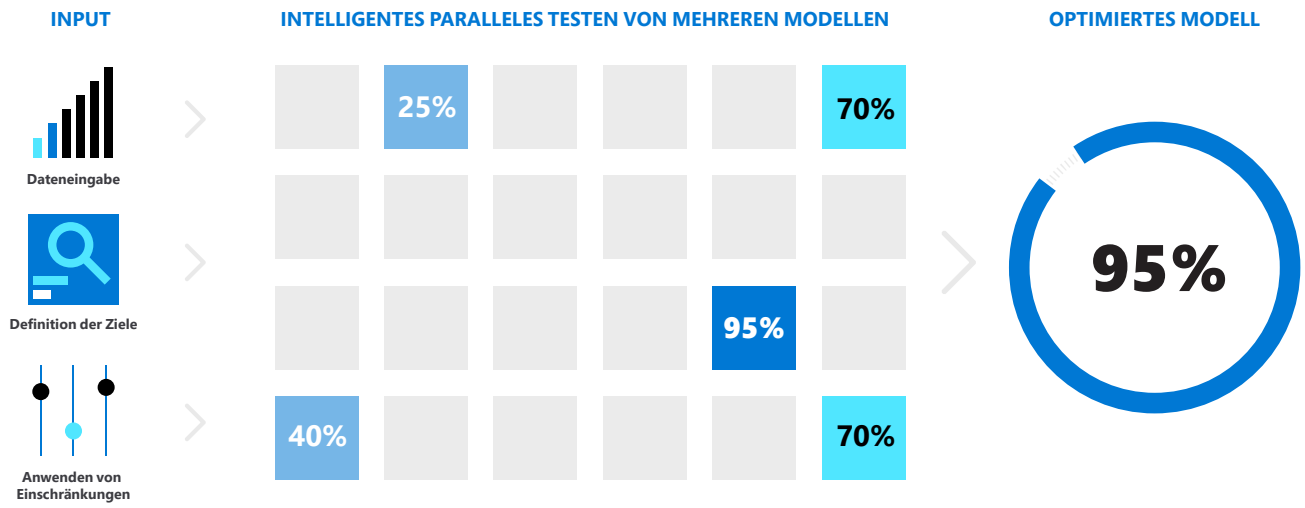


Abbildung 3. Mithilfe von automatisiertem Machine Learning können die besten Modelle für ein bestimmtes Problem und Dataset gefunden werden.

Automatisiertes ML unterstützt Klassifizierung, Regression und Prognose. Es beinhaltet auch Features für die Behandlung fehlender Werte, die vorzeitige Beendigung durch eine Stoppmetrik, das Blockieren von Algorithmen, die Sie nicht untersuchen möchten, und andere Optimierungen. Darüber hinaus kann der neu eingeführte UI-Modus (ähnlich einem Assistenten) für eine höhere Produktivität von Einsteigern oder nicht professionellen Data Scientists sorgen. So können Unternehmen mehr ML-Initiativen durchführen, ohne dass wichtige Projekte aufgrund einer begrenzten Zahl an qualifizierten Mitarbeitern gefährdet sind.

Zwar können Sie automatisiertes ML für die Hyperparameter-Optimierung nutzen, dies ist jedoch nicht die einzige Möglichkeit. Azure Machine Learning bietet auch einen Dienst zur Hyperparameter-Optimierung, der Zufalls-, Raster- und Bayessches Sampling unterstützt. Zur Konfiguration des Experiments definieren Sie einfach den Suchbereich, geben die primäre Metrik zusammen mit Richtlinien für die vorzeitige Beendigung an und weisen Ressourcen zu. Wenn Sie das Experiment übermitteln, kümmert sich der Dienst für die Hyperparameter-Optimierung um banale Aufgaben wie das Erstellen und Überwachen von Aufträgen. Sie können den Fortschritt Ihrer Trainingsausführungen mit einem Notebook-Widget visualisieren, das im Azure Machine Learning SDK bereitgestellt wird.

In dem Dokument Experimentation using Azure Machine Learning werden automatisiertes ML, der Dienst zur Hyperparameter-Optimierung und andere Features im Zusammenhang mit Machine Learning-Experimenten behandelt.

Verpacken des Modells

Bevor Sie Ihr Modell bereitstellen (oder testen) können, müssen Sie es für seine Zielausführungsumgebung verpacken, also alle für die Ausführung benötigten Komponenten erfassen, darunter Conda-Umgebungen, Python-Bibliotheken mit Versionsverwaltung und andere Bibliotheken, auf die das Modell verweist. Container sind der De-Facto-Standard hierfür. Es handelt sich hierbei um die standardmäßige Ausführungseinheit in fast allen Rückschlussumgebungen.

Wenn Sie Azure Machine Learning für die Bereitstellung eines Modells verwenden, wird es in ein Docker-Image gepackt, das einen Webserver für die Verarbeitung eingehender Anfragen enthält. Azure Machine Learning bietet ein standardmäßiges Docker-Basisimage, Sie müssen sich also nicht um das Erstellen eines Image kümmern, wenn Sie dies nicht möchten. Über die Azure Machine Learning-Umgebungen können Sie ein bestimmtes Basisimage auswählen.

Verwenden eines benutzerdefinierten Docker-Basisimage
Sie können auch ein benutzerdefiniertes Basisimage verwenden, das Sie bereitstellen. In der Regel stellen Sie ein Modell mithilfe eines benutzerdefinierten Docker-Basisimage bereit, wenn Docker für die Verwaltung von Abhängigkeiten, eine striktere Kontrolle der Komponentenversionen oder eine schnellere Bereitstellung verwendet werden soll. Es könnte beispielsweise sein, dass Sie eine Standardisierung auf eine bestimmte Version von Python, Conda oder einer anderen Komponente vornehmen möchten. Oder Sie möchten die für Ihr Modell erforderliche Software vorinstallieren – insbesondere wenn eine solche Installation lange dauert – damit Sie sie nicht für jede Bereitstellung separat installieren müssen.

ONNX
Beim Verpacken Ihres Modells sollten Sie die Verwendung von Open Neural Network Exchange (ONNX), einem offenen Format, als Möglichkeit zur Maximierung der Leistung bei Rückschlüssen in Erwägung ziehen. Warum? Die Optimierung von ML-Modellen für Rückschlüsse kann schwierig sein, da Sie das Modell und die Rückschlussbibliothek optimieren müssen, um die Hardwarefunktionen optimal nutzen zu können. Noch schwieriger wird es, wenn Sie für verschiedene Arten von Plattformen (Cloud/Edge, CPU/GPU usw.) eine Optimierung vornehmen möchten, da die jeweiligen Funktionen und Merkmale unterschiedlich sind. Wenn Sie Modelle aus einer Vielzahl von Frameworks verwenden, die auf einer Vielzahl von Plattformen ausgeführt werden müssen, wird das Ganze noch komplexer. Eine Optimierung für alle verschiedenen Kombinationen ist dann sehr zeitaufwendig.

Mit ONNX haben Sie die Möglichkeit, Ihr Modell einmal in einer Umgebung mit Ihrem bevorzugten Tool-Stack zu trainieren und es dann für Rückschlüsse und Prognosen in einer anderen Umgebung bereitzustellen. Modelle aus vielen Frameworks (einschließlich PyTorch, MXNet, Caffe2, TensorFlow, Core ML, scikit-learn, Keras, Chainer, MXNet, MATLAB und anderen) können exportiert oder in das Standard-ONNX-Format konvertiert werden.

Wenn die Modelle im ONNX-Format vorliegen, können sie auf zahlreichen Plattformen und Geräten ausgeführt werden. Die ONNX-Runtime, eine leistungsstarke Rückschluss-Engine für die Bereitstellung von ONNX-Modellen für die Produktion, ist für Cloud und Edge optimiert

und kann unter Linux, Windows und Mac verwendet werden. Sie ist in C++ geschrieben, verfügt jedoch auch über C-, Python- und C#-APIs. Die ONNX-Runtime bietet Support für die gesamte ONNX- ML-Spezifikation und kann auch mit Beschleunigern auf anderer Hardware, wie TensorRT auf NVIDIA- GPUs, integriert werden.

Die ONNX-Runtime kommt in weit verbreiteten Diensten von Microsoft wie Bing, Office und Cognitive Services zum Einsatz. Zwar hängen die spezifischen Leistungssteigerungen, die Sie erzielen werden, von einer Reihe von Faktoren ab, diese Microsoft-Dienste konnten jedoch durchschnittlich eine Verdopplung der Leistung bei den CPUs verzeichnen. Die ONNX-Runtime wird zudem als Teil von Windows ML auf Hunderten Millionen von Geräten verwendet. Natürlich können Sie die ONNX- Runtime auch mit Azure Machine Learning nutzen – sie ermöglicht Laufzeitbindungen und eine einfache Portabilität zu ONNX-Modellen – und von den umfangreichen Optimierungen, Tests und laufenden Verbesserungen für Produktionsumgebungen profitieren.

Validieren des Modells

Nachdem Ihr Modell verpackt ist, können Sie es testen. Da es in einem Container verpackt ist, lässt es sich ideal mit Azure Container Instances testen, einem Dienst, der einen einfachen, kostengünstigen Mechanismus für die Containerbereitstellung bietet. Es ist problemlos möglich, eine Bereitstellung in einem ACI-Container vorzunehmen und dann zu Testzwecken Rückschlüsse daraus zu ziehen. Sie müssen nicht einmal einen Container im Voraus erstellen – dies geschieht automatisch im Rahmen des Bereitstellungsprozesses.

Validierung und Profilerstellung

Mit den Funktionen für die Modellvalidierung und Profilerstellung in Azure Machine Learning können Sie Beispieleingabeabfragen bereitstellen, um sicherzustellen, dass Ihr Modell bei der Bereitstellung wie erwartet funktioniert. Azure Machine Learning stellt das gepackte Modell automatisch auf einer Vielzahl von Rückschluss-CPU-/Speicherkonfigurationen bereit und testet es, um die optimalen Einstellungen für die Bereitstellung Ihres Modells zu ermitteln. Die Modellvalidierung erfolgt im Rahmen dieses Prozesses. Dabei werden Daten verwendet, die Sie für die Profilerstellung bereitstellen.

Interpretierbarkeit des Modells

Beim Testen Ihres Modells werden Sie sich vielleicht fragen, aus welchem Grund die getroffenen Vorhersagen erfolgt sind. Durch die Verwendung der verschiedenen Interpretierbarkeitspakete im Azure Machine Learning Python SDK können Sie Hypothesen überprüfen, feststellen, ob das Modellverhalten Ihren Zielen entspricht, und prüfen, ob das Modell voreingenommen ist. All das kann letztlich dazu beitragen, Vertrauen bei den Stakeholdern aufzubauen. Mit den Klassen und Methoden in dem SDK können Sie Relevanzwerte für unformatierte und verarbeitete Merkmale (die Datenfelder, mit denen ein Zieldatenpunkt vorhergesagt wird) abrufen. Während des Trainings können Sie die Interpretierbarkeitsklassen und -methoden anwenden, um das globale Verhalten des Modells (als globale Erklärung bezeichnet) oder bestimmte Vorhersagen (als lokale Erklärung

bezeichnet) zu verstehen. Es können auch Explainer zusammen mit dem Modell bereitgestellt werden und für die lokale Erklärung zur Rückschlusszeit verwendet werden.

Bereitstellen Ihres Modells

Nach einer gründlichen Validierung sind Sie, wenn alles gut aussieht, bereit, Ihr Modell in der Produktion bereitzustellen. An diesem Punkt haben Sie mehrere Optionen:

- Wenn Sie Skalierbarkeit, Flexibilität sowie umfangreiche Protokollierung und Überwachung wünschen, sollten Sie die Bereitstellung auf Azure Kubernetes Services durchführen, einem Dienst, der nach Bedarf dimensioniert werden kann.
- Wenn Ihre Modelle klein sind und sicher nicht skaliert werden müssen, können Sie eine Bereitstellung auf einer Azure Container Instance vornehmen, die ebenfalls umfangreiche Überwachung und Protokollierung bietet.
- Bei Verwendung von Azure IoT Edge können Sie IoT Edge-Module direkt auf Linux-basierten IoT-Geräten bereitstellen.
- Wenn die Rückschlussleistung entscheidend ist, können Sie eine Bereitstellung in Field Programmable Gate Arrays (FPGAs) auf Azure – der weltweit größten Cloudinvestition – durchführen. Bevor Sie dies planen, sollten Sie jedoch überprüfen, ob Ihr spezifischer Anwendungsfall unterstützt wird.

Sie können auch eine Bereitstellung in verschiedenen anderen Umgebungen vornehmen, darunter GPU, Azure App Service und Notebook-VMs. Wenn Sie über ein vorhandenes Modell verfügen, das anderweitig trainiert wurde, können Sie dennoch für die Bereitstellung Azure Machine Learning verwenden.

Überwachen Ihres Modells

Nach der Bereitstellung sollten Sie Ihr Modell überwachen, um zu verstehen, welche Daten an Ihr Modell gesendet werden, welche Vorhersagen es zurückgibt und schließlich, wie es verwendet wird. Sie sollten auch eine Überwachung auf Datendrifts durchführen, damit Sie wissen, ob und wann Ihr Modell neu trainiert werden muss.

Zu diesem Zweck müssen Sie viele Metriken erfassen und analysieren. Hierfür können Sie die Datensammlung aktivieren. Gesammelte Daten werden in Azure Blob Storage gespeichert, sodass Sie problemlos eine Validierung und Analyse mit Ihren bevorzugten Tools vornehmen können. Sie können auch dafür sorgen, dass Azure Machine Learning eine Überwachung auf Datendrift – einen der Hauptgründe für eine Verringerung der Modellgenauigkeit im Laufe der Zeit – vornimmt und Sie per E-Mail benachrichtigt, wenn ein Datendrift festgestellt wird. Schließlich können Sie Azure Application Insights mit Azure Machine Learning verwenden, um Anforderungsraten, Abhängigkeitsraten, Antwortzeiten, Fehlerraten und mehr zu überwachen.

Kontinuierliches erneutes Training

Im Laufe der Zeit müssen Sie Ihr Modell möglicherweise neu trainieren, um seine Genauigkeit und/oder Leistung zu verbessern. Idealerweise sollte es sich hierbei um einen vollständig automatisierten CI-/CD-Prozess handeln. Wenn Sie die Azure DevOps-Erweiterung für Azure Machine Learning verwenden, können Sie sowohl die Azure Machine Learning-Modellregistrierung als auch das GitHub-Repository mit Ihren Python-Notebooks und -Skripten überwachen und anschließend Azure Pipelines auslösen, einen Dienst, der Ihr Modell automatisch neu trainiert und bereitstellt, auf der Grundlage von Codecommits oder bei Registrierung neuer Versionen eines Modells.

Solche Funktionen können extrem leistungsfähig sein. Mithilfe dieser Funktionen können Data-Science-Teams Phasen für Build- und Releasepipelines innerhalb von Azure DevOps für ihre ML-Modelle konfigurieren und damit den Prozess vollständig automatisieren. Doch damit nicht genug: Da Azure DevOps zur Verwaltung von App-Lebenszyklen verwendet wird, schließt sich nun der Kreis, da Data-Science-Teams und App Development Teams problemlos zusammenarbeiten und neue Versionen ML-gestützter Apps in der DevOps-Umgebung auslösen können, wenn während des MLOps-Lebenszyklus bestimmte Bedingungen erfüllt sind.

Abschließende Arbeiten

Mit Azure Machine Learning haben Sie alles zur Hand, was Sie für die Implementierung eines umfassenden MLOps-Ansatzes benötigen, einschließlich der Möglichkeit, jede Ressource im Lebenszyklus des ML-Modells nachzuverfolgen, zu versionieren, zu überwachen und wiederzuverwenden. Darüber hinaus verfügen Sie über die nötigen Tools, um diesen Lebenszyklus durchgängig zu optimieren, zu automatisieren und in vorhandene DevOps-Prozesse einzubinden. So können Data Scientists, ML Engineers und App-Entwickler kooperieren und bei der Bereitstellung ML-gestützter Apps im gleichen Rhythmus arbeiten.

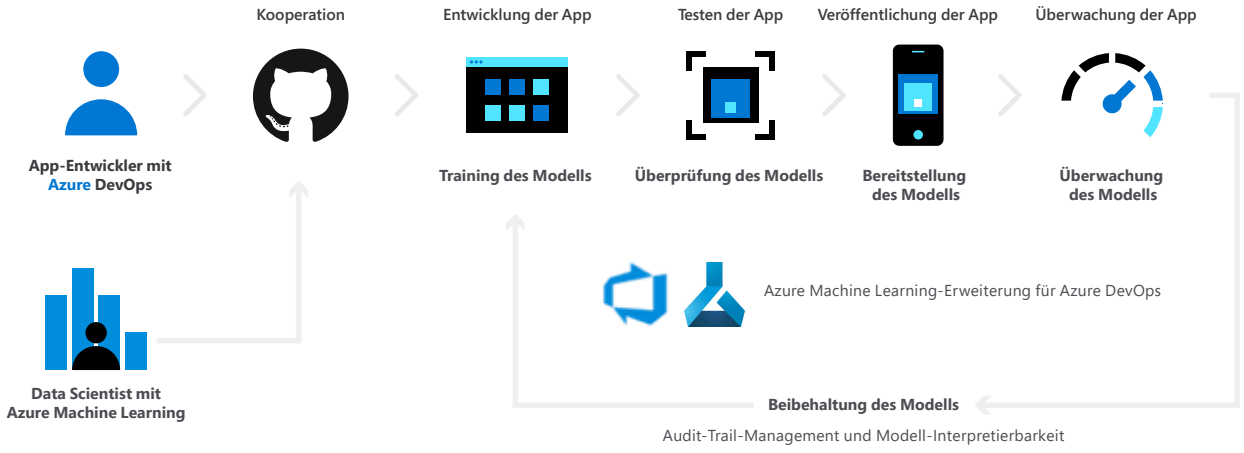


Abbildung 4. Azure Machine Learning bietet Data Scientists und ML Engineers komplexe Tools und optimierte Verfahren, wie sie Entwickler in einer modernen DevOps-Umgebung kennen.

Für sich genommen mögen viele der hier besprochenen Features und Funktionen logisch und intuitiv erscheinen. Bis jetzt ist es jedoch noch niemandem gelungen, all das auf eine Weise zu vereinigen, die für alle Beteiligten funktioniert und enorme Produktivitätssteigerungen ermöglicht.

Wenn Sie Azure Machine Learning gerne selbst testen möchten, wechseln Sie einfach zum Abschnitt Erste Schritte. Wir erklären Ihnen die Vorgehensweise und erläutern, wie Sie sich für eine kostenlose Testversion anmelden können. Wenn Sie sich hierfür noch nicht bereit fühlen, finden Sie im Folgenden einige weitere Ressourcen:

- Erfahren Sie mehr über Azure Machine Learning.
- Sehen Sie sich das Azure Machine Learning-Service-MLOps-Repository auf GitHub an.
- Sehen Sie sich die Sitzung auf der Microsoft Build 2019 zu MLOps an.
- Sehen Sie sich die Sitzung auf der Microsoft Ignite 2019 zu MLOps an.
- Laden Sie unsere kostenfreien E-Books herunter:
    - Prinzipien der Data Science
    - Durchdachtes Machine Learning mit Python

05 /

Einstieg in Azure Machine Learning

Wir haben in diesem Leitfaden kurz einige wichtige Konzepte von Azure Machine Learning behandelt und dabei auch jeweils auf die umfassende Onlinedokumentation hingewiesen. Wenn Sie etwas über ein Thema lesen möchten, das wir hier nicht behandelt haben, finden Sie alle Artikel online. Den ersten Artikel können Sie hier abrufen, die restlichen sind im Navigationsbereich unmittelbar darunter aufgeführt. Bevor Sie beginnen, sollten Sie sich zumindest die Übersicht über den Azure Machine Learning-Service und sein Workflowmodell ansehen.

Schritt-für-Schritt-Tutorials

Bereit, mit einem unserer Tutorials zu beginnen? Wenn Sie noch kein Azure-Abonnement haben, können Sie vor dem Einstieg ein kostenfreies Konto erstellen.

Zunächst sollten Sie versuchen, mithilfe des Python SDK Ihr erstes Experiment zu erstellen, indem Sie einen Arbeitsbereich und eine Entwicklungsumgebung einrichten und anschließend Ihr erstes Modell trainieren. Sie können auch eines unserer anderen Tutorials zum Python SDK ausprobieren:

- Trainieren eines einfachen Bildklassifizierungsmodells mit logistischer Regression, und anschließende Bereitstellung des Modells.
- Erstellen eines Regressionsmodells mit automatisiertem Machine Learning.
- Batchbewertung eines Klassifizierungsmodell mit dem Python SDK und Pipelines.

Wenn Sie mit der grafischen Benutzeroberfläche beginnen möchten, können Sie sich die Tutorials zur Verwendung der Oberfläche zum Trainieren eines Regressionsmodells und zur anschließenden Bereitstellung des Modells ansehen. Sie können auch versuchen, ein automatisiertes Machine Learning-Experiment über die Landing Page des Arbeitsbereichs zu erstellen, ohne eine einzige Codezeile zu schreiben.

Beispiele

Für den schnellen Einstieg steht auch eine Vielzahl von Python SDK-basierten Codebeispielen zur Verfügung. Wenn Sie das zuvor erwähnte Tutorial zum Einrichten der Umgebung und des Arbeitsbereichs abgeschlossen haben, verfügen Sie bereits über einen dedizierten Notebook-Server mit dem SDK und dem Beispiel-Repository. Wenn Sie lieber Ihren eigenen Notebook-Server für die lokale Entwicklung nutzen oder Beispiele für die Data Science Virtual Machine, ein angepasstes VM-Image für Data Science, abrufen möchten, sind einige zusätzliche Schritte erforderlich.

Schrittanleitungen

Nach Durchführung einiger Tutorials und Beispielexperimente können Sie die Möglichkeiten von Azure Machine Learning auf eigene Faust erkunden. Unsere Onlineanleitungen enthalten Schritt-für-Schritt-Anweisungen zu vielen möglichen Maßnahmen. Nicht alle diese Anleitungen können hier aufgeführt werden, die erste davon finden Sie jedoch hier, die anderen im Navigationsbereich unmittelbar darunter.

Literatur

Wenn Sie Referenzmaterialien benötigen, sehen Sie sich unsere Dokumentation zum Azure Machine Learning SDK für Python und zur Azure Machine Learning-CLI an. Es gibt auch eine umfassende Reihe von Referenzmaterialien für die Module der grafischen Benutzeroberfläche. Auch hier finden Sie eine vollständige Liste im Navigationsbereich.