Automatisiertes Bewerten bei der praktischen Vermittlung von Methoden des Maschinellen Lernens

Katharina Holstein[1], Nata Kozaeva[2] und Korinna Bade[3]

Abstract: Im Kontext der Informatiklehre für die Einführung in das Maschinelle Lernen an Hochschulen für angewandte Wissenschaften liegt ein wesentlicher Fokus auf Anwendungsszenarien und Datensätzen mit aktuellem Problembezug. Zeitgleich ist der Stand der Vorkenntnisse vor allem der Masterstudierenden sehr divers. Dazu wird im folgenden Paper dargestellt, wie die praktischen Vermittlung von Methoden des Maschinellen Lernens mit Python durch automatisiertes Testen unterstützt werden kann. Dabei liegt ein besonderer Fokus auf Lösungen für die speziellen Anforderungen der Programmierausbildung für das Maschinelle Lernen.

Keywords: Selbstlernprogrammierkurs; Maschinelles Lernen; Automatisiertes Testen; VPL

1 Einleitung

An der Hochschule Anhalt werden in verschiedenen Studiengängen und Modulen Kenntnisse zu Themen des Maschinellen Lernens und des Data Science vermittelt. Für die im Folgenden präsentierte Arbeit ist insbesondere der Masterstudiengang Data Science von Interesse, in welchen Studierenden beliebiger Bachelor-Abschlüsse zugelassen werden. Daraus ergibt sich eine sehr heterogene Studierendenschaft mit unterschiedlich vorhandenen Kompetenzen im Bereich der Programmierung. Von Studienanfängern ohne jegliche Programmiererfahrung bis hin zu Studierenden mit einem Informatikabschluss ist alles vertreten. Entsprechend didaktisch anspruchsvoll ist der Umgang damit. Grundsätzlich wird in einem speziell dafür konzipiertem ersten Semester Studienanfängern mit Bachelor-Abschlüssen jenseits der Informatik ein Lehrangebot unterbreitet, welches darauf abzielt, die wichtigsten Grundkompetenzen der Informatik und so auch der Programmierung zu vermitteln. Nichtsdestotrotz muss innerhalb der Module der unterschiedlichen Lernkurve der einzelnen Studierenden Rechnung getragen werden. Studierende brauchen auf Grund ihres unterschiedlichen Wissensstandes meist sehr individuelle Unterstützung und gleichzeitig viel Programmierpraxis. Da dies mit den vorhandenen personellen Ressourcen sehr schwierig zu leisten ist, soll automatisiertes Bewerten von Programmieraufgaben ein zeitnahes Feedback und damit Unterstützung bei der Erlangung von Programmierpraxis geben.

In der vorliegenden Arbeit wird insbesondere die Vermittlung von Programmierkenntnissen im Umfeld des Maschinellen Lernens und des Data Science betrachtet. Hieraus ergeben

[1] Hochschule Anhalt, Fachbereich Informatik und Sprachen, Lohmannstraße 23, 06366 Köthen, Deutschland; Katharina.Holstein@hs-anhalt.de
[2] Hochschule Anhalt, Fachbereich Informatik und Sprachen, Lohmannstraße 23, 06366 Köthen, Deutschland
[3] Hochschule Anhalt, Fachbereich Informatik und Sprachen, Lohmannstraße 23, 06366 Köthen, Deutschland; Korinna.Bade@hs-anhalt.de; https://orcid.org/0000-0001-9139-8947

doi:10.18420/abp2023-10

sich spezifische Anforderungen an das automatische Testen, welche im Folgenden näher betrachtet werden sollen. Die genutzte Programmiersprache ist Python, die in dieser Domäne hohe Relevanz hat. Im Bezug zu anderen Arbeiten ist festzuhalten, dass es zwar verschiedene Systeme zum automatisierten Testen von Python-Code gibt, die speziellen Anforderungen im Bereich des Maschinellen Lernens und des Data Science bisher unseres Wissens nach aber in der Literatur noch nicht explizit betrachtet wurden. Im Folgenden werden zunächst unsere Lernszenarien genauer vorgestellt. Anschließend werden die bisher von uns identifizierten Herausforderungen an das automatisierte Testen im Data Science Umfeld sowie unser Lösungsansatz an einzelnen Beispielen dargestellt.

2 Lernszenarien und Stand der Arbeit

Es werden zwei Lernszenarien betrachtet, die unterschiedlich stark das automatisierte Testen als Bestandteil integrieren. Dabei dient das erste Szenario dazu, Studierende an das Arbeiten mit automatisierten Tests heranzuführen und eine „herkömmliche“ Lehrveranstaltung anzureichern sowie in der Interaktion mit den Studierenden Erfahrungen im Umgang mit automatisierten Tests zu sammeln und weitere Anforderungen abzuleiten. Im zweiten Lernszenario soll dies entsprechend weitergeführt werden, so dass hier dem automatisierten Testen eine zentrale Rolle zukommt. Dabei soll das automatisierte Testen nicht zum Zwecke der Notenfindung eingesetzt werden. Stattdessen dient es der Unterstützung des Selbststudiums und soll mit möglichst sinnvollen und konkreten Fehlermeldungen direktes, zeitnahes Feedback an die Studierenden weitergeben.

Lernszenario 1: Unterstützung der Praktika des Moduls Maschinelles Lernen

Die Lehrveranstaltung Maschinelles Lernen wird bereits im ersten Mastersemester, zeitgleich mit einem Modul zur Vermittlung von Programmierfähigkeiten, durchgeführt. Programmierfertigkeiten der Studierenden werden auch in diesem Modul mit entwickelt, weiter vertieft und ausgebaut. Ein umfangreiches Verständnis von entsprechenden Bibliotheken aus dem Bereich Data Science kann nicht vorausgesetzt werden. Inhaltlich gibt die Vorlesung eine Einführung in die Grundlagen des Maschinellen Lernens und stellt einzelne Ansätze des überwachten und unüberwachten Lernens sowie deren praktische Umsetzung mit Python vor. Bestandteil der Lehrveranstaltung ist ein Programmierpraktikum, welches auf verschiedenen Anforderungsniveaus unterschiedliche Programmieraufgaben vorsieht. Um die Präsenzzeiten der Praktika zielgerichteter zu gestalten und den Studierenden auch bei der Arbeit zu Hause eine Unterstützung zu geben, sollen die Lösungen zu den Praktikumsaufgaben mit automatisierten Tests geprüft werden. Das Feedback soll typische Fehler der Studierenden zeitnah identifizieren, damit die Studierenden ausreichend Zeit für die eigentliche Lösung des Problems einsetzen können.

Lernszenario 2: Selbstlernkurs zur Vermittlung von Grundlagen der Python-Programmierung im Umfeld von Data Science

Das zweite Lernszenario ist als Ergänzungsmaterial konzipiert und soll daher unabhängig für sich ohne Betreuung durch eine Lehrperson absolvierbar sein. In diesem Kurs soll insbesondere der Umgang mit speziellen Bibliotheken aus dem Umfeld des Data Science, wie z. B. numpy, pandas, matplotlib oder ähnliches trainiert werden. Grundsätzlich lassen sich diese Inhalte auch über Tutorials im Web erarbeiteten. Dies stellt jedoch vor allem Studierende ohne einen Informatikhintergrund vor große Herausforderungen, da sie häufig auf Grund der Fülle der verfügbaren Informationen und fehlendem Wissen Tutorials nicht gezielt recherchieren, Dokumentationen nicht nachvollziehen und ähnliche Problemstellungen nicht übertragen können sowie wesentliche Bibliotheken nicht betrachten. Daher soll das vorliegende Lernszenario die Studierenden beim Selbststudium unterstützen und idealerweise einen Übergang in das selbstgesteuerte Lernen mit Internetressourcen bilden.

Aktueller Stand der Umsetzung

Als erster Schritt erfolgte bisher die Umsetzung des ersten Lernszenarios. Dieses wird im kommenden Semester mit den Studierenden praktisch erprobt, evaluiert und wo notwendig, verbessert und erweitert. Daran anschließend wird das Lernszenario 2 umgesetzt. Alle Materialien sollen als Open Educational Ressource (OER) umgesetzt werden und stehen damit anderen Lehrenden (und natürlich auch Lernenden) zur Verfügung.

3 Automatisches Testen für Szenarien des Maschinellen Lernens

Für die Umsetzung kann prinzipiell ein beliebiges Tool zum Bewerten von Python-Code zum Einsatz kommen. Die besonderen Herausforderungen müssen durch die Umsetzung der eigentlichen Tests gelöst werden. Da an der Hochschule Anhalt Moodle als Lernmanagementsystem mit dem Plugin „Virtual Programming Lab (VPL)“ [Ro23] bereits in einer anderen Lehrveranstaltung für Tests von Java-Programmen im Einsatz ist, wurde diese Umgebung für die Umsetzung gewählt. Das Plugin stellt im Browser einen Editor bereit, in welchem die Studierenden Ihren Quellcode in Form von Python-Skripten eingeben und die Tests über die Nutzerschnittstelle durchführen können. Das VPL-Plugin nutzt im Hintergrund den Zugriff auf einen Jail-Server, welcher die Tests der eingereichten Python-Skripte mittels Standardunittests [Ro01]. übernimmt. Dazu werden auf dem Server die jeweiligen Test-Skripte sowie weitere benötigte Ressourcen wie etwa Standard-Datensätze hinterlegt.

Im Lernszenario 1 erfolgt der Einstieg in das automatisierte Testen begleitet. Im ersten Praktikum erfolgt zunächst eine *Einführung in die grundsätzlich verwendeten Tools*. Für den Einstieg in das automatisierte Testen wird der Zugriff auf den Jailserver sowie die Verwendung des VPL-Plugins in Moodle an einem einfachen „Hello World“ Testskript erläutert. Der Fokus liegt zunächst auf der Einbindung der eigenen Skripte, deren Evaluierung, dem Verständnis der Rückgabewerte bzw. Fehlermeldungen sowie dem grundsätzlichen Verständnis zur Handhabung des automatisierten Testens. Darüber hinaus wird die Integration eigener Datensätze besprochen, welches ebenfalls über das Plugin möglich ist.

Erste einfache Aufgaben werden im Rahmen des Praktikums selbst gelöst, um die Einstiegshürde zu senken, da bei Problemen direkt die Lehrkraft zur Verfügung steht. Weiterführende Programmieraufgaben müssen zum nächsten Praktikum im Selbststudium vorbereitet und dort präsentiert werden. Dabei sollen die automatisierten Tests unterstützen. In den folgenden Praktika präsentieren verschiedene Studierende ihre Lösungen zur Anregung einer Diskussion über Lösungsstrategien und somit der Schaffung des Rahmens eines Flipped Classroom [Sp16]. Der Fokus liegt dabei nicht auf dem Finden des besten Lösungsansatzes, sondern in der Bildung von Verständnis für verschiedene Ansätze, deren Vor- und Nachteile sowie der Schaffung einer Lernumgebung, in der unterschiedliche Herangehensweisen gleich honoriert werden. Darüber hinaus soll diese Herangehensweise verhindern, dass Studierende Lösungsstrategien lediglich konsumieren oder kopieren und stattdessen auch eigene Strategien präsentieren.

Alle nachfolgenden Praktikumseinheiten folgen dem gleichen **Grundaufbau**, um den Lernprozess der Studierenden zu unterstützen. Basis bildet das jeweilige Vorlesungsskript zum aktuellen Thema, welches auch jeweils ein praktisches Beispiel enthält. Auf dessen Basis soll zunächst ein Skript mit einem anderen, durch die Studierenden frei gewählten Datensatz mit gleicher Funktionalität umgesetzt werden (Schritt I). Im nächsten Schritt (Schritt II) sollen kleinere Veränderungen am Code vorgenommen werden, um ein Verständnis für die genutzten Bibliotheken und Tools, Auswirkungen von Parametern etc. aufzubauen. Hierfür sollen die Studierenden sowohl auf die Dokumentation der Bibliothek zurückgreifen als auch entsprechende Standardforen bzw. Hilfsseiten nutzen. Im letzten Schritt (Schritt III) sollen dann einzelne Algorithmen selbst implementiert werden, ohne auf vorgefertigte Bibliotheken zurückzugreifen, beginnend bei einzelnen Funktionen wie z. B. die Berechnung der Entropie eines Knotens in Entscheidungsbäumen bis hin zur vollständigen Umsetzung eines Algorithmus wie etwa der gesamte ID3-Algorithmus.

Aus diesem Grundaufbau ergeben sich spezielle Anforderungen an die konkrete Ausgestaltung des automatischen Bewertens, welche im Folgenden näher betrachtet und mit Umsetzungsbeispielen untersetzt werden sollen. Eine große Rolle im Bereich des Maschinellen Lernens spielt die **Arbeit mit Datensätzen**. Dabei sollen die Studierenden mit möglichst unterschiedlichen Datensätze in Berührung kommen. Im Praktikum wird dies durch die beliebige Wahl von Datensätzen ermöglicht. Neben Standarddatensätze, wie z. B. aus dem UCI-Repository [KLN23], sollen auch eigene Datensätze bzw. Datensätze aus der Industrie, von wissenschaftlichen Partnereinrichtungen oder aus den Fachbereichen der Hochschule verwendet werden können. Dies stellt für das automatisierte Bewerten eine besondere Herausforderung dar, da die Korrektheit der Lösung unabhängig vom spezifischen Datensatz geprüft werden soll. Notwendige Schritte, z. B. in der Datenvorverarbeitung, sind aber ggf. stark vom Datensatz, der Ausprägung der Feature u. ä. abhängig. Die Testumgebung muss möglichst generisch die notwendigen Schritte im Zusammenhang mit dem jeweils genutzten Datensatz bewerten können. Durch die Verwendung unterschiedlicher Datensätze können verschiedene Schwierigkeitsgrade für die Studierenden realisiert und unterschiedlichen Vorkenntnissen, aber auch Interessen Rechnung getragen werden. Die Verwendung generischer

automatisierter Tests reduziert zusätzlich den Aufwand des Lehrpersonals in der Betreuung, da der Datensatz selbst nicht in der kompletten Tiefe durchdrungen oder vorbereitet werden muss.

Im Folgenden sollen zwei Herausforderungen im Umgang mit generischen Datensätzen sowie die Umsetzung der Lösung gezeigt werden. Für Klassifikationsaufgaben ist die *Identifizierung der Klassenspalte* in den Daten erforderlich. Grundsätzlich könnte man natürlich verlangen, dass Studierende ihre Daten vor der Nutzung in ein bestimmtes Format überführen müssen, aus dem das Zielattribut direkt hervorgeht, bzw. das Zielattribut direkt benennen müssen. Dies erfordert jedoch ggf. bereits eine gewisse Expertise im Umgang mit Datensätzen. Um die Einstiegshürde zu senken und Fehler bei der händischen Behandlung der Daten zu vermeiden bzw. erkennen zu können, sollen die Test-Skripte eine automatisierte Erkennung des Zielattributs umsetzen. So kann auch überprüft werden, ob die Studierenden überhaupt das richtige Zielattribut identifiziert haben und daraus abgeleitet Zielattribute und Merkmalsklassen korrekt erzeugt und kodiert haben sowie Trainings- und Testdaten korrekt abgeleitet wurden. Dazu wurde basierend auf den uns vorliegenden Datensätzen eine Heuristik abgeleitet, die typische Ausprägungen von Datensätzen berücksichtigt. Die entsprechende Funktion, die in den Test-Skripten zum Einsatz kommt, ist in List 1 im Pseudocode dargestellt. Im Falle des Fehlschlags der Klassenerkennung wird eine Fehlermeldung ausgegeben, die eine entsprechende Merkmalsbenennung durch den Studierenden erfordert. Um Fehlern vorzubeugen, überprüft das Skript aber auch die formelle Eignung der benannten Klasse. Gleichzeitig werden diese Datensätze dann betrachtet, um die Heuristik weiter zu verbessern.

Ein weiterer wichtiger Schritt der Datenvorverarbeitung ist das *Feature Encoding*. Nicht numerische Feature müssen für viele Algorithmen zunächst in numerische Feature umgewandelt werden. Auf der anderen Seite dürfen bereits numerische Feature nicht noch einmal encodiert werden. Dies ist zugleich eine typische Fehlerquelle bei Studierenden beim Übertragen von Skripten auf neue Datensätze. List 2 enthält den Pseudocode des eingesetzten Tests. Dabei wird das korrekte Encoding durchgeführt und dann mit der Studierendenlösung verglichen.

Insgesamt wird in Schritt I aller Praktikumsaufgaben ein automatisiertes Testen einzelner Arbeitsschritte der gesamten **Datenvorverarbeitungskette** mittels Unittests vorgenommen. Diese Überprüfung folgt dem Schema: Laden von benötigten Bibliotheken, Verwendung der korrekten Funktion zur Bearbeitung von Datensätzen, Aufspaltung des Datensatzes in Zielattribute und Feature (vgl. List 1), korrekte Verwendung eines Encodings (vgl. List 2), Aufteilung des Datensatzes in Trainings- und Testdaten. Neben den gezeigten Pseudocodes kommen in dieser Verarbeitungskette z. B. die explizite Überprüfung von Datentypen bzw. Python-Klassen oder die Überprüfung der Shapes der Teildatensätze zum Einsatz.

Eine weitere Herausforderung beim automatisierten Testen [Bo17; Fa15; Ly12] liegt in der **Überprüfung unspezifischer Aufgabenstellungen**, wie der Änderung von Visualisierungen zum Explorieren der Bibliotheksfunktionen in Schritt II. Dies kann über die Bibliothek

```
def data_preprocessing_X_y(data: pd.DataFrame) -> List:
  Wenn Datensatz data weniger als 2 Spalten enthält:
    Gib entsprechende Fehlermeldung zurück
  Wenn Datensatz data mind. eine Spalte 'class', 'klasse' oder 'target'
      enthält:
    Wenn es genau eine solche Spalte gibt:
      Wenn diese Spalte die Eigenschaften einer Klassenspalte erfüllt (nur
          diskrete Werte):
        Verwende diese Spalte als Zielattribut y und den Rest als Daten X
            und gib [X,y] zurück
      Sonst: Gib Fehlermeldung zur Spaltenauswahl zurück
    Sonst: Gib Fehlermeldung der Uneindeutigkeit zurück
  Wenn Datensatz data mind. eine einzelne Spalte vom Datentyp Object
      enthält:
    Wenn es genau eine solche Spalte gibt:
      Verwende diese Spalte als Zielattribut y und den Rest als Daten X und
          gib [X,y] zurück
    Sonst: Gib Fehlermeldung der Uneindeutigkeit zurück
  Wenn Datensatz data mindestens eine Spalte enthaelt, die nur
      Integer-Werte besitzt:
    Verwende von allen Spalten, die diese Bedingung erfüllen, diejenige,
        die die wenigsten unterschiedlichen, eindeutigen Werte hat, als
        Zielattribut y und den Rest als Daten X und gib [X,y] zurück
  Gib Fehlermeldung zur Klassenidentifizierung zurück
```

List. 1: Generische Bestimmung des Zielattributs in einem Datensatz.

Mock bzw. MagicMock der Unittest Bibliothek gelöst werden. Diese erlaubt es, über eine zentrale Mock-Klasse die Verwendung von Methoden bzw. Attributen und ihrer spezifischen Funktionsaufrufe zu testen, ohne spezifische Abfragen vorzunehmen. Falls eine Änderung der Methoden nicht geschehen ist, werden über entsprechende Fehlermeldungen die Studierenden mit aussagekräftigen Hinweisen auf die korrekte Lösung hingewiesen werden. Der Vorteil liegt hier in der generischen Anwendbarkeit der Testumgebung auf herausgelöste Problemstellungen sowie auf unspezifischen Aufgabenstellungen. In einigen Fällen können Funktionsaufrufe auch über regex Ausdrücke getestet werden. Dies erfordert keine extra Funktion im Studierendenskript, die gemockt werden muss. Allerdings ist diese Methode weniger flexibel, da ein String-Vergleich nur in wenigen Fällen möglich ist.

Als letztes soll die **Bewertung kleinerer Teilfunktionen** im Schritt III für Ergebnisse auf unbekannten Datensätzen betrachtet werden. Dies kann in der Regel darüber evaluiert werden, ob das in der Lösung berechnete Ergebnis in etwa mit dem korrekten Ergebnis übereinstimmt, was durch die „assertAlmostEqual“ Funktion ermöglicht wird. Können die Studierenden mit beliebigen Datensätzen arbeiten, kann hierfür aber nicht ein festes Ergebnis hinterlegt werden. Stattdessen muss, wie oben bereits beschrieben, eine generische Funktion hinterlegt werden, die für jeden möglichen Datensatz das korrekte Ergebnis bestimmen kann. Zeitgleich muss überprüft werden, dass Studierende nur zulässige Bibliotheken

```
def umkodierung_X(attribute_X, data):
  Für jedes Feature in attribute_X:
    Wenn das Feature vom Datentyp Object ist:
      Encodiere die Daten dieser Spalte mit dem LabelEncoder
    Wenn das Feature vom Datentyp Integer oder Float ist:
      Verändere die Daten nicht
    Gib die Umkodierten Daten zurück

def test_encoding_X(self):
  studierende_attribute = die kodierten Daten der Studierendenlösung
  korrekte_umkodierung_X = korrektes Encoding über umkodierung_X
  Für jedes Feature in studierende_attribute:
    Wenn das Feature vom Datentyp Object ist:
      Gib Fehlermeldung zu fehlender Kodierung zurück
  Für jedes Feature in korrekte_umkodierung_X:
    Wenn der Datentyp des Features in studierende_attribute und
        korrekte_umkodierung_X unterschiedlich ist:
      Gib Fehlermeldung zu fehlerhaft durchgeführter Kodierung zurück
    Wenn der Datentyp des Features in studierende_attribute und
        korrekte_umkodierung_X jeweils Integer, die Werte aber
        unterschiedlich sind:
      Gib Fehlermeldung zu fehlerhaft durchgeführter Kodierung zurück
```

List. 2: Überprüfung auf richtige Verwendung des Feature Encodings.

und Funktionen nutzen und nicht etwa eine bereits fertige Funktion eine Bibliothek. Hier helfen sogenannte Search Patterns, mit denen der Einsatz bestimmter Methoden getestet werden kann. Anspruchsvoller wird dieser Abgleich, wenn statt Teilfunktionen *ganze Algorithmen* wie etwa der ID3-Algorithmus umgesetzt werden. Hier muss der Abgleich auf dem gesamten Klassifikationsergebnis erfolgen. Als Zwischenschritt kann der Vergleich von Evaluationsmaßen (wie z. B. Accuracy, Precision, Recall) von Musterlösung und Studierendenlösung einen Hinweis geben, ob sich die Lösung zumindest in der erwarteten Spanne bezogen auf die Qualitätsmaße befindet.

4 Fazit und Ausblick

Für das hier vorgestellte und bereits umgesetzte Lernszenario 1 erfolgt im kommenden Semester eine Evaluierung im Praxistest im Rahmen der Lehrveranstaltung. Unter Einbezug der dort gewonnenen Erkenntnisse, wird nicht nur dieses Szenario weiter verbessert, sondern parallel dazu auch Lernszenario 2 umgesetzt. In diesem werden die Ansätze aus Lernszenario 1 erweitert, um sowohl Grundlagenwissen zum Programmieren in Python mit dem Fokus auf Data Science und Maschinelles Lernen, Datensatz- und Programmierbibliotheksspezifische Inhalte, wie z. B. Visualisierung von Lernergebnissen auf Hyperspektralbildern, Anwendung unterschiedlicher Methoden für unterschiedliche Datensatztypen (Zeitreihen, visuelle oder akustische Datensätze, . . .), als auch die korrekte Auswertung und Darstellung der

Lernergebnisse im jeweiligen datensatzspezifischen Kontext zu vermitteln. Zusätzlich soll die Bearbeitung unscharfer, unbekannter oder heterogener Daten sowie deren Verarbeitung vermittelt werden. Die Verwendung entsprechend realer, größerer Datensätze wird darüber hinaus die Verwendung von rechenstarken Hardwaretools vermitteln, deren Effekte sich bei der Verwendung kleiner Standarddatensätzen nur schwer darstellen lassen.

Die hier vorgestellten Lerninhalte sind als Open Educational Resource konzipiert und werden nach der Evaluierung anderen Lehrenden zur Verfügung gestellt. Darüber hinaus ist eine Datensatzbibliothek aus Datensätzen der angewandten Forschung als auch der Industrie für die Studierenden geplant, welche u. a. für den Selbstlernkurs zur Verfügung stehen wird. Diese Daten decken einen großen Bereich an möglichen mehrdimensionalen und heterogenen oder unvollständigen Daten ab, sodass nach dem Erlangen eines Grundverständnisses zu Datensätzen eine tiefgreifendere Problembehandlung möglich wird. Dies dient auch zur Vorbereitung für die spätere Projektarbeit im Studium, da Studierenden bereits frühzeitig an praxisrelevante Arbeit herangeführt werden.

Literaturverzeichnis

[Bo17] Bott, O. J.; Fricke, P.; Priss, U.; Striewe, M.: Automatisierte Bewertung in der Programmierausbildung. Waxmann, 2017, ISBN: 9783830936060.

[Fa15] Fangohr, H.; O'Brien, N. S.; Prabhakar, A.; Kashyap, A.: Teaching Python programming with automatic assessment and feedback provision. ArXiv abs/1509.03556/, 2015.

[KLN23] Kelly, M.; Longjohn, R.; Nottingham, K.: The UCI Machine Learning Repository, last accessed: 2023-09-07, 2023, URL: https://archive.ics.uci.edu.

[Ly12] Lynch, C.; Ashley, K. D.; Aleven, V.; Pinkwart, N.: Ill-Defined Domains and Adaptive Tutoring Technologies. In: Adaptive Technologies for Training and Education. Cambridge University Press, 2012.

[Ro01] van Rossum, G.: unittest – Unit testing framework, last accessed: 2023-06-08, 2001, URL: https://docs.python.org/3/library/unittest.html.

[Ro23] Rodríguez-del-Pino, J. C.: VPL, the Virtual Programming lab for Moodle, last accessed: 2023-09-07, 2023, URL: https://vpl.dis.ulpgc.es/.

[Sp16] Spannagel, C.: Flipped Classroom, last accessed: 2023-06-07, 2016, URL: https://cspannagel.wordpress.com/category/flippedclassroom-2/.