Fakultät für Wirtschaftswissenschaften

DIPLOMARBEIT

KNIME, TensorFlow und scikit-learn
- Vergleich anhand Bildern, Texten und Daten -

Diplomarbeit zur Erlangung des Grades eines
Diplom-Wirtschaftsinformatikers (FH)
der Hochschule Wismar

eingereicht von:	Andreas Eckl geboren am 19. März 1980 in Schwandorf Studiengang Wirtschaftsinformatik, WI 2018
Betreuer:	Prof. Dr.-Ing. Sabine Schumann
weitere Gutachter:	Prof. Dr. rer. nat. Jürgen Cleve

Unterhaching, den 17. Mai 2019

Vorwort

Um die Nichtnotwendigkeit des Vorworts wissend ist dieses im Sinne aller kurzgehalten.

Die Existenz dieser Arbeit basiert auf der Zusammenarbeit von Alan Turing und Tommy Flowers. Deren gemeinsame Arbeit zur Erschaffung des Colussus regte mich zur weiteren Auseinandersetzung mit der Informatik an. Diese führte zu einem Studium der Wirtschaftsinformatik.

Auch die vorliegende Diplomarbeit kann von der Arbeit der beiden abgeleitet werden, wurde doch damals aus dem doppelten Versand einer Nachricht das Wissen um die Verschlüsselungsmethode und deren Entschlüsselung eruiert.

Natürlich möchte ich mich an dieser Stelle bei Prof. Dr.-Ing. Sabine Schumann und Prof. Dr. rer. nat. Jürgen Cleve für die Betreuung dieser Arbeit bedanken.

Ich widme diese Arbeit meiner Frau, ohne deren Verständnis und Unterstützung es gar nicht erst bis zum Schreiben einer Diplomarbeit gekommen wäre.

Inhalt

I. Abbildungsverzeichnis

II. Tabellenverzeichnis

III. Abkürzungsverzeichnis

Abkürzung	Bedeutung
AMA	Ask Me Anything
API	Application Programming Interface
ARM	Advanced RISC Machines (ursprünglich Acorn RISC Machines)
ARRF	Attribute-Relation File Format
ATI	ATI Technologies Incorporated
AVX	Advanced Vector Extensions
BMP	Bitmap
BSD	Berkeley Software Distribution
CNN	Convolutional Neural Network
CNTK	Microsoft Cognitive Toolkit
CPU	Central Processing Unit
CSV	Comma Separated Value
CUDA	Compute Unified Device Architecture
DARPA	Defense Advanced Research Projects Agency
DNN	Deep Neural Network
EEG	Elektroenzephalogramm
FAQ	Frequently Asked Questions
GB	Gigabyte
GIF	Graphics Interchange Format
GNU	GNU's Not Unix
GPU	Graphics Processing Unit
GUI	Graphical User Interface
ID	Identification
IDE	Integrated Development Environment
IDS	Intrusion Detection System

ILSVRC	ImageNet Large Scale Visual Recognition Challenge
INRIA	Institut National de Recherche en Informatique et en Automatique
JDBC	Java Data Base Connectivity
JEP	Java Embedded Python
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
KI	Künstliche Intelligenz
knar	KNIME Archive File
KNIME	Konstanz Information Miner
knwf	KNIME Workflow Files
LFW	Labeled Faces in the Wild
MB	Megabyte
MINA	Montreal Institute for Learning Algorithms
ML	Machine Learning
MNIST	Mixed National Institute of Standards and Technology
NAN	Not A Number
NCSU	North Carolina State University
PIP	Pip Installs Packages
PMML	Predictive Model Markup Language
PNG	Portable Network Graphics
RAM	Random-Access Memory
RCV1	Reuters Corpus Volume 1
RELU	Rectified Linear Unit
REST	Representational State Transfer
RGB	Rot Grün Blau
RISC	Reduced Instruction Set Computer
RMSE	Rooted Mean Sqared Error
RNN	Rekurriertest neuronales Netz

SQuAD	Stanford Question Answering Dataset
SVG	Scalable Vector Graphics
SVM	Support Vector Machine
TED	Technology Entertainment Design
TFRC	TensorFlow Research Cloud
TPU	Tensor Processing Unit
UCI	University of California
UINT	Unsigned Integer
URL	Uniform Resource Locator
US	United States
USD	US-Dollar
XLA	Accelerated Linear Algebra
XLS	Excel Spreadsheet
XLSX	Excel Microsoft Office Open XML Format Spreadsheet
XML	Extensible Markup Language
µs	Mikrosekunde

1. Einleitung und Problemstellung

1.1 Gegenstand und Ziel

In jedem Unternehmen, das im Verlauf seiner Unternehmensprozesse Computer nutzt, existiert eine gewisse Anzahl an computerlesbar gespeicherten Daten, welche für das jeweilige Unternehmen von Relevanz sind. Laut statistischem Bundesamt nutzten im Jahr 2018 95 Prozent aller deutschen Unternehmen Computer [COMP_DE].

Je nach Art des Unternehmens, sowie dessen Größe, liegen diese Daten in unterschiedlichen Formaten und auf eine unterschiedliche Anzahl an Quellen verteilt vor. Die Unterschiedlichkeit der Formate basiert auf der Unterschiedlichkeit der Inhaltstypen. So wird ein Abschleppdienst beispielsweise unter anderem auch mehrere Bilddateien vorhalten, um den Zustand abgeschleppter Fahrzeuge zu dokumentieren. Diese Dokumentation des Fahrzeugzustandes vor und nach Erbringung der Dienstleistung dienen der Absicherung der abschleppenden Unternehmen gegen eventuelle Regressansprüche durch unfreiwillig Abgeschleppte. Um beispielsweise im Falle des besagten Regresses, das Wissen über einen eventuellen Zustandswechsel während des Abschleppvorganges zu extrahieren, bedarf es menschlicher Interaktion. So ergeben die Daten einer Bilddatei erst durch Öffnen, anschließendes menschliches Betrachten und Beurteilen die Information, welches Fabrikat abgebildet ist. Diese Information reicht jedoch nicht aus, um zu wissen, ob ein Fahrzeug während des Transports beschädigt wurde. Die Schaffung dieses Wissens erfordert zum einen die Sichtung aller vorhandenen Abbildungsperspektiven des Fahrzeugs vor Abtransport durch die betrachtende Person. Zum anderen muss die betrachtende Person auch alle vorhandenen Abbildungsperspektiven des Fahrzeugs nach dessen Abladung sichten, um das Wissen über einen Zustandswechsel zu gewinnen.

Dieser eben beschriebene Prozess lässt sich mit den Mitteln des klassischen Programmierens partiell unterstützen, aber nicht vollumfänglich lösen. Bei der bildbasierten Erkennung von Hersteller und Modell werden bereits, unter Nutzung eines CNN, Ergebnisse mit akzeptablem Klassifikationswert erzielt [MAKE_MODEL].

Um maschinenbasiert Wissen aus Daten zu gewinnen, bedarf es unter anderem einer passenden Software, beziehungsweise eines Programmiergerüsts. Ein Faktor in der Findung des geeigneten Werkzeuges ist dessen Bedienbarkeit, welche beispielsweise aus der Nutzung und Notwendigkeit von Code auf unterschiedlichen Systemen ableitbar wird. Weitere Faktoren stellen auch Funktionsumfang, Kompatibilität, Leistungsdifferenzen untereinander, die Art der zu analysierenden Daten, und natürlich auch finanzielle Aspekte dar. Um den finanziellen Aspekt ausblenden zu können, werden in dieser Arbeit nur quelloffene Lösungen verglichen. Bei den verglichenen Lösungen handelt es sich um „KNIME“, „TensorFlow“ und „scikit-learn“, diese werden bezüglich der vorgenannten Kriterien analysiert. Das Ziel dieses Vergleichs stellt die Beantwortung der Frage nach einem für die lesende Person individuell geeigneten Arbeitsmittel dar.

Die aus diesem Ziel abgeleitete Forschungsfrage lautet demnach:

Welche der untersuchten Lösungen eignet sich für welchen Anwender?

1.2 Motivation

Das Bundeskabinett hat am 15.11.2018 die „Strategie Künstliche Intelligenz der Bundesregierung“ beschlossen [KI_KABINETT]. Im Zuge dieser soll neben der Schaffung von mindestens 100 zusätzlichen neuen Professuren auch die „KI-spezifische Unterstützung von mittelständischen Unternehmen“ ausgeweitet werden [KI_STRATEGIE, S..6]. Daraus folgend ergeben sich für Unternehmen, die Werkzeuge des maschinellen Lernens implementieren wollen, nicht nur die im vorhergehenden Abschnitt beispielhaft thematisierten zeitökonomischen Vorteile, sondern auch klar pekuniäre. Somit besteht für 95 Prozent der deutschen Unternehmen die Möglichkeit, von der Klärung der Forschungsfrage zu profitieren.

Die Nutzung von computerlesbaren Informationen mit dem Ziel von dem daraus gewonnenen Wissen zu profitieren, ist auch die Begründung für die Zuordnung der Thematik zur Wirtschaftsinformatik. Diese stellt ein Bindeglied zwischen der oft technisch ausgerichteten Informatik und der anwendungsorientierten Betriebswirtschaftslehre dar. In dieser Wissenschaft werden zur Unterstützung von Geschäftsprozessen in Unternehmen computergestützte Informations- und Kommunikationssysteme sowohl geplant, als auch entwickelt und angewandt [WI_MERTENS_ET, S. 2f.]. Neben den eben bezeichneten Teilgebieten nutzt die Wirtschaftsinformatik auch das Informationsmanagement zur Generierung neuer ökonomischer Lösungen. Die folgende Grafik gibt den Bezug der einzelnen Arbeitsgebiete der Wirtschaftsinformatik zueinander wieder,

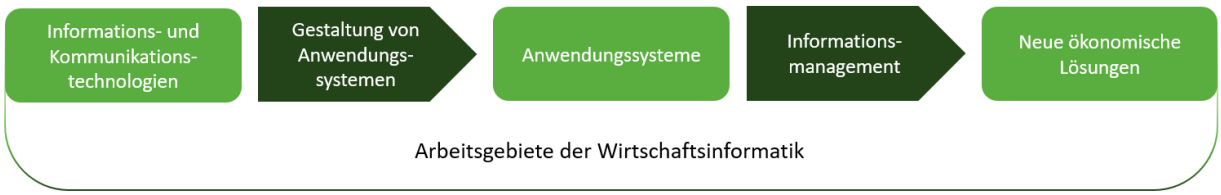


Abbildung 1: Themenfelder der Wirtschaftsinformatik
(eigene Darstellung, basierend auf [WI_MERTENS_ET, S. 3])

Die zur Generierung von Wissen benötigten Informationen finden sich in Bestandteilen der Informations- und Kommunikationstechnologien wieder. Diese Bestandteile sind Datenträger auf denen die Informationen vorgehalten werden. Die Integration und Nutzung der besprochenen Lösungen wirkt sich einerseits auf die Gestaltung, und in Folge der Integration auch auf die Nutzung, der Anwendungssysteme aus. Decken diese Systeme neue Bereiche des Informationsbedarfs ab, so helfen sie bei der Schaffung neuer ökonomischer Lösungen. Der Informationsbedarf ist dem Informationsmanagement zuzuordnen. Damit berührt die Forschungsfrage alle Bestandteile der Wirtschaftsinformatik.

1.3 Vorgehensweise bei der Bearbeitung

Die Arbeit stellt zur Kenntnis theoretischer Grundlagen zuerst kurz Hilfsmittel vor, die die Vergleichskandidaten unterstützen, oder deren Nutzung vereinfachen. Anschließend werden die eigentlichen Kandidaten kurz vorgestellt, bevor deren Entstehungsgeschichte, sowie die Information unter welcher Lizenz diese veröffentlicht werden, Erwähnung findet. Um die Eigenschaften bezüglich der Bedienbarkeit der Lösungen zu untersuchen werden die unterschiedlichen Schritte zur Installation auf verschiedenen Systemen, sowie die jeweiligen grafischen Benutzeroberflächen betrachtet. Dafür nutzt der Autor zwei verschiedene handelsübliche Rechner: eine Windows 10 Maschine mit einem i7 920, 24 GB RAM und einer ATI Radeon HD 5700, sowie einen Raspberry Pi 3 B+. Die Hardware des Windows-Rechners wird im Lauf der Arbeit auch mit Ubuntu 18.04.2 LTS genutzt um Antworten für die Nutzung unter Linux zu geben. Der Raspberry Pi wird primär mit dem Betriebssystem Raspbian betrieben. Der Funktionsumfang wird beispielhaft anhand der in den Lösungen integrierten Musterdaten, und den unterstützten Programmiersprachen betrachtet. Die Basis dafür stellt eine Literaturrecherche dar. Als Quellen dienen neben den Dokumentationen der Produzenten der eingesetzten Lösungen die Lösungen selbst, sowie weitere relevante Publikationen. Um Differenzen bezüglich der Kompatibilität abzubilden, werden ebenfalls in Sekundärforschung gewonnene Informationen zu Interoperabilität und unterstützten Quellen der Lösungen aufbereitet. Die Leistungsdifferenzen werden anhand der Aufbereitung von Informationen über die jeweiligen Fähigkeiten der Hardwarenutzung, sowie der Erstellung eines Leistungsvergleichs wiedergegeben. Dieser Vergleich findet unter der Verwendung von Bildern, Texten und Daten statt. Da der finanzielle Aspekt ausgeblendet wird, ist nur das cloudbasierte Training von Modellen unter der Nutzung von FloydHub Bestandteil der Ausführungen. Diese Plattform wirbt in der Beschreibung seiner „Beginner“-Option, unter der Prämisse der Veröffentlichung, mit der kostenlosen Erstellung unbegrenzt vieler Projekte und Datensätze [FLOYD_PLAN].

1.4 Abgrenzung

Um dem Umfang einer Diplomarbeit zu entsprechen, behandelt diese Arbeit weder die mathematischen Hintergründe, noch erläutert sie die Arten und Funktionsweisen künstlicher neuronaler Netze. Dass es sich bei den in dieser Arbeit thematisierten neuronalen Netzen um künstliche handelt findet keine weitere Erwähnung, da die Tatsache als Schussfolgerung aus der Wahl des Titels angesehen wird. Ebenfalls nicht erwähnt werden die,zur Installation der jeweiligen Betriebssysteme nötigen Schritte. Deren Auslassung findet, zum einen aus Platzgründen, zum anderen weil der Autor von der Existenz eines bereits aufgesetzten Systems beim voraussichtlichen Nutzer ausgeht, statt. Auch ausgelassen werden die Aspekte der Distribution und anschließenden Nutzung von generierten Modellen. Diese Punkte finden keine Erwähnung, da der hier mögliche Umfang als Material zur Erarbeitung einer eigenen Arbeit angesehen wird. Diese Ansicht basiert darauf, dass mögliche Anwendungsfälle der Distribution sowohl in der festen Integration in neu zu generierende Softwarelösungen, der Erstellung einer Web-Applikation mit Serveranbindung, oder auch in der Integration in Firmware liegen können. Dabei erhebt diese Aufzählung keinerlei Anspruch auf Vollständigkeit.

2. Theoretische Grundlagen

Da diese Arbeit sich des Vergleiches von Werkzeugen zur Gewinnung von Wissen aus Daten widmet, werden an dieser Stelle zuerst diese Begrifflichkeiten definiert. Basis beider Begriffe sind die zu Grunde liegenden Zeichen. Bei diesen handelt es sich um Buchstaben, Zahlen und Sonderzeichen wie „/“ oder „$“. Enthält beispielsweise eine Zelle einer Tabelle die Zeichen „2“ und „0“ zu „20“ verkettetet, so stellt dies ein Datum dar. Bei diesem Begriff handelt es sich nicht um die mit dem gleichen Begriff belegte Kalender- oder Zeitangabe, sondern um den Singular von Daten. Somit sind Daten einzelne oder verkettete Zeichen eines Zeichenvorrates nach definierten Syntaxregeln.

Wird ein Datum durch Semantik oder Kontext an eine Bedeutung gekoppelt, so wird es zur Information. Im gegebenen Beispiel kann es sich dabei um eine Zeilenüberschrift der Tabelle mit dem Inhalt „Tempoüberschreitung in km/h“ handeln. Also handelt es sich bei einer Information um ein an eine Bedeutung gekoppeltes Datum.

Um aus diesen Informationen Wissen zu generieren sind im Regelfall mehrere Informationen zu betrachten. Wenn aus dieser Betrachtung eine Nutzbarkeit resultiert, wurde Wissen geschaffen. Wissen bezeichnet eine Information in Verbindung mit der Fähigkeit diese zu benutzen (vgl. [DATA_MINING, S. 37,38]; [BODEN_DATEN, S. 1]).

2.1 Jupyter

Das Projekt Jupyter ist ein nicht gewinnorientiertes, quelloffenes Projekt, welches sich vom IPython Projekt abkoppelte. Dieser Entwicklungsprozess war dadurch bedingt, dass man interaktive Datenwissenschaft, ebenso wie wissenschaftliche Berechnungen, über alle Programmiersprachen hinweg unterstützen wollte. Das sechzehn Mitglieder umfassende Lenkungsgremium des Projekts will Jupyter dauerhaft kostenlos und hundertprozentig quelloffen halten. Die finanziellen Mittel dafür stammen von elf Sponsoren und zehn institutionellen Partnern. Letztere zahlen die Gehälter des Lenkungsgremiums. In der Quelle werden neben den Mitgliedern beider Gruppen auch die des Lenkungsgremiums genau benannt [JUPYTER, /about].

Den ersten Platz des Grundlagenkapitels verdankt Jupyter der Tatsache, dass eine Vielzahl dessen Lösungen, wenn Python auf dem genutzten System installiert ist, in der Lage sind mit allen drei Vergleichskandidaten zu interagieren. Die schwammige Formulierung „Lösungen“ wird aufgrund des beträchtlichen Umfangs des Projektportfolios genutzt. Zum Erstellungszeitpunkt dokumentiert die Projektseite die auf der folgenden Seite abgebildeten Lösungen. Die Betrachtung der Lösungen konzentriert sich in dieser Arbeit auf „Jupyter Notebook“ und „JupyterLab“. Beide arbeiten mit dem „ipynb“ Format, diese Abkürzung stand ursprünglich für „IPython Notebook“, dieses wurde vom „Jupyter Notebook Format“ beerbt.

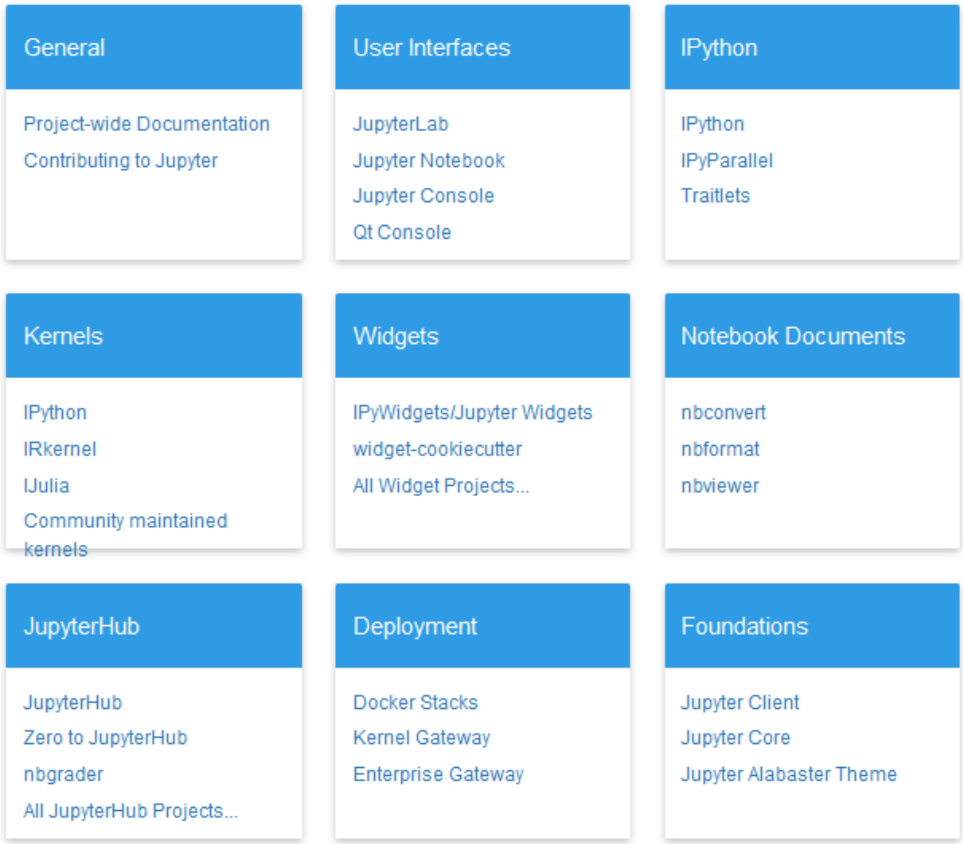


Abbildung 2: Projekt Jupyter Produktportfolio [JUPYTER, /documentation]

Hinter dem Format verbergen sich einfache JSON Dokumente die Text, Quellcode, Metadaten und reichhaltige Medienausgaben beinhalten können [JUPY_FORMAT]. Unter Letztgenanntem ist beispielsweise die Einbindung von online befindlichen Mediendateien wie Audio- oder Videodaten zu verstehen.

2.1.1 Jupyter Notebook

„Jupyter Notebook“ bezeichnet eine quelloffene Webanwendung die es ermöglicht, Dokumente zu erstellen und zu teilen. Diese Dokumente können neben Quellcode auch Gleichungen, Visualisierungen und textliche Erläuterungen enthalten. Der Vorteil, den die Nutzung von Jupyter Notebook bringt, liegt in der Tatsache, dass der jeweilige Code unverzüglich in der Software ausgeführt werden kann. Dies geschieht beispielsweise durch gleichzeitiges Drücken der Tasten Steuerung und Enter. Im Unterschied zu klassischen IDEs kann die anwendende Person den Code auch auf mehrere sogenannte „Zellen“ verteilen, um somit beispielsweise die Ausführung zu steuern. Die denkbaren Anwendungszwecke von Jupyter Notebook reichen von Datenbereinigung und numerischer Simulation über Datenvisualisierung bis hin zu maschinellem Lernen und mehr [JUPYTER]. Nachfolgend ist eine beispielhafte Instanz von Jupyter Notebook mit fünf befüllten Feldern, den eben thematisierten „Zellen“, vor deren Ausführung abgebildet.

Das Jupyter Notebook dazu befindet sich im Stammverzeichnis des mit der Arbeit gelieferten USB-Sticks als „Jupyter-Musterdarstellung.ipynb“.

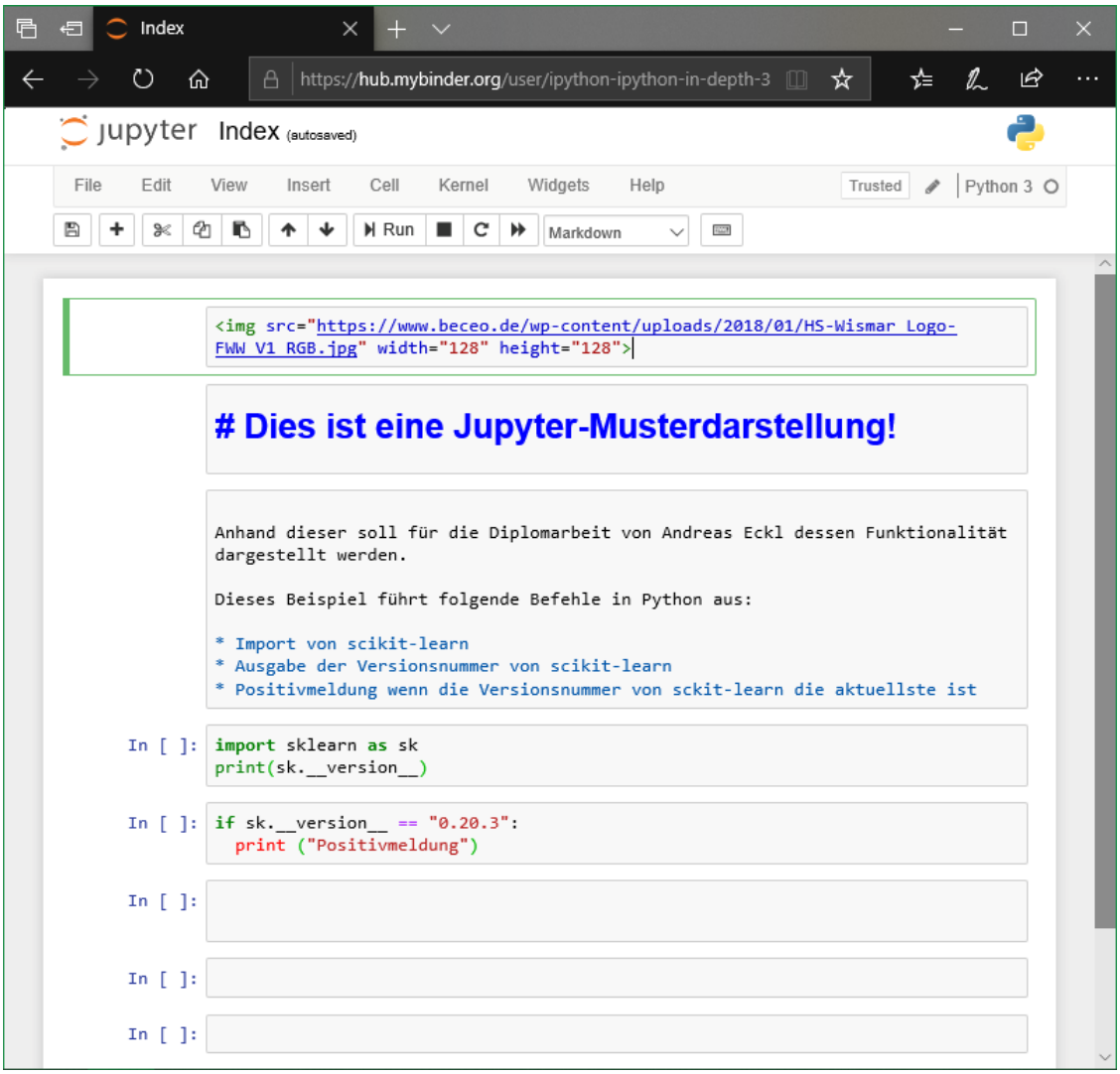


Abbildung 3: Ein Jupyter Notebook mit beispielhaftem Inhalt (Bildschirmfoto samt Inhalt selbst in Jupyter Notebook erstellt)

Die Bezeichnung „Zellen“ steht für die grau hinterlegten, mit Code in unterschiedlichen Sprachen befüllten Felder. Wird der Code durch Klicken der Schaltfläche „Cell“, mit anschließendem Wählen der Option „Run all“ ausgeführt, stellt sich das Notebook wie in der Abbildung auf der Folgeseite dar.

Das in den Abbildungen demonstrierte Sprachverständnis der Jupyter Notebooks bildet nur einen Bruchteil der tatsächlich möglichen Sprachen ab, anhand des Installierens zusätzlicher Systemkerne können weitere Sprachen genutzt werden. Die GitHub-Seite der Kernels listet zum Zeitpunkt der Erstellung dieser Zeilen sechsundachtzig individuelle Sprachen auf [JUPY_KERNELS].

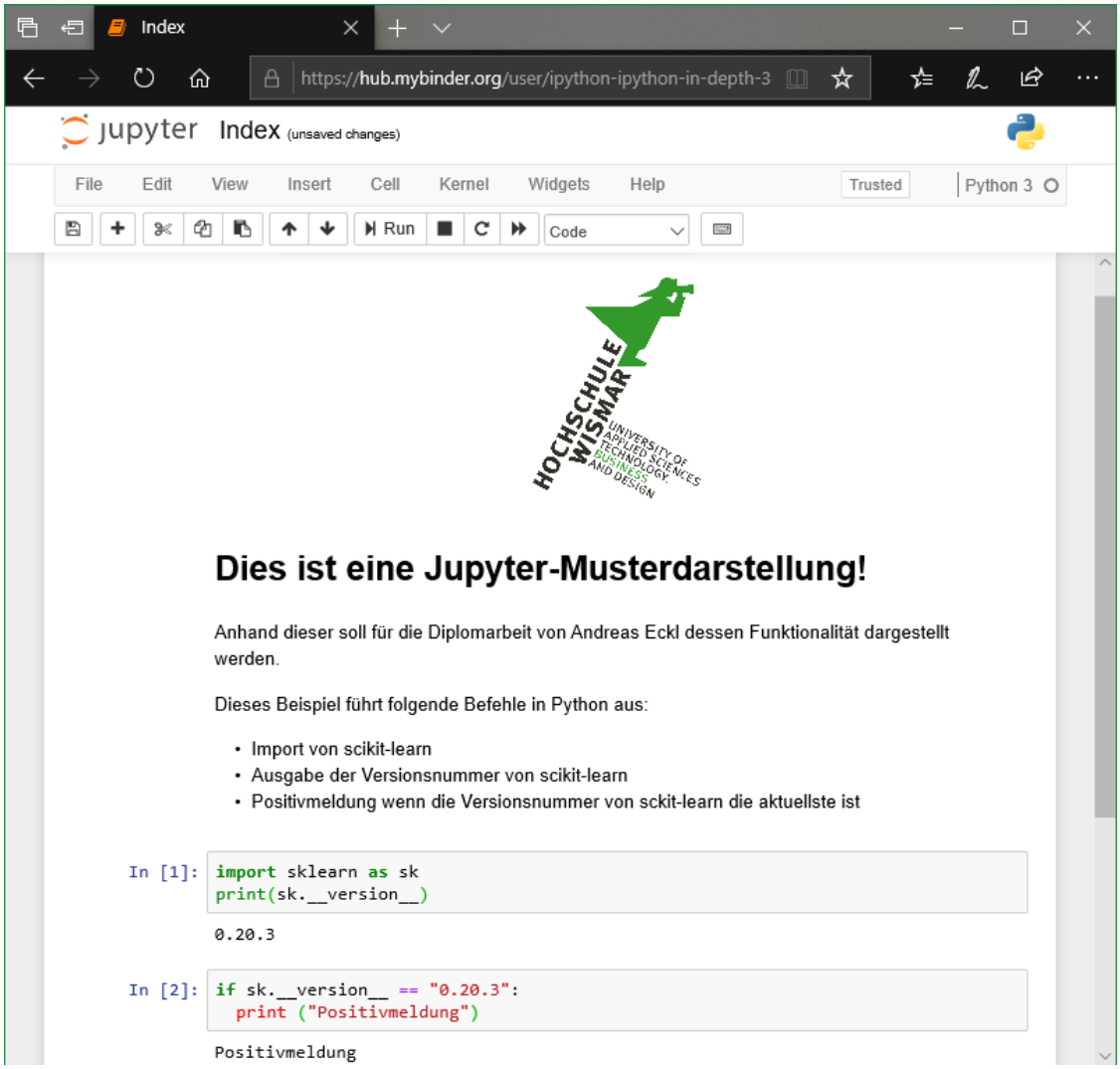


Abbildung 4: Das Jupyter Notebook aus Abbildung 3 nach Ausführung des Codes der Zellen (Bildschirmfoto samt Inhalt selbst erstellt)

Jupyter Notebook findet sowohl in der Dokumentation von TensorFlow [TF_DOCKER], als auch in der von scikit-learn Verwendung [SK_IRIS]. Seit der Version 3.7 ist KNIME in der Lage Python Code aus Jupyter Notebooks zu nutzen. Zusätzlich können auch KNIME Workflows in Jupyter Notebooks genutzt werden. Letzteres erfordert die Installation des KNIME Python Paketes [KN_JUPYTER].

Die Installation von Jupyter Notebook findet in den Abschnitten der Systemeinrichtung von TensorFlow und scikit-learn Verwendung. Unter Linux bewirkt der folgende Befehl die Installation [JUPYTER_INSTALL].

```
pip3 install jupyter
```

Nach dieser Eingabe ist das Prüfen der Versionsnummer empfehlenswert, da die Version 5.7.7 noch Teile einer angreifbaren Lücke enthielt. Diese wurde in der Version 5.7.8 nachhaltiger geschlossen [SK_VULN]. Die Überprüfung der Version erfolgt über die erste, das eventuell notwendige Upgrade, anhand der zweiten Zeile des folgenden Codes.

```
jupyter notebook –version
pip3 install –upgrade 'notebook>=5.7.8'
```

2.1.2 JupyterLab

Da JupyterLab letztendlich Jupyter Notebook ablösen wird findet auch dieses hier Erwähnung [JUPY_LAB_OV]. Jason Grout, ein Mitglied des Lenkungsgremiums und Jupyter Entwickler, teilte bereits auf der QCon.ai 2018 mit, dass die Anzahl an Entwicklungsbeiträgen denen des Jupyter Notebooks entspräche. Bei der QCon.ai handelt es sich um eine jährlich stattfindende Entwicklerkonferenz für angewandte KI Software [QCONAI]. Die folgende Abbildung zeigt die entsprechende Folie und den Entwickler auf besagter Konferenz.

Abbildung 5: Jason Grout auf der QCon.ai 2018
(selbst erstellte Momentaufnahme aus [GROUT_QCONAI])

JupyterLab erinnert an IDEs, da es die Funktionen aus Jupyter Notebook um einen Dateibrowser und konfigurierbare Fenster ergänzt. In diese können Texteditoren, Terminals, Dokumente und weitere Objekte nebeneinander eingebunden werden. Neben anpassbaren Tastenkürzeln ist die Nutzung von Tastenzuordnungen der Texteditoren vim, emacs und Sublime Text möglich. Für eine weitere Möglichkeit der Anpassung an individuelle Vorlieben oder Bedürfnisse sorgen die Erweiterungen von JupyterLab. Diese können neben rein kosmetischen Änderungen auch neue Dateieditoren oder maßgeschneiderte Komponenten umfassen. [JUPY_LAB_OV]. Die Entwicklung dieser Individualkomponenten wird auch beispielhaft auf der Projektseite erläutert [JUPY_LAB_EXT]. Selbst ohne die Installation von Erweiterungen ist JupyterLab in der Lage mit Material verschiedensten Ursprunges zu arbeiten. Diese Fähigkeit bildet die Darstellung auf der folgenden Seite ab.

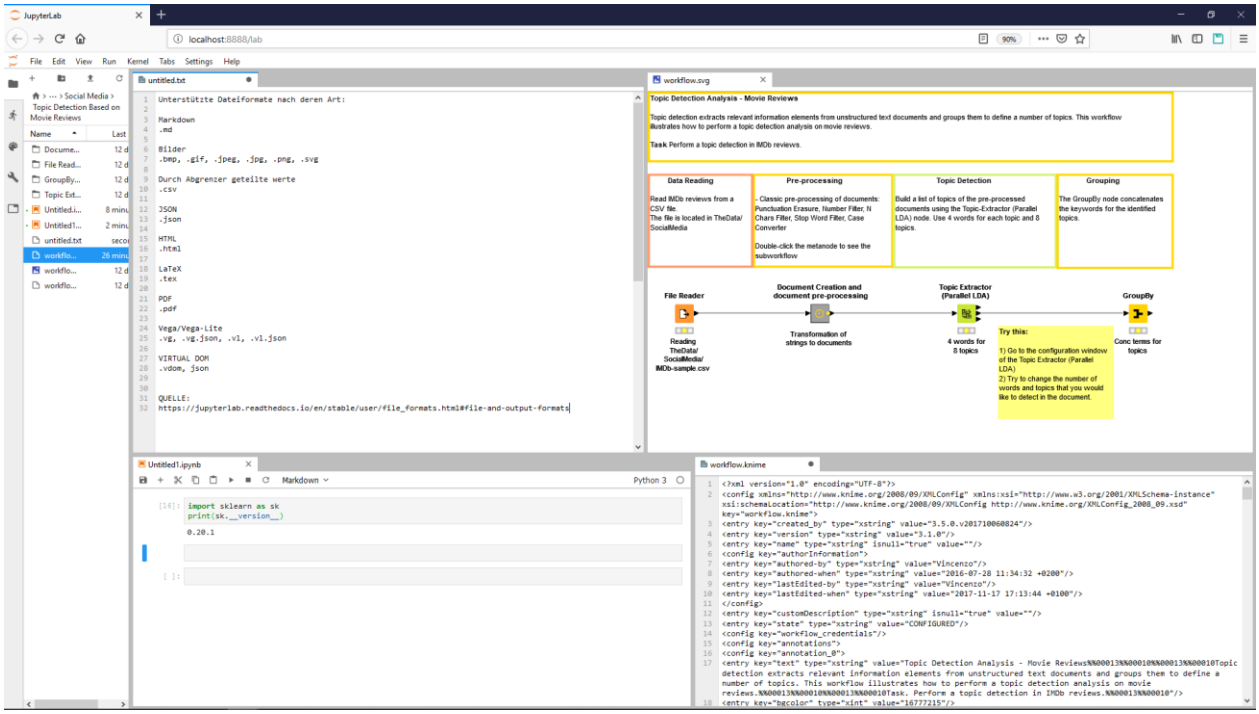


Abbildung 6: Bildschirmfoto einer beispielhaft belegten JupyterLab Instanz (Bildschirmfoto samt Inhalt in JupyterLab selbst erstellt)

In der Grafik findet sich links die ausblendbare Navigationsleiste. Diese kann wahlweise mit einem Dateibrowser, der Anzeige der laufenden Terminals und Kernels, Kommandos, einem „Zellinspektor“ oder einer Darstellung der offenen Reiter befüllt werden. Der Zellinspektor stellt ein Werkzeug zur Erstellung von Präsentationen auf Basis von Jupyter Notebooks dar.

Die in der Grafik abgebildete Textdatei listet die nativ von JupyterLab unterstützten Dateiformate auf [JUPY_LAB_FILES], welche aufgrund der schlechten Lesbarkeit in der Anlage E1 noch einmal vergrößert abgebildet wurde. Die untere linke Hälfte zeigt ein Jupyter Notebook mit einem Python-Kernel. Die obere rechte Hälfte der Grafik belegt eine bildliche Darstellung eines KNIME-Workflows, während die rechte untere Hälfte die geöffnete XML Datei desselben Workflows wiedergibt.

JupyterLab ist in Anaconda, einer Freemium Python Distribution, bereits in der Version 0.35.4 enthalten [ANACONDA_NEW]. Auf Linux basierten Systemen kann JupyterLab auch ohne Anaconda oder conda genutzt werden. Dafür ist lediglich die Installation anhand pip nötig, diese startet der folgende Befehl [JUPY_LAB_INSTALL].

```
pip3 install jupyterlab
```

Wird der Befehl, wie in der Quelle beschrieben, mit pip ausgeführt, moniert das System eine zu niedrige Python Version. Es bedarf mindestens der Version 3.5 um JupyterLab nutzen zu können. Die Ausführung mit pip3 bedarf auf dem Raspberry zwei Minuten und fünfundvierzig Sekunden, unter Ubuntu 18.04.2 LTS zwei Minuten und sieben Sekunden.

2.2 NumPy

NumPy ist laut Bezeichnung auf der Projektseite, das fundamentale Paket für wissenschaftliche Berechnungen in Python. Der Inhalt des unter der BSD Lizenz veröffentlichten Werkzeugs umfasst verschiedene Fähigkeiten. Diese beinhalten neben der Bereitstellung eines n-dimensionalen Array-Objekts, Sendefunktionen, Werkzeugen der Integration von C/C++ und Fortran Code auch mathematische Funktionen. Zu diesen Funktionen zählen Berechnungen der linearen Algebra, die Fourier Transformation und die Erstellung von Zufallszahlen.

Zusätzlich zu den bisher genannten Fähigkeiten kann NumPy auch als multidimensionaler Container generischer Daten fungieren. Dabei können beliebige Datentypen definiert werden, weswegen NumPy in der Lage ist, ein breites Spektrum an Datenbanken zu integrieren [NUMPY]. Diese Tatsache führt zur Möglichmachung der Nutzung von NumPy seitens der Entwickler, sowohl derer von scikit-learn, als auch derer von TensorFlow. Bei scikit-learn wird NumPy als vorhandene Voraussetzung auf dem System gefordert [SK_INSTALL]. In TensorFlow existieren außer zwei dedizierten Klassen [TF_PY, /contrib/checkpoint/NumpyState; /contrib/timeseries/NumpyReader] weitere Möglichkeiten der Nutzung von NumPy Objekten [TF_PY, /estimator/inputs/numpy_input_fn; TF_NUMPY_MORE]. Damit KNIME NumPy nutzen kann, benötigt es eine parallele Python Installation, die KNIME AG nutzt in ihrem Installationsleitfaden die Anaconda Python Distribution von Continuum Analytics [KNIME_NUMPY]. Deren Installation und Installationsumfang werden unter dem Punkt 4.3.1 geschildert.

Um NumPy eigenständig auf einem Linux basierten System zu installieren, reicht laut der Python Software Foundation folgender Code [NUMPY_INSTALL].

```
pip install numpy
```

NumPy ist Bestandteil des SciPy Ökosystems. Dieses wird im weiteren Verlauf des folgenden Abschnittes thematisiert. Im Abschnitt zu scikit-learn findet sich eine kontextuelle Einordnung von NumPy.

2.3 SciPy

Im Kontext dieser Arbeit bezeichnet SciPy die gleichnamige Bibliothek. Die SciPy-Bibliothek ist eines der Kernpakete des SciPy-Stacks, und liefert viele nutzerfreundliche und effiziente numerische Routinen. Unter diesen finden sich beispielsweise Routinen zur numerischen Integration, ebenso wie solche zur numerischen Optimierung [SCIPY]. Die Bibliothek benötigt NumPy, da sie für die Arbeit mit dessen n-dimensionalen Array-Objekten erstellt wurde [SCIPY_INSTALL]. Die Relevanz von SciPy für diese Arbeit ergibt sich aus der Tatsache, dass scikit-learn das Vorhandensein dessen auf dem System voraussetzt [SK_INSTALL]. In TensorFlow existiert eine Klasse, die die Nutzung der SciPy-Minimierungsfunktion „optimize.minimize“ implementiert [TF_SCIPY].

Unter Windows wird SciPy beispielsweise mit der Installation von Anaconda Python nutzbar. Die Installation dieser Distribution ist unter dem Punkt 4.3.1 genau beschrieben.

Alternativ kann auch im Windows Subsystem für Linux, wie unter anderen auf Linux basierenden Systemen, die folgende Eingabe zur Installation genutzt werden [SCIPY_INSTALL].

```
pip install scipy.
```

Eine grafische Einordnung der Bibliothek findet sich im Abschnitt zu scikit-learn.

Der Eigenname „SciPy“ wird, neben der hier thematisierten Bibliothek, mit drei weiteren Entitäten in Verbindung gebracht. Dazu zählt zum einen das SciPy Ökosystem, welches eine Sammlung von quelloffener Software für wissenschaftliche Berechnungen in Python darstellt. Auch die Gemeinschaft von Entwicklern und Benutzern des Stacks wird damit bezeichnet. Gleichlautend benannte Konferenzen, die sich dem Thema mathematischer Berechnungen in Python widmen stellen, nach der Bibliothek, die letzte Entität dar [SCIPY_ABOUT]. Neben der 2019 in Texas stattfindenden „SciPy“ [SCIPY_2019] existieren auch ein japanischer, europäischer, sowie ein indischer Ableger. Diese werden regional zuordenbar mit „SciPy Japan“, „EuroSciPy“ und „SciPy.in“ betitelt [SCIPY_ABOUT].

2.4 Pandas

Pandas ist ein Bestandteil des SciPy Ökosystems und liefert performante, einfach nutzbare Datenstrukturen [SCIPY_ABOUT]. Eine grafische Zuordnung ist im Abschnitt zu scikit-learn ersichtlich. Zusätzlich zur Lieferung der Datenstrukturen stellt pandas auch ein Werkzeug zur Datenanalyse in Python dar, welches die Arbeit mit relationalen oder etikettierten Daten einfach und intuitiv machen soll. Ein gesetztes Ziel des Entwicklerteams ist es, mit pandas das sprachenübergreifend mächtigste und flexibelste, quelloffene Werkzeug zur Analyse und Bearbeitung von Daten zur Verfügung zu stellen. Die Dokumentation von pandas liefert neben der Information über die auf der Nutzung von Cython basierende, hohe Geschwindigkeit unter anderem auch die Tatsache, dass pandas auf NumPy aufbaut. Eine weitere Eigenheit stellt die Nutzung zweier primärer Datenstrukturen dar. Die eindimensionale wird mit „Series“ bezeichnet, die zweidimensionale mit „DataFrame“. Um der Leserschaft weitere Schachtelsätze zu ersparen, nennt die folgende Aufzählung die für die Nutzung mit pandas geeigneten Daten [pandas, S. 13].

- Tabellarische Daten mit Spalten unterschiedlichen Inhalts wie in SQL Tabellen oder Excel Tabellen
- Daten von Zeitreihen, unsortiert und sortiert (diese bedürfen keiner festen Frequenz)
- Zufällige Daten einer Matrix mit bezeichneten Spalten und Zeilen, die Datentypen können sowohl homogen als auch heterogen sein
- Jede andere Form beobachteter oder statistischer Datensätze, die Daten müssen nicht gekennzeichnet sein

Die folgende Tabelle gibt „Dinge, die pandas gut kann“ wieder [pandas, S. 13f].

Tabelle 1: Dinge, die pandas gut kann
(schematische Darstellung basierend auf [pandas, S. 13f])

Objekt	Bearbeitungsmöglichkeiten der Objekte seitens pandas
Daten	◦ Einfacher Umgang mit fehlenden Daten (die als NAN dargestellt werden) bei Daten mit, und ohne Fließkomma ◦ Eindeutige oder automatische Datenjustage: -Objekte können eindeutig einem Kennzeichnungssatz zugeordnet werden -Objekte können durch Ignorieren der Kennzeichnungen seitens des Nutzers in den Berechnungen automatisch durch Series, DataFrame, etc. koordiniert werden ◦ Einfaches Konvertieren ungepflegter und unterschiedlich indizierter Daten aus anderen Python und NumPy Strukturen in DataFrame-Objekte ◦ Robuste Ein- und Ausgabewerkzeuge um Daten aus .csv, Excel und Datenbanken zu laden ◦ Speichern und Lesen im sehr schnellen HDF5 Formats
Datensätze	◦ Flexibles Umformen und Pivotisieren ◦ Intuitives Mergen und Joinen ◦ Intelligentes, kennzeichnungsbasiertes Zerlegen, Fancy indexing und Unterteilen Reguläres Indizieren: [x[3], x[7], x[2]] Fancy indexing: ind = [3, 7, 4] x[ind] ◦ Mächtige, flexible Gruppierungsfunktionalität um Datensätze nach Zerlegung und Bearbeitung der Einzelmengen wieder zusammenzusetzen - sowohl zur Aggregierung, als auch zur Transformation der Daten
Zeitreihen	◦ Erstellen von Datumsbereichen ◦ Frequenzumwandlung von Datumsbereichen ◦ Statistiken gleichbleibender Zeitrahmen ◦ lineare Regression gleichbleibender Zeitfenster ◦ Datumsverschiebung und Verzögerung
Achsen	◦ Hierarchische Achsbeschriftung (um mehrere Bezeichnungen auswählen zu können)
Spalten	◦ Größenmutation ermöglicht Einfügen und Löschen von Spalten aus DataFrames und höher dimensionierten Objekten

Um pandas auf Windows zu nutzen, ist die unter Punkt 4.3.1 beschriebene Installation der Python Distribution Anaconda durchzuführen. Deren Installation wird auch unter Linux von den Entwicklern empfohlen. Alternativ kann pandas dort auch anhand des folgenden Befehls installiert werden. Weitere Wege listet die Quelle auf [pandas, S. 9].

```
pip install pandas
```

Damit pandas funktioniert setzt es aktuell setuptools mindestens in der Version 24.2.0, NumPy ab Version 1.12.0, python-dateutil mindestens in Version 2.5.0, sowie pytz voraus. Es existieren weitere optionale Abhängigkeiten, welche die Dokumentation auflistet [pandas, S. 10].

2.5 Keras

Keras findet an dieser Stelle Erwähnung, da es zum einen ein Modul von TensorFlow darstellt [TF_PY, /keras], und zum anderen auch von KNIME eingebunden werden kann [KNIME_DEEP_L]. Des Weiteren existieren in Keras zwei Klassen, die Funktionen aus scikit-learn in diesem nutzbar machen [KERAS, /scikit-learn-api/].

Bei Keras handelt es sich um eine quelloffene, in Python verfasste, Programmierschnittstelle beziehungsweise Bibliothek [KERAS], deren ursprünglicher Autor François Chollet sie als Framework bezeichnet [DL_WITH_PY, preface]. In der Quelle beschreibt der Autor auch, dass ein hohes Maß an Zugänglichkeit schnell zu einem wichtigen Ziel bei der Entwicklung von Keras wurde. Ursprünglich hatte er das Werkzeug nur zur Erleichterung seines eigenen Arbeitsalltages geschrieben. Als er jedoch im weiteren Zeitverlauf feststellte, wie viele Nutzer, und auf welche unerwarteten Weisen diese Keras nutzten, ergab sich dieses Ziel. Für Chollet folgte aus der Feststellung, dass bereits Zehntausende Keras nutzten auch der Wunsch, mit dem Werkzeug eine Demokratisierung des Feldes der künstlichen Intelligenz zu schaffen.

Keras ist in der Lage sowohl auf TensorFlow, CNTK, als auch auf Theano aufzubauen [KERAS]. Ersteres wird unter 2.7 behandelt, CNTK ist die Abkürzung für Microsofts Cognitive Toolkit. Dabei handelt es sich um einen vereinheitlichten Werkzeugsatz für Linux und Windows, welcher neuronale Netze, unter der Nutzung gerichteter Graphen, als eine Serie von Berechnungsschritten beschreibt [CNTK]. Eine sehr grobe Zusammenfassung der Möglichkeiten die CNTK bietet, bilden die folgenden drei Punkte ab. Details finden sich in der Quelle.

- Vereinfacht die Verkettung beispielsweise von CNNs, vorwärts gerichteten DNNs und RNNs miteinander
- Implementiert Rückwärtspropagierung mit automatischer Unterscheidung
- Kann Berechnungen über mehrere GPUs und Server parallel betreiben

Theano stellt eine quelloffene Python Bibliothek dar und wurde primär von Mitarbeitern des Montrealer Instituts zur Erlernung von Algorithmen (MINA) erstellt [THEANO]. Die Werkzeuge der Bibliothek sollen eine effiziente Definition, Optimierung und Einschätzung mathematischer Ausdrücke, vor allem derer, die multidimensionale Felder beinhalten, bieten [THEANO_DL]. Aufgrund der Möglichkeit Grafikkarten zu nutzen, kann Theano die Berechnungsgeschwindigkeit die die Nutzung von CPUs liefert um viele Größenordnungen übertreffen. Theano bleibt weiterhin auf GitHub verfügbar [THEANO], allerdings ließ der MILA-Leiter Yoshua Bangio am 28. September 2017 das Ende der Entwicklung durch MILA verlautbaren. Dieser Schritt basiert auf der Feststellung, dass die Unterstützung Theanos nicht mehr der beste Weg sei, Entstehung und Anwendung neuartiger Forschungsansätze zu unterstützen [THEANO_QUITS].

Um selbst Keras nutzen zu können, setzt dieses zuerst die Installation von TensorFlow, Theano oder CNTK voraus, wobei TensorFlow empfohlen wird.

Optionale Abhängigkeiten werden - ebenso wie die zur eigentlichen Installation nötige nachfolgende Codezeile - auf der Dokumentationsseite unter https://keras.io/ aufgeführt.

```
pip3 install keras
```

2.6 KNIME

Wenn in dieser Arbeit die Abkürzung KNIME verwendet wird, bezeichnet sie die quelloffene „KNIME Analytics Platform“ [KNIME_OPEN]. Diese Plattform dient der Wissensgewinnung aus Informationen sowie der Integration der gewonnenen Modelle. Modelle bezeichnet dabei das computerlesbare Wissen. Allgemeingültig kürzt KNIME die Wortgruppe “Konstanz Information Miner” ab. Wie die Langform schon andeutet, entstand die damit verbundene quelloffene Plattform an der Konstanzer Universität (genauere Details zur Entstehung finden sich unter dem Punkt 3.1).

Dieser Abschnitt der theoretischen Grundlagen beschäftigt sich mit den für die nutzende Person ersichtlichen Grundlagen von KNIME. Die Basis jeglicher Erkenntnisgewinnung in KNIME stellen die sogenannten Nodes dar. Als Nodes bezeichnet die KNIME AG die kleinsten Programmiereinheiten in KNIME. Diese sind jeweils zur Erfüllung einer spezifischen Aufgabe gedacht. Der Umfang dieser Einzelaufgaben reicht von der trivialen Änderung einer Spaltenbezeichnung bis hin zu komplexen Klassifikationsverfahren. Die Verknüpfung der Nodes ergibt dann den eigentlichen Arbeitsablauf, welchen die KNIME AG schlichtweg mit dessen englischen Begriff als „Workflow“ bezeichnet. Die nachfolgende Abbildung zeigt ein typisches Fenster einer geöffneten Instanz von KNIME und die darin enthaltenen Bestandteile. Da diese Bestandteile Teile der grafischen Benutzeroberfläche darstellen werden sie in Punkt 4.2.1 genauer erläutert.

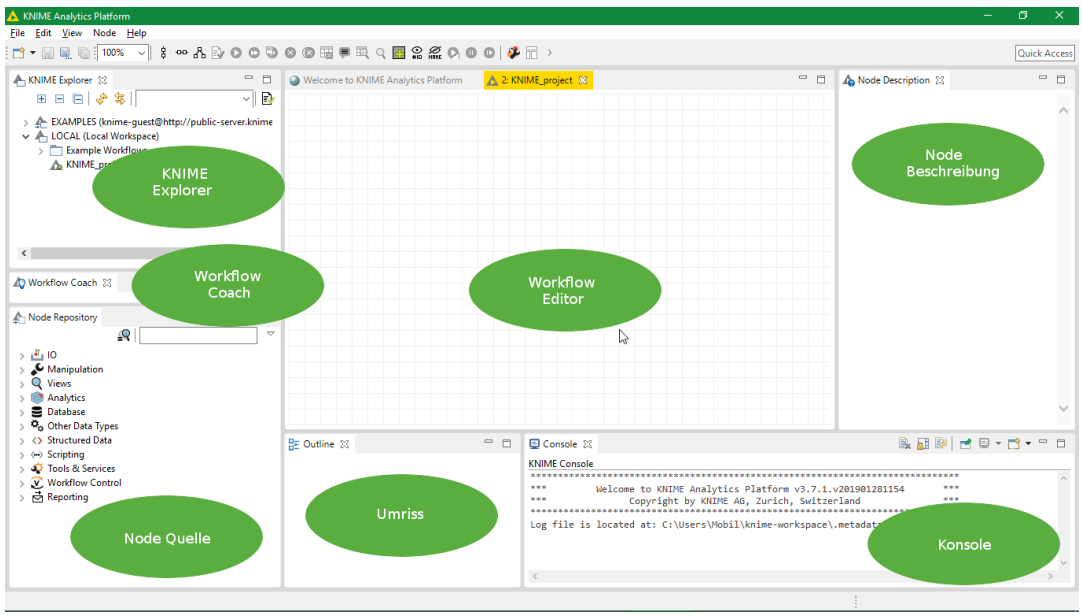


Abbildung 7: Bildschirmfoto mit übersetzten Beschriftungen (eigene Darstellung, basierend auf [KNIME_WORKBENCH])

KNIME bietet neben Lage Deep Learning, beispielsweise anhand der Einbindung von Keras und TensorFlow, auch den Umgang mit sehr großen Datenmengen, unter anderem mit Apache Spark [KNIME_PRODUCTS].

Die KNIME AG gliedert ihr Portfolio, neben der in dieser Arbeit thematisierten Plattform, in Integrationen und Erweiterungen auf. Bei letzteren differenziert das Unternehmen zwischen selbst, von der Gemeinschaft und von Partnern entwickelten. Zuletzt sei das Produkt „KNIME Server“ erwähnt [KNIME_PRODUCTS], diese Bezeichnung steht für eine umfangreichere Unternehmenslösung. Deren jährliche Gebühr startet bei 7.500 € und endet bei 45.500 €. In der Quelle findet sich neben einer mittleren Preisalternative auch eine Leistungsübersicht der einzelnen Varianten [KNIME_SERVER].

2.7 TensorFlow

„TensorFlow“ bezeichnet Googles gleichnamige quelloffene Plattform zur Wissensgewinnung und Distribution. Diese verdankt ihren Namen dem zugrundeliegenden Arbeitskonzept [TF_GUIDE, /tensors]. Das Konzept fußt auf der Definition und Berechnung von Tensoren. Tensoren stellen Größen dar, unter deren Zuhilfenahme man Skalare, Vektoren, Matrizen und weitere Größen analoger Struktur einheitlich schematisiert einordnen kann. Eine derart schematisierte Einordnung dient der Beantwortung zweier Fragen, die im Umfeld des Erkenntnisgewinns aus Daten als sehr relevant angesehen werden dürfen. Ausformuliert lauten diese: „Was ändert sich und was ändert sich nicht, wenn das Bezugssystem sich ändert?“ [TENSOR]. TensorFlow stellt Tensoren intern als n-dimensionale Felder von Basis-Datentypen dar, wobei jedes Element eines Tensors dem gleichen, bekannten Datentypen zugehörig ist [TF_GUIDE, /tensors].

TensorFlow wurde mit dem Ziel des großflächig verteilten Trainierens und Schlussfolgerns entwickelt [TF_GUIDE, /extend/architecure]. Es verbuchte seine ersten Erfolge - zu diesem Zeitpunkt der Öffentlichkeit noch unbekannt - auch in dieser Domäne. Explizit handelte es sich dabei um die Optimierung der von Google produzierten Übersetzungen [NYT_AI]. Dennoch ist TensorFlow auch flexibel genug, um in kleinerem Umfeld nutzbar zu sein. Deshalb eignet es sich beispielsweise auch für das Experimentieren mit neuen Modellen, oder das Optimieren auf System-Ebene [TF_GUIDE, /extend/architecure]. Einen Beleg für die Flexibilität von TensorFlow liefert die später folgende Abbildung 8, die dessen Programmier-Stack wiedergibt. In diesem finden sich, neben diversen integrierten Werkzeugen, auch APIs für verschiedene Programmiersprachen.
Neben Schnittstellen in verschiedenen Programmiersprachen existieren auch welche für „Layers“, „Datasets“, Metrics“ und „Estimators“. „Layers“ steht dabei für den gleichnamigen Namensraum, welcher über 19 Klassen zur Interaktion mit Schichten neuronaler Netze verfügt [TF_PY, /layers]. Des Weiteren existiert ein gleichlautendes Keras-Modul, welches Bestandteil von TensorFlow ist und zeitgleich eine API darstellt. Dieses „Layers“ bietet neunundneunzig Klassen zur Interaktion mit den oben bereits bezeichneten Schichten an. Das Spektrum dessen Klassen reicht von vorkonfektionierten Umwandlungsschichten, über die nutzerdefinierte Konfiguration einer Schicht bis hin zur Schaffung eines RNNs [TF_PY, /keras/layers].

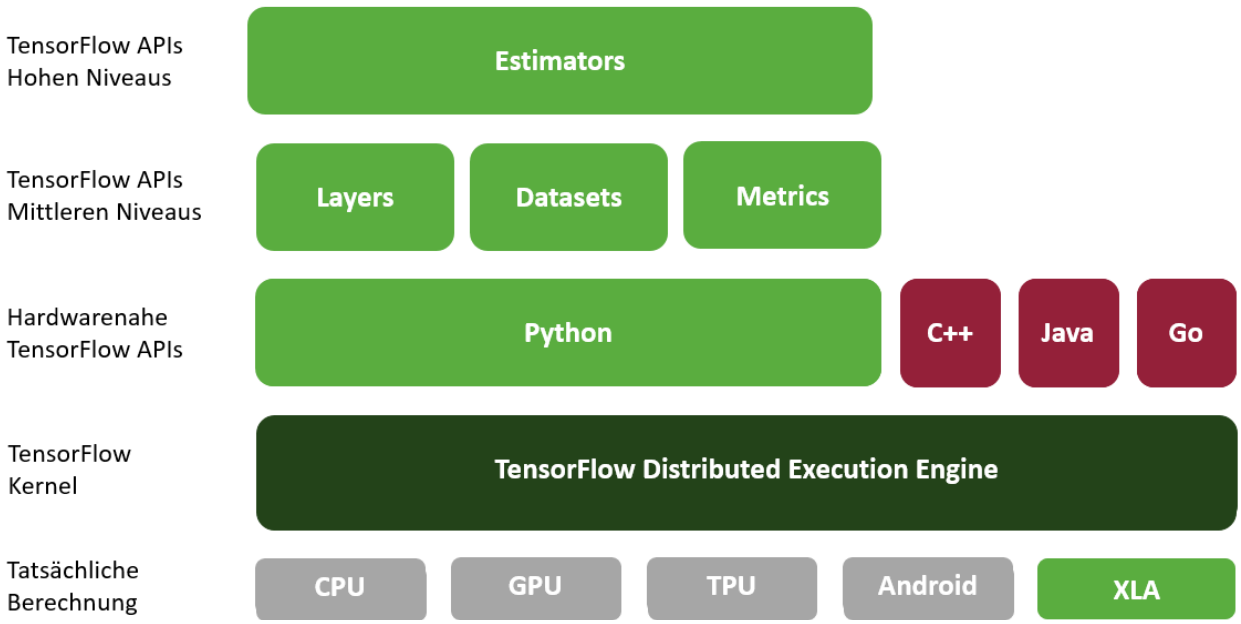


Abbildung 8: TensorFlows Programmierstack
(Eigene Abbildung, Vermengung aus TF_GUIDE, /premade_estimators] und [TF_STRUCTURE])

Der zweite Begriff „Datasets“ bezieht sich auf die in TensorFlow integrierte API welche der Bereitstellung und Nutzung von Eingabeleitungen dient [TF_PY, /data]. Diese Eingabeleitungen repräsentieren, als Bestandteil von TensorFlows „tf.data“-API, eine Reihe von Elementen. Jedes dieser Elemente enthält einen oder mehrere Tensoren [TF_GUIDE, /datasets].

Der dritte auf dieser Ebene befindliche Begriff „Metrics“ repräsentiert auswertungsbezogene Metriken. Die Dokumentation der API listet dreiunddreißig Funktionen auf, anhand derer die nutzende Person beispielsweise die Gesamtanzahl der falschen Negativen, oder die mittlere quadratische Abweichung zwischen den Bezeichnungen und Vorhersagen berechnen lassen kann [TF_PY, /metrics].

Bei den „Estimators“ handelt es sich um eine API die laut Dokumentation, die Programmierung maschinellen Lernens stark vereinfachen soll, da sie verschiedene Aktionen dessen umfasst. Dabei handelt es sich um die folgend dargestellten vier Teilschritte.

- Training
- Auswertung
- Vorhersage
- Export

Diese Estimators - eingedeutscht Schätzer - basieren wiederum auf den im Vorfeld beschriebenen „Layers“, und stehen der anwendenden Person zum einen vorgefertigt, zum anderen aber auch völlig personalisierbar zur Verfügung. Die Vorteilhaftigkeit vorgefertigter Schätzer liegt darin, dass diese die Verquickung verschiedener notwendiger Bestandteile des Trainingsprozesses und auch relevante Entscheidungen bezüglich der Hardwarenutzung übernehmen. Für erfahrene Personen ist die Erstellung individualisierter Schätzer gedacht. Diese können entweder bereits existente Schätzer oder Modelle aus Keras, aber auch komplett

eigens erstellten Code als Basis nutzen [TF_ESTIMATORS]. Für letztere eignet sich statt der vorgenannten Quelle die Seite https://www.tensorflow.org/guide/custom_estimators.

Die als „Tatsächliche Berechnung“ bezeichnete Schicht unterhalb der des Kernels wurde vom Autor aus einer anderen Quelle ergänzt, um an dieser Stelle bereits auf die TensorFlow immanente Fähigkeit der verteilten Berechnung hinzuweisen. Dies bezeugt unter anderem auch die zehnte Abbildung dieser Arbeit unter Punkt 3.2, welche den EEG Debugger zeigt, der dort auch beschrieben wird.

2.8 Scikit-learn

Scikit-learn stellt - anders als die im Titel der Arbeit bezeichneten Vergleichskandidaten - keine Plattform dar. Vielmehr handelt es sich dabei um eine quelloffene Python-Bibliothek zur computerlesbaren Gewinnung von Wissen. Das Team dahinter strebt auch in der nahen Zukunft keine Erweiterung in diese Richtung an. Diese Schlussfolgerung ergibt sich aus der Reaktion auf häufig gestellte Fragen. Darunter finden sich die nach einer möglichen GPU-Nutzung, den Gründen des Entfernens des Markov Modells und die nach dem Hinzufügen grafischer Modelle oder der Vorhersage von Sequenzen. Die Frage nach der GPU Nutzung wird im Punkt 5.3.1 geklärt, die letzten beiden identisch mit einem Verweis auf die Absicht, eine einheitliche API zur Verfügung zu stellen, beantwortet. Dabei sollen die grundsätzlichen Aufgaben des maschinellen Lernens abgedeckt werden. Dies will das Team hinter scikit-learn durch die Nutzung einer Menge datenverarbeitender Elemente, und Meta-Algorithmen gewährleisten. Zusätzlich dazu weist das Team auf die Anforderungen des strukturellen Lernens hin, welche vom Funktionsumfang, den scikit-learn bietet, abweichen. Diese Diskrepanz bestehe bezüglich der Konzepte, API, Algorithmen und Expertise, weswegen man auf zwei andere Projekte verweist. Die Empfehlung der Projekte geschieht mit der Ergänzung, dass deren APIs der eigenen ähnelten.

Der Verweis darf auch den hohen Ansprüchen zugerechnet werden, die sich das Team hinter scikit-learn selbst auferlegt hat. Es werden dort nur Algorithmen aufgenommen, deren Publikation mindestens drei Jahre zurückliegt, die mindestens 200 Mal in wissenschaftlichen Arbeiten zitiert wurden, und die über ein breites Anwendungsspektrum sowie eine hohe Nützlichkeit verfügen. Doch selbst Algorithmen, die diese Hürden erklimmen würden, werden nur dann in den Zenit der scikit-learn Algorithmen eingelassen, wenn sie gut in die gegenwärtige API passen. Diese Formulierung meint, dass neben Anforderungen bezüglich der Schnittstellen auch solche mit Bezug auf Ein- und Ausgabeformate zu erfüllen sind. Bei den Formaten sind Felder aus NumPy oder dünnbesetzte Matrizen aus SciPy zu unterstützen [SCIKIT_FAQs]. Die letzte Information aus den häufig gestellten Fragen offenbart auch eine wesentliche Eigenschaft von scikit-learn, nämlich dessen Aufbau auf NumPy und SciPy [SK_INSTALL], weswegen diese bereits vorhergehend thematisiert wurden. Die nachfolgende Grafik bereitet unter anderen den Zusammenhang zwischen scikit-learn, NumPy und SciPy auf. Auch weitere Werkzeuge werden dort, ohne einen Anspruch auf Vollständigkeit erheben zu wollen, eingegliedert. Dabei bauen weiter oben befindliche Instrumente auf darunter platzierten auf.

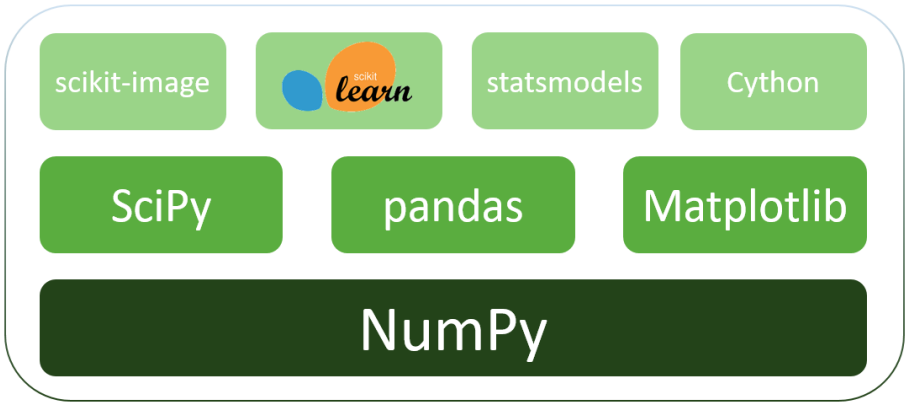


Abbildung 9: Einordnung von scikit-learn im Bezug zu weiteren Werkzeugen (eigene Darstellung, basierend auf [SK_SP_PD])

3. Historie und allgemeine Details

Die nachfolgenden Abschnitte geben jeweils den Entstehungshintergrund, die beteiligten Unternehmen, die aktuelle Version und die Information, unter welcher Lizenz die Lösungen angeboten werden, wieder.

3.1 KNIME

Anfang 2004 begann eine Gruppe von Entwicklern eines aus dem Silicon Valley stammenden und auf pharmazeutische Anwendungen spezialisierten Unternehmens mit der Arbeit an einer quelloffenen Plattform. Diese war als Instrument zur Forschung und Zusammenarbeit gedacht. Da den Entwicklern bewusst war, dass ihr Produkt eine sehr große Menge unterschiedlicher Daten bearbeiten und integrieren müsse, erlegten sie sich strenge Entwicklungsstandards auf. Dies diente der Produktion einer robusten, modularen und hochgradig skalierbaren Plattform, welche verschiedene Modelle des Ladens, Transformierens, Analysierens und visuellen Erforschens unterschiedlicher Daten beinhalten sollte. Schon kurz nach der Veröffentlichung der ersten Version der Plattform im Jahr 2006 begannen Softwarelieferanten mit der Entwicklung auf KNIME basierender Werkzeuge.

Zu Beginn der Erstellung dieser Diplomarbeit lag KNIME in der Version 3.7.1 vor, während der Erstellung wurde Version 3.7.2 veröffentlicht. Die Software wird in über fünfzig Ländern in diversen Einrichtungen genutzt. Diese setzen KNIME in einem breiten Spektrum von Anwendungsgebieten ein. Das Spektrum reicht von Finanzdienstleistungen über Einzelhandel, Verlagswesen, Beratung und Regierung bis hin zur Forschung.

KNIME wurde in Java verfasst und basiert auf Eclipse. Letzteres wurde entschieden, um einen Kompromiss zwischen einer integrierten Entwicklungsumgebung (IDE) und einem ausbaufähigen Erweiterungssystem zu ermöglichen. Da die KNIME Dachgesellschaft, die KNIME AG, laut eigener Aussage stark an Quelloffenheit und die Macht der Gemeinschaft

glaubt, stellt sie KNIME vollumfänglich, also ohne jedwede technische Einschränkung, zur Verfügung. Darin ist auch immer Software von Eclipse enthalten.

KNIME wird unter einer Open Source GPLv3 Lizenz veröffentlicht. Zusätzlich erlaubt ist die Nutzung der exakt definierten Programmierschnittstelle, der sogenannten „Nodes“, falls dies dem Hinzufügen zusätzlicher proprietärer Erweiterungen dient [KNIME_HP].

3.2 TensorFlow

TensorFlow entstand laut einem New York Times Artikel [NYT_AI] dank Googles „20% time“ Regel. Diese besagt, dass Mitarbeiter zwanzig Prozent ihrer Arbeitszeit in Projekte, die sie für besonders nützlich für Google erachten, investieren mögen [IPO_2014]. Jeff Dean, der bei Erstellung dieser Arbeit Googles KI Abteilung leitet, investierte ab dem Frühjahr 2011 seine zwanzig Prozent in die Unterstützung von Andrew Ng. Letzterer war zu diesem Zeitpunkt als Berater für Google im Bereich künstlicher neuronaler Netze im Projekt Marvin tätig. Das Duo wurde bald durch Gregg Corrado ergänzt, da dieser über einen neurowissenschaftlichen Hintergrund verfügt. Zum Quartett wurde das Team durch das Hinzukommen von Quoc Le, einem der besten Absolventen Ngs. Zwischenzeitlich wurde das Team von anderen Google Ingenieuren intern schon als „Google Brain“ bezeichnet – dem heutigen Teamnamen. Dessen Teammitglieder haben zum Verfassungszeitpunkt dieser Arbeit 805 Veröffentlichungen mit KI-Bezug getätigt [GB_PUB_DB].

Am 09.11.2015 wurde nach eifriger, erfolgreicher interner Nutzung eine funktionell eingeschränkte Variante veröffentlicht. Welche Werkzeuge, außer des im TensorFlow Whitepaper [TF_WHITEPAPER] beschriebenen EEG Debuggers, in der veröffentlichten Version fehlen konnte der Autor nicht verifizieren. Jeff Dean bestätigte lediglich das Fehlen des besagten Debuggers im zweiten Reddit AMA [RED_AMA_J] des Google Brain Teams im Jahr 2017. Das nachfolgende Bild zeigt eine EEG-Visualisierung paralleler CPU Berechnungen, die X-Achse stellt die Zeit in µs dar. Darauf werden, statt wie bei dem namensgebenden Vorbild aus der Hirnforschung, Aktivitäten der einzelnen CPUs grafisch wiedergegeben.

Zu Beginn der Arbeit stellt die Version 1.13 die aktuellste dar, welche noch als Freigabekandidat bezeichnet wurde. Im weiteren Verlauf schritt deren Entwicklung so weit fort, dass sie als stabil bezeichnet wird [TF_VERSIONS]. Version 2.0 wird momentan immer noch als Vorschauversion bezeichnet angeboten. TensorFlow wurde in C++ verfasst und unter einer Apache 2.0 Lizenz veröffentlicht. Ein unter dieser Lizenz veröffentlichtes Softwareprodukt unterliegt dadurch der eingeschränkten Nutzung seiner Marke. Ausgeschlossen werden durch die Veröffentlichenden auch Garantie und Haftung.

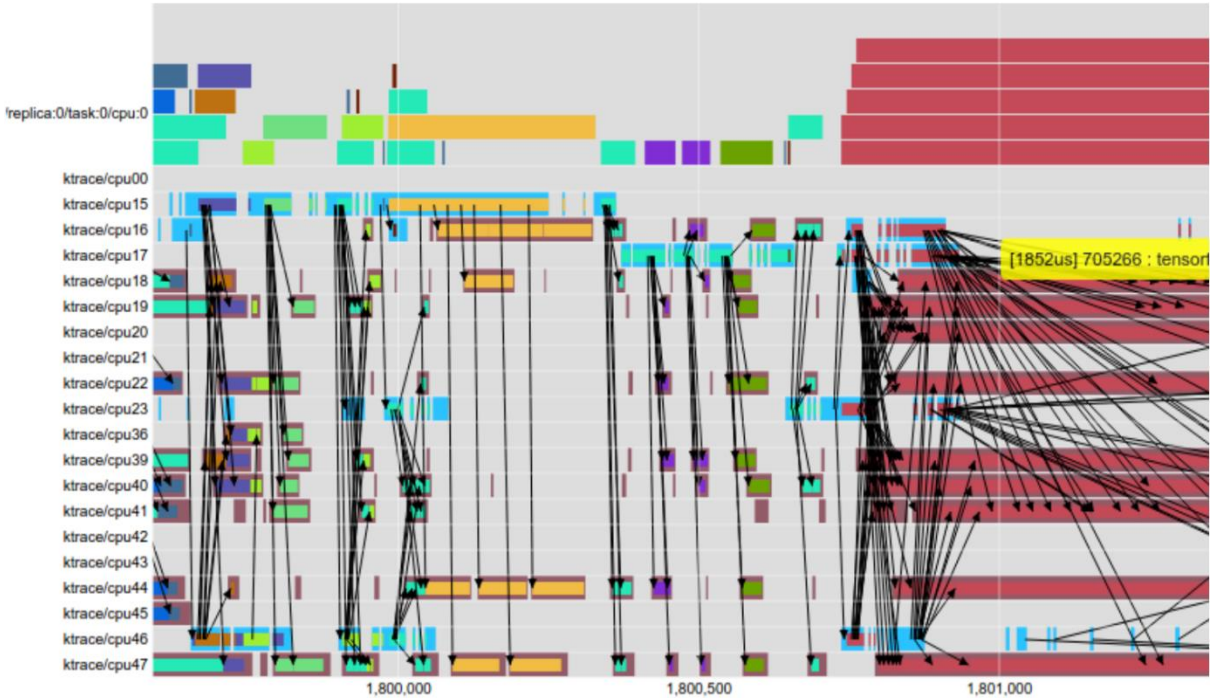


Abbildung 10: Bildschirmfoto des fehlenden EEG Debuggers in TensorFlow [TF_WHITEPAPER, S. 15]

3.3 Scikit–learn

David Cournapeau begann mit dem damals noch „scikits.learn“ benannten Projekt 2007 im Rahmen von Googles jährlich stattfindendem Programmierstipendium „Summer of Code“. Im selben Jahr begann Mathieu Brucher seine Arbeit am Projekt im Zuge seiner Abschlussarbeit.

2010 übernahmen Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort und Vincent Michel die Führung des Projekts. Dieses Quartett war für das INRIA tätig, welches sich sowohl der Grundlagen-, als auch der angewandten Forschung widmet. Es forscht in den Bereichen der Informations- und Kommunikationswissenschaften und deren Techniken. Als staatliche Einrichtung ist es französischen Ministerien, dem Forschungsministerium und dem Wirtschaftsministerium, unterstellt. Unter der neuen Leitung fand am ersten Februar des Übernahmejahres die erste frei verfügbare Veröffentlichung statt.

Seitdem finden in ungefähr dreimonatigen Zyklen Veröffentlichungen neuer Versionen statt. Während der Erstellung dieser Diplomarbeit stellt die Version 0.20.3 die aktuellste Version dar. Zu besagtem Zeitpunkt listet die Projektseite fünfzig beteiligte Autoren namentlich auf. Zusätzlich zu der Auflistung der Autoren findet sich dort auch eine Aufstellung über die finanziellen Unterstützer des Projekts. Darunter befinden sich neben dem INRIA auch Hochschulen aus Paris, New York und Sydney, eine wissenschaftlich orientierte Stiftung sowie Google.

Scikit-learn ist zur Erreichung einer besseren Lesbarkeit des Quellcodes größtenteils in Python verfasst.

Begründet wird diese Entscheidung mit einem besseren Verständnis der nutzenden Personen für das Verhalten der Algorithmen und einer einfacheren Wartbarkeit für die Entwickler. Des Weiteren ist Code in Cython direkt, aber auch als Hülle für C und C++ Algorithmen, nutzbar. Dessen Einsatz findet primär zur Geschwindigkeitssteigerung statt.

Scikit-learn wird unter BSD-Lizenz veröffentlicht und kann somit frei verwendet werden. Die einzige Auflage ist, dass der Hinweis auf beinhaltete Software von scikit-learn nicht entfernt werden darf [SCIKIT_ABOUT].

4. Bedienbarkeit

Bezüglich der Bedienbarkeit werden die jeweils notwendigen Schritte erläutert, derer es bedarf, um die Lösungen zu installieren. Aufgrund der Tatsache, dass nicht alle unter Raspbian nutzbar sind, findet auch die Installation unter Ubuntu 18.04.2 LTS auf dem ursprünglich mit Windows betriebenen System Erwähnung.

4.1 Installation

4.1.1 KNIME

Die KNIME AG stellt auf Ihrer Webpräsenz verschiedene Versionen von Installationsroutinen der Software zur Verfügung. Es werden keine Abhängigkeiten oder Voraussetzungen an den zu nutzenden Rechner benannt. Lediglich für Nutzer von Apple-Geräten besteht der Hinweis, dass Mac OSX 10.11 erforderlich ist.

4.1.1.1 Windows

Unter Windows gestaltet sich die Installation von KNIME als sehr anwenderfreundlich. Nach der Eingabe einer beliebigen E-Mail-Adresse und Zustimmung zu den Datenschutzregelungen sowie den Lizenzbedingungen erhält man Zugriff auf die Installationsdatei. Diese findet man auf der Internetpräsenz der KNIME AG durch Aufruf von https://www.knime.com/downloads .

Dort stellt die KNIME AG für Windows drei verschiedene Optionen zur Verfügung, welche jeweils sowohl in 32, als auch in 64 Bit, angeboten werden. Zur Auswahl stehen die vorgenannte Installationsdatei, ein selbstextrahierendes Archiv sowie ein gezipptes Archiv. Der Einfachheit halber hat sich der Autor für die erste Option entschieden. Diese ist, nach nochmaliger Zustimmung zu den Datenschutzregelungen und den Lizenzbedingungen herunterladbar. Die zeitgleich angezeigte Information, dass eventuell Schwierigkeiten aufgrund des Microsoft Sicherheitsmechanismus „SmartScreen-Filter“ entstehen könnten, trifft im beschriebenen Anwendungsfall nicht zu.

Im Zuge der Installation finden verschiedene Abfragen statt. Diese beginnen mit der Zustimmung der Lizenzbedingungen, gefolgt von der Wahl des Installationsordners für die zu installierende Datenmenge von 822,6 MB bei Version 3.7.1.

In der Version 3.7.2 hat sich der Speicherplatzverbrauch um 0,1 MB gesteigert. Als nächstes schließt sich die Abfrage bezüglich eines Startmenüeintrages an. Darauffolgend kann der Desktopverknüpfung und der Kopplung mit Dateiendungen des Typs „knwf“ sowie „knar“ zugestimmt werden. Dabei steht knwf für Dateien, die Arbeitsabläufe in KNIME beschreiben, die so genannten „Workflows“. Die Dateiendung knar bezeichnet Archivdateien von KNIME. Im gleichen Dialogfenster kann ebenfalls der Verknüpfung von URLs beginnend mit „knime://“ zugestimmt werden. Die auf diesen Absatz folgende Grafik spiegelt die getroffene Auswahl wider.

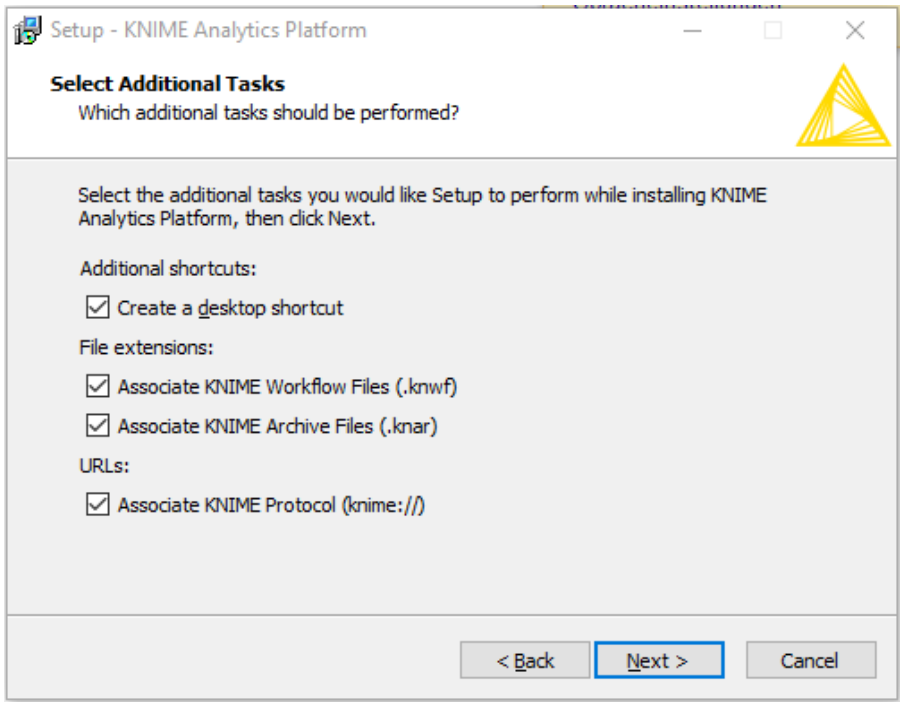


Abbildung 11: Auswahlmöglichkeiten im Zuge der Installation von KNIME (Bildschirmfoto selbst erstellt)

Im Anschluss an die abgebildete Abfrage schlägt KNIME die maximal zur eigenen Ausführung zur Verfügung stehende Speichermenge im RAM in Megabyte vor. Auf dem eingangs beschriebenen Testsystem liegt diese Empfehlung bei 12284 MB. Im Anschluss an die Bestätigung der bisher getätigten Eingaben findet die eigentliche Installation statt. Als letzter Schritt kann KNIME dann direkt aus der Installationsroutine heraus gestartet werden. Der gesamte Installationsprozess nimmt in der vorhandenen Computerkonfiguration eine Minute und neun Sekunden in Anspruch bei Version 3.7.1, in der Version 3.7.2 steigert sich der Zeitbedarf um knappe vier Sekunden.

4.1.1.2 Raspberry und Ubuntu

Auf dem, ursprünglich mit Raspbian betriebenen, Raspberry Pi 3 Model B+ gestaltet sich die Installation weniger einfach. Die Nutzung des von der KNIME AG zur Verfügung gestellten gzip Archivs führt nicht zum Erfolg. Auch ein mehrfach wiederholtes Herunterladen, und

Neuinstallieren des Betriebssystems ändert daran nichts. Den Grund des Scheiterns auf dem Raspberry stellt das offiziell empfohlene Betriebssystem des Minicomputers Raspbian dar.

Dieses wurde ursprünglich für eine 32-Bit-Architektur entwickelt, deshalb wird der 64-Bit-Prozessor standardmäßig im sogenannten AArch32 Modus betrieben.

Dieser Umstand führt dazu, dass sich die verbaute vierkernige 64-Bit-CPU vom Typ BCM2837B0 bei der Eingabe des Befehls

```
cat /proc/cpuinfo
```

in ein LXTerminal, als sein fünf Jahre älterer Vorgänger BCM2835 ausgibt. Zusätzlich wird falsch ausgegeben, dass der Prozessor über eine ARMv7 Architektur verfügt. Richtig wäre beim verbauten Prozessor eine ARMv8 Architektur, der einkernige Urvater wurde in der ARMv6 Architektur gebaut.

Eine Installation von KNIME auf dem Raspberry mit der experimentellen 64-Bit-Version von Ubuntu-Mate 18.04.2 [UBUNTU_MATE] scheitert nicht an der geringen Geschwindigkeit der Distribution, sondern liefert eine indirekte Fehlermeldung. Diese besteht in der Erstellung einer Datei, welche auf eine unpassende Codierung schließen lässt. Den Namen dieser Datei, und den Inhalt des Ordners in dem sie erstellt wird, gibt die folgende Abbildung wieder.

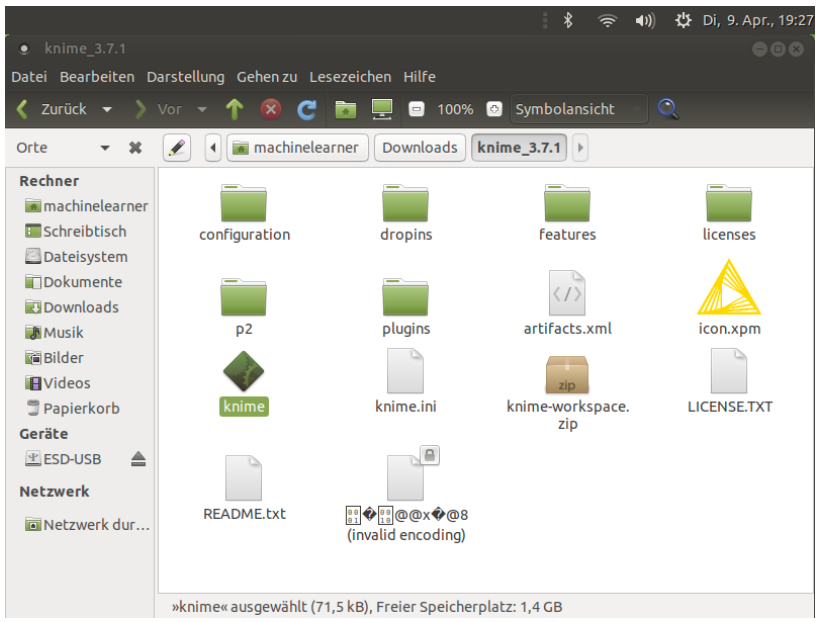


Abbildung 12: Generierte Datei, welche auf unpassendes Codieren schließen lässt (Bildschirmfoto selbst erstellt)

In einer Testinstallation auf einem Ubuntu 18.04.2 LTS im eingangs beschriebenen Desktop-Rechner lässt sich mit dem Archiv KNIME in unter einer Minute problemlos installieren.

Als letzter zeitfreundlicher Weg eine einigermaßen aktuelle Version von KNIME auf dem Kleinstrechner nutzbar zu machen galt, laut KNIME-Forum, die Installation von Eclipse [KNIME_RASPBERRY]. Dieses befindet sich nach den folgenden beiden Eingaben in ein LXTerminal auf dem System [ECLIPSE_PI].

```
sudo apt-get update
sudo apt-get install eclipse
```

Die Abarbeitung des ersten Befehls ist nach zwei Minuten vollendet. Der zweite beansprucht, inklusive der Bestätigung des Verbrauchs von 394 MB Plattenplatz und dem Herunterladen aller Teilkomponenten, sieben Minuten bis sich Eclipse auf dem System befindet.

Das Befolgen der auf der Homepage beschriebenen Installation von KNIME [KNIME_INSTALL] führte jedoch auch unter Zuhilfenahme des Forums und darin gegebener Tipps [KNIME_RASPBERRY] zu keiner zeitkonformen Lösung.

Der Autor erspart den Lesern an dieser Stelle Details zu weiter unternommenen Anläufen wie dem direkten Download und Installationsversuch diverser 32-Bit-Versionen, der vielfältig versuchten Nutzung der Installationsfunktion innerhalb von Eclipse, und der Nutzung einer angeblichen 64-Bit-Version von Raspbian. Im Zuge der Sekundärforschung ergab sich ein weiterer, zu zeitaufwändiger Weg, welcher in der Erstellung eines passenden Kernels liegt [COMPILE_KERNEL].

4.1.2 TensorFlow

TensorFlow stellt verschiedene Möglichkeiten der Installation beziehungsweise Nutzung bereit. Es werden prinzipiell zwei unterschiedliche Versionen angeboten, eine die nur die CPU nutzt, sowie eine die auch die GPU nutzen kann. Diese Varianten liegen in zwei Qualitätsstufen vor. Als zuverlässige, bewährte Version und als eventuell instabile. Auch die Nutzung eines vorhandenen Docker-Images ist möglich. Unter Raspbian im Speziellen, und Linux im Allgemeinen, reicht die Nutzung einer Terminal-Eingabe zur Installation.

4.1.2.1 Windows

Es existieren unterschiedliche Wege um TensorFlow unter Windows zu nutzen. Einen stellt die unter 4.3.1 beschriebene Nutzung von Anaconda dar. Ein anderer basiert auf der Verwendung der Docker-Container. Die beiden folgenden Unterunterabschnitte stellen diese dar.

4.1.2.1.1 Anaconda Nutzung unter Windows

Da Anaconda standardmäßig scikit-learn beinhaltet, ist dessen Installation im Punkt 4.3.1 detailliert dargestellt. Nach Durchlaufen der dort beschriebenen Schritte ist zuerst der sogenannte „Anaconda Navigator“ zu öffnen. Dieser ist leicht durch Druck der Windows Taste und anschließende Eingabe des Namens auffindbar. Es empfiehlt sich eine neue Umgebung zu erstellen. Dazu ist im linken Teil des Fensters auf „Environments“, und in der sich nun darstellenden Ansicht auf die Schaltfläche „Create“ zu klicken. Die im daraufhin erscheinenden Dialog getätigte Auswahl zeigt die nächste Abbildung.

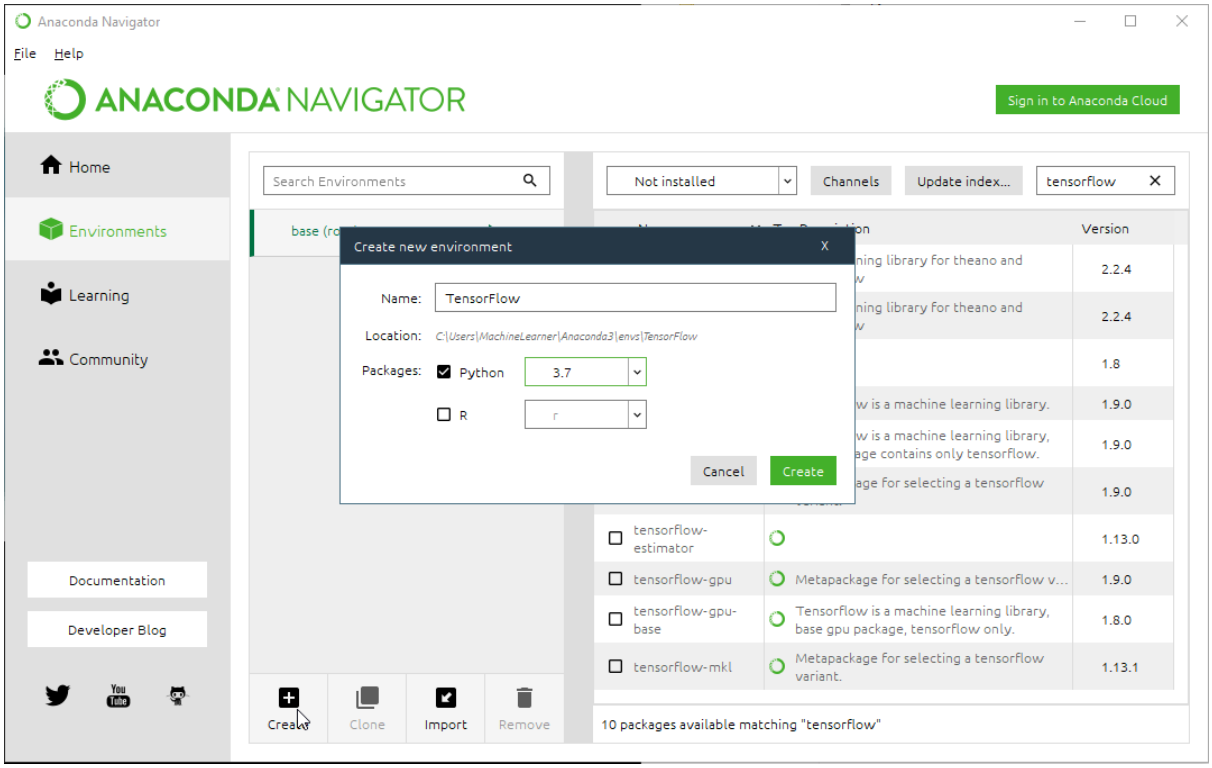


Abbildung 13: Dialog zur Erstellung einer neuen Umgebung in Anaconda (Bildschirmfoto selbst erstellt)

Dem Klick auf „Create“ folgen weitere vierzig Sekunden und die neue Umgebung wurde fertiggestellt. Um in diese TensorFlow zu installieren ist diese zuerst durch Klick darauf auszuwählen. Anschließend bedarf es der im Hintergrund der vorhergehenden Abbildung ersichtlichen Auswahl von „Not installed“ und der Eingabe des Suchbegriffes „tensorflow“. Im Testsystem werden „tensorflow-estimator“ und „tensorflow“ per Hakensetzung gewählt, auf GPU unterstützenden Systemen wird der zweite Haken bei „tensorflow-gpu“ gesetzt. Inklusive der Bestätigung der Installation dauert der Vorgang circa zwei Minuten und sechsunddreißig Sekunden.

Wird nach Klick auf die Abspielschaltfläche die Option „Open with Python“ gewählt, gibt TensorFlow nach Eingabe der folgenden Codezeilen samt Druck der Taste Enter seine Versionsnummer aus.

```
import tensorflow as tf;print(tf.__version__)
```

Ein Nutzen von Jupyter Notebook anhand der alternativen Option “Open Terminal” und dortiger Eingabe von „jupyter notebook“ wird nicht empfohlen. Dieses Vorgehen bringt TensorFlow dazu, die angebliche Unfähigkeit des Prozessors AVX-Befehle auszuführen zu bemängeln. Wird Jupyter Notebook auf dieselbe Art wie TensorFlow in der Umgebung installiert klappt die Nutzung problemlos.

4.1.2.1.2 Docker Nutzung unter Windows

Folgend wird der Einsatz des stabilen Docker-Containers ohne GPU-Unterstützung dargestellt. Der Container ohne GPU-Unterstützung wird genutzt, da die in der Konfiguration beschriebene GPU nicht unterstützt wird [TF_GPU]. Zur Nutzung des Containers ist auf dem System zuerst Docker zu installieren. Die Software findet sich nach Erstellen eines Nutzerkontos anhand des Aufrufs von https://hub.docker.com/signup, und anschließender Verifikation der E-Mail unter https://download.docker.com/win/stable/Docker%20for%20Windows%20Installer.exe . Der Installationsprozess der Container-Software dauert, inklusive des Downloads und der notwendigen Ab- und Anmeldung, eine Minute und zwölf Sekunden. Im Zuge dessen finden zwei Interaktionen mit dem Benutzer statt. Die erste klärt die Präferenz bezüglich einer Desktopverknüpfung und der Nutzung von Windows-Containern anstelle von Linux-Containern. Hier ist kein Haken bei der zweiten Option zu setzen, da alle TensorFlow-Docker-Container auf Linux basieren [TF_WINDOCK]. Nach dem ersten Start von Docker wurden noch „Hyper-V“ und „Container“ Funktionen abgefragt, deren Installation einen zweifachen Neustart bedingt. Anschließend kann, im mit Hochfahren des Rechners gestarteten Programm, die Übermittlung von Nutzungsstatistiken deaktiviert werden. Der Abruf des TensorFlow-Containers findet mit der Eingabe des folgenden Codes in ein Eingabeaufforderungsfenster statt. Dieses bedarf keiner Administratorenrechte.

```
docker pull tensorflow/tensorflow
```

Der Abruf dauert im genutzten System eine Minute und sorgt für das Vorhandensein der aktuellsten stabilen Version. Um TensorFlow bequem nutzen zu können, empfiehlt sich der Aufruf anhand eines Jupyter Notebooks. Dazu ist die folgende Zeichenfolge in das bereits geöffnete Eingabeaufforderungsfenster einzugeben [TF_DOCKER].

```
docker run –it –p 8888:8888 tensorflow/tensorflow:latest-py3-
jupyter
```

Als Reaktion auf die Eingabe stellt Docker fest, dass noch siebzehn weitere Komponenten nachzuladen und zu installieren sind. Im Anschluss an diesen Vorgang ist TensorFlow über den mit dem genannten Befehl gestarteten Jupyter-Notebook-Server nutzbar. Die Nutzung erfordert die Eingabe der Adresse http://127.0.0.1:8888 in einen Webbrowser.

In das daraufhin erscheinende Eingabefenster ist noch das Token aus dem Kommandozeilenfenster zu kopieren, um TensorFlow mittels Jupyter nutzen zu können. Die folgende Abbildung zeigt die beiden betroffenen Fenster - in der linken Hälfte das besagte Kommandozeilenfenster - in welchem das Token anhand weißer Hinterlegung markiert ist. Die rechte Bildhälfte zeigt die Wiedergabe der aufgerufenen URL im Browser Firefox und innerhalb dieser im oberen Bildbereich die grün umrandete Eingabefläche.

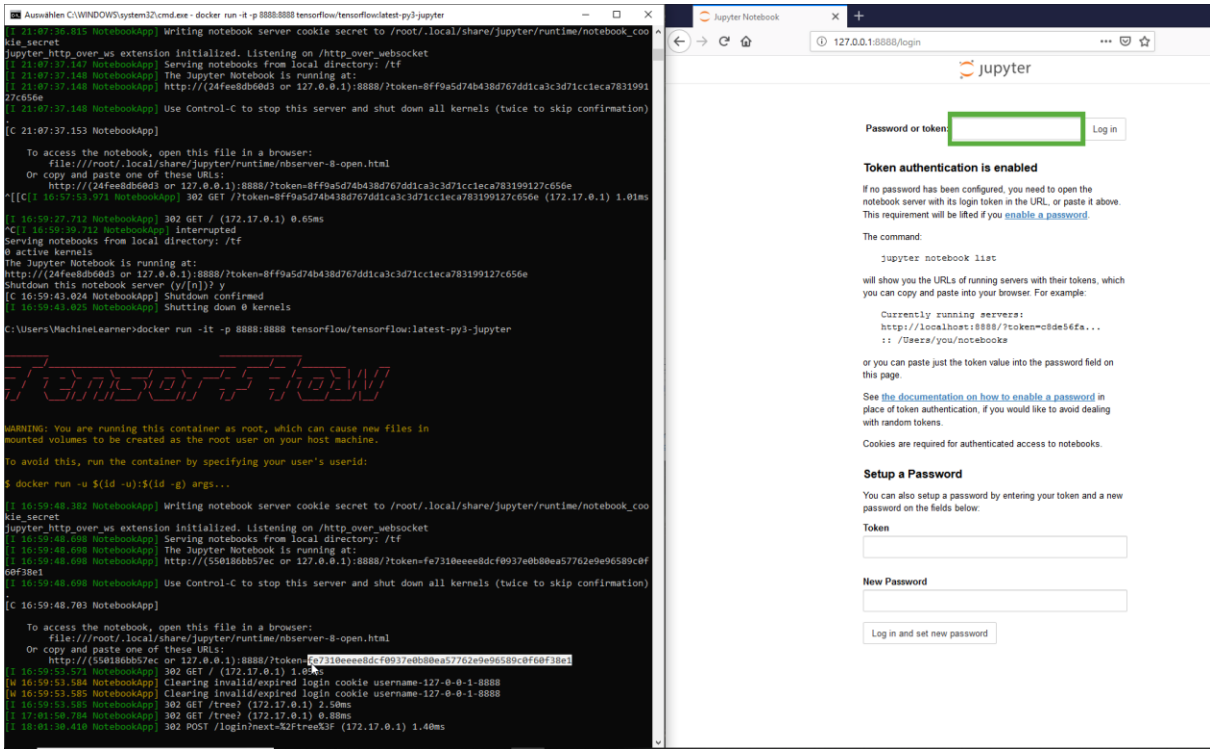


Abbildung 14: Jupyter basierte Docker Instanz von TensorFlow (selbst angefertigtes Bildschirmfoto)

4.1.2.2 Raspberry und Ubuntu

Zum Erstellungszeitpunkt der Arbeit hat Raspbian selbst in der Version „Stretch Lite“ Python 3 bereits in der Version 3.5.3 vorinstalliert. Das zur einfachen Installation notwendige Paketverwaltungsprogramm pip ist jedoch erst ab der Version „Stretch“ verfügbar, weswegen diese genutzt wird. Basis dieser Erkenntnis ist die Ausgabe des jeweiligen Systems, welche auf die Eingabe der nachfolgenden Codezeile in ein LXTerminal folgt.

```
python3 -m pip --version
```

In der „Stretch“ Version von Raspbian gibt diese die Versionsnummer 9.0.1 aus, während die „Stretch Lite“ Variante lediglich das Nichtvorhandensein des Moduls meldet.

Die eigentliche Installation von TensorFlow wird durch die Eingabe des folgenden Befehles im LXTerminal angestoßen.

```
pip3 install tensorflow
```

Der gesamte Installationsprozess zieht sich über zwölf Minuten und fünfzig Sekunden, dabei benötigt das Aufsetzen des inkludierten NumPy die meiste Zeit. Neben NumPy werden noch diverse andere Pakete installiert. Eine komplette Aufstellung dieser findet sich für interessierte Leser in der Anlage B1. Auf einem performanteren Rechner mit unterstützter GPU unter Linux empfiehlt sich die Nutzung des Docker Images zur Installation. Laut Google stellt dies den einfachsten Weg der GPU Nutzung unter Linux dar [TF_DOCKER]. Unter Ubuntu 18.04.2 LTS ist dazu der Start von „Ubuntu-Software“ erforderlich, wo sich nach Eingabe des Suchbegriffs „Docker“ eben dieses findet.

Nach einem Klick auf die Schaltfläche „Installieren“ und der Eingabe des Benutzerpassworts dauert es 20 Sekunden bis Docker installiert ist. Um das aktuellste TensorFlow Containerabbild auf das System zu ziehen, ist die folgende Terminaleingabe nötig.

```
sudo docker pull tensorflow/tensorflow
```

Bei Nutzungsabsicht einer GPU ist der erste Parameter („-gpu“) der nächsten Terminaleingabe erforderlich - ohne diese kann er weggelassen werden.

```
sudo docker run –it –p 8888:8888 tensorflow/tensorflow:latest-gpu-
py3-jupyter
```

Nach drei Minuten dreiundvierzig Sekunden ist das Jupyter Notebook mit GPU Nutzung unter http://127.0.0.1:8888 erreichbar. Ohne GPU Nutzung steht es bereits nach einer Minute dreiundzwanzig Sekunden zur Verfügung. Wie unter der Installation unter Windows ist zur Nutzung des Notebooks noch die Eingabe des Tokens nötig.

Falls Docker und GPU Nutzung nicht erwünscht sind, reichen unter Ubuntu 18.04.2 LTS zwei Terminaleingaben. Die erste holt pip, welches nicht installiert ist, in das System.

```
sudo apt install python3-pip
```

Abzüglich der Zeit für die Passworteingabe und die der Bestätigung der Installation, benötigt dieser Vorgang circa vierzig Sekunden. Die zweite Terminaleingabe installiert TensorFlow.

```
pip3 install tensorflow
```

Nach der Eingabe und weiteren siebenundvierzig Sekunden befindet sich TensorFlow auf dem System. Ist optional noch Jupyter Notebook erwünscht, so findet sich dieses nach der Eingabe von

```
pip3 install jupyter
```

auf dem System wieder [JUPYTER_INSTALL].

4.1.3 Scikit-learn

Scikit-learn setzt Python voraus. Zum Zeitpunkt der Erstellung dieser Arbeit wird entweder eine Version größer gleich 2.7, oder eine größer gleich 3.4 erwartet. Ab Version 0.21 von scikit-learn wird jedoch mindestens Python 3.5 vorausgesetzt.

Zusätzlich zu Python benötigt scikit-learn zwei Python-Bibliotheken. Die erste geforderte Bibliothek ist NumPy. Hiervon benötigt scikit-learn zur Ausführung momentan mindestens Version 1.8.2. Die zweite geforderte Bibliothek ist SciPy, deren Version größer gleich 0.13.3 sein muss [SK_INSTALL].

4.1.3.1 Windows

Selbst Windows 10 Pro verfügt standardmäßig über keine der erforderlichen Bedingungen. Eine mögliche Lösung dieses Mangels würde die Nutzung eines optionalen Linux Subsystems darstellen. Da die Installation jedoch Windows basiert sein soll, entscheidet sich der Autor für die Nutzung einer Python-Distribution. Da Anaconda über eine umfangreiche GUI verfügt findet dieses - und nicht Canopy - Verwendung [ANA_VS_CANOPY].

Anaconda wird von der amerikanischen Anaconda Incorporated in verschiedenen Versionen online zur Verfügung gestellt. Der Downloadbereich findet sich aktuell unter https://www.anaconda.com/distribution/#download-section .

Da im Zuge dieser Arbeit ein Update von Anaconda stattfand, beschreibt der folgende Ablauf den Installationsprozess der 64-Bit-Variante von Anaconda mit zweierlei Versionen. Der Ablauf ist sowohl für die Anaconda3-Version mit der Python Version 3.6, als auch für die mit 3.7 gültig. Die erste installierte Version ist die „2018.12“, bei der anderen handelt es sich um die „2019.03“. Der nachfolgende Text schildert jeweils dabei getätigte Auswahloptionen und die Unterschiede, welche sich im Zuge der Installation ergaben.

Die Installationsroutine beginnt in beiden Fällen mit der Empfehlung, alle anderen offenen Programme zu schließen. Darauffolgend ist der Lizenzvereinbarung zuzustimmen und festzulegen, für welche Nutzer Anaconda installiert werden soll. Hier wurde der Empfehlung, nur für den angemeldeten Nutzer zu installieren, Folge geleistet. Ebenfalls übernommen wurde der Installationspfad, welcher im Unterordner „Anaconda3“ im Quellpfad des angemeldeten Benutzers empfohlen wird.

Die nächste Dialogbox ermöglicht das Hinterlegen von Anaconda in der Umgebungsvariable des Systems, sowie die Registrierung als Standard für Python auf dem System. Hier wurde nur die Registrierung als Standard gewählt.

Nach sechs Minuten und dreiunddreißig Sekunden ist die eigentliche Installation der Version 2018.12 beendet und der Download von Visual Studio Code wird empfohlen. Wird dort dieser Empfehlung gefolgt, sind nach weiteren dreiundvierzig Sekunden der Download und die Installation abgeschlossen. In der Version 2019.03 benötigt die Installation neun Minuten und fünfunddreißig Sekunden und dauert somit über zwei Minuten länger als die Installation inklusive Visual Studio Code. In der Version 2019.03 kooperiert Anaconda Incorporated mit JetBrains und hat nun statt des Visual Studio Code Downloads den Link zur Freemium IDE PyCharm eingefügt. Wird die Version 2019.03 in einem System mit installiertem Visual Studio Code installiert wird, eine Verknüpfung zu letzterem automatisch im Startbildschirm des Anaconda Navigator eingefügt.

Die Installation von Anaconda beinhaltet außer Python Jupyter in der Version 1.0.0, NumPy in der Version 1.16.2, die Version 1.2.1 von SciPy sowie scikit-learn in Version 0.20.3. Eine vollständige Übersicht der 281 beinhalteten Pakete ist in der Anlage B2 ersichtlich. Diese beinhaltet auch eine indirekte Gegenüberstellung der Versionen 2019.03 und 2018.12.

4.1.3.2 Raspberry und Ubuntu

Das Vorhandensein von Python in Raspbian wurde bereits im Punkt 4.2.2 bei der Installation von TensorFlow erläutert. Auch NumPy ist schon in Raspbian stretch verfügbar, im genutzten Abbild in der Version 1.16.2. Diese Information gibt das System nach der folgenden Eingabe in ein LXTerminal preis.

```
python3 -c "import numpy;print(numpy.version.version)"
```

Somit ist neben scikit-learn noch SciPy zu installieren. Dies wird durch die Eingabe des folgenden Befehls in ein LXTerminal gestartet:

```
pip3 install –U scikit-learn scipy .
```

Nach Bestätigen des Befehls beginnt pip die benötigten Inhalte herunterzuladen und zu installieren. Dafür werden während der Tests minimal zwei Minuten und sechs Sekunden, maximal elf Sekunden mehr, beansprucht. Ein anschließendes Aktualisieren, welches zusätzliche Pakete auf das System bringt, wird durch nachfolgende Eingabe in das LXTerminal initiiert. Unter diesen Paketen befindet sich das in den Grundlagen nicht thematisierte matplotlib. Dabei handelt es sich um eine Python-Bibliothek zur Anfertigung veröffentlichungsfähiger mathematischer Abbildungen [MATPLOTLIB].

```
pip3 install --upgrade jupyter numpy pandas scipy scikit-learn
matplotlib
```

Dieser Vorgang benötigt sechs Minuten und dreißig Sekunden. Wie der Befehl schon vermuten lässt, sind während des besagten Zeitraumes unter anderem Jupyter 1.0.0, NumPy 1.16.2, pandas 0.24.2, scipy 1.2.1, scikit-learn 0.20.3 und matplotlib 3.0.3 installiert worden. Die Zahlenkürzel hinter den Paketnamen stellen die Versionsnummern dar. Eine vollständige Übersicht findet sich in der Anlage C1.

Unter Ubuntu 18.04.2 LTS benötigt man zuerst noch pip, welches durch die folgende Codezeile zu installieren ist.

```
sudo apt install python3-pip
```

Nach ungefähr vierzig Sekunden ist pip installiert, anschließend ist die Beschreibung für Raspbian auch unter Ubuntu nutzbar. Hier dauert die Installation von scikit-learn fast siebzehn Sekunden, das Upgrade benötigt fünfundvierzig Sekunden. Alternativ kann auch nur der folgende Befehl genutzt werden, falls nur scikit-learn inklusive NumPy und SciPy auf dem System landen soll.

```
pip3 install scikit-learn
```

Auf potenteren Linux Maschinen als dem Raspberry Pi kann alternativ zu den bisher beschriebenen Verfahren auch Anaconda genutzt werden. Eine Installation erfolgt durch Ausführen des Shell-Skripts. Dieses findet sich unter der in 4.1.3. genannten Adresse. Sollte dieses sich nicht ausführen lassen, sind dessen Zugriffsrechte anhand der ersten nachfolgenden Codezeile anzupassen. Ist Anaconda installiert, kann der Navigator anhand der Eingabe der folgenden Zeichenkette in ein Terminal-Fenster gestartet werden.

```
sudo chmod 755
anaconda-navigator
```

4.2 Grafische Benutzeroberfläche

Unter den verglichenen Kandidaten besitzt einzig KNIME nativ eine vom Hersteller angebotene und integrierte GUI. Bei den anderen beiden Kandidaten lässt sich das Fehlen unterschiedlich leicht und umfangreich nachrüsten.

4.2.1 KNIME

Dank der GUI ist das Konzept hinter KNIME schnell abbild- und anwendbar. Unter Punkt 2.5 wurde bereits eine Abbildung der GUI gezeigt. Die folgende Abbildung zeigt dasselbe Fenster einer offenen KNIME Instanz ohne zusätzliche Beschriftungen. Diese Ansicht stellt sich der nutzenden Person nach dem Erstellen eines neuen Workflows dar. Dazu ist im zentral positionierten „Workflow Editor“ ein Klick auf die Schaltfläche „Create new workflow“ zu tätigen.

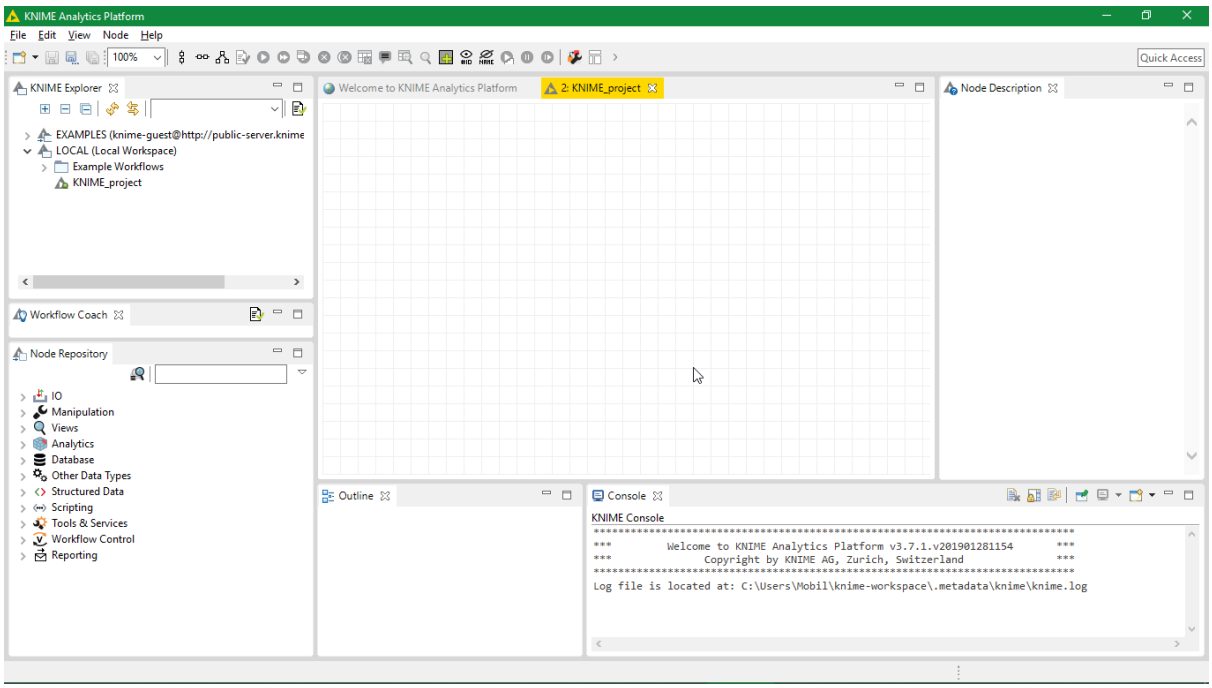


Abbildung 15: Die KNIME GUI bei leerem Workflow
(selbst in KNIME angefertigtes Bildschirmfoto)

Hier finden sich links unten die Nodes thematisch gegliedert. Werden Nodes in das zentrale Fenster gezogen, können sie miteinander verknüpft und konfiguriert werden. Anhand des beschriebenen Ablaufs lassen sich ohne das Schreiben einer einzigen Zeile Code Workflows erstellen. Der „Workflow Coach“ der obigen Abbildung enthält keine Empfehlungen, da dessen Nutzung das Senden anonymisierter Nutzungsdaten an die KNIME AG erfordert. Diese Information ergibt sich aus einer im Startprozess von KNIME stattfindenden Abfrage in einer Dialogbox.

Die übertragenen Daten werden in den FAQs [KNIME_FAQs] auf dreizehn Zeilen verteilt aufgeführt. Um welche es sich im Detail handelt gibt die folgende Tabelle komprimiert wieder.

Tabelle 2: Vom Nutzer zu synchronisierende Informationen zur Nutzung des Workflow-Coach (eigene Darstellung, basierend auf [KNIME_FAQs])

Informations-subjekt	Übertragene Detailinformation
KNIME:	- Genutzte Version der Plattform
	- Wie lange die Plattform läuft
	- Wie oft KNIME gestartet wurde
	- Wie oft KNIME erflogreich beendet wurde
Nodes:	- Welche Nodes genutzt wurden
	- Zu jedem Node wie oft er benutzt wurde
	- Zu jedem Node wie oft er ausgeführt wurde
	- Zu jedem Node wie viel Zeit für dessen Ausführung verbraucht wurde
	- Zu jedem Node dessen wahrscheinlichster Nachfolger im Workflow

Entscheidet sich die nutzende Person für die Übertragung der Informationen, hat sie bezüglich des Sendens ihrer Daten die Wahl zwischen drei Optionen. Diese beziehen sich auf die Frequenz, aber nicht auf die Inhalte der Übertragung. Möglich sind die komplett manuelle, wöchentliche, oder die monatliche Übertragung der eigenen Daten. Im Austausch gegen die Daten liefert die KNIME AG beispielsweise die in der folgenden Abbildung gezeigten Informationen.

Da es sich bei der Abbildung um die Empfehlungen handelt, die gegeben werden, wenn ein leerer Workflow vorliegt, kann man daraus folgern, dass vierundzwanzig Prozent der Nutzer mit dem Einlesen unklarer Daten starten. Achtzehn Prozent der Nutzer beginnen mit dem Import von Daten des Dateiformates csv, siebzehn beginnen mit Inhalten aus Excel-Dateien.

Das in der unteren Mitte des Bildschirmfotos in Abbildung 15 befindliche (vom Autor als „Umriss“ übersetzt) Fenster „Outline“ zeigt den Umriss des aktuellen Workflows und dient der schnelleren Navigation in diesem. Die Skalierung des Umrisses passt sich der Skalierung des Workflows in dessen dediziertem Fenster an.

Die rechts oben verortete „Node Description“ liefert die Beschreibung zu gewählten Bestand-teilen des links unten platzierten Fensters,

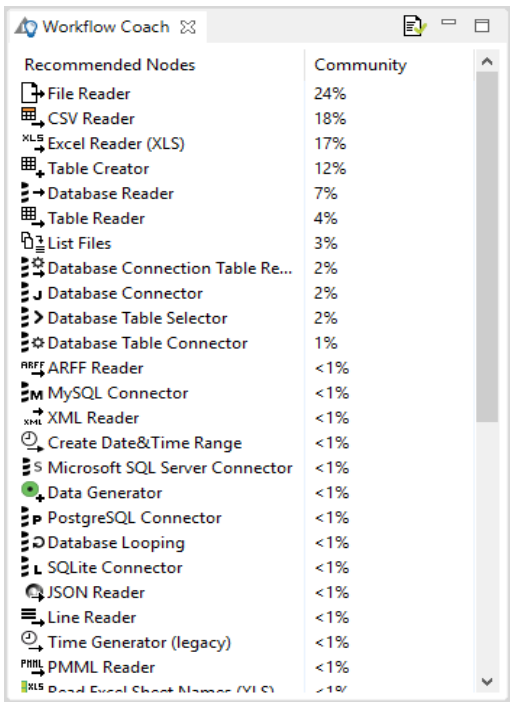


Abbildung 16: Informationen des Workflow Coach (selbst in KNIME angefertigtes Bildschirmfoto)

welches von der KNIME AG als „Node Repository“ bezeichnet wird. Diese Beschreibungen werden - bei Auswahl eines Nodes - um sogenannte „Dialog Options“ und Ports angereichert. Unter „Dialog Options“ werden die, bei Rechtsklick und anschließendem Konfigurieren des Nodes, möglichen Einstellungsoptionen erläutert. Wählt die anwendende Person einen Ordner im „Node Repository“ aus, so werden dessen enthaltene Nodes im „Node Description“ aufgeführt und deren Funktion kurz erläutert.

Der in der Benutzeroberfläche links oben befindliche „KNIME Explorer“ fungiert als KNIME-bezogener Dateimanager. Er stellt, nach Rechtsklick auf die Objekte, zusätzliche Funktionalitäten mit Workflow-Bezug zur Verfügung. Das Erstellen neuer Ordner ist mit ihm nicht möglich. Diese können mit üblicher Software im Ordner „knime-workspace“ erzeugt werden, um später auch im „KNIME Explorer“ angezeigt zu werden.

Die rechts unten angezeigte Konsole gibt standardmäßig Warnungen und Fehlermeldungen aus. Diese Ausgabe kann aber anhand Klicks auf die Schaltflächen „File“ → „Preferences“ →“KNIME“ auch angepasst werden.

4.2.2 TensorFlow

Jede Installation von TensorFlow beinhaltet auch TensorBoard [TF_INSTALL_BOARD], welches man im weitesten Sinne als TensorFlows GUI verstehen kann. Die sprachliche Einschränkung erfolgt, da mit TensorBoard keine Programmierung in TensorFlow stattfindet, sondern lediglich dessen Ausgabe visualisiert und untersucht werden kann [TF_USEBOARD]. Dazu werden Inhalte an TensorBoard übergeben und von diesem dann standardmäßig auf dem Port 6006 des aktuellen Rechners (unter http://localhost:6006) zur Verfügung gestellt.

Des Weiteren existiert unter https://playground.tensorflow.org eine abgespeckte digitale Spielwiese, die Beispieldatensätze nutzen kann aber auf einer anderen Codebasis steht.

Als echte Option zur grafischen Nutzung aller Funktionen von TensorFlow bewirbt Deep Cognition sein „Deep Learning Studio“. In diesem findet sich unter anderem die Möglichkeit der Nutzung von TensorFlow [DLS_PRODUCT]. Die Abbildung zu Beginn der folgenden Seite wurde der genannten Quelle entnommen und zeigt den visuellen Model-Editor. Deep Learning Studio ist unter der Adresse https://deepcognition.ai/products/desktop/ abrufbar und ist dort aktuell in der Version 2.5.0 für Mac und Windows verfügbar.

Die Installation des Deep Learning Studios ermöglicht genauere Einblicke in die Software, welche sich bei einer ersten Analyse als eine um einige Funktionen angereicherte JupyterLab-Oberfläche erweist. Bei zweitem Blick offenbart sie sich als eine Kombination, die sich unter anderem aus vielen quelloffenen Instrumenten zusammensetzt. Die Software erfordert trotz lokaler Installation zuerst eine Registrierung, und mit jedem Start eine Anmeldung unter https://deepcognition.ai/. Eine weitere Besonderheit ist das unverschlüsselte Protokollieren getätigter Nutzeraktivitäten, welches anhand einer Logdatei – unter anderem inklusive Mailadresse und Benutzername – nachvollziehbar ist. Der letzte Grund warum diese Lösung nicht weiter untersucht wird, ist nicht die Tatsache, dass Dateien des Typs csv selbst manuell

zu befüllen sind, sondern dass die Nutzung eigener Datensätze deren komplettes Hochladen auf die Server des Anbieters erfordert [DLS_VID].

Genauere Details zur Software finden sich in der Anlage D1.

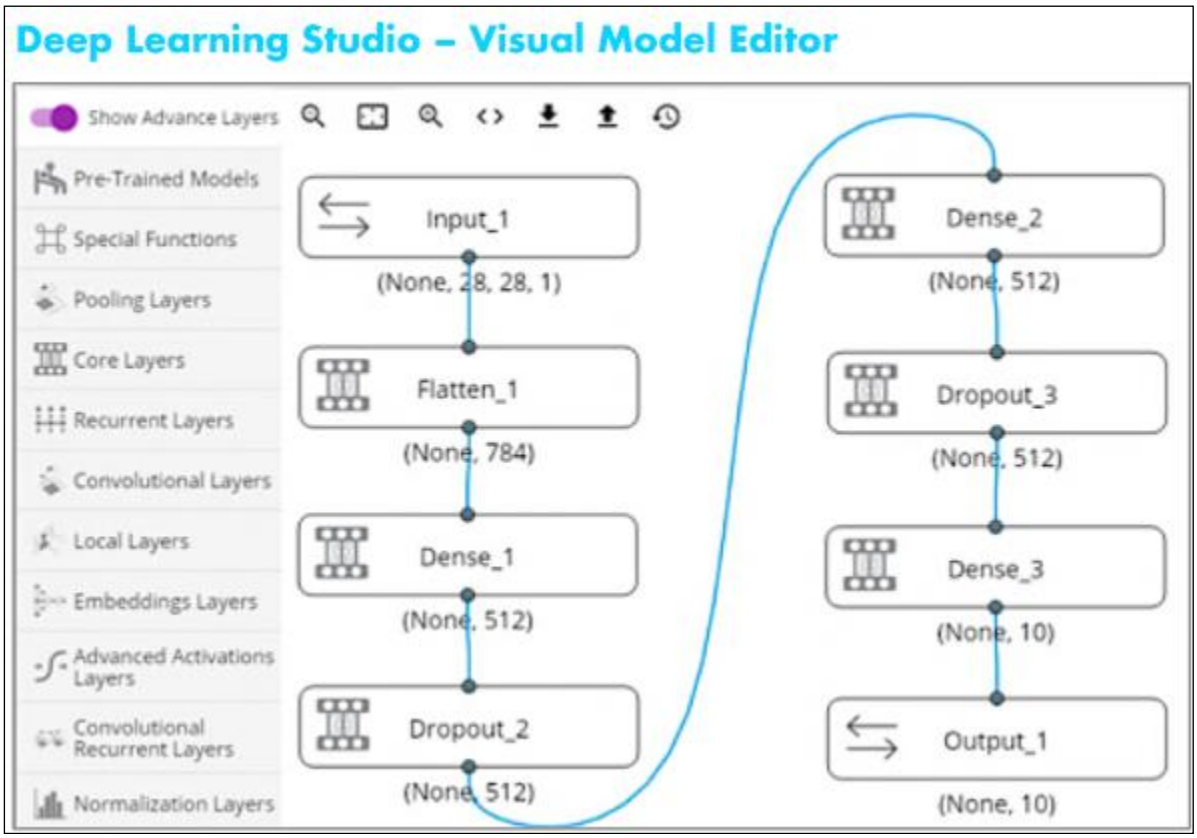


Abbildung 17: Visueller Model-Editor von Deep Learning Studio (Abbildung aus [DLS_PRODUCT, S. 2])

4.2.3 Scikit-learn

Vom Team hinter scikit-learn wurde keine GUI erstellt. Es existieren jedoch zwei Optionen grafikbasiert mit scikit-learn zu arbeiten. Die erste, laut eigener Aussage für Rechner mit Mac OSX konzipierte, Option wurde von drei GitHub-Nutzern erstellt und ist unter https://github.com/bsuhagia/Pykit-Learn verfügbar. In der dort einsehbaren Übersicht wird die Lösung auch „Scikit-learn GUI“ genannt. Der letzte Beitrag zum Projekt fand am 12.11.2015 statt [SK_GH_GUI], weswegen sich die zweite Option anbietet.

Die zweite Option namens „ClassificaIO“ wird von der Python Software Foundation unter https://pypi.org/project/ClassificaIO/, in Form eines Python-Paketes bereitgestellt. Entwickelt wurde die Lösung von zwei Studenten der Michigan State University [CLASSIFICAIO]. Eine Installation der GUI ist dort für Mac, Windows und Linux beschrieben. Um das Paket zu installieren, benötigt es maximal dreier Codezeilen. Unter Windows und Mac befindet sich ClassificaIO nach der Eingabe des folgenden Befehls in ein Anaconda-Terminal-Fenster auf dem System.

```
pip install ClassificaIO
```

In Anaconda unter Windows dauern Download und Installation circa dreizehn Sekunden. Nach dem Öffnen eines Python Fensters und dortiger Eingabe der folgenden beiden Zeilen erscheint die nüchterne Eingabemaske von ClassificaIO [CLASSIFICAIO].

```
from ClassificaIO import ClassificaIO
ClassificaIO.gui()
```

Unter Linux bedarf es neben dem Vorhandensein von Python und pip noch der GUI Erweiterung Tkinter, deren Verfügbarkeit der folgende Einzeiler in Python prüft.

```
import tkinter; tkinter.TkVersion
```

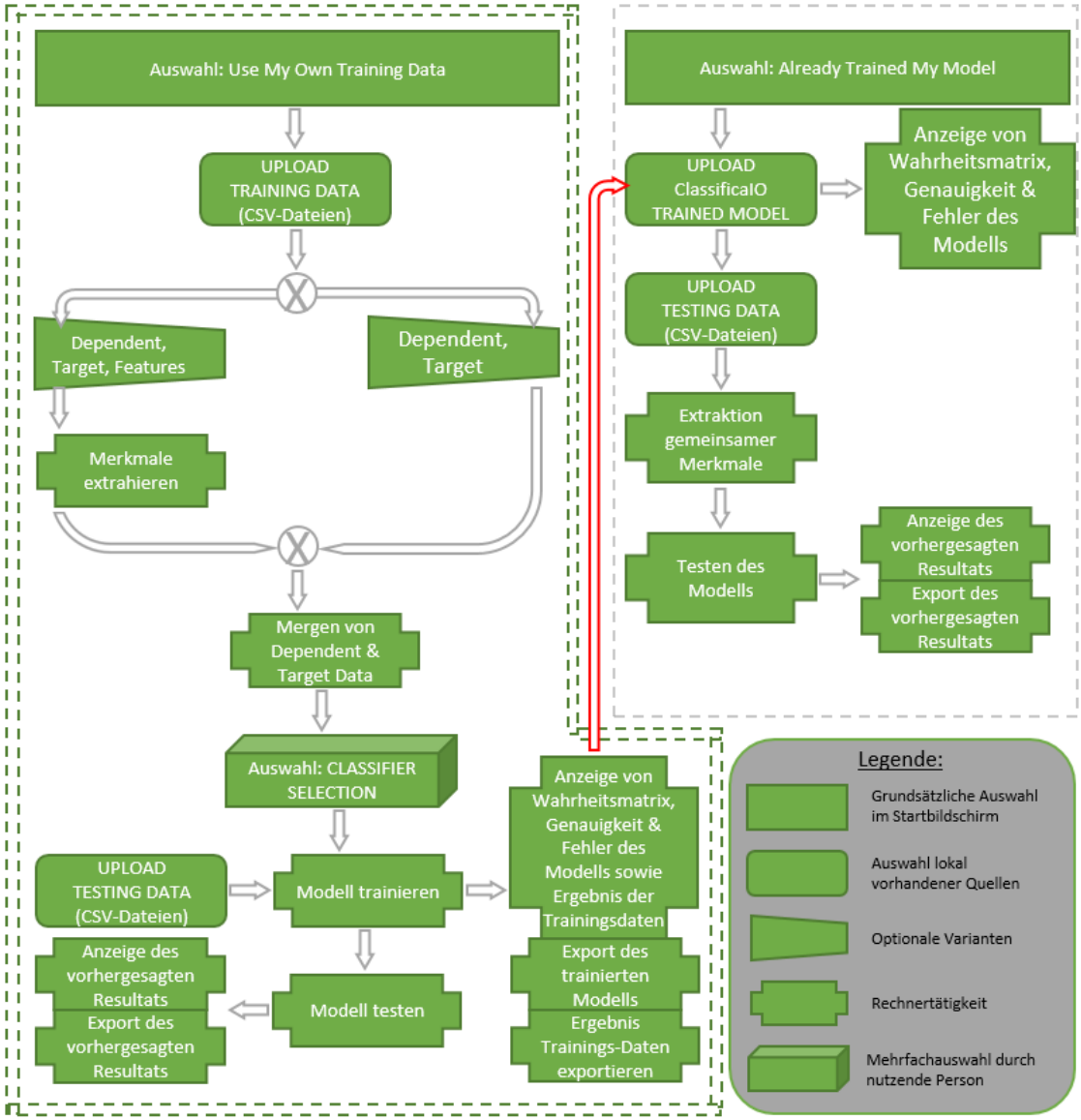
Ist Tkinter bereits auf dem System, so funktioniert die eigentliche Installation unter Linux anhand der minimal angepassten Codezeile, wie sie bereits für Anaconda erwähnt wurde. Dabei ist anstelle der Zeichenkette „pip“ die Zeichenkette „pip3“ zu verwenden. Die Quelle [CLASSIFICAIO] nennt mehrere Installationsmethoden, wovon unter Raspbian keine zum Erfolg führt. Dabei macht es keinen Unterschied, ob das System bereits mehr Software beinhaltet, oder bis auf die Installation von scikit-learn im Auslieferungszustand ist. Für die Gewinnung dieser Information bedarf es neben den bereits benannten Schritten im Schnitt eine Stunde und dreizehn Minuten bis die Fehlermeldung erscheint. Dabei handelt es sich um einen Konflikt bei der Installation der Python Bibliothek namens „Pillow“. Es ist auch irrelevant ob der Installationsversuch mit Administratorenrechten unternommen wurde. Unter Ubuntu 18.04.2 LTS scheint die Installation mit der modifizierten Zeile nach siebzehn Sekunden zum Erfolg zu führen, allerdings lässt sich die GUI trotz Hochfahrens erst nach nochmaliger Installation anhand der folgenden Eingabe auch nutzen.

```
pip install git+https://github.com/gmiaslab/ClassificaIO/
```

Nach der Wahl zwischen der des Trainings der eigenen Daten, beziehungsweise der Option, dass man schon ein Modell trainiert habe, erscheinen die jeweiligen Unterfenster, welche in der Anlage C2 abgebildet sind, wie sie sich nach der Auswahl von Musterdaten darstellen. Beim Training der eigenen Daten reduziert sich die Auswahl der nutzbaren Quellformate auf .csv, was sich nach Klick auf die Schaltflächen zum Import von Trainings- oder Testdaten zeigt. Diese und die folgende Tatsache geben die Abbildungen im Ordner „ClassificaIO“ auf dem Begleit-Datenträger wieder. Bereits trainierte Modelle können in Form eines sogenannten „pickle“ exportiert und auch nur in diesem Format wieder importiert werden. Dieses Dateiformat ist anhand der Endung „pkl“ erkennbar und wurde im Regelfall durch Nutzung des Python Moduls „pickle“ erstellt. Dieses Modul nutzt binäre Protokolle um Python-Objektstrukturen abzubilden. Eine derart als Zeichenstrom gestaltete Abbildung dient dem Zweck an anderer Stelle eine exakte Nachbildung der konservierten Abbildung vollständig wiederherzustellen. Der Grund des ausführlichen Exkurses ist die Tatsache, dass das Modul „pickle“ nicht sicher gegenüber fehlerhaften oder böswillig gestalteten Daten ist [PICKLE]. Würde ein Nutzer ein Modell im pickle-Format, welches er beispielsweise per digital signierter Mail [SIG_SPOOF] erhalten hat, öffnen, liefe er Gefahr, schadhaften Code auszuführen. Die Erforschung der pickle Schwachstelle ist umfangreich dokumentiert und reicht bis ins Jahr 2011 zurück [PICKLE_ROT], weswegen auch wenig versierte Angreifer mit geringem Aufwand bösartige Pickle generieren könnten.

Die nachfolgende Abbildung gibt die Abläufe innerhalb ClassificaIO wieder. Dabei rahmt die grün gestrichelte Linie die Abläufe innerhalb des Trainingsprozesses ein, die graue die der Prozesse bei vorhandenem importierten Modell.

Abbildung 18: Abläufe innerhalb von ClassificaIO
(eigene Abbildung, basierend auf [ClassificaIO])



Wie die erste Abbildung der Anlage C2 zeigt ergeben sich nach der Auswahl von lokal hinterlegten Trainings- und Testdaten weitere Trainingsoptionen, die die GUI der nutzenden Person ansonsten vorenthält. Unter diesen Optionen findet sich unter anderem der prozentuale Anteil der zu nutzenden Trainingsmenge, die Anzahl an Trainingsdurchläufen und die Wahl des Strafmaßes. Ohne Eingabe von Daten besteht nur die Möglichkeit ein Klassifikationsverfahren zu wählen. Das Duo hinter der Software ordnet die insgesamt fünfundzwanzig zur Verfügung stehenden Verfahren den nachfolgenden zehn Überschriften zu.

Dabei entspricht die arabische Zahl vor dem Titel der im Original römisch nummerierten und englisch bezeichneten Zuordnung innerhalb von ClassificaIO.

1 Lineare Modelle	2 Diskriminan-tenanalyse	3 SVMs	4 Nachbarn	5 Gaußscher Prozess
6 Naive Bayes	7 Bäume	8 Ensemble	9 Halb überwacht	10 Neuronales Netzwerk

ClassificaIO nutzt dabei nicht alle Möglichkeiten die scikit-learn bietet. So finden sich beispielsweise unter den linearen Modellen der GUI fünf nutzbare Modell, wobei scikit-learn aber laut dessen Dokumentation über sechzehn lineare Modelle verfügt[SCIKIT, S. 173-194]. Auch bei den restlichen neun Zuordnungen fehlt mindestens jeweils ein Punkt, welcher sich in der Dokumentation und somit im eigentlichen Funktionsumfang von scikit-learn wiederfindet.

5 Funktionsumfang

5.1 Integrierte Musterdaten

Dieser Abschnitt beschreibt die unterschiedlichen Möglichkeiten der Kandidaten bei der Auswahl und dem Import von integrierten Daten. Somit verschafft er zum einen den Überblick über die in den Lösungen integrierten Musterdaten und zum anderen über die unterstützten Dateiformate und nutzbare Quellen nach deren Ursprung.

Jede Lösung verfügt über Musterdaten unterschiedlichen Umfangs. Diese werden für einen schnelleren Zugang des Lesers zu nutzbaren Daten an erster Stelle erwähnt. Ein Überblick findet sich in den folgenden Unterunterabschnitten. Der Umfang der Beispiele des ersten Kandidaten ist so groß, dass eine detaillierte vollständige Untersuchung nicht erbracht wird. Eine Abbildung dieser findet sich auf dem beigefügten USB-Stick im Verzeichnis „Musterdatensätze in KNIME“. Die Begründung über deren Unvollständigkeit liefert die Anlage A1

5.1.1 KNIME

Ein Überblick über die extern abrufbaren Musterdaten in KNIME wird nach Ausklappen des „EXAMPLES“ Ordners im KNIME Explorer ermöglicht. Anhand eines Doppelklicks auf die erscheinende Schaltfläche wird die thematisch strukturierte Übersicht der Beispiele geladen. Diese Beispiele bestehen in den ersten zwölf Hauptordnern hauptsächlich aus kompletten beispielhaften Workflow-Dateien. Ab dem dreizehnten Ordner namens „50_Applications“ steigt die Anzahl anderer Formate. Da die beinhalteten Dateien der Beispiele erst nach dem Öffnen der eigentlichen Workflows ersichtlich sind, beschränkt sich die folgende Untersuchung auf die Beispiele eines Hauptordners. Für eine genauere Untersuchung wird der vorletzte extern abrufbare Ordner namens „_Example_Workflows_from_Installation“ gewählt. Dies geschieht aufgrund seiner Deckungsgleichheit mit dem standardmäßig lokal installierten

Ordner mit der Bezeichnung „Example Workflows“. Diese Behauptung belegt der bildliche Vergleich im Anschluss.

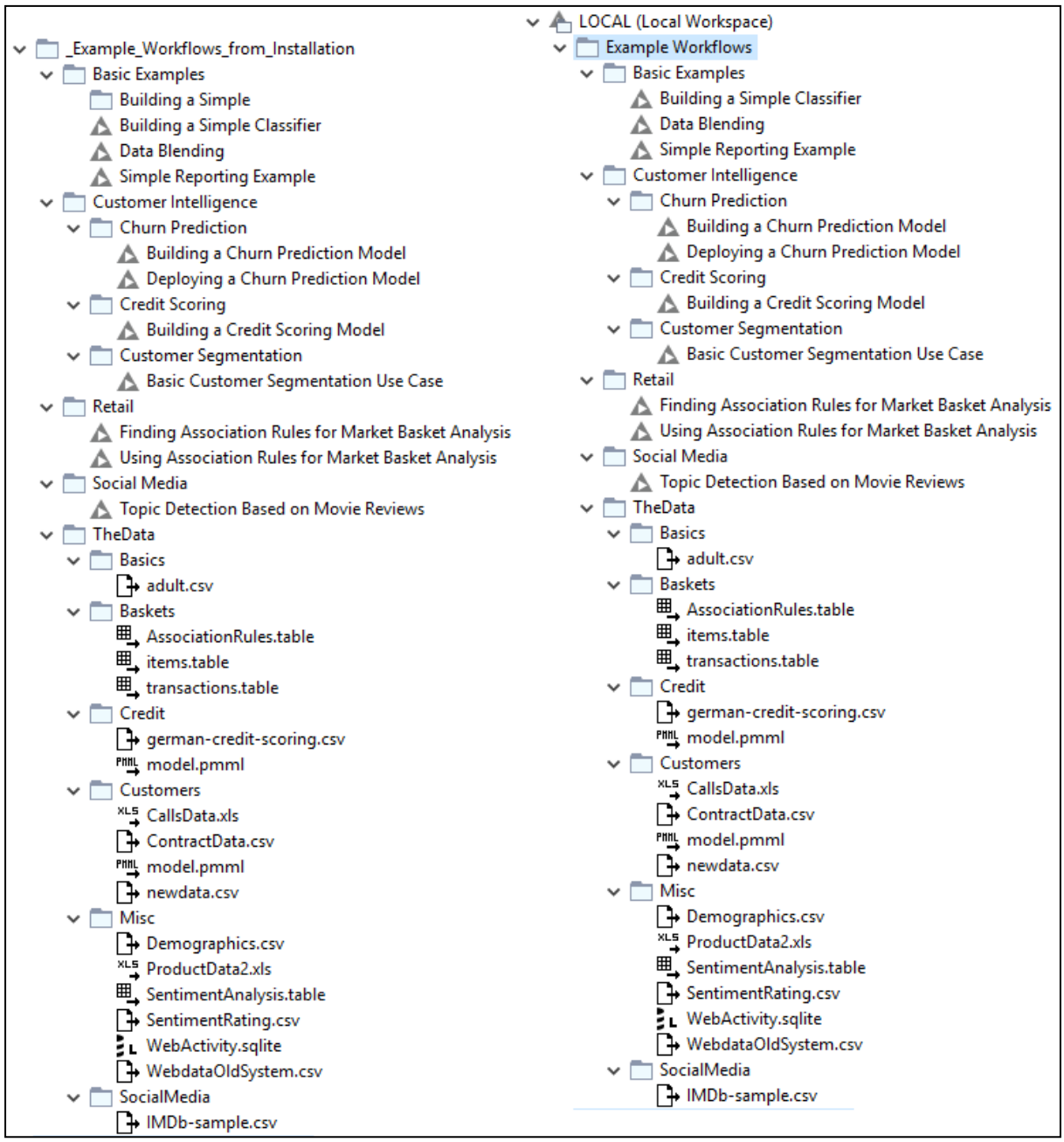


Abbildung 19: Vergleich zweier KNIME Beispielsordner
(Selbst angefertigte, zusammengefügte, Bildschirmfotos aus KNIME)

Die Abbildung gibt linkerhand den Inhalt des erst herunterzuladenden Ordners wieder, rechts findet sich der Inhalt des Ordners, welcher mit der Installation von KNIME bereits auf dem System verfügbar gemacht wird, wieder. Somit sind die in der nachfolgenden Tabelle aufgezählten Inhalte auch ohne eine Verbindung zum Internet sofort nach Installation von KNIME nutzbar.

Tabelle 3: Lokal verfügbare KNIME Beispieldaten
(eigene Darstellung, basierend auf Verfügbarkeit in KNIME)

Ordner	Unterordner	Workflow	Quelldatei	Anzahl Zeilen	Beschreibung des Inhalts
Basic Examples		Building a Simple Classifier	adult.csv	32.560	Unvollständige, anonymisierte Informationen zu Personen
		Data Blending	WebDataOldSystem.csv	9.381	Attribute CustomerKey und First(WebActivity)
			WebActivity.sqlite	2.858	Attribute CustomerKey und WebActivity
			SentimentAnalysis.table	20.724	Attribute CustomerKey und Sentiment Analysis
			SentimentRating.csv	5	Zuordnung nominale Bezeichnungen der Sentiment Analysis zu numerischen Werten namens "SentimentRating"
			Demographics.csv	18.483	Details zu Personen mit Ehestatus, Geschlecht, Einkommen p.a. und mehr
			ProductData2.xls	18.483	Attribute Customer Key und Products
		Simple Reporting Example	adult.csv	32.560	Unvollständige, anonymisierte Informationen zu Personen
Customer Intelligence	Churn Prediction	Building a Churn Prediction Model	CallsData.xls	3.333	16 Spalten zum Telefonverhalten, über enthaltene Telefonnumern referenzierbar
			ContractDate.csv	3.333	7 Attribute welche anhand einer Telefonnummer deanonymisierbar sind
		Deploying a Churn Prediction Model	newdata.csv	1	19 Spalten zum Telefonverhalten, über enthaltene Telefonnumern referenzierbar
	Credit Scoring	Building a Credit Scoring Model	german-credit-scoring.csv	1000	Schufa-artige Datensätze mit 21 Attributen
	Customer Segmentation	Basic Customer Segmentation Use Case	CallsData.xls	3.333	16 Spalten zum Telefonverhalten, über enthaltene Telefonnumern referenzierbar
			ContractDate.csv	3.333	7 Attribute welche anhand einer Telefonnummer deanonymisierbar sind
Retail		Finding Association Rules for Market Basket Analysis	transactions.table	2.869	Transaktionszeilen mit unterschiedlicher Anzahl an Produkten, alle in einem Feld
			items.table	246	Produkte mit Artikelnummer, Preis und englischer Produktbezeichnung
Social Media		Topic Detection Based On Movie Reviews	IMDb-sample.csv	2.000	Nutzerkommentar-Strings zu unterschiedlichen Filmen sowie deren URL, ein numerischcer Index und Die Einteilung "POS"

*(Sämtliche falsch getätigten Großschreibungen im Englischen sind direkt aus KNIME übernommen.)

Anders als bei den Vergleichspartnern sind die Daten bereits automatisch in einen analytischen Kontext eingebunden, um deren Analyse beispielhaft darzustellen. Die Auswahl erfolgt in den jeweiligen Kontext eingebunden durch Doppelklick auf den die Daten beinhaltenden Workflow.

Ein Beispiel für diese Behauptung ist der in der Tabelle beschriebene Inhalt des Ordners „Customer Intelligence“. Die hier hinterlegten kommaseparierten Werte der Datei „ContractDate.csv“ werden zuerst durch einen sogenannten „Joiner“ mit den Daten der Excel-Datei „CallsData.xls“ verbunden. Nach Vollziehen dieses Schrittes wird der Node „Number To String“ genutzt, um den Typ der Werte, der Postleitzahlen und der Abwanderung von Integer in String umzuwandeln. Der eigentliche Inhalt bleibt davon unberührt. Die Abwanderung bezeichnet in den Beispieldaten den Wechselzustand eines Kunden. Hat diese den Wert null blieb dieser beim Anbieter, wohingegen eine eins die Kündigung des Vertrages repräsentiert. Anhand des Wertes, welcher im Feld der Abwanderung angegeben ist werden anschließend die Zeilen der Tabelle anhand des Nodes „Color Manager“ rot eingefärbt. Der Node „Partitioning“ folgt auf diesen und teilt die Daten in ein Test- und ein Trainings-Set ein.

Hier teilt sich der Verlauf der getrennten Daten auf. Die des Trainings-Sets werden an einen Entscheidungsbaum-Lern-Node weitergeleitet, dessen Ergebnis in Form eines Modells im PMML Format vom „PMML Writer“ Node gespeichert wird. Das Entscheidungsbaummodell

wird aber auch an einen Vorhersage-Node weitergeleitet, um Vorhersagen für die Daten des Test-Sets zu generieren. Diese Vorhersagen werden einerseits mit den realen Werten durch den „Scorer“ Node verglichen, aber auch an einen Node zur grafischen Aufbereitung weitergeleitet.

Diese JavaScript-basierte Aufbereitung ist auf die Kurvendarstellung von Klassifikationsproblemen mit zwei Klassen spezialisiert. Eine grafische Wiedergabe liefert die folgende Abbildung, welche sich etwas breiter dargestellt mit der Bezeichnung „Building a Churn Prediction Model“ im Unterordner „Churn Prediction“ des eingangs genannten Ordners „Customer Intelligence“ befindet. Der Import dieser Daten ist denkbar einfach. Die Verknüpfungen sind in den jeweiligen Workflows hinterlegt und das eigentliche Laden der Daten wird durch Auswahl des importierenden Nodes und anschließenden Druck der Taste F7 angestoßen.

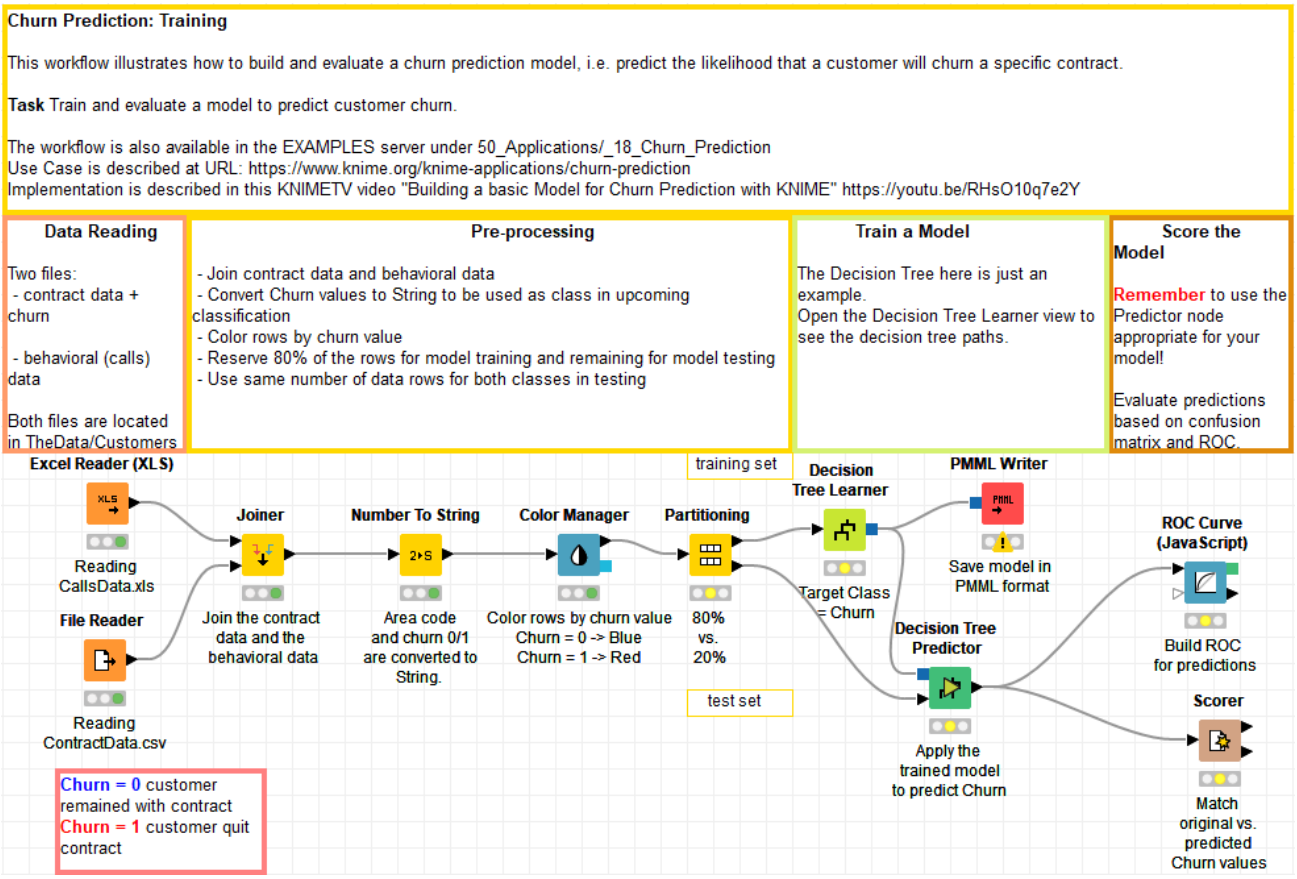


Abbildung 20: Workflow zur Erstellung eines Vorhersagemodells über die Abwanderung von Kunden (selbst angefertigtes Bildschirmfoto eines leicht gestauchten Musterbeispiels)

Der analysierte Beispielordner ist, wie die Abbildungen auf dem beigelegten USB-Stick ersichtlich machen, einer der wenigen Ordner, deren Unterordner die Datensätze auch einzeln enthalten. Falls ein Musterbeispiel einer bestimmten Erweiterung bedarf, vermeldet dies KNIME sofort und ermöglicht eine unkomplizierte Installation.

Als letzte Möglichkeit der Nutzung integrierter Daten sind die Nodes „Data Generator“ und „Time Generator“ anzuführen. Erster befindet sich im Unterordner „Other“ des Ordners „IO“ und ist in der Lage, Zufallsdaten zu generieren. Diese enthalten laut der Beschreibung in der

Node Description „einige Cluster für ein Paralleluniversum“ mit einem definierten Anteil an Rauschmustern. Öffnet man den Unterordner „Time Series“ des Ordners „Other Data Types“, und anschließend den Ordner „Time series (legacy)“, so findet sich dort der Node „Time Generator“. Dieser ist in der Lage, Zeitwerte mit einer bestimmten Anzahl abstandsgleicher Einträge innerhalb einer konfigurierbaren Zeitspanne zu erstellen. Generatoren nachträglich installierbarer Erweiterungen wurden nicht untersucht, weswegen diese hier keine Aufzählung finden.

5.1.2 TensorFlow

Die für TensorFlow verfügbaren Musterdatensätze sind anhand ihres Ursprungs sortiert. Die Arten der Ursprünge reichen von Bildern über „Strukturiert“, Texte, Töne, Übersetzungen bis hin zu Videos. Unter „Strukturiert“ findet sich dabei ein Datensatz, welcher den Überlebensstatus einzelner Passagiere der Titanic darlegt. Eine vollständige Übersicht über die für TensorFlow verfügbaren Datensätze gibt die Tabelle gegen Ende dieses Unterabschnittes wieder. Vollständig mit der Einschränkung, dass der Vollständigkeitsanspruch für den Zeitpunkt des Erstbesuchs am 22.03.2019 gültig ist. Aufgrund der Möglichkeit der nutzenden Personen neue Datensätze anzulegen, existieren am 01.05.2019 bereits sechsundzwanzig in der Tabelle nicht wiedergegebene Musterdatensätze. Eine aktuelle Übersicht findet sich unter https://www.tensorflow.org/datasets/datasets. Anhand der in der Quelle benannten Datensätze findet die eigentliche Auswahl in TensorFlow statt.

Die Nutzung der TensorFlow Musterdaten erfordert ein vorheriges Installieren des Pakets „TensorFlow Datasets“. Manche Datensätze benötigen wiederum zusätzliche Bibliotheken. Je nachdem ob eine unterstützte GPU genutzt wird oder nicht, ist der Zusatz „-gpu“ des folgenden Befehls für die Installation irrelevant oder sinnvoll [TF_DATASET].

```
pip install tensorflow-datasets-gpu
```

Um die Datensätze nutzbar zu machen, sind diese in Python vorher zu importieren. Dies geschieht mit der folgenden Codezeile, dabei wird das Paket dem Kürzel „tfds“ zugewiesen.

```
import tensorflow_datasets as tfds
```

Die eigentlichen Musterdatensätze können dann heruntergeladen und - beispielsweise mit den zugehörigen Informationen - Variablen zugewiesen werden. Die notwendige Eingabe gibt die folgende Codezeile wieder [TF_SAMPLE_USE].

```
daten, informationen = tfds.load("mnist", with_info=True)
```

Hierbei wird die Variable „daten“ mit den Daten des Datensatzes der handgeschriebenen Ziffern gefüllt, die Variable „informationen“ enthält anschließend Detailinformationen zum importierten Datensatz. Ein bildliches Beispiel dieser Informationen gibt die Anlage B3 wieder. Darunter befinden sich beispielsweise Name, Version, Beschreibung und Zitierweise des Datensatzes. Alternativ können die Daten auch direkt einer Variablen zugewiesen werden, dies geschieht mit der Nutzung der folgenden Eingabe [TF_SAMPLE_USE].

```
daten = tfds.builder("mnist")
```

Tabelle 4: Übersicht der Musterdaten in TensorFlow und Keras (eigene Darstellung, basierend auf [TF_DATASOURCE], [KERAS, /datasets/])

Datentyp	Bezeichnung	Kurzbeschreibung	Dateigröße	Keras
Bilder	cats_vs_dogs	Großes Set mit Bildern von Katzen und Hunden	786,68 MB	
	celeb_a	Mehr als 200.000 Bilder mit je 40 Attributen	1,38 GB	
	celeb_a_hq	30.000 Bilder as "celeba" in maximal 1024*1024 Pixeln	abhängig von Setauswahl	
	chexpert	224.316 Brust-Röntgenaufnahmen von 65.240 Patienten	nicht angegeben	
	cifar10	60000 32*32 Pixel Farbbilder in 10 Klassen á 6000 Bilder	162,17 MB	identisch
	cifar100	Aufgeteiltes cifar 10 – 100 Klassen á 600 Bilder	160,71 MB	identisch
	coco2014	Bilddatensatz mit Bildern und objektbezeichnenden Boxen	37,57 GB	
	colorectal_histology	8 Klassen von 150*150*3 RGB Bildern der Gewebelehre koloteraler Karzinome	246,14 MB	
	colorectal_histology_large	10 Bilder der Gewebelehre koloteraler Karzinome in 5000*5000 Pixeln	707,65 MB	
	diabetic_retinopathy_detection	Großes Set hochaufgelöster Bilder von Retinas	nicht angegeben	
	fashion_mnist	70.000 28*28 Pixel Bilder von Artikeln die Zalando führt, Testset 60' Train 10'	29,45 MB	identisch
	horses_or_humans	Großes Set mit Bildern von Pferden und Menschen	153,59 MB	
	image_label_folder	Exemplarischer Datensatz zur Klassifikation von Bildern	nicht angegeben	
	imagenet2012	Bilddatensatz, Bilder mit Wörtern oder Wortgruppen beschrieben (ILSVRC)	nicht angegeben	
	lsun	Bilder verschiedener, kategoriebezogener Kategorien (Die angegebene Größe ist die der Bilder von Schlafzimmern)	42,77 GB	
	mnist	MNIST Datenbank handgeschriebener Ziffern	11,06 MB	identisch
	omniglot	1623 unterschiedliche handgeschriebene Zeichen aus 50 Alphabeten	17,95 MB	
	open_images_v4	~9 Millionen Bilder mit Beschreibungen und objektumschliessenden Boxen	565,11 GB	
	quickdraw_bitmap	50 Millionen Zeichnungen aus 345 Kategorien als 28*28 grau-skalierte Bilder	36,82 GB	
	rock_paper_scissors	Bilder von Händen deren Besitzer "Schere, Stein, Papier" spielen	219,53 MB	
	svhn_cropped	Datensatz zur Erkennung von Ziffern in Bildern mit >600' Bildern in 32*32 Pixeln	1,47 GB	
	tf_flowers	Großer Blumenbildersatz	218,21 MB	
Strukturiert	titanic	Bearbeiteter Satz der den Überlebensstatus einzelner Titanic Passagiere darlegt	114,98 KB	
Text	imdb_reviews	Großer Datensatz von Filmrezensionen, 25.000 Trainings- und 25.000 Testsätze	80,32 MB	ähnlich
	lm1b	Fast eine Milliarde Wörter Trainingsdaten in unklarer Sprache, um den Fortschritt statistischer Sprachmodellierung zu messen	1,67 GB	
	multi_nli	433.000 Satzpaare mit textueller Beschreibung der Folgebeziehung	216,34 MB	
	squad	Antworten eines Leseverständnistests, basierend auf Crowdworkerantworten (SQuAD)	33,51 MB	
Töne	nsynth	Datenset mit ~300.000 Musiknoten	73,07 GB	
Übersetzung	flores_translate_neen	Datensatz der Beurteilung maschineller Übersetzung mit geringen Ressourcen von Nepali ins Englische	984,65 KB	
	flores_translate_sien	Datensatz der Beurteilung maschineller Übersetzung mit geringen Ressourcen von Sinhala ins Englische	984,65 KB	
	ted_multi_translate	Multilinguale Transkripte von TED-Vorträgen in 60	335,91 MB	
	wmt_translate_ende	Übersetzungsdatensatz vom Englischen ins Deutsche der auf den Daten von statmt.org basiert	1,60 GB	
	wmt_translate_enfr	Übersetzungsdatensatz vom Englischen ins Französische der auf den Daten von statmt.org basiert	nicht angegeben	
Video	bair_robot_pushing_small	Zirka 44.000 Beispiele der Drückbewegungen von Robotern in 64*64 Pixeln	30,06 GB	
	moving_mnist	Bewegte Variante der MNIST Datenbasis handgeschriebener Zahlen	781,25 MB	
	starcraft_video	Unterschiedliche Videos aus dem Computerspiel Starcraft in einer Auflösung von 64*64 oder 128*128 Pixeln (Kleinstes Video 1,77 GB, grösstes 20,67 GB)	20,67 GB	

Die Tabelle über diesem Satz beinhaltet auch Informationen zu den am 27.04.2019 in Keras hinterlegten Datensätzen. Bei den Filmrezensionen wurde der Datensatz dahingehend angepasst, dass die Wörter der eigentlichen Texte durch deren Häufigkeit innerhalb des Datensatzes - und somit durch einen numerischen Wert - ersetzt wurden. Aufgrund dieser

Anpassung bezeichnet beispielsweise die Zahl 1 das am meisten über alle Texte des Datensatzes hinweg genutzte Wort. Zusätzlich zu den Datensätzen der Tabelle finden sich in Keras auch ein Datensatz mit 11.228 Agenturmeldungen von Reuters die 46 Themen zugeordnet wurden, sowie ein Datensatz der jeweils dreizehn Attribute zu Häusern aus der Gegend der Bostoner Vororte der späten 1970er Jahre wiedergeben. Den Import dieser Daten bewirkt die erste Zeile Code, die zweite die Zuordnung von Test- und Trainingsdaten. Das Beispiel gibt den Prozess für die Daten aus Boston wieder.

```
from keras.datasets import boston_housing
(x_train, y_train), (x_test, y_test) = boston_housing.load_data()
```

5.1.3 Scikit-learn

Die Auswahl des gewünschten Musterdatensatzes findet in scikit-learn durch das Konsultieren der Dokumentation statt. Laut dieser verfügt scikit-learn über drei Hauptgruppen an beinhalteten Musterdatensätzen. Die erste Gruppe nennen die Entwickler „Toy datasets“. Der Vorteil dieser Datensätze liegt darin, dass sie bereits mit der Installation von scikit-learn auf dem jeweiligen System hinterlegt werden. Ihr Laden erfolgt über den Aufruf des Inhalts der Spalte „SK-Funktion“ der nachfolgenden Abbildung. Dabei sind zum Laden der gesamten Daten die eckigen Klammern und deren Inhalt zu ignorieren [SK_IMPORT].

Tabelle 5: Aufstellung der "Toy datasets" aus scikit-learn (eigene Darstellung, basierend auf [SK_IMPORT #toy-datasets])

Empfohlenes Analyseverfahren	SK-Funktion	Beschreibung des Sets
Klassifikation	load_breast_cancer([return_X_y])	UCI ML Datensatz zu Brustkrebs in Wisconsin (Diagnostic)
	load_digits([n_class, return_X_y])	UCI ML Test-Setz hangeschriebener Zahlen
	load_iris([return_X_y])	Iris Datenbanke aus Sir R.A. Fisher's Arbeit
	load_wine([return_X_y])	Modifizierter UCI ML Weindatensatz
Regression	load_boston([return_X_y])	Datensatz der Bostoner Hauspreise
	load_diabetes([return_X_y])	Modifizierter Diabetes-Datensatz den die NCSU vorhält
Multivariate Regression	load_linnerud([return_X_y])	Linnerud Datensatz

Folgende Codezeile bildet repräsentativ das Füllen der Variable „beispiel“ mit den Datensätzen der Bostoner Hauspreise ab [SK_IMPORT_BOSTON]

```
from sklearn.datasets import load_boston
beispiel = load_boston()
```

Das Zuordnen der Daten geschieht dabei so schnell, dass bei allen sieben Beispieldatensätzen aus Tabelle 5 der Vorgang jeweils circa eine Sekunde benötiget. Scikit-learn enthält auch zwei integrierte Bilder “china“ und „flower“ [SK_SAMPLE_PICS]. Diese können anhand des folgenden Befehls der Variable „beispiel“ zugeordnet werden.

```
from sklearn.datasets import load_sample_images
beispiel = load_sample_images()
```

Die Dokumentation beschreibt auch eine Methode namens „load_sample_image“, welche den namentlich benannten Einzelimport eines der beiden Bilder ermöglichen sollte.

Diese Methode funktioniert allerdings mit keinem der beiden Namen. Weder die Nutzung von Hochkommata oder Anführungszeichen, noch Großschreibevarianten machen die Bilder einzeln abrufbar.

Die zweite Gruppe der einfach nutzbaren Beispieldatensätze bezeichnet das Team hinter scikit-learn als "Real world datasets", da diese ihrem Umfang nach den Datenmengen realer Beispiele entsprächen [SK_IMPORT]. Deren Benennung, Inhalt und Umfang gibt die folgende Tabelle wieder.

Tabelle 6: Aufstellung der "Real world datasets" aus scikit-learn (eigene Darstellung, basierend auf [SK_IMPORT #real-world-datasets])

Empfohlenes Analyseverfahren	SK-Funktion	Beschreibung des Sets	Klassen	Gesamt-proben-anzahl
Klassifikation	fetch_olivetti_faces([data_home, shuffle, ...])	Olivetti Gesichts-Datensatz von AT&T	40	400
	fetch_20newsgroups([data_home, subset, ...])	Dateinamen und Daten des 20 Newsgroups-Datensatzes	20	18.846
	fetch_20newsgroups_vectorized([subset, ...])	In Vektoren umgewandelte Zählung der Zeichen des Newsgroups-Datensatzes	20	18.846
	fetch_lfw_people([data_home, funneled, ...])	Bilder berühmter Persönlichkeiten mit Beschreibungen	5.749	13.233
	fetch_lfw_pairs([subset, data_home, ...])	Lädt den oberhalb beschriebenen Bilddatensatz in Bildpaaren	5749	13.233
	fetch_covtype([data_home, ...])	Bilder von 30*30m Abschnitten von US Wäldern	7	581.012
	fetch_rcv1([data_home, subset, ...])	RCV1 - ein Archiv manuell kategorisierter Meldungen der Presseagentur Reuters	102	804.414
Regression	fetch_kddcup99([subset, data_home, shuffle, ...])	Der kddcup99 Datensatz - basiert auf dem tcpdump-Teil des DARPA IDS von 1998	-	4.898.431
	fetch_california_housing([data_home, ...])	DateHausspezifische Daten Kaliforniens aus der amerikanischen Volkszählung des Jahres 1990	-	20.640

Als dritte Gruppe sind die Methoden zur Beispielgenerierung zu nennen. Diese erstellbaren Beispieldatensätze werden in der Tabelle ab Ende dieses Unterunterabschnittes nach ihren spezifischen Anwendungsfällen angeordnet, aufgelistet und beschrieben [SK_IMPORT]. Das Generieren der Musterdatensätze geschieht jeweils in unter einer Sekunde. Die eckigen Klammern der Aufrufe sind nicht notwendig. Die Eingabe der folgenden Zeichenkette generiert den oben beschriebenen Umfang an Musterdatensätzen isotroper gaußscher BLOBs für das Clustering.

```
from sklearn.datasets import make_blobs as clusterBlobmacher
X, y = clusterBlobmacher ()
```

Dadurch wird die Variable X mit den eigentlichen Daten der BLOBs befüllt. In der Variable y finden sich danach die dazugehörigen Attribute. Die Anzahl standardmäßig erstellter Beispieldaten und Attribute wurden vom Autor in einem Jupyter Notebook unter Windows recherchiert. Dies lässt sich beispielsweise anhand der Eingabe der folgenden drei Zeilen in eine Zelle und deren anschließender Ausführung reproduzieren.

```
from sklearn.datasets.samples_generator import make_blobs
X, y = make_blobs()
print(X.shape)
```

Diese Eingabe bewirkt die Anzeige der Zeichenkette „(100, 2)“, die erste Zahl steht für die Anzahl an erstellten Beispielen, die zweite gibt die Attribute je Beispiel wieder. Diese Ausgaben dienen als Basis der in den beiden rechten Spalten ausgewiesenen Werte.

Tabelle 7: Aufstellung der "Generated datasets" aus scikit-learn (Eigene Darstellung, basierend auf [SK_IMPORT #generated-datasets] und eigener Forschung)

Empfohlenes Analyseverfahren	Ergänzungsinformation Datensätzen Klassifikation und Clustering	SK-Funktion	Beschreibung des Sets	Beispiel ohne Eingabe	Anzahl standardmäßig erstellter Beispieldaten	Attribute je Beispiel
Klassifikation und Clustering	Einfach bezeichnet	datasets.make_blobs([n_samples, n_features, ...])	Generiert isotrope gaußsche BLOBs für das clustering	✓	100	2
		datasets.make_circles([n_samples, shuffle, ...])	Erstellt zweidimensional einen kleinen Kreis innerhalb eines größeren	✓	100	2
		datasets.make_classification([n_samples, ...])	Generiert ein zufälliges, n-klassiges Klassifikationsproblem	✓	100	20
		datasets.make_gaussian_quantiles([mean, ...])	Erstellt isotrope gaußsche (sic) und beschriftet diese mit Quantilen	✓	100	2
		datasets.make_hastie_10_2([n_samples, ...])	Schafft Musterdaten zur Binärklassifikation wie in Hastie et al.	✓	12000	10
		datasets.make_moons([n_samples, shuffle, ...])	Erzeugt Punkte zweier, verzahnter Halbkreise	✓	100	2
	Mehrfach bezeichnet	datasets.make_multilabel_classification([...])	Generiert ein zufälliges mehrfach bezeichnetes multilabel	✓	100	20
	Biclustering	datasets.make_biclusters(shape, n_clusters)	Erstellt ein Feld mit konstanter Blockdiagonalmatrix für das			
		datasets.make_checkerboard(shape, n_clusters)	Erstellt ein Feld mit schachbrettartiger Blockstruktur für das Biclustering			
Regression		datasets.make_friedman1([n_samples, ...])	Erstellt Daten des “Friedman #1” Regressionsproblems	✓	100	10
		datasets.make_friedman2([n_samples, noise, ...])	Erstellt Daten des “Friedman #2” Regressionsproblems	✓	100	4
		datasets.make_friedman3([n_samples, noise, ...])	Erstellt Daten des “Friedman #3” Regressionsproblems	✓	100	4
		datasets.make_regression([n_samples, ...])	Schafft Daten eines zufälligen Regressionsproblems	✓	100	100
		datasets.make_sparse_uncorrelated([...])	Generiert ein zufälliges Regressionsproblem mit	✓	100	10
Manifold		datasets.make_s_curve([n_samples, noise, ...])	Erstellt den Datensatz einer S-Kurve	✓	100	3
		datasets.make_swiss_roll([n_samples, noise, ...])	Schafft einen Datensatz dessen Punkte einer Biskuitrolle ähneln ("swiss roll")	✓	100	3
Dekompositon		datasets.make_low_rank_matrix([n_samples, ...])	Generiert eine hauptsächlich geringfügige Matrix mit	✓	100	100
		datasets.make_sparse_coded_signal(n_samples, ...)	Erstellt ein Signal als dünnbesetzte Kombination von Elementen eines			
		datasets.make_sparse_spd_matrix([dim, ...])	Generiert eine dünnbesetzte symmetrisch positiv definite Matrix	✓	1	1
		datasets.make_spd_matrix(n_dim[, random_state])	Erstellt eine zufällige symmetrische positiv definite Matrix			

Zusätzlich zu den benannten Beispielen ist scikit-learn auch nativ in der Lage Datensätze von openml.org herunterzuladen [SK_IMPORT].

5.2 Unterstützte Programmiersprachen

Je nach Präferenz und Vorkenntnis der nutzenden Person stellt das Spektrum an unterstützten Programmiersprachen eine relevante Information im Entscheidungsprozess dar, weswegen diese folgend betrachtet werden. Die Betrachtung umfasst dabei die von den Herstellern in ihren Informationen benannten Sprachen, andere verfügbare Erweiterungen zur Nutzung weiterer Sprachen finden keine Erwähnung. KNIME verfügt in diesem Punkt über ein absolutes Alleinstellungsmerkmal – hier kann Wissensgewinnung ohne das Schreiben von Code erfolgen. Scikit-learn benötigt dazu die unter 4.2.3 beschriebene GUI ClassificaIO.

Eine Grafik am Ende des Abschnittes stellt die verglichenen Lösungen gegenüber, um einen schnellen Überblick zu gewähren.

5.2.1 KNIME

Neben der Eclipse-bedingten Unterstützung von Java ist in KNIME auch die standardmäßige Verwendung anderer Programmiersprachen möglich. Diese dienen unterschiedlichen Einsatzzwecken zum Verfassen analytischer Arbeitsabläufe dienen R und Python. Um Daten zu visualisieren benutzt KNIME JavaScript [KN_CODING]. Neben den bereits angesprochenen Sprachen unterstützt KNIME laut der KNIME AG auch C und C++ [KN_CODE_2].

5.2.2 TensorFlow

Der Umfang der von TensorFlow unterstützten Sprachen umfasst neben Python auch JavaScript, C++, Java, Go und Swift. Allerdings sind nur Python und C abwärtskompatibel [TF_BACK]. Ein Stabilitätsversprechen existiert exklusiv für Python [TF_LINGOS].

Zusätzlich zu den APIs in den eben genannten Sprachen existieren auch noch sogenannte „Bindungen“ für andere Sprachen. Diese Bindungen stellen keine echte Übersetzung dar, sondern binden einen Befehl in einer Sprache an einen Befehl in einer anderen Sprache. Derartige Bindungen existieren aktuell für C#, Haskell, Julia, Ruby, Rust und Scala [TF_LINGOS]. Die beschriebene Unterstützung an Sprachen und Bindungen wird auch für die Version 2.0 übernommen [TF_LINGOS_2.0].

5.2.3 Scikit-learn

Da scikit-learn eine Python Bibliothek ist, wird es auch in dieser Sprache genutzt. Es besteht über Umwege beispielsweise die Möglichkeit scikit-learn in Java zu nutzen. Dies funktioniert über CPython, unter Einsatz von JEP. JEP stellt allerdings keine native Unterstützung dar [JEP_GH].

Tabelle 8: Übersicht unterstützter Sprachen
(eigen Darstellung, basierend auf [TF_LINGOS], [SCIKIT, S.1], [KN_CODING])

Sprache	C	C++	Java	JavaScript	Go	python	R
KNIME	✓	✓	✓	✓		✓	✓
TensorFlow		✓	✓		✓	✓	
scikit-learn						✓	

6. Kompatibilität

Das Kapitel der Kompatibilität umfasst zum einen die Betrachtung der Interoperabilität der einzelnen Lösungen untereinander. Zum anderen beschäftigt sich der Inhalt mit den

unterstützten Quellen; wobei diese Formulierung auf dem Rechner vorhandene Dateiformate bezeichnet. Ebenfalls betrachtet werden nativ unterstützte Netzwerkquellen.

6.1 Interoperabilität

Bezüglich der Interoperabilität wird nachfolgend sowohl die direkte, als auch die indirekte Interoperabilität der Kandidaten untereinander betrachtet. Indirekte Interoperabilität bezeichnet dabei den Import oder Export einer Lösung in eine andere durch vermittelnde Software wie beispielsweise Keras. Wird nachfolgend von Modellen zu lesen sein, so bezeichnet dieser Ausdruck Vorhersagemodelle, die in einer der Lösungen trainiert und somit generiert wurden.

6.1.1 KNIME

Aus Keras-Modellen ist KNIME in der Lage „TensorFlow SavedModels“ abzuleiten. Dies erfordert die Installation der TensorFlow Integration. Wurde diese installiert kann KNIME diese Modelle auch unabhängig von Keras lesen und speichern [KNIME_TF]. Die benötigte Erweiterung findet sich unter den „KNIME Labs Extensions“ und ist unabhängig von der Bitbreite der Rechenarithmetik verfügbar. Diese Ergänzung ist relevant, da manche Erweiterungen nur für 64-Bit-Systeme existieren. Die importierten TensorFlow-Modelle können anschließend in KNIME genutzt, trainiert und bearbeitet werden [KNIME_TF_USE].

Die Lösung der KNIME AG ist auch in der Lage scikit-learn einzubinden, was unter Verwendung der „KNIME Python Integration“ möglich ist. Besagte Integration findet sich unter „KNIME & Extensions“ bei den zusätzlichen Nodes. Damit dies auch funktioniert, erfordert das System eine Python Umgebung. Hierfür empfiehlt die KNIME AG Anaconda und beschreibt in der Quelle auch dessen Installation unter MacOS und Linux [KNIME_SK_PY]. Für die Installation unter Windows muss die Quelle nicht gelesen werden. Dieser Prozess wurde unter dem Punkt 4.1.3.1 beschrieben

6.1.2 TensorFlow

TensorFlow verfügt dank der Integration von Keras über das Modul „tf.keras.models“, welches unter anderem über vier Funktionen zum Import und eine Funktion zur Speicherung von Modellen verfügt. Anhand der Funktion „load_model“ können unter anderem aus KNIME exportierte Keras Modelle im hierarchischen Datenformat HDF5 importiert werden. Die anderen Ladefunktionen importieren Modelle im „yaml“ oder „json“ Format, sowie aus einem sogenannten „Configuration dictionary“ [TF_PY, /keras/models].

Um mit scikit-learn zu interagieren, bedarf es nicht zwangsläufig dieses Umwegs, da beide Lösungen parallel in einem Python-Skript importierbar sind. Des Weiteren existierte in TensorFlow eine Python-API die zu scikit-learn kompatibel war [ML_SL_TF, S. 228]. Diese - „TF.Learn“ benannte - API war aus einem ursprünglich eigenständigen Projekt namens „Scikit Flow entstanden und wird unter anderem auf der GitHub Seite von TensorFlow als veraltet bezeichnet. Erwähnung findet die API, die als Modul namens „tf.contrib.learn“ noch in der TensorFlow Dokumentation zu finden ist [SKF_REMAINS], da Teile daraus in anderen

Modulen implementiert wurden und somit noch nutzbar sind. Dies betrifft die Clusterbildung mit k-means und neun „Estimators“. Diese unter Punkt 2.7 genauer erläuterten Schätzer sind nun durch die Zeichenkette „tf.estimator.Estimator” statt der bisherigen Nutzung von „tf.contrib.learn.Estimator“ im Code aufrufbar [SKF_GIT].

6.1.3 Scikit-learn

Es findet sich keine Quelle die den Import eines Modells aus KNIME für scikit-learn beschreibt. Inhalte aus TensorFlow lassen sich unter der gemeinsamen Nutzung von Python bearbeiten. Dies bezieht sich jedoch nur auf Variablen und deren Inhalt. Der Import eines in TensorFlow erarbeiteten Modells in scikit-learn ist nicht dokumentiert.

6.2 Unterstützte Quellen

Unter dieser bewusst unscharf formulierten Überschrift finden lokale, beispielsweise in Form spezifischer Dateiformate, aber auch nicht auf dem System befindliche Quellen Erwähnung. Explizit im Quellmaterial der jeweiligen Lösung erwähnte Netzwerkquellen werden somit ebenfalls spezifisch benannt. Netzwerkquellen die erst mit einer nachträglichen Erweiterung der ursprünglich durch die Installation erhaltenen Lösung nutzbar sind, werden nicht weiter benannt.

6.2.1 KNIME

In KNIME existieren mehrere Möglichkeiten des Imports von Daten. Die einfachste stellt das Ziehen der benötigten Datei aus dem KNIME Explorer in den Workflow Editor dar. Wenn dieser Vorgang umgesetzt wird, öffnet sich beispielsweise der sogenannte „File Reader“ Node, um eventuell nötige Eingaben seitens des Anwenders abzufragen. Diese können das Mitlesen von Spaltentiteln, Zeilen-IDs, der Art des Trennzeichens und weitere importrelevante Einstellungen betreffen. Je nach Art der zu importierenden Daten kann auch ein anderer Eingabe-Node automatisch aufgerufen werden. Um eigene Daten auf diesem Weg in einem KNIME Workflow zu nutzen, sind diese zuerst in einen innerhalb des Quellordners von KNIME zu erstellenden Ordner zu hinterlegen.

Der alternative Weg des Datenimports besteht darin, einen der datenspezifischen Nodes direkt in den Workflow Editor zu ziehen. Die nachfolgende Tabelle gibt die in KNIME vorhandenen Nodes zum Datenimport wieder, allerdings nur die, für die ein unterstütztes Dateiformat innerhalb der Software angegeben wird. Zusätzlich zu deren Bezeichnung sind deren Symbol, beinhaltender Ordner und die von Ihnen unterstützten Dateiformate aufgeführt.

Tabelle 9: Import-Nodes mit ausgewiesenen Dateiformaten (Eigene Darstellung basierend auf Informationen in KNIME)

Unterordner	Bezeichnung	Icon	Unterstützte Formate	Ergänzende Hinweise
IO \ Read	Excel Reader		xls, xlsx	Liest nur die Daten einer Tabelle, unterstützt werden die Datentypen String, TimeAndDate, Double, Boolean und Integer.
	File Reader		csv, data	Als Abgrenzungszeichen zwischen den Spalten muss nicht bindend ein Komma genutzt worden sein. Die Quelldatei kann auch online sein.
	ARFF Reader		ARFF	Liest ARFF Dateien anhand der Angabe einer URL.
	CSV Reader		csv	Im Unterschied zum File Reader Node ist dieser für eine Serverumgebung, oder für die Stapelverarbeitung gedacht, wenn die Dateistruktur sich während der Aufrufe ändert.
	Table Reader		table	Liest Dateien die mit dem "Table Writer", in einem KNIME-eigenen Format, erstellt wurden.
	PMML Reader		pmml	Liest Modelle aus der beschreibenden XML Datei aus.
	Model Reader		zip	Liest KNIME Modell Port Objekte die mit dem "Model Writer" mode erstellt wurden.
	Fixed Width File Reader		csv, txt	Wie "File Reader", nur dass hier Spalten nicht von einem Trennzeichen, sondern von der Anzahl an Zeichen definiert werden.
	Read Images		png, svg	Liest das erste Bild eines zip Ordners, oder mehrere per URL verknüpfte Bilder, und fügt sie als neue Spalten hinzu.
Database \ Read/Write	Database Reader		jdbc	Benötigt die Definition des händlerspezifischen Datenbank Treibers.
Structured Data	JSON		json	Hat ausser Lese- auch noch Konvertierungsfunktion von bzw. nach json.
	XML		XML	Hat ausser Lese- auch noch Konvertierungsfunktion von bzw. nach XML.
Other Data Types	DML Dcoument Parser		xml	Dursucht Verzeichnisse nach Dateitypen links die dml formatierten Text enthalten und liest diesen, auf Wunsch auch aus Unterverzeichnissen, aus.
\Text Processing	PDF Parser		pdf	Liest pdf Dokumente inklusive der Metadaten Titel und Autor aus und erstellt daraus Dokumente ohne Übernahme der Ursprungsstruktur.
\IO	Sdml DocumentParser		xml, gz, zip	Duchrsucht Verzeichnisse nach Dateitypen links die sdml formatierten Text enthalten und liest diesen, auch aus Unterverzeichnissen, aus.
	Word Parser		doc, docx, docm	Liest die Dokumente der betreffenden Typen aus und nutzt den ersten Satz als Basis zur Erstellung von Dokumenten je Datei.

Einige Nodes, die von der KNIME AG in den Unterordner der Einlese-Nodes abgelegt wurden, hat der Autor nicht erwähnt, da diese seines Erachtens Funktionen des Datenerhebens beziehungsweise Vorverarbeitens abbilden. Dies betrifft die Nodes „List Files“, „Read Excel Sheet Names“ und „Explorer Browser“. Der „Line Reader“ Node findet hier keine Erwähnung, da er mit keinem Dateiformat verknüpft ist.

Was allerdings noch erwähnenswert ist, ist die Tatsache, dass anhand des jdbc Treibers auch andere Datenbanken als Importquellen genutzt werden können. Mögliche aufgelistete Typen sind hierbei H2, Microsoft SQL, MySQL, Postgre SQL, SQLite und HP Vertica. Die Integrationsseite der KNIME AG listet neben den vorgenannten Kommunikationspartnern noch Google Drive, Azure, AWS, PubMed Dokumente, RSS Feeds und sogar Twitter als mögliche Quellen auf [KNIME_PRODUCTS].

Als Schlusssatz dieses Unterunterabschnittes dient die Information, dass aufgrund des erweiterungsorientierten Konzepts der Software auch weitere Formate und Quellen nachrüstbar sind [KN_JAR; KN_MMI; KN_HTS].

6.2.2 TensorFlow

Da TensorFlow, wie unter 2.6 beschrieben, auf Tensoren basiert, erfordert der Import von Daten auch eine Konvertierung in diese. TensorFlow besitzt unter anderem den Namensraum „tf.io“ der dem Vorgang des Umwandelns gewidmet ist. Dessen Funktionen sind Basis der nachfolgenden Tabelle.

Tabelle 10: Funktionen des Namensraums "tf.io" mit direkter Umwandlung eines Formats in einen Tensor (eigene Darstellung, basierend auf [TF_IO])

Funktion	Dateityp	Erläuterung zur Funktion
decode_and_crop_jpeg(...)	jpeg	Schneidet ein JPEG-codiertes Bild zu und konvertiert es in einen uint8 Tensor.
decode_bmp(...)	bmp	Wandelt den ersten Bildrahmen eines BMP-codierten Bildes in einen uint8 Tensor um.
decode_csv(...)	csv	Wandelt csv Dateien in Tensoren um, Jede Spalte wir in einen Tensor übersetzt.
decode_gif(...)	gif	Wandelt den ersten Bildrahmen eines GIF-codierten Bildes in einen uint8 Tensor um.
decode_image(...)	bmp, gif, jpeg	Kombiniert die Funktionen für bmp, gif und jpeg in eine.
decode_jpeg(...)	jpeg	Wandelt ein JPEG-codiertes Bild in einen uint8 Tensor um.
decode_png(...)	png	Kodiert ein PNG-codiertes Bild in einen uint8 oder uint16 Tensor.
extract_jpeg_shape(...)	jpeg	Extrahiert die Informationen bezüglich der Form aus einem JPEG-codierten Bild.

In obiger Tabelle werden nur Funktionen des Namensraums aufgezählt, die eine direkte Umwandlung eines Formats in einen Tensor ermöglichen.

In einem Python Skript können TensorFlow auch andere Quellformate zugeführt werden. Beispielformate dafür werden in der Tabelle des Abschnittes für scikit-learn benannt.

6.2.3 Scikit-learn

In scikit-learn besteht die Möglichkeit, Felder aus NumPy („numpy arrays“), dünnbesetzte Matrizen aus SciPy, sowie andere Datentypen zu nutzen. Die Datentypen müssen in numerische Felder konvertierbar sein, um genutzt werden zu können. Darunter fällt beispielsweise der DataFrame aus Pandas. Die Importfähigkeiten der Bibliothek finden sich auch in der später folgenden Tabelle wieder [pandas, S. 197]. Die Importfähigkeiten von scikit-image [SCIKIT_IMAGE] und imageio[IMAGE_IO] finden über die Nutzung der Felder von NumPy ebenfalls Verwendung in scikit-learn und flossen auch in die Tabelle ein.

Damit die Daten auch in scikit-learn genutzt werden können, müssen diese als numerische Merkmale vorliegen. Alternativ können kategorische und nominelle Merkmale anhand

verschiedener Funktionen umgewandelt werden. Dazu zählen beispielsweise „sklearn.preprocessing.OneHotEncoder“ und „sklearn.preprocessing.OrdinalEncoder“ [SK-LEARN, S. 629]. Ist dieser Schritt vollzogen oder nicht nötig, können die Daten einer Variablen zugewiesen werden. In letzterem Fall ist beispielsweise der nachfolgende Code nutzbar. Er bildet musterhaft den Import einer Datei namens „MUSTER“ mit kommaseparierten Werten ab. Er nutzt für die Zuweisung an die Variable „daten“ pandas. Dieses wird mit der ersten Codezeile importiert, um in der zweiten dessen Funktion „read_csv“ als Einlesewerkzeug zu nutzen [PANDAS_CSV].

```
import pandas
daten = pandas.read_csv('MUSTER.csv')
```

Tabelle 11: Von scikit-learn direkt und indirekt unterstützte Dateiformate (Eigene Darstellung basierend auf Quellen aus obigem Abschnitt)

Importweg	nativ	SciPy	scikit-image		pandas	
Formate:	libsvm	mat	tiff	fits	CSV	Parquet
	openml	arff	gdal	pil	JSON	Msgpack
	svmlight	wav	gtk	simpleitk	HTML	Stata
	txt		imread	qt	lokale Zwischenablage	SAS
			imageio	matplotlib	MS Excel	Python Pickle
					HDF5	SQL
					Feather	Google Big Query

Importweg	imageio				
Formate:	BUFR-PIL	JPEG2000-PIL	BMP-FI	PCX-FI	XBM-FI
	CUR-PIL	MCIDAS-PIL	CUT-FI	PFM-FI	XPM-FI
	DCX-PIL	MIC-PIL	DDS-FI	PGM-FI	ICO-FI
	DIB-PIL	MPO-PIL	EXR-FI	PGMRAW-FI	GIF-FI
	EPS-PIL	MSP-PIL	G3-FI	PICT-FI	BSDF
	FITS-PIL	PIXAR-PIL	HDR-FI	PNG-FI	DICOM
	FLI-PIL	SPIDER-PIL	IFF-FI	PPM-FI	NPZ
	FPX-PIL	SUN-PIL	J2K-FI	PPMRAW-FI	FEI
	FTEX-PIL	TGA-PIL	JNG-FI	PSD-FI	FITS
	GBR-PIL	TIFF-PIL	JP2-FI	RAS-FI	ITK
	GRIB-PIL	WMF-PIL	JPEG-FI	RAW-FI	GDAL
	HDF5-PIL	XBM-PIL	JPEG-XR-FI	SGI-FI	LYTRO-LFR
	ICNS-PIL	XPM-PIL	KOALA-FI	TARGA-FI	LYTRO-ILLUM-RAW
	IM-PIL	XVTHUMB-PIL	PBM-FI	TIFF-FI	LYTRO-LFP
	IMT-PIL	SCREENGRAB	PBMRAW-FI	WBMP-FI	LYTRO-F01-RAW
	IPTC-PIL	CLIPBOARDGRAB	PCD-FI	WEBP-FI	SPE

Ergänzend zur vorherigen Abbildung ist darauf hinzuweisen, dass die Bezeichnung „gdal“ bei den Importmöglichkeiten dank scikit-image die GDAL-Bibliothek repräsentiert, die allein 250 Raster- und Vektor-Formate beherrscht [GDAL].

7. Leistungsdifferenzen

In diesem Kapitel werden einerseits die jeweiligen Fähigkeiten der drei Vergleichskandidaten bezüglich der Nutzung von CPUs, GPUs und dedizierter Hardware erläutert. Bei Nutzungsfähigkeit von GPUs findet sich in den jeweiligen Unterabschnitten die Empfehlung einer Grafikkarte, die den jeweiligen Leistungskriterien entspricht und auch den Preiseinstieg in der jeweiligen Kategorie darstellt.

Andererseits wird im zweiten Abschnitt des Kapitels auch der Umgang der Lösungen mit Datensätzen verschiedenen Ursprunges betrachtet. Dort werden Bilder, Texte und Daten als Quellmaterial genutzt und dann mit diesen Vorhersagemodelle trainiert.

7.1 Hardwarenutzung

Alle betrachteten Kandidaten sind in der Lage, mehrere Kerne einer CPU parallel zu nutzen. Es ist möglich, diese Parallelität durch den Einsatz von GPUs nochmals deutlich zu steigern. Alle besprochenen Lösungen arbeiten mit dem von NVIDIA erdachten CUDA Programmiermodell - weswegen die Karten dieses Anbieters empfohlen werden [NVIDIA_CUDA]. Der Autor von Keras empfiehlt in „Deep Learning with Python“ zwar beispielsweise eine Nvidia Titan X zu nutzen [DL_WITH_PY, S. 14], aber diese Empfehlung ist zum Zeitpunkt der Erstellung dieser Arbeit bereits veraltet. Dies liegt an der Tatsache, dass deren Berechnungsfähigkeiten für einen der Vergleichskandidaten nicht ausreichend sind. Will man im Mai 2019 eine zukunftssichere GPU kaufen, empfiehlt sich eine „GeForce RTX 2080Ti“, welche in der Version vom Hersteller Palit für 1.047,03 € auf amazon.de angeboten wird [PALIT_RTX2080Ti]. Die Empfehlung basiert zuerst auf der Tatsache, dass diese GPU weniger als die Hälfte einer „NVIDIA TITAN RTX“ kostet, welche über gleichwertige CUDA Berechnungsfähigkeiten der Stufe 7.5 verfügt. An zweiter Stelle kostet die RTX etwas über ein Achtel des Kaufpreises einer nur doppelt so performanten „Nvidia Tesla V100“. Zu guter Letzt handelt es sich dabei um eine GPU, die für die Nutzung bei Computerspielen beworben wird [RTX_2080TI], was dank größerer Nachfrage eine höhere Produktionsmenge, und somit einen geringeren Verkaufspreis mit sich bringt.

Alternativ zu NIVIDIA Karten existiert mit „GPU Ocelot“ ein Projekt, welches Karten anderer Hersteller befähigt CUDA-Kernels bis zur Version 4.0 auszuführen. Details finden sich auf der seit September 2014 nicht mehr aktualisierten Seite http://gpuocelot.gatech.edu/ .

7.1.1 KNIME

In KNIME sind laut einem KNIME Teammitglied mit Zugang zur Codebasis von KNIME einige der Nodes standardmäßig dazu in der Lage, auf CPUs parallel zu arbeiten. Die Nodes,

die dies nicht beherrschen, können beispielsweise zwischen die Nodes „Parallel Chunk Start“ und „Parallel Chunk End“ platziert werden, um eine parallele Ausführung zu erhalten. Anstelle des individuellen Verpackens einzelner Nodes zwischen diese Parallelunterstützer ist es empfehlenswert, mehrere Nodes innerhalb eines Workflows zwischen die eben genannten zu platzieren [KN_MULTICPU]. Alternativ können - je nach Art des Nodes -, auch Flussvariablen genutzt werden, um paralleles Arbeiten umzusetzen. Dies zeigt beispielhaft der aus der Quelle [KN_PARALLEL] stammende Workflow auf dieser Seite. Der Workflow wurde zugunsten einer besseren Lesbarkeit um dessen unteren Teil reduziert. Dort finden sich in der Originaldarstellung weitere vier „Database SQL Executor“ Nodes. Davon werden drei, wie die abgebildeten oberen drei „Database SQL Executor“ Nodes anhand eines „Merge Variables“ Nodes wieder verbunden. Die Verbindungen des obersten abgeschnittenen Nodes zu den mittig platzierten „Database SQL Executor“ und „Merge Variables“ Nodes sind mittig am unteren Bildrand ersichtlich. Im abgebildeten Beispiel wird das Kopieren von Datenbankelementen parallelisiert. Die „Merge Variables“ Nodes fügen die Inhalte wieder zusammen.

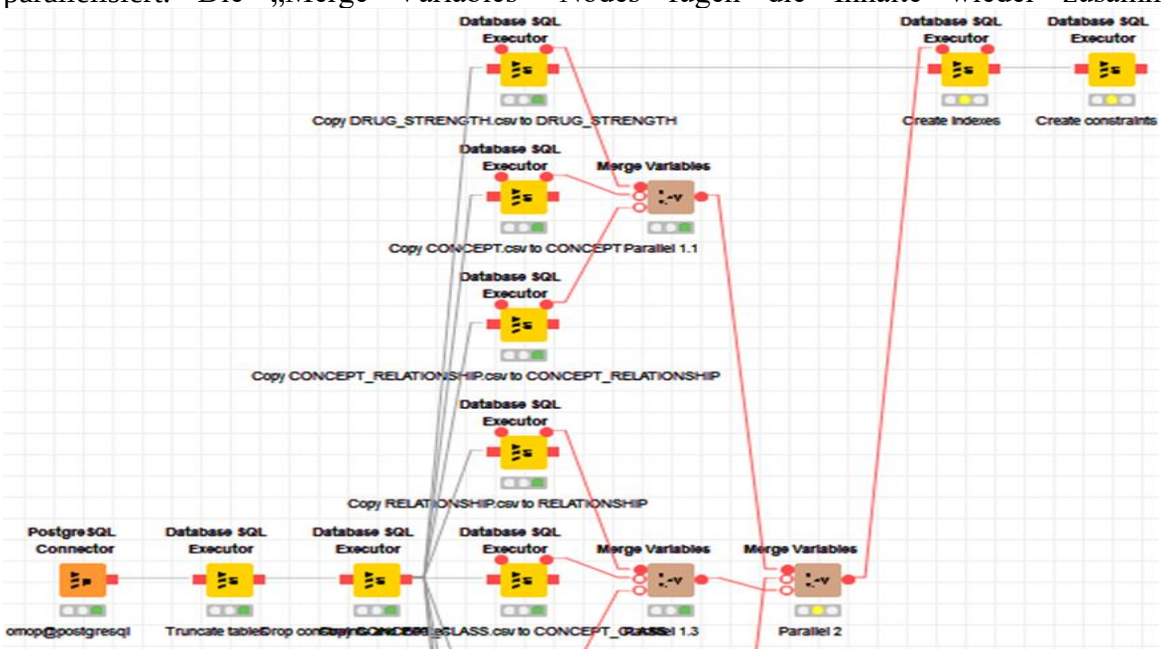


Abbildung 21: Parallelisierung in KNIME anhand von Flussvariablen (zugeschnittene Abbildung aus [KN_PARALLEL])

KNIME ist nativ nicht fähig GPUs zu nutzen. Diese Funktion erfordert die Installation der CUDA Bibliotheken und einer zusätzlichen, direkt innerhalb von KNIME installierbaren Erweiterung. Die Quelle der CUDA Bibliotheken und deren Installation sind in der Quelle dieser Information beschrieben [KNIME_GPU]. Um Zugriff zu allen installierbaren Erweiterungen innerhalb von KNIME zu erhalten ist ein Klick auf die Schaltfläche „Get additional nodes“ notwendig. Dort findet sich die benötigte Erweiterung namens „KNIME DeepLearning4J Integration“ im Verzeichnis der KNIME Labs Extensions. Die Funktionalität der Erweiterung ist in 64-Bit-Betriebssystemen mit CUDA 7.5- oder 8.0- fähigen GPUs und einer 64-Bit-Version von KNIME gegeben. Den Preiseinstieg für GPUs dieser Kategorie bildet die Gigabyte GeForce RTX 2060 OC [CUDA_GPUs] mit einem Preis von 369 € bei Amazon [RTX_2060_OC]. KNIME ist - wie auch TensorFlow - in der Lage auf mehreren Servern

parallel Berechnungen anzustellen [KNIME_CLUSTER]. Bei KNIME ist dafür allerdings die kostenpflichtige Nutzung von KNIME Server erforderlich[KNIME_SERVER].

7.1.2 TensorFlow

Die parallele CPU-Nutzung wurde in TensorFlow durch eine parallele Implementierung der einzelnen Operationen geschaffen und erfordert keine Aktivität des Nutzers.

TensorFlow kann unter Linux und Windows auch GPUs nutzen, sogar mehrere auf einem System [TF_MULTI_GPU]. Damit diese auch nutzbar sind, müssen sie idealerweise von Nvidia sein und CUDA Berechnungen der Stufe 3.5 oder höher durchführen können [TF_GPU]. Grafikkarten mit den beschriebenen Eigenschaften, beispielsweise die EVGA GeForce GTX 780, sind ab 64,89 € bei Drittanbietern auf Amazon erhältlich [GTX_780].

TensorFlow kann, wie auch KNIME, durch die Erstellung eines Clusters parallel auf mehreren Servern Berechnungen anstellen [TF_CLUSTER]. Jedoch entstehen bei TensorFlow, außer deren für die zusätzliche Hardware, keine zusätzlichen Kosten.

Ein Alleinstellungsmerkmal, welches TensorFlow von den beiden Vergleichskandidaten unterscheidet, ist die Existenz dedizierter Hardware [TPU_ARTICLE]. Diese sogenannten TPUs, welche aktuell in ihrer dritten Iteration als „TPU 3.0“ bezeichnet werden, wurden bei Google intern entwickelt. Die Nutzung dieser Einheiten ist im Augenblick nur durch Verwendung der Google Cloud möglich und grundsätzlich stundenbasiert zu bezahlen. Hierbei differieren die Preise je nach genutzter TPU-Variante, Unterbrechungsfreiheit und Region der nutzenden Person. Unterbrechungsfreiheit meint hierbei das Fehlen von für die nutzende Person willkürlich erscheinenden Unterbrechungen der Berechnung. Diese führt die Google Cloud durch, falls die Ressourcen für andere Zwecke benötigt werden. [TPU_PREEMPT]. Für den asiatisch-pazifischen Raum ist aktuell nur das ältere TPU Modell namens „TPU v2“ verfügbar, sowohl mit, als auch ohne Unterbrechungen. Zusätzlich zu dieser Tatsache sind Nutzer aus dieser Region auch diejenigen die dafür am meisten bezahlen. Der stündliche Obolus unterbrechungsfreier Nutzung beträgt dort 5,22 USD während in Europa 4,95 USD verlangt werden. Am günstigsten ist die Nutzung aller Varianten in den USA. Dort werden beispielsweise für die unterbrechungsfreie Nutzung der TPU v2 4,50 USD und für die der TPU 3.0 8,00 USD berechnet. Europäische Nutzer zahlen dafür zehn Prozent mehr [TPU_PRICE].

Ausgenommen von der Bezahlpflicht sind Mitglieder des TensorFlow Research Cloud Programms, kurz TFRC, sowie die Nutzung per Google Colab [TPU_TFRC]. Um die TFRC nutzen zu können, bedarf es der Beantwortung von neun Fragen bezüglich der voraussichtlichen Nutzung der TPUs. Des Weiteren werden neben dem eigenen Vor- und Zunamen auch eine E-Mail-Adresse und der Name der Organisation, sowie das Land als Zwangseingabe abgefragt. Diese Details sind im Zuge einer Bewerbung unter https://services.google.com/fb/forms/tpusignup/ notwendig, um eventuell Zugang zu erhalten. Google erwartet von den Teilnehmern des Programms neben der Veröffentlichung ihrer Forschungsergebnisse auch Feedback.

Eine kostenfreie TPU Nutzung per Google Colab ist auf zehn definierte Lernbeispiele, welche unter https://cloud.google.com/tpu/docs/colabs einsehbar sind, beschränkt. Deswegen scheint diese für Experimente von Studenten und Neulingen interessant zu sein. Die einzige existente Hürde hierfür ist das Vorhandensein eines Google-Kontos.

7.1.3 Scikit-learn

Scikit-learn ist anhand der Nutzung der Klasse „joblib.Parallel“ in der Lage CPUs parallel zu nutzen [SK_MULTICPU], wird aber laut Aussage der Entwickler zumindest in der nahen Zukunft nativ keine GPUs unterstützen. Diese Entscheidung begründen sie mit aus der Nutzung von GPUs resultierenden Softwareabhängigkeiten und plattformspezifischen Problemen [SCIKIT_FAQs]. Das Team hinter scikit-learn verweist lediglich auf Theano im Kontext der GPU-Nutzung [SCIKIT_GPU]. Mit H2O4GPU existiert eine auf der scikit-learn Python API basierende Möglichkeit der GPU-Nutzung. Dieses Python Paket ist auf GitHub für Systeme mit Linux verfügbar. Für dessen Funktionalität müssen auf dem Linux-System CUDA mit Display-Treibern, die GNU-C-Bibliothek ab der Version 2.17, Python Bibliotheken und eine passende GPU vorhanden sein. Die GPU muss generell mindestens CUDA Berechnungen der Stufe 3.5 beherrschen. Um komplexere Anwendungen, wie beispielsweise mehr als 2.097.152 Zeilen in k-means, zu nutzen, ist mindestens eine Berechnungsfähigkeit der Stufe 5.2 erforderlich [SK_H2O4GPU]. Die aktuell günstigste GPU-Lösung für alle möglichen Anwendungen stellt die Gigabyte GeForce GTX 1050 GV-N1050D5 dar. Diese wird ab 126,99 € bei Amazon angeboten [GTX_1050].

7.2 Leistungsvergleich

Für die Entscheidung, welcher der drei Kandidaten die individuell richtige Lösung darstellt, ist neben den bisher geführten Vergleichen auch die Anwendung in der Praxis relevant. Deshalb wird in den folgenden Unterabschnitten das Trainieren eines Modells innerhalb der Lösungen mit Daten unterschiedlichen Ursprungs geschildert. Die nachfolgend beschriebenen Beobachtungen ergaben sich bei der Nutzung unter Windows auf dem unter dem Punkt 1.3 beschriebenen System. Im Zuge des Vergleiches war auch angedacht, die Kandidaten unter Zuhilfenahme der als frei nutzbar dargestellten Plattform FloydHub zu analysieren. TensorFlow ist der einzige Vergleichskandidat welcher ab dessen Version 0.12 von FloydHub als eigenständige Entwicklungsumgebung angeboten wird. Am 29.04.2019 reicht die Unterstützung bis hin zur nicht mehr aktuellsten Version 1.12 von TensorFlow. In allen Entwicklungsumgebungen findet sich auch scikit-learn [FH_SUPPORTED], über dessen Version der Autor keine Aussage tätigen kann. FloydHub erwartet von den Personen, die es nutzen wollen, eine als Verifizierung bezeichnete Hinterlegung von Finanzdaten. Dieser Verifizierungsprozess erfordert explizit die Verknüpfung mit einer Kreditkarte. Das Hinterlegen eines PayPal Kontos reicht nicht aus. Diese Verifizierung ist nicht nur die Grundvoraussetzung, um die versprochenen zwanzig Stunden CPU-Zeit [FLOYD_PLAN] nutzen zu können, sondern auch erforderlich, um überhaupt Projekte erstellen zu können.

Das Löschen eines Kontos bei FloydHub erfordert nach Eingabe des Nutzernamens im Untermenü "Billing" der persönlichen Einstellungen nur noch einen Klick auf die Schaltfläche

"Delete Data!". Die getroffenen Aussagen belegen die angefertigten Bildschirmfotos dazu. Diese befinden sich auf dem mitgelieferten Datenträger im Unterordner "FLOYDHUB_PROOF".

7.2.1 Leistungsvergleich anhand von Bildern

Bei dem in der Arbeit genutzten Vergleichsmaterial handelt es sich zum einen um den zur Erkennung von händisch geschriebenen Zahlen produzierten MNIST-Datensatz. Da dieser häufig in der Literatur Verwendung findet, wird er auch mehrfach als das „Hello World“ des maschinellen Lernens bezeichnet. [DIGIT_REC; CNN_OVERVIEW].

Zum anderen kommt bei funktionierender Bildanalyse der Fashion-MNIST Datensatz zur Verwendung, welcher Abbildungen von Kleidungsstücken aus zehn Kategorien enthält. Die zu trainierenden Modelle sollen nach Lernen anhand der bezeichneten Trainingsdaten Vorhersagen auf die Testdaten abgeben und diese somit einer von zehn Kategorien zuordnen.

7.2.1.1 KNIME

Nach Eingabe des Suchbegriffs „MNIST“ im Dateiexplorer finden sich in den herunterladbaren Beispielen vier Workflows, die mit dem Datensatz in Verbindung stehen. Bei dem folgend besprochenen Workflow handelt es sich um den als „03_Train_MNIST_classifier“ bezeichneten im Ordner „03_TensorFlow“. Dieser Ordner wiederum findet sich im Unterordner „14_DeepLearning“ des Ordners „04_Analytics“.

Um aus den Daten des MNIST Datensatzes ein Modell zu generieren, bedarf es zuerst der Installation dreier Erweiterungen. Deren Download und Installation gestaltet sich in KNIME einfach und geschieht anhand des Klickens auf vier Schaltflächen und eine Optionssetzung zur Zustimmung zu den Lizenzbedingungen. Nach einem Neustart von KNIME ist unter Windows, falls es noch nicht installiert ist, Anaconda zu installieren. Dieses ist erforderlich, da KNIME, um das Modell zu generieren eine lokale Python Installation inklusive TensorFlow benötigt. Der notwendige Ablauf wird unter 4.1.2.1.1 beschrieben. Ist diese Voraussetzung erfüllt ist eine neue Entwicklungsumgebung anhand einer in der Quelle verknüpften Konfigurationsdatei zu erstellen [KNIME_PY]. Diese findet sich auch auf dem mitgelieferten Datenträger im Ordner „KNIME_CONDA“. Liegt dieser Ordner im Verzeichnis C, führt die Eingabe des nachfolgenden Codes in ein „Anaconda Prompt“ zur Erstellung der Umgebung. Das „Anaconda Prompt“ findet sich durch Eingabe der Zeichenkette „anaconda“ in die Windows-Suche.

```
conda env create -f C:/KNIME_CONDA/py36_knime.yml
```

Die Erstellung der Umgebung inklusive Download der erforderlichen Elemente, benötigt vier Minuten und sechs Sekunden. Die bisherigen Schritte waren neben Windows auch für Mac und Linux deckungsgleich nutzbar, bei der Erstellung eines Startskriptes besteht die Überschneidung nur noch für Mac und Linux.Das folgende Windows-Skript ist als Inhalt einer .bat Datei zu speichern. Der Autor hat es im Pfad der oben erstellten Datei abgelegt.

```
@SET PATH= C:\Users\MachineLearner\Anaconda3\Scripts;%PATH%
```

```
@CALL activate py36_knime || ECHO Activating python environment
failed
@python %*
```

Nun ist in KNIME auf das erstellte Skript zu verweisen dieses wird durch Klicken auf die Schaltflächen „File“ und „Preferences“ mit anschließendem Erweitern der Optionen bei „KNIME“ gestartet. Weitergeführt wird der Eintragungsprozess nach Klick auf „Python“ unterhalb der Auswahlpunkte bei „KNIME“ und Eintrag des Pfades in die, Python 3 zugeordnete, mittig platzierte weiße Fläche. Nach Klick auf „Apply and Close“ ist der Eintrag hinterlegt. Wird anschließend in die KNIME Umgebung, die automatisch wie die bat Datei benannt wird, noch TensorFlow installiert, könnte ein Model aus den MNIST Daten gewonnen werden. Der Konjunktiv wird genutzt, da ein Ausführen des anfangs bezeichneten Workflows nach einigen Warnmeldungen mit der Fehlermeldung eines Ausführungsfehlers von „org/tensorflow/TensorFlow/“ abgebrochen wird. Der Workflow im Unterordner „Keras“ bricht an der Stelle des „Keras Network Learner“ Nodes ab. Eine vorherige Installation von Keras in die Anaconda Umgebung von KNIME wurde selbstverständlich im Vorfeld erledigt. Beide Workflows nutzen 10.000 Trainings- und 1.500 Testbilder, wie eine genauere Analyse des Metanodes „Prepare training data“ beweist. Metanodes kapseln mehrere einzelne Nodes und helfen somit einen Workflow grafisch einfacher darzustellen. Im besprochen Metanode beider Workflows findet sich unter anderem der Node „List Files“, welcher die Bilder herunterlädt und nach seiner Ausführung per Rechtsklick und Auswahl von „File List“ auch den lokalen Pfad der Bilder übergeben kann.

Auch der letzte Versuch, mit zur Verfügung gestellten Mitteln der KNIME AG ein Modell anhand der MNIST Bilder zu erstellen, scheitert. Bei der Nutzung des im Ordner „KNIME_CONDA“ des abgegebenen Datenträgers abgelegten Workflows, stellt sich der Umgang mit fehlenden Werten des Nodes „Missing Value“ als Problem heraus [KN_MNIST_DL4J]. Das Problem besteht darin, dass die vom Node genutzten Methoden in PMML 4.2 laut einer Fehlermeldung nicht abbildbar sind. In diesem Workflow wird vom Node „HTTP Connection“ innerhalb des Metanodes „Download dataset and convert to CSV“ eine Verbindung mit http://yann.lecun.com/ initiiert, um den folgenden Nodes den Download des gesamten MNIST-Datensatzes zu ermöglichen. Dies belegt die Abbildung unter A2.

7.2.1.2 TensorFlow

Die erste Zeile des folgenden Codes importiert TensorFlow. Der Import des hinterlegten Keras MNIST Beispiels wird anhand der zweiten Codezeile bewerkstelligt.

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
```

```
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Die dritte Zeile ordnet die enthaltenen NumPy-Felder den Trainings- und Testdaten zu. Dabei enthält die Variable „x_train“ anschließend Daten zu 60.000 Testbildern handgeschriebener Ziffern von null bis neun in einer Auflösung von achtundzwanzig mal achtundzwanzig Bildpunkten. Daten zu identisch aufgelösten 10.000 Testbildern werden der Variable „x_test“ zugeordnet. In den anderen Variablen befinden sich die numerischen Zuordnungen zu den Trainings- und Test-Daten. Da die Pixel der Datensätze nach Download noch mit Werten zwischen null und 255 wiedergegeben werden, sind diese mit dem Teilen durch 255 in Zeile vier zu normalisieren, um später mit der Aktivierungsfunktion RELU genutzt werden zu können.

Der folgende Block gibt das Konfigurieren der Schichten des CNNs wieder. Die erste Zeile besagt, dass es sich um ein linear gestapeltes Modell handelt. Die zweite Zeile des Blocks wandelt das zweidimensionale Feld der Bildpunkte in ein eindimensionales Feld um. Die verbleibenden drei Zeilen definieren den weiteren Verlauf des Schichtenstapels. Die nächste Schicht besteht aus 512 voll verbundenen Neuronen. Auf diese folgt eine Schicht, die vor dem zu starken Anpassen an die Testdaten schützen soll. Dazu werden im Beispiel zwanzig Prozent der in dieser Schicht ankommenden Werte zufällig auf null gesetzt. Die letzte Schicht ist eine zehn Knotenpunkte umfassende Softmax Schicht, die ein Feld mit zehn Wahrscheinlichkeiten erstellt. Die Summe dieser Wahrscheinlichkeiten ergibt eins Die Felder entsprechen dabei den zehn Klassen der zehn Ziffern des MNIST Datensatzes.

Der dritte Abschnitt sorgt für das Kompilieren des Modells, dabei werden von oben nach unten den Zeilen folgend:

- der stochastische Optimierungsalgorithmus ADAM [ADAM] genutzt
- die dünnbesetzte Kreuzentropie („sparse_categorical_crossentropy“) als Verlustfunktion gewählt
- als Messwert die Häufigkeit der richtig vorhergesagten Zuordnungen bestimmt.

Die vorletzte Zeile weist an, über fünf sogenannte „Epochen“ anhand der Testdaten zu trainieren. Dabei bezeichnet der Begriff in Anführungszeichen die Anzahl an Durchläufen die dem gesamten Trainingsdatenset widerfahren soll. Die letzte Zeile sorgt für einen Testlauf des trainierten Modells anhand der vorher definierten Testdaten. Wird der angegebene Code in einem Jupyter Notebook ausgeführt ist - bei bereits heruntergeladenen Daten - nach einer Minute und fünfzig Sekunden ein Modell trainiert, welches auf den 10.000 Testbildern eine

Vorhersagegenauigkeit von fast achtundneunzig Prozent erzielt. Während der Durchläufe wird der nutzenden Person der Fortschritt minimalistisch dargestellt, die Darstellung nach fertigem Durchlauf zeigt die Abbildung auf der nächsten Seite .

```
Epoch 1/5
60000/60000 [==============================] - 21s 352us/sample - loss: 0.2201 - acc: 0.9351
Epoch 2/5
60000/60000 [==============================] - 21s 344us/sample - loss: 0.0968 - acc: 0.9707
Epoch 3/5
60000/60000 [==============================] - 21s 344us/sample - loss: 0.0689 - acc: 0.9785
Epoch 4/5
60000/60000 [==============================] - 21s 346us/sample - loss: 0.0517 - acc: 0.9836
Epoch 5/5
60000/60000 [==============================] - 21s 344us/sample - loss: 0.0424 - acc: 0.9858
10000/10000 [==============================] - 1s 132us/sample - loss: 0.0721 - acc: 0.9772

[0.07209290307170013, 0.9772]
```

Abbildung 22: Ausgabe durch TensorFlow nach Training anhand des MNIST Datensatzes (Selbst erstelltes Bildschirmfoto)

Wird die zweite Zeile des Beispielcodes [TF_MNIST] um die Zeichenkette „fashion_“ vor der Buchstabenfolge „mnist“ ergänzt, so werden statt der Bilder handgeschriebener Zahlen die Bilder von Kleidungsstücken des Onlinehändlers Zalando als Basis genutzt. Diese sind so aufbereitet, dass sie zehn Kategorien entstammen und in gleicher Auflösung vorliegen wie der ursprüngliche MNIST Datensatz [FASHION_MNIST]. Auf diesem Datensatz erzielt das konfigurierte Netz, bei bereits heruntergeladenen Daten, ebenfalls nach einer Minute und fünfzig Sekunden eine Zuordnungsgenauigkeit von 87,08 %.

7.2.1.3 Scikit-learn

Im folgenden scikit-learn Beispiel dient die erste Zeile der Nutzbarmachung von Datensätzen der Seite https://www.openml.org, die zweite macht die Nutzung eines neuronalen Netzes möglich.

```
from sklearn.datasets import fetch_openml
from sklearn.neural_network import MLPClassifier

X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
X = X / 255.

X_train, X_test = X[:60000], X[10000:]
y_train, y_test = y[:60000], y[10000:]

print('Stoppuhrstart für Chancengleichheit')
mlp = MLPClassifier(hidden_layer_sizes=(512,), activation='relu',
max_iter=5, solver='adam', verbose=1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```

In der dritten Codezeile findet neben dem Download des Datensatzes auch die Zuordnung dessen Inhalts zu den Variablen X und y statt. Dabei finden sich in X die Werte der Bildpunkte der einzelnen Bilder, y werden die numerischen Werte der codierten Bilder des Datensatzes zugeordnet. Die vierte Zeile skaliert die Werte der Pixel auf solche zwischen null und eins, da diese bis dahin Werte zwischen null und 255 hatten. Das nächste Zeilenduo teilt die Daten in 60.000 Trainings- und 10.000 Test-Bilder, dabei steht die obere Zeile für die Werte der Bildpunkte und die untere für die Zuordnungen.

Die siebte Zeile weist den Autor durch Ausgabe der Zeichenkette darauf hin, dass jetzt die Stoppuhr zu starten sei um einen fairen Vergleich zu schaffen. Dies ist deshalb nötig, da die Ausgabe von scikit-learn erst nach der ersten Iteration angezeigt wird. Einen bildlichen Eindruck liefert die Grafik am Ende dieses Unterunterabschnittes. Die darauffolgenden Zeilen definieren den „Mulit-layer Perceptron classifier“, dabei wurde der Beispielcode der Dokumentation [SCIKIT, S. 1285] so adaptiert, dass er dem von TensorFlow ähnelt. Hierbei bewirken die Zuordnungen innerhalb der Klammern gemäß ihrer Reihenfolge:

- die Erstellung einer verdeckten Schicht mit 512 Neuronen
- die Nutzung von RELU als Aktivierungsfunktion
- das Setzen der Höchstgrenze an Iterationen auf fünf
- die Nutzung von ADAM als Optimierungsalgorithmus
- die Ausgabe des Fortschrittes während des Trainings,

Die ersten sieben Zeichen der zehnten Zeile entsprechen der Trainingsanweisung, die Klammer umschließt die Daten anhand derer trainiert werden soll.

Die Ausgabe der durchschnittlichen Zuordnungsgenauigkeit des Modells mit den Trainingsdaten bewirkt die vorletzte Zeile. Die letzte Zeile gibt die Zuordnungsgenauigkeit bei den Testdaten aus. Die folgende Grafik zeigt die Ausgabe des Skripts, welches nach einer Minute und zweiundzwanzig Sekunden ein Modell mit höherer Zuordnungsgenauigkeit als das in TensorFlow generierte erzielt.

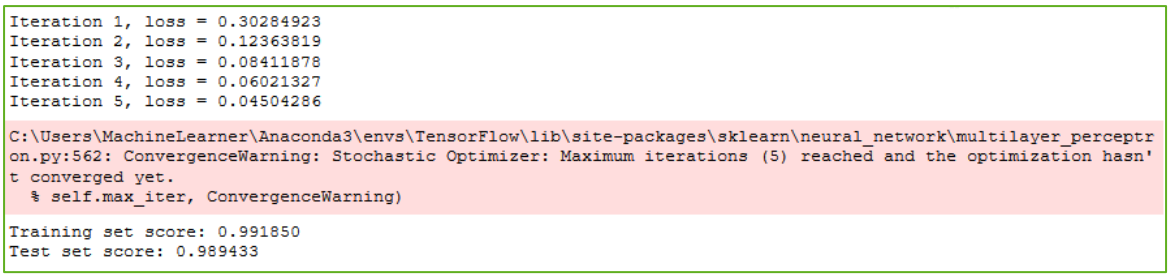


Abbildung 23: Ausgabe durch scikit-learn nach Training anhand des MNIST Datensatzes (Selbst erstelltes Bildschirmfoto)

Da auf https://www.openml.org auch der Fashion-MNIST Datensatz vorgehalten wird reicht in der dritten Zeile das Ändern der Zeichenkette „mnist_784“zu „Fashion-MNIST“, um mit dessen Daten zu trainieren. Damit erreicht das Modell in scikit-learn nach einer Minute und

zweiundzwanzig Sekunden eine Zuordnungsgenauigkeit von 89,65 % auf die Trainingsdaten und eine Zuordnungsgenauigkeit von 89,31 % auf die Testdaten.

7.2.2 Leistungsvergleich anhand von Texten

Um die Kandidaten bezüglich des Umgangs mit Texten zu vergleichen, wurde der Datensatz der zwanzig Usenet Nachrichtenforen ausgewählt. Dabei handelt es sich um einen Datensatz der, laut den Informationen auf http://qwone.com/~jason/20Newsgroups/, ursprünglich von Ken Lang erstellt worden sein soll. Dieser ist sowohl auf der genannten Seite, als auch in den Musterdaten von scikit-learn verfügbar. Das Ziel in diesem Anwendungsfall ist die Zuordnung der Testdaten zu einem der bekannten Nachrichtenforen. Diese wurden zur Simplifizierung auf drei reduziert: „alt.atheism“, „sci.crypt2“ und „comp.graphics“.

7.2.2.1 KNIME

Für Training und Test des Modells in KNIME wurde das gzip Archiv „20news-bydate.tar.gz“ der oben genannten Quelle genutzt. Nach Extraktion der Daten aus diesem wurden nur die Inhalte der bereits benannten Nachrichtenforen in den Trainings- und Testdaten belassen und an alle Dokumente mit der Software „Total Commander“ die Endung „.txt“ angehängt. Dies geschah zur Auslesbarkeit der Daten in KNIME mit dem Node „Tika Parser URL Input“, da dieser die Dateien ohne die Endung nicht auslesen konnte.

Um in KNIME Texte wie nachfolgend abgebildet zu klassifizieren ist zuerst die Installation der sogenannten „Palladian“ Nodes erforderlich. Diese finden sich anhand der Suche nach der Zeichenkette unter den zusätzlichen Nodes in „KNIME Community Contributions – Other“. Unter den in „Palladian“ enthaltenen Nodes befindet sich neben den Nodes zum Erlernen und Beschneiden des Modells auch der zur Vorhersage genutzte Node. Diese Nodes wurden in der nachfolgenden Grafik zur schnelleren Auffindbarkeit grün eingerahmt.

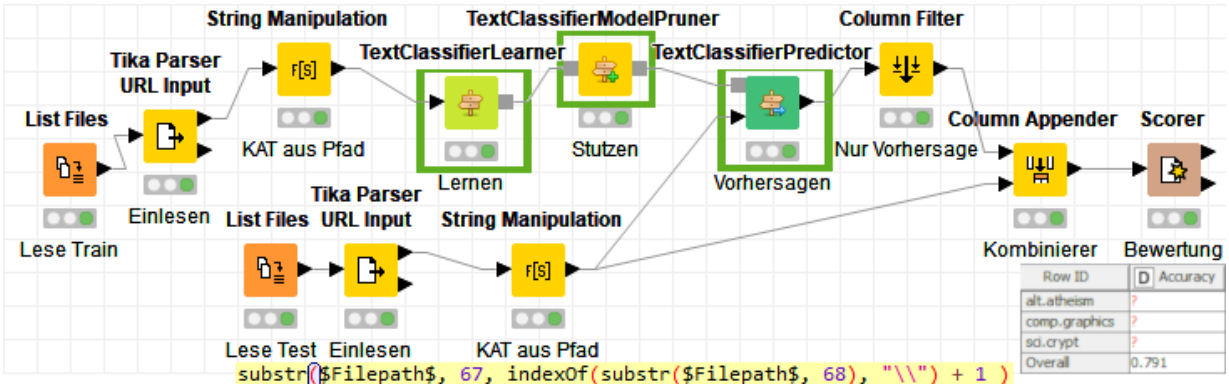


Abbildung 24: Workflow zur Klassifikation der Texte des Datensatzes der Nachrichtenforen in KNIME (selbst erstelltes Bildschirmfoto)

Der in der Grafik links oben befindliche „List Files“ Node erstellt eine Liste der URLs der Dateien innerhalb der Dateipfade der Trainingsdaten, der untere die der Testdaten. Diese Listen geben sie an die danach platzierten „Tika Parser URL Input“ Nodes weiter, welche den Inhalt der Dateien, deren Autoren und Dateipfade zeilenweise in eine Tabelle einlesen.

Die Inhalte der erhaltenen Tabellen nutzen die folgenden „String Manipulation“ Nodes, sie beschneiden die Dateipfade, um daraus die Kategorisierung abzuleiten. Der Code dafür ist der in der Abbildung gelb hinterlegte.

Die anhand der bisherigen Schritte bearbeiteten Trainingsdaten werden dem Node zur Erlernung des Modells übergeben, dessen Ausgabe wiederum vom „TextClassifierModelPruner“ bearbeitet wird. Die hier einstellbaren Optionen sind die minimale Häufigkeit von Termen, sowie der minimale Informationsgewinn daraus. Der erste Wert wurde zur Erreichung der oben abgebildeten Genauigkeit von 79,1% auf 2 und der untere auf 0,001 gesetzt. Um die Genauigkeit durch den „Scorer“ Node berechnen lassen zu können, werden zuerst durch den „Column Filter“ die Vorhersagen auf die getroffene Zuordnung reduziert. Diese werden mit der Ausgabe des unteren „String Manipulation“ Nodes durch den „Column Appender“ kombiniert. Die Ausführung des gesamten Workflows dauert circa vierunddreißig Sekunden.

7.2.2.2 TensorFlow

Der für TensorFlow recherchierte Code zeigt einen Vergleich ohne die Nutzung von Keras und gibt somit einen tieferen Einblick in die Nutzung von TensorFlow. Er wurde von Juan Carlos Sastre auf der Software-Versionsverwaltungsplattform GitHub [TF_NEWS_GIT] veröffentlicht und vom Autor dieser Arbeit angepasst. Die Anpassungen umfassen neben der Tilgung von Kommentaren und Testausgaben die Ersetzung der Nachrichtenforen „rec.sport.baseball“ und „sci.space“ durch „alt.atheism“ und „sci.crypt“. Der Code wird aufgrund seines Umfanges nicht im Ganzen, sondern auf eine Beschreibung wie folgend aufgeteilt wiedergegeben. Die ersten vier Zeilen sorgen für den Import des Musterdatensatzes von scikit-learn, welcher auf die eingangs beschriebenen Foren mit der zweiten Zeile reduziert wird.

```
from sklearn.datasets import fetch_20newsgroups
categories = ['comp.graphics','alt.atheism','sci.crypt']
newsgroups_train = fetch_20newsgroups(subset='train',
categories=categories)
newsgroups_test = fetch_20newsgroups(subset='test',
categories=categories)
```

Die eben den mit “newsgroups_” beginnenden Variablen zugeordneten Datensätze werden anhand der folgenden beiden Schleifen bearbeitet. Durch diese Bearbeitung werden die Zeichenketten innerhalb der Datensätze anhand der Leerzeichen separiert und die jeweilige Anzahl an Vorkommnissen anhand des importierten Zählwerkzeuges mitgezählt. In der Mitte des Blocks wird zuerst die Gesamtmenge an Wörtern der Variable „total_words“ zugewiesen, bevor die Zeichenketten anhand ihrer Häufigkeit von der Funktion „get_word_2_index“ sortiert und dergestalt der Variablen „word2index“ zugewiesen werden.

```
from collections import Counter
vocab = Counter()
for text in newsgroups_train.data:
    for word in text.split(' '):
        vocab[word.lower()]+=1
```

```
for text in newsgroups_test.data:
    for word in text.split(' '):
        vocab[word.lower()]+=1
total_words = len(vocab)

def get_word_2_index(vocab):
    word2index = {}
    for i,word in enumerate(vocab):
        word2index[word.lower()] = i
    return word2index
word2index = get_word_2_index(vocab)
```

Der nächste Block zeigt bei einem Vergleich mit dem Code der MNIST Analyse klar die Vorteile der Nutzung von Keras auf. Dort benötigt die Definition einer Schicht des neuronalen Netzes eine Zeile Code. Im folgenden Codeblock steht die Anzahl an Rauten in der Zeile für die Netzschicht, die sie repräsentieren. Dabei stellt der gesamte Block eine Funktion dar, die zwei versteckte Schichten sowie eine Ausgabeschicht definiert.

```
def multilayer_perceptron(input_tensor, weights, biases):
    layer_1_multiplication = tf.matmul(input_tensor,
weights['h1'])                                          #
    layer_1_addition = tf.add(layer_1_multiplication,
biases['b1'])                                           #
    layer_1_activation = tf.nn.relu(layer_1_addition)  #
    layer_2_multiplication = tf.matmul(layer_1_activation,
weights['h2'])                                           ##
    layer_2_addition = tf.add(layer_2_multiplication,
biases['b2'])                                            ##
    layer_2_activation = tf.nn.relu(layer_2_addition)   ##
    out_layer_multiplication = tf.matmul(layer_2_activation,
weights['out'])
    out_layer_addition = out_layer_multiplication + biases['out']
    return out_layer_addition
```

Anschließend erstellt Sastre eine Funktion zur Aufteilung der Datensätze, welche er später für die eigentliche Lernphase nutzen wird. Da diese die Daten in NumPy Felder schreibt, wird dieses vor der Definition der Funktion importiert.

```
import numpy as np
def get_batch(df,i,batch_size):
    batches = []
    results = []
    texts = df.data[i*batch_size:i*batch_size+batch_size]
    categories = df.target[i*batch_size:i*batch_size+batch_size]
    for text in texts:
        layer = np.zeros(total_words,dtype=float)
        for word in text.split(' '):
```

```
                layer[word2index[word.lower()]] += 1
            batches.append(layer)
        for category in categories:
            y = np.zeros((3),dtype=float)
            if category == 0:
                y[0] = 1.
            elif category == 1:
                y[1] = 1.
            else:
                y[2] = 1.
            results.append(y)
        return np.array(batches),np.array(results)
```

Auf diesen Schritt folgt das Setzen verschiedener Parameter wie beispielsweise Lernrate, Anzahl der Epochen, Größe der Datenbündel und Anzahl der Klassen. Die dritte und vierte Zeile gibt die Definition des Ein- und Ausgabetensors wieder. Die letzten Zeilen geben die Gewichtung und Maßabweichungen vor.

```
learning_rate = 0.01 ; training_epochs = 10 ; batch_size = 150 ;
display_step = 1
n_hidden_1 = 100 ; n_hidden_2 = 100; n_input = total_words ;
n_classes = 3
input_tensor = tf.placeholder(tf.float32,[None,
n_input],name="input")
output_tensor = tf.placeholder(tf.float32,[None,
n_classes],name="output")
weights = {
    'h1': tf.Variable(tf.random_normal([n_input, n_hidden_1])),
    'h2': tf.Variable(tf.random_normal([n_hidden_1,
n_hidden_2])),
    'out': tf.Variable(tf.random_normal([n_hidden_2,
n_classes]))}
biases = {
    'b1': tf.Variable(tf.random_normal([n_hidden_1])),
    'b2': tf.Variable(tf.random_normal([n_hidden_2])),
    'out': tf.Variable(tf.random_normal([n_classes]))}
```

Nachdem diese Schritte unternommen wurden, werden im folgenden Codeblock von oben nach unten folgende Schritte abgearbeitet: das Modell definiert, Verlust- und Kostenfunktion definiert, die Optimierungsfunktion gewählt und die Variablen initiiert. Der Ablauf des eigentlichen Lernprozesses startet mit der Zeile „sess.run(init)“, da diese den Graphen startet. Die erste mit „for“ beginnende Zeile kennzeichnet den Beginn der Programmierschleife über die Anzahl an Trainingsepochen. Die letzte Zeile sorgt für die Ausgabe des Verlustes je Epoche.

```
prediction = multilayer_perceptron(input_tensor, weights, biases)
```

```
entropy_loss =
tf.nn.softmax_cross_entropy_with_logits(logits=prediction,
labels=output_tensor)
loss = tf.reduce_mean(entropy_loss)
optimizer =
tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss
)
init = tf.global_variables_initializer()
with tf.Session() as sess:
  sess.run(init)
  for epoch in range(training_epochs):
    avg_cost = 0.
    total_batch = int(len(newsgroups_train.data)/batch_size)
    for i in range(total_batch):
    batch_x, batch_y = get_batch(newsgroups_train, i ,batch_size)
    c,_ = sess.run([loss,optimizer], feed_dict={input_tensor:
batch_x, output_tensor:batch_y})
    avg_cost += c / total_batch
     if epoch % display_step == 0:
      print("Epoch:", '%04d' % (epoch+1), "loss=",
"{:.9f}".format(avg_cost))
```

Die mit "print" beginnenden Zeilen des letzten Codeblocks verursachen einerseits die Ausgabe der Zeichenkette „Fertig optimiert" und andererseits die der Zuordnungsgenauigkeit auf die erstellten Testsätze. Diese wird von den Zeilen dazwischen berechnet.

```
    print("Fertig optimiert!")
    correct_prediction = tf.equal(tf.argmax(prediction, 1),
tf.argmax(output_tensor, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction,
"float"))
    total_test_data = len(newsgroups_test.target)
    batch_x_test,batch_y_test =
get_batch(newsgroups_test,0,total_test_data)
    Genauigkeit = accuracy.eval({input_tensor: batch_x_test,
output_tensor: batch_y_test})
    print("Accuracy:", Genauigkeit)
```

Das Ausführen des Skripts benötigt eine Minute siebenundzwanzig, da die Trainingsdaten anhand der Funktion „get_batch" in jedem Lauf neu gemischt werden schwankt die Zuordnungsgenauigkeit. Sie liegt zwischen 69,52% und 74,72%.

7.2.2.3 Scikit-learn

Der nachfolgend wiedergegebene Code stammt ursprünglich vom Nutzer „Code Wrestling", der diesen auf GitHub veröffentlicht hat [SK_NEWS_GIT]. Der Autor hat diesen zum besseren

Vergleich mit den anderen Lösungen in der Auswahl der Nachrichtenforen angepasst, sowie zur besseren Lesbarkeit die Importschritte gruppiert.

```
import numpy as np
from sklearn.datasets import fetch_20newsgroups
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB

categories = ['alt.atheism','sci.crypt', 'comp.graphics']
news_train = fetch_20newsgroups(subset='train',categories=
categories)
news_test = fetch_20newsgroups(subset='test',categories=
categories)

text_clf = Pipeline([('vect', TfidfVectorizer()), ('clf',
MultinomialNB()) ])

text_clf.fit(news_train.data, news_train.target)
predicted = text_clf.predict(news_test.data)

print('Accuracy achieved is ' + str(np.mean(predicted ==
news_test.target)))
```

Die erste Zeile importiert NumPy, die darauf folgenden vier verschiedene Funktionen von scikit-learn. Die sechste Zeile legt die Auswahl der zu nutzenden Nachrichtenforen fest, welche sich in den Zeilen sieben und acht bei der Zuweisung der Daten an die Variablen auswirkt. Dabei werden der Variablen „news_train“ die Trainingsdaten des bereits aufgeteilten Datenbestandes zugewiesen. Mit den Testdaten und der Variable „news_test“ verhält es sich identisch. Die mit „text_clf“ beginnende Zeile nutzt das Konzept von „pipelines“ innerhalb von scikit-learn. Dieser Begriff bezeichnet dort die Möglichkeit mehrere Transformationsschritte miteinander zu kombinieren. Zum Abschluss dieser Transformationen muss an diese ein sogenannter „Estimator“ angehängt werden. Diese bereits in 2.7 für TensorFlow thematisierten Objekte definiert das Team hinter scikit-learn als „jedes Objekt, das aus Daten lernt“ [SCIKIT, S. 136]. Im obigen Code wird die Transformationsmethode „TfidfVectorizer“ mit dem multinomialen Naive Bayes Klassifikator als „pipeline“ bestimmt. Die Methode entspricht einer Aneinanderreihung zweier Methoden, wobei die erste die Zeichenketten von Texten in eine Matrix verwandelt die die jeweilige Anzahl an Vorkommnissen der einzelnen Zeichenketten in den Dokumenten wiedergibt. Die zweite wandelt diese Matrix in eine normalisierte „tf-idf“ Darstellung um, wobei das Buchstabenkürzel die eingedeutschte Term-Frequenz pro invertierter Dokument-Frequenz bezeichnet. Diese Begriffsansammlung bildet demnach den Wert ab, der sich ergibt, wenn man die Häufigkeit der Zeichenkette in einem Dokument durch die Häufigkeit der Zeichenkette in allen Dokumenten teilt. Je höher dieser Wert ist, desto relevanter ist der von der Zeichenkette dargestellte Begriff.

Die zehnte Zeile nutzt die durch die zuvor definierte pipeline transformierten Trainingsdaten zur Wissensgewinnung, bevor in der nächsten Zeile das gewonnene Modell auf die transformierten Testdaten angewandt wird. Die letzte Zeile sorgt für die Ausgabe der Zuordnungsgenauigkeit des Modells, die nach vier Sekunden Berechnungszeit mit 91,84% angegeben wird.

7.2.3 Leistungsvergleich anhand von Daten

Als Datensatz für den Vergleich des Umgangs der Kandidaten mit strukturierten Daten wird der Datensatz der Bostoner Hauspreise genutzt. Dieser in 5.1.2 bereits beschriebene Datensatz umfasst 506 Instanzen mit jeweils dreizehn Attributen zu Häusern der Bostoner Vororte aus den späten 70er Jahren des letzten Jahrhunderts. Für TensorFlow und scikit-learn wird der aus scikit-learn stammende Datensatz genutzt, welcher eine Kopie des von der UCI vorgehaltenen Datensatzes darstellt [SK_IMPORT, #boston-dataset]. In KNIME findet der Datensatz der UCI Verwendung, welcher unter https://archive.ics.uci.edu/ml/machine-learning-databases/housing/ abgerufen werden kann. Das Lernziel im beschriebenen Fall ist es, den Preis eines Hauses anhand der vorhandenen Informationen vorherzusagen.

7.2.3.1 KNIME

Der Inhalt des von der UCI zur Verfügung gestellten Datensatzes ist nicht kommasepariert, sondern ist in Spalten eingeteilt, wodurch die Werte teils von mehreren Leerzeichen getrennt werden. Deswegen meldet der KNIME File Reader Node einen Fehler in Zeile 376. Aufgrund der fehlenden Spaltenüberschriften hat der Autor die Datei mit Excel bearbeitet und als .csv exportiert, um sie mit KNIME nutzbar zu machen. Basis des genutzten Workflows war ein von der KNIME AG vorgehaltener Workflow [KNIME_HOUSING], der allerdings zur Analyse ein mehrlagiges Perzeptron nutzte, weswegen dessen Nodes durch die entsprechenden Nodes der linearen Regression ersetzt wurden. Auch wurden die Einstellungen der Datenaufteilung und der linearen Regression angepasst. Die einzelnen Nodes erfüllen entlang des Ablaufs den Datenimport, die Normalisierung, die zufällige Aufteilung in Trainings- und Testdaten, den eigentlichen Lernprozess, die Vorhersage anhand des erlernten Modells auf die Testdaten, die Berechnung der Beurteilungsgüte der Vorhersagen anhand von Metriken und deren Ausgabe je Lern-Iteration.

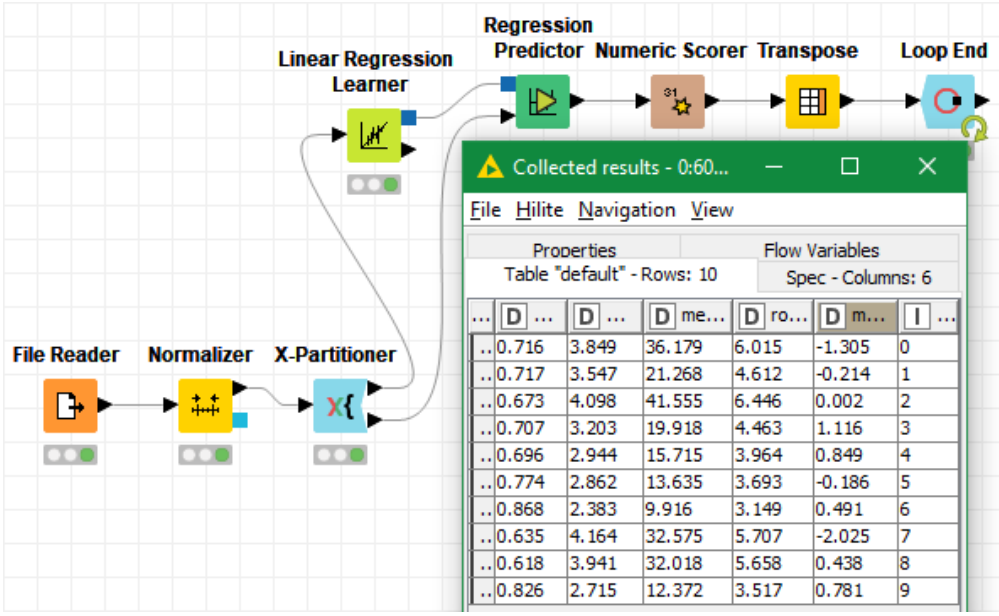


Abbildung 25: KNIME Workflow mit dem Bostoner Datensatz (selbst erstelltes Bildschirmfoto)

Bei den in der rechten unteren Ecke des Bildes angezeigten Werten handelt es sich von links nach rechts um das Bestimmtheitsmaß, den mittleren absoluten Fehler, den mittleren quadratischen Fehler, die gewurzelte mittlere quadratische Abweichung und den mittleren vorzeichenbehafteten Unterschied. Diese werden pro Lerndurchlauf angegeben Für zehn Iterationen werden circa vier Sekunden benötigt.

7.2.3.2 TensorFlow

Der nachfolgende Code wurde im Original von Giancarlo Zaccone geschrieben [TF_BOSTON_LIN]. Vom Autor dieser Arbeit wurden diverse Anpassungen vorgenommen, um einen lauffähigen und deutlich weniger Zeilen umfassenden Code zu erhalten. Aus diesen Änderungen ergibt sich der Code dieses Unterunterabschnittes, der komplett einzugeben ist, um lauffähig zu sein. Die ersten beiden Zeilen des folgenden Blocks sorgen für den Import von NumPy und TensorFlow. Mit der Eingabe der verbleibenden drei Zeilen werden aus dem Umfang von scikit-learn die Fähigkeiten zur Nutzung des Bostoner Datensatzes, zur Aufteilung der ursprünglichen Daten in Trainings- und Testdaten, sowie die zur Berechnung des Bestimmtheitsmaßes importiert

```
import numpy as np
import tensorflow as tf
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

Anschließend werden die Daten und die Verkaufspreise der Häuser den Variablen „features“ und „labels“ zugewiesen.

```
features, labels = load_boston(return_X_y=True)
```

Die nachfolgend definierte Methode dient dem Anhängen des zufälligen Fehlerterms an die normalisierten Daten.

```
def bias_vector(features,labels):
 n_training_samples = features.shape[0]
 n_dim = features.shape[1]
 f= np.reshape(np.c_[np.ones(n_training_samples),features],
[n_training_samples,n_dim + 1])
 l = np.reshape(labels,[n_training_samples,1])
 return f, l
```

Der folgende Code sorgt dafür, dass die durch ihn normalisierten Daten und die Preise von der obigen Methode um den Fehlerterm angereichert werden. Die derartig bearbeiteten Daten werden anschließend den Variablen „data“ und „label“ zugeordnet.

```
data, label = bias_vector(((features -
(np.mean(features,axis=0))) / np.std(features,axis=0)),labels)
n_dim = data.shape[1]
```

Der Code im Anschluss nutzt die von scikit-learn bereitgestellte Funktion der Aufteilung der Daten in Trainings- und Testdaten. Dabei sorgen die Parameter dafür, dass achtzig Prozent der Daten zufällig als Trainingsdaten genutzt werden.

```
train_x, test_x, train_y, test_y =
train_test_split(data,label,test_size = 0.20,random_state = 5)
```

Folgend werden die von TensorFlow zu nutzenden Datenstrukturen anhand der Zuweisung an Variablen definiert.

```
log_loss = np.empty(shape=[1],dtype=float)
X = tf.placeholder(tf.float32,[None,n_dim]); Y =
tf.placeholder(tf.float32,[None,1]); W =
tf.Variable(tf.ones([n_dim,1]))
```

Diese drei Zeilen bilden das Modellieren der linearen Regression ab. Die erste multipliziert die Werte der Matrix mit deren Gewichten, die zweite sorgt für die Berechnung der Kosten, die dritte sorgt für die Nutzung des Gradientenverfahrens zur Minimierung der quadratischen Abweichung.

```
y_ = tf.matmul(X, W)
cost_op = tf.reduce_mean(tf.square(y_ - Y))
training_step =
tf.train.GradientDescentOptimizer(0.01).minimize(cost_op)
```

Die ersten beiden Zeilen des vorletzten Codeblocks starten die eigentliche TensorFlow Sitzung und initialisieren danach alle Variablen. Die restlichen stellen den eigentlichen Trainingsprozess dar, wobei die mit „for“ beginnende Zeile eine Schleife über 10.000 Epochen auslöst, und die verbleibenden zwei das eigentliche Training abbilden.

```
sess = tf.Session()
sess.run(tf.initialize_all_variables())
for epoch in range(10000):
```

```
 sess.run(training_step,feed_dict={X:train_x,Y:train_y})
 log_loss = np.append(log_loss,sess.run(cost_op,feed_dict={X:
train_x,Y: train_y}))
```

Die letzten Codeschnipsel sorgen für die Vorhersage unter der Nutzung des eben erlernten Modells auf die Testdaten, sowie die Ausgabe der gewurzelten quadrierten mittleren Abweichung und des Bestimmtheitsmaßes.

```
pred_y = sess.run(y_, feed_dict={X: test_x})
print("Mittlere gewurzelte quadrierte Abweichung: %.4f" %
np.sqrt(int(sess.run(tf.reduce_mean(tf.square(pred_y -
test_y))))), "\n Bestimmtheitsmaß: ", r2_score(test_y, pred_y) )
```

Die Laufzeit des Skriptes beträgt fast vierzehn Sekunden bis zur Ausgabe der folgend abgebildeten erreichten Werte.

```
Mittlere gewurzelte quadrierte Abweichung: 4.4721
 Bestimmtheitsmaß:  0.7334515389400453
```

Abbildung 26: RSME und Bestimmtheitsmaß eines Prognosemodells Bostoner Hauspreise in TensorFlow (eigenes Bildschirmfoto)

7.2.3.3 Scikit-learn

Die Basis des wiedergegebenen Codes stammt von Animesh Argawal [SK_BOSTON_LIN]. Der Autor dieser Arbeit hat daraus Analyseschritte der Daten entfernt und neben den Importfunktionen auch die Berechnung der Vorhersagegenauigkeit und deren Ausgabe gruppiert. Des Weiteren hat er die vom ursprünglichen Autor vorgenommene Einschränkung der Daten auf die mit dem höchsten Zusammenhang zum Zielwert - den prozentualen Anteil an Bevölkerung mit niedrigem Status, und die Anzahl der Räume der Behausung – getilgt und nutzt somit alle vorhandenen Daten.

Um ein Vorhersagemodell anhand der Nutzung der linearen Regression mit scikit-learn zu trainieren, ist der komplette Code dieses Unterunterabschnittes einzugeben. Die folgenden Zeilen sorgen ihrer Reihenfolge nach für den Import von NumPy, pandas und verschiedenen Werkzeugen aus scikit-learn.

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_boston
from sklearn import linear_model ; lin_model =
linear_model.LinearRegression()
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
```

Dabei bewirkt die in dieser Darstellung umgebrochene vierte Zeile die Nutzung der linearen Regression. Wollte man das Modell beispielsweise anhand der Nutzung spärlicher Koeffizienten trainieren, wäre lediglich die Zeichenkette „LinearRegression()“ durch „Lasso(alpha=0.1)“ zu ersetzen. Die folgende Zeile sorgt für die Zuweisung der Daten des Bostoner Datensatzes an die Variable X und die Übergabe der Zielwerte an die Variable Y.

```
X, Y = load_boston(return_X_y=True)
```

Der folgende Befehl nutzt die Funktion „train_test_split“ aus dem Repertoire von scikit-learn, um die Daten in Trainings- und Testdaten zu teilen. Die gewählten Optionen sorgen dabei für die zufällige Nutzung von zwanzig Prozent der Daten als Testdaten, die verbleibenden ergeben die Trainingsdaten.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size = 0.2, random_state=5)
```

Die folgende Zeile sorgt für das eigentliche Training des Modells anhand der Trainingsdaten.

```
lin_model.fit(X_train, Y_train)
```

Die vorletzte Codeansammlung repräsentiert den Abgleich der tatsächlichen Werte der Trainings- und Testdaten mit den Vorhersagen des eben trainierten Modells. Dazu werden zuerst Vorhersagen auf die eben genannten Datenmengen erstellt. Für diese wird dann die gewurzelte mittlere quadratische Abweichung („rmse“) kalkuliert, und Variablen zugewiesen. Die letzten beiden Zeilen sorgen für die Zuweisung des Bestimmtheitsmaßes der jeweiligen Vorhersagen an die mit „r2“ beginnenden Variablen.

```
y_train_predict = lin_model.predict(X_train)
y_test_predict = lin_model.predict(X_test)
rmse = (np.sqrt(mean_squared_error(Y_train, y_train_predict))) ;
testse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
r2 = r2_score(Y_train, y_train_predict) ; testr2 =
r2_score(Y_test, y_test_predict)
```

Der letzte Codeblock sorgt für die Ausgabe der eben berechneten Beurteilungsmetriken.

```
print('Vorhersagegenauigkeit auf Trainings-                und
auf Testdaten\n' +
'-------------------------------------              ------------
-------------------------\n' +
'RMSE is ' + format(rmse) + "                            " +
format(testse) + '\n' +
'R2 score is ' + format(r2) + "                        " +
format(testr2))
```

Die gesamte Ausführung des Codes lässt sich bei lokalem Vorhandensein des Datensatzes nicht exakt stoppen und liegt bei circa einer Sekunde, die erzielten Ergebnisse bildet die folgende Grafik ab.

```
Vorhersagegenauigkeit auf Trainings-              und auf Testdaten
-------------------------------------             -------------------------------------
RMSE is 4.741000992236517                         4.56829204230318
R2 score is 0.7383393920590519                    0.7334492147453108
```

Abbildung 27: RSME und Bestimmtheitsmaß eines Prognosemodells Bostoner Hauspreise in scikit-learn (eigenes Bildschirmfoto)

8. Ergebnis und Ausblick

8.1 Bewertung der eingesetzten Lösungen

Wie die Arbeit im ihr möglichen Rahmen wiedergibt, handelt es sich um drei ähnliche, aber im Detail doch auch sehr unterschiedliche Lösungen. Ein behandelter Aspekt, in dem eine hohe Ähnlichkeit zwischen zwei Lösungen besteht, ist die Installation. Unter Windows sind TensorFlow und scikit-learn dank der Anaconda Python Distribution einfach zu installieren, mit einem Vorteil für scikit-learn, welches bereits mit der Installation der Distribution installiert ist. Beide kommen aber an die Einfachheit von KNIME unter Windows nicht heran. Auf potenten Systemen, die mit 64-Bit Linux Distributionen betrieben werden, ist die individuelle Interpretation von Einfachheit ausschlaggebend. Wer es als einfacher empfindet Installationsdateien manuell herunterzuladen und auszuführen, empfindet hier KNIME besser. Liegt der nutzenden Person eher das Eingeben eines Einzeilers in ein Terminal-Fenster, so liegen TensorFlow und scikit-learn auf Augenhöhe. Anders sieht die Bewertung auf dem Raspberry aus: KNIME wird hier laut Ivan Pazin, einem Moderator des KNIME-Forums mit „leader“ Status, nicht unterstützt [KNIME_RASPBERRY]. Dies musste der Autor anhand der Erprobung vieler zeitfressender, erfolgloser alternativer Lösungswege auch feststellen. Deshalb sieht der Autor hier scikit-learn aufgrund der existenten GUI und der einfacheren Programmierung vor TensorFlow.

Die für TensorFlow untersuchte GUI ist zwar optisch sehr ansprechend, da der Autor allerdings Wert auf Datensparsamkeit legt, hat er die Untersuchung des „Deep Learning Studios“ mit der Feststellung des unverschlüsselten lokalen Speicherns aller Nutzereingaben abgebrochen. Diese Feststellung alleine wäre noch tolerabel, da allerdings selbst zur lokalen Nutzung ein Onlinezwang herrscht reichte diese Kombination an Fakten zum Forschungsabbruch. Der Onlinezwang besteht darin, dass der Nutzer sich immer erst im Portal des Herstellers anmelden muss um die GUI zu nutzen. Belege für diese Äußerungen stellen die Abbildungen im Ordner „FLOYDHUB_PROOF“ des mitgelieferten Datenträgers dar. Eine Trennung des Rechners vom Internet vor Starten der Software ändert nichts an der Tatsache. Der Autor hat den Netzwerkverkehr nicht genauer untersucht, hier wäre beispielsweise die Analyse mit dem Netzwerkwerkzeug „Wireshark“ interessant, um die tatsächlich vom Deep Learning Studio übertragenen Pakete zu untersuchen.

Beim Umfang der integrierten Musterdaten ist KNIME aktuell klarer Favorit, sowohl mengenmäßig, aber auch da die Daten automatisch in einem Analyseablauf eingebunden werden und somit einen einfachen Lernprozess ermöglichen. TensorFlow holt was den Umfang anbelangt auf. Im Zeitraum von circa fünf Wochen wuchs der Umfang um sechsundzwanzig Beispieldatensätze an.

Eine numerische Auswertung bezüglich der unterstützten Programmiersprachen gewinnt KNIME vor TensorFlow mit einem Vorsprung von zwei Sprachen, scikit-learn muss sich mit einer unterstützten Sprache von beiden geschlagen geben.

Was die Interoperabilität untereinander anbelangt, ist KNIME nativ schon im Vorteil und anhand der Installation zusätzlicher Erweiterungen ist dieser Vorsprung weiter ausbaubar. Diese Erweiterungen sind allerdings auch für manche Arbeitsabläufe zwingende Voraussetzung und funktionierten nicht immer. Ein absolutes Negativbeispiel stellt der Versuch des Lernens anhand von Bildern dar. Dabei griff der Autor auf einen bereits vorhandenen Arbeitsablauf zurück, welcher nicht funktionierte. Dafür ist die native Unterstützung von Quellformaten ein klarer Trumpf, den KNIME ausspielen kann.

Bei der Hardwarenutzung stellt sich TensorFlow als flexibelste Lösung heraus, da es nicht nur mehrkernige CPUs und GPUs unterstützt, sondern auch eigens dafür entwickelte TPUs befehligen kann. Bezüglich des eigentlichen Lernens ist anhand der getätigten Untersuchungen kein pauschales Urteil ableitbar, dem Autor gelang das Zuordnen von Bildern in KNIME nicht. Scikit-learn liegt bei der Klassifizierung von Bildern sowohl mit dem MNIST, als auch mit dem Fashion-MNIST Datensatz über der Zuordnungsgenauigkeit des mit TensorFlow erlernten Modells. Bei der Zuordnung von Texten liegt die Genauigkeit des von scikit-learn produzierten Modells mit 91,84% vor dem von KNIME generierten mit 79,1 %. Von TensorFlow generierte Modelle liegen im Mittel bei 72,12 %. Bei der Vorhersage der Bostoner Hauspreise anhand des auf linearer Regression basierend erlernten Modells liegen scikit-learn und TensorFlow bis zur vierten Nachkommastelle des Bestimmtheitsmaßes gleichauf mit 0,7334. Ab der fünften besteht ein Unterschied zugunsten von scikit-learn. Dafür ist der Wert den die Vorhersagewerte des mit TensorFlow erlernten Modells auf die Testwerte bei der gewurzelten mittleren Abweichung erzielt um einen Wert von 0,096 besser als der des Modells von scikit-learn. Das Modell das von KNIME erstellt wurde, unterliegt beiden Lösungen bei beiden Werten.

KNIME wird vom Autor als die Lösung mit der niedrigsten Einstiegsschwelle angesehen, da deren Bestandteile strukturiert und leicht nachvollziehbar angeordnet wurden. Sie eignet sich sogar für nutzende Personen ohne jegliche Programmiererfahrung. Bei Unklarheiten kann man schnell durch einen Blick in die standardmäßig im gleichen Fenster angeordnete Beschreibung viele Details in Erfahrung bringen. Auch erübrigt sich mit KNIME, dank der Integrierbarkeit der anderen beiden Lösungen, der Zwang eine Entscheidung treffen zu müssen.

Verfügt eine nutzende Person bereits über Programmiererfahrung in Python und will mit sehr umfangreichem Datenmaterial komplexe Zusammenhänge ableiten kann diese TensorFlow in KNIME, oder alternativ mit Keras, nutzen. Möchte eine nutzende Person desselben Typs Projekte geringeren Umfangs realisieren eignet sich scikit-learn eher.

Zusammenfassend stellt KNIME die erste Wahl für Menschen ohne Programmierkenntnisse dar. Mit steigender Komplexität der Problemstellung, sowie Programmierkenntnissen führt die Wahl über scikit-learn zu TensorFlow.

8.2 Fazit und Ausblick

Diese Arbeit liefert keine allgemeingültige Antwort auf die Frage nach dem geeigneten Arbeitsmittel um Wissen aus Daten zu extrahieren. Die Fähigkeiten der Lösungen bezüglich der essentiell wichtigen Schritte der Datenanalyse, -vorverarbeitung und -transformation werden, wie auch die Fähigkeiten zur grafischen Aufbereitung von Erkenntnissen, nicht

beleuchtet. Findet sich allerdings in der Prioritätenliste der lesenden Person einer oder mehrere der folgenden Punkte, so liefert diese Arbeit relevante Informationen.

- Installation unter Windows 10, Raspbian oder Ubuntu 18.04.2 LTS
- Art oder Importweise der integrierten Musterdaten
- Interoperabilität untereinander
- Hardwarenutzungsfähigkeiten
- Prognose eines Wertes in Abhängigkeit von anderen
- Nutzung einer grafischen Benutzeroberfläche
- unterstützte Programmiersprachen
- unterstützte Quellformate
- Klassifikation von Texten und Bildern

Unter Windows ist KNIME am einfachsten zu installieren, unter Linux ist es eine Frage der Präferenz. Soll ein Raspberry Pi genutzt werden, scheidet KNIME mangels Unterstützung aus und scikit-learn ist zu empfehlen. Die einzige native GUI liefert KNIME. Bei den integrierten Musterdaten mausert sich TensorFlow mittelfristig zum Primus, wobei KNIME aufgrund seiner Möglichkeit, TensorFlow zu nutzen davon ebenfalls profitiert. KNIMEs Beispielsammlung ist momentan noch umfangreicher, sie umfasst aber auch teilweise die Nutzung verwaister Nodes. Die Priorisierung der Lösung nach Programmiersprachen liegt im Auge der lesenden Person. Da der Autor vor Erstellung der Arbeit noch nicht in Kontakt mit Python kam, war hier bei TensorFlow und scikit-learn Lernbedarf. Die Lernkurve von scikit-learn empfand der Autor als angenehmer, wird TensorFlow mit Keras genutzt erleichtert dies den Umgang damit. Klarer Sieger im Punkt Interoperabilität ist KNIME, braucht hierzu allerdings zusätzliche Software um beispielsweise TensorFlow integrieren zu können. Bei den nativ unterstützten Quellformaten hat KNIME die Nase vorn. Für einen aussagefähigeren Vergleich sind neben den behandelten Wegen auch die Möglichkeiten anhand der Nutzung von Python für TensorFlow und scikit-learn, sowie alle Erweiterungen von KNIME zu untersuchen. Bei der Hardwarenutzung ist TensorFlow nativ am besten aufgestellt, KNIME kann anhand der Interoperabilität wieder profitieren. Für die nutzenden Personen, die sich eigene Hardware zur Nutzung der Lösungen anschaffen möchten, gibt die Arbeit sowohl spezifische, als auch eine allgemeingültige Hardwareempfehlung. Die Ergebnisse des Vergleichs anhand von Bildern, Texten und Daten reichen aus, um einen Überblick der zu unternehmenden Schritte zu gewähren. Sie lassen auch Rückschlüsse auf den jeweiligen Aufwand zu, um einen tiefen Leistungsvergleich zu schaffen reichen die unternommenen Untersuchungen hier nicht aus. Material für weitere Analysen lässt sich zwischenzeitlich anhand einer spezifischen Suchmaschine unter https://toolbox.google.com/datasetsearch suchen, Google veröffentlicht periodisch interessante Forschungsdatensätze unter https://ai.google/tools/datasets/. Für Erkenntnisse bezüglich Distribution und Integration der erlernten Modelle sind weitere Untersuchungen anzustellen. Diese werden in der Arbeit nicht behandelt. Amazon und Google bieten beispielsweise kostenpflichtige, skalierbare Lösungen an. Unter https://services.google.com/fb/forms/tpusignup/ kann die Anmeldung zur kostenlosen Forschungsnutzung der dedizierten Hardware von Google für TensorFlow beantragt werden.

Literaturverzeichnis

[ADAM]	Kingma, Diederik; Ba, Jimmy: Adam: A Method for Stochastic Optimization [Online] https://arxiv.org/abs/1412.6980v8 , Abruf: 02.05.2019
[ANACONDA_NEW]	Anaconda: Packages for 64-bit Windows with Python 3.7 [Online] http://docs.anaconda.com/anaconda/packages/py3.7_win-64/ , Abruf: 10.05.2019
[ANACONDA_PACKS]	Anaconda: Packages for 64-bit Windows with Python 3.6 [Online] http://docs.anaconda.com/anaconda/packages/py3.6_win-64/ , Abruf: 07.04.2019
[BODEN_DATEN]	Bodendorf, Freimut: Daten- und Wissensmanagement Zweite, aktualisierte und erweiterte Auflage, Berlin Heidelberg: Springer, 2. Auflage, 2006
[CLASSI_SAMPLES]	Roushangar, Raeuf; Mias, George I: gmiaslab / manuals [Online] https://github.com/gmiaslab/manuals , Abruf: 25.04.2019
[CLASSIFICAIO]	Roushangar, Raeuf; Mias, George I.: ClassificaIO: machine learning for classification graphical user interface [Online] https://doi.org/10.1101/240184 , Abruf: 24.04.2019
[CNN_OVERVIEW]	Bambharolia, Purvil: Overview of Convolutional Neural Networks [Online] https://www.semanticscholar.org/paper/OVERVIEW-OF-CONVOLUTIONAL-NEURAL-NETWORKS-Bambharolia/b482013815fc813c98695e72afd894b5dd169fd5 , Abruf: 19.04.2019
[CNTK]	Pettersson, Joe: README.md [Online] https://github.com/Microsoft/cntk , Abruf: 18.04.2019
[COMP_DE]	Statista: Anteil der Unternehmen mit Computernutzung nach Wirtschaftszweigen in Deutschland im Jahr 2018 [Online] https://de.statista.com/statistik/daten/studie/3871/umfrage/computernutzung-in-deutschen-unternehmen/ , Abruf: 18.04.2019

[COMPILE_KERNEL]	Kunst, Eva-Katharina; Quade, Jürgen: Den Raspberry Pi 3 im 64-Bit-Modus betreiben [Online] http://www.raspberry-pi-geek.de/Magazin/2017/04/Den-Raspberry-Pi-3-im-64-Bit-Modus-betreiben , Abruf: 22.04.2019
[CUDA_GPUs]	Nvidia: CUDA GPUs [Online] https://developer.nvidia.com/cuda-gpus , Abruf: 21.02.2019.
[DATA_MINING]	Cleve, Jürgen; Lämmel, Uwe: DATA MINING, Oldenburg: De Gruyter, 2. Auflage, 2016
[DIGIT_REC]	Jain, Subhi; Chauhan, Rahul: Recognition of Handwritten Digits Using DNN, CNN, and RNN: Second International Conference, ICACDS 2018, Dehradun, India, April 20-21, 2018
[DL_WITH_PY]	Chollet, François: Deep Learning with Python, Manning Publications Co., 2018
[DLS_VID]	Deep Cognition: How to upload custom Dataset [Online] https://player.vimeo.com/video/199905597 , Abruf: 27.04.2019
[DLS-PRODUCT]	Deep Cognition: Deep Learning Studio [Online] https://deepcognition.ai/wp-content/uploads/2018/07/DLS-ProductOverview.pdf , Abruf: 23.03.2019.
[ECLIPSE_PI]	MAKER.IO STAFF: How to install Eclipse on a Raspberry Pi [Online] https://www.digikey.com/en/maker/blogs/2018/how-to-install-eclipse-on-a-raspberry-pi , Abruf: 09.04.2019
[FASHION_MNIST]	Xiao, Han: Fashion-MNIST [Online] https://github.com/zalandoresearch/fashion-mnist/blob/master/README.md , Abruf: 02.05.2019
[FLOYD_PLAN]	FloydHub: Plans [Online] https://docs.floydhub.com/faqs/plans/ , Abruf: 21.04.2019
[FLOYD_SUPPORT]	FloydHub: Environments / List of Available Environments [Online] https://docs.floydhub.com/guides/environments/ , Abruf: 29.04.2019
[GB_PUB_DB]	Google AI: Publication Database [Online] https://ai.google/research/pubs/ , Abruf: 04.04.2019.
[GDAL]	GDAL: Geospatial Data Abstaction Library [Online] https://www.gdal.org/, Abruf: 13.04.2019
[GROUT_QCONAI]	Grout, Jason: JupyterLab: The Evolution of the Jupyter Notebook [Online] https://www.youtube.com/watch?v=ctOM-Gza04Y , Abruf: 02.04.2014

[GTX_1050]	Cosmic Shovel, Inc.: Amazon-Preisverlauf für Gigabyte GeForce GTX 1050 GV-N1050D5-2GD 2GB Grafikkarte schwarz [Online] https://de.camelcamelcamel.com/Gigabyte-GeForce-GV-N1050D5-2GD-Grafikkarte-schwarz/product/B01MG1MP73 , Abruf: 29.04.201*
[GTX_780]	Cosmic Shovel, Inc.: Amazon-Preisverlauf für WEIWEITOE-DE EVGA GeForce GTX 780 FTW Graphics Card 3GB GDDR5 [Online] https://de.camelcamelcamel.com/WEIWEITOE-EVGA-GeForce-Graphics-GDDR5/product/B07Q4TRWNS?context=search , Abruf: 29.04.2019
[IMAGE_IO]	Imageio contributors: Imageio formats [Online] https://imageio.readthedocs.io/en/latest/formats.html , Abruf: 10.04.2019
[IPO_2014]	Page, Larry; Brin, Sergey: *Founder's Letter* [Online] https://abc.xyz/investor/founders-letters/2004-ipo-letter/ , Abruf: 04.04.2019
[JEP_GH]	Bsteffensmeier: README.rst [Online] https://github.com/ninia/jep , Abruf: 24.03.2019.
[JUPY_FORMAT]	Jupyter development team: The Jupyter Notebook Format [Online] https://nbformat.readthedocs.io/en/latest/ , Abruf: 01.04.2019
[JUPY_KERNELS]	Syme, Don: Jupyter kernels [Online] https://github.com/jupyter/jupyter/wiki/Jupyter-kernels , Abruf: 02.04.2019
[JUPY_LAB_EXT]	Jupyter: Extension Developer Guide [Online] https://jupyterlab.readthedocs.io/en/latest/developer/extension_dev.html , Abruf: 05.04.2019
[JUPY_LAB_FILES]	Jupyter: JupyterLab File and Output Formats [Online] https://jupyterlab.readthedocs.io/en/latest/user/file_formats.html#file-and-output-formats , Abruf: 05.04.2019
[JUPY_LAB_INSTALL]	Jupyter: JupyterLab Installation [Online] https://jupyterlab.readthedocs.io/en/latest/getting_started/installation.html , Abruf: 05.04.2019
[JUPY_LAB_OV]	Jupyter: JupyterLab Overview [Online] https://jupyterlab.readthedocs.io/en/latest/getting_started/overview.html , Abruf: 02.04.2019

[JUPYTER]	Jupyter: Jupyter [Online] https://jupyter.org/ , Abruf: 02.04.2019
[JUPYTER_INSTALL]	Jupyter: Installing Jupyter Notebook [Online] https://jupyter.readthedocs.io/en/latest/install.html , Abruf: 09.04.2019
[KERAS]	KERAS: Keras: The Python Deep Learning library [Online] https://keras.io/ , Abruf: 18.09.2019
[KI_KABINETT]	Bundesministerium für Bildung und Forschung: Bundesregierung beschliesst Strategie Künstliche Intelligenz [Online] https://www.bmbf.de/de/bundesregierung-beschliesst-strategie-kuenstliche-intelligenz-7337.html , Abruf: 19.04.2019
[KI_STRATEGIE]	Die Bundesregierung: Strategie Künstliche Intelligenz der Bundesregierung [Online] https://www.bmbf.de/files/Nationale_KI-Strategie.pdf , Abruf: 19.04.2019
[KN_CODE_2]	KNIME AG: KNIME for Developers [Online] https://www.knime.com/knime-for-developers , Abruf: 24.03.2019.
[KN_CODING]	Berthold, Michael: To Code or not to code – Is that the question? [Online] https://www.knime.com/blog/to-code-or-not-to-code-is-that-the-question , Abruf: 24.03.2014.
[KN_HTS]	KNIME AG: HTS Data Mining/ SPARQL Nodes [Online] https://www.knime.com/book/hts-data-mining-sparql-nodes , Abruf: 13.04.2019
[KN_JAR]	KNIME AG: JMS Connector for KNIME [Online] https://www.knime.com/book/jmsconnector-for-knime , Abruf: 13.04.2019
[KN_JUPYTER]	Landrum, Greg: KNIME and Jupyter [Online] https://www.knime.com/blog/knime-and-jupyter , Abruf: 25.03.2019.
[KN_MMI]	KNIME AG: MMI Labs Nodes [Online] https://www.knime.com/book/mmi-labs-nodes , Abruf: 13.04.2019
[KN_MNIST_DL4J]	Fuller, Jon: Classifying handwritten digits using KNIME, DL4J and a LeNet variant [Online] https://hub.knime.com/knime/workflows/Examples/04_Analytics/14_Deep_Learning/01_DL4J/14_DeepLearningTutorial_MNI

ST/01_Using_DeepLearning4J_to_classify_MNIST_Digits*-gi4GZC_H9hpW0om , Abruf: 01.05.2019

[KN_MULTICPU] Gabriel: Which Image Nodes already use multiple cores? [Online] https://forum.knime.com/t/which-image-nodes-already-use-multiple-cores/10143 , Abruf: 23.03.2019.

[KN_PARALLEL] Martin, Anna: Parallel execution of nodes [Online] https://forum.knime.com/t/parallel-execution-of-nodes/9692 , Abruf: 23.03.2019.

[KNIME_CLUSTER] KNIME AG: KNIME Cluster Executor [Online] https://www.knime.com/knime-cluster-executor , Abruf: 14.04.2019

[KNIME_DEEP_L] KNIME AG: KNIME Deep Learning [Online] https://www.knime.com/deeplearning , Abruf: 17.04.2019

[KNIME_FAQs] KNIME AG: FAQ [Online] https://www.knime.com/faq#usage_data , Abruf: 30.03.2019

[KNIME_GPU] Jonfuller: Learning Deep Learning. A tutorial in KNIME Deeplearning4J Integration [Online] https://www.knime.com/blog/learning-deep-learning , Abruf: 21.03.2019.

[KNIME_HOUSING] DaveK: Housing Value Prediction using Regression [Online] https://hub.knime.com/knime/workflows/Examples/04_Analytics/14_Deep_Learning/01_DL4J/11_Housing_Value_Prediction_Using_Regression*W8bpaLswZ2ftRRTv , Abruf: 06.05.2019

[KNIME_HP] KNIME AG: KNIME Open Source Story [Online] https://www.knime.com/knime-open-source-story , Abruf: 16.03.2019.

[KNIME_INSTALL] KNIME AG: Building KIME and Update Site [Online] https://www.knime.com/building-knime-and-update-sites , Abruf: 22.04.2019

[KNIME_NUMPY] KNIME AG: KNIME Python Integration Installation Guide [Online] https://docs.knime.com/latest/python_installation_guide/index.html , Abruf: 07.04.2019

[KNIME_OPEN] KNIME AG: KNIME Analytics Platform [Online] https://www.knime.com/knime-software/knime-analytics-platform , Abruf: 20.04.2019

[KNIME_PRODUCTS]	KNIME AG: KNIME Integrations [Online] https://www.knime.com/knime-software/knime-integrations , Abruf: 17.04.2019
[KNIME_PY]	KNIME AG: KNIME Python Integration Installation Guide [Online] https://docs.knime.com/latest/python_installation_guide/index.html , Abruf: 01.05.2019
[KNIME_RASPBERRY]	KNIME Forum: Compiling Knime to run on Raspberry Pi? [Online] https://forum.knime.com/t/compiling-knime-to-run-on-raspberry-pi/5958 , Abruf: 16.04.2019
[KNIME_SERVER]	KNIME AG: KNIME Server [Online] https://www.knime.com/knime-software/knime-server-pricing , Abruf: 08.04.2019
[KNIME_SK_PY]	KNIME AG: KNIME Python Integration Installation Guide (2019) [Online] https://docs.knime.com/2018-12/python_installation_guide/python_installation_guide.pdf , Abruf: 24.03.2019.
[KNIME_TF]	KNIME AG: KNIME Deep Learning – TensorFlow Integration [Online] https://www.knime.com/deeplearning/tensorflow , Abruf: 21.03.2019.
[KNIME_TF_USE]	Walter, Alison: README.md (2018) [Online] https://github.com/knime/knime-tensorflow , Abruf: 23.03.2019.
[KNIME_WORKBENCH]	KNIME AG: Build a workflow [Online] https://www.knime.com/getting-started , Abruf: 10.04.2019
[MAKE_MODEL]	Satar, Burak; Dirik, Ahmet Emir: Deep Learning Based Vehicle Make-ModelClassification [Online] https://arxiv.org/pdf/1809.00953.pdf , Abruf: 19.04.2019
[MATPLOTLIB]	Hunter, John; et al.: matplotlib [Online] https://matplotlib.org/ , Abruf: 22.04.2019
[ML_SK_TF]	Géron, Aurelion: Machine Learning mit Scikit-Learn & TensorFlow, Heidelberg: dpunkt.verlag GmbH, 1. Auflage, 2018
[NUMPY]	NumPy developers (2019): Numpy [Online] http://www.numpy.org/ , Abruf: 07.04.2019
[NUMPY_INSTALL]	Oliphant,Travis E.; et al.: numpy 1.16.2 [Online] https://pypi.org/project/numpy/ , Abruf: 07.04.2019

[NVIDIA_CUDA]	Nvidia: CUDA Zone [Online] https://developer.nvidia.com/cuda-zone , Abruf: 25.03.2019
[NYT_AI]	The great A.I. Awakening [Online] https://www.nytimes.com/2016/12/14/magazine/the-great-ai-awakening.html , Abruf: 16.03.2019.
[PALIT_RTX2080Ti]	Cosmic Shovel, Inc.: Amazon-Preisverlauf für PALIT GeForce RTX 2080Ti Dual 11 GB GDDR6/PCI Express 3.0 Grafikkarte [Online] https://de.camelcamelcamel.com/PALIT-GeForce-2080Ti-Express-Grafikkarte/product/B07JKVV7WN , Abruf: 29.04.2019
[PANDAS]	McKinney, Wes ; et al.: pandas: powerful Python data analysistoolkitRelease 0.24.2 [Online] https://pandas.pydata.org/pandas-docs/stable/pandas.pdf , Abruf: 08.04.2019
[PANDAS_CSV]	Pandas: pandas 0.24.2 documentation » API Reference » Input/Output »pandas_read.csv [Online] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html , Abruf: 10.05.2019
[PICKLE]	Python Software Foundation: pickle — Python object serialization [Online] https://docs.python.org/3/library/pickle.html , Abruf: 24.04.2019
[PICKLE_ROT]	Slaviero, Marco: Sour Pickles A serialized exploitation guide in one part [Online] http://www.hakim.ws/BHUS2011/materials/Slaviero/BH_US_11_Slaviero_Sour_Pickles_Slides.pdf , Abruf: 24.04.2019
[PY_DISTRO]	5 Python distributions for mastering machine learning [Online] https://clusterdata.nl/bericht/news-item/5-python-distributions-for-mastering-machine-learning/ , Abruf: 18.03.2019.
[QCONAI]	InfoQ: Qcon.ai [Online] https://qcon.ai/ , Abruf: 02.04.2019
[RED_AMA_J]	Dean, Jeff:We are the Google Brain team. We'd love to answer your questions (again) [Online] https://www.reddit.com/r/MachineLearning/comments/6z51xb/we_are_the_google_brain_team_wed_love_to_answer/dmylath/?utm_source=share&utm_medium=web2x , Abruf: 10.04.2019.

[RTX_2060_OC]	Cosmic Shovel, Inc.: Amazon-Preisverlauf für Gigabyte GeForce RTX 2060 OC [Online] https://de.camelcamelcamel.com/Gigabyte-GeForce-RTX-2060-OC/product/B07MJGCPW5?context=search , Abruf: 29.04.2019
[RTX_2080TI]	NVIDIA: GEFORCE® RTX 2080 Ti [Online] , https://www.nvidia.com/de-de/geforce/graphics-cards/rtx-2080-ti/ , Abruf: 01.05.2019
[SCIKIT]	Scikit-learn developers: scikit-learn user guid Release 0.20.3 [Online] https://scikit-learn.org/stable/_downloads/scikit-learn-docs.pdf , Abruf: 19.04.2019
[SCIKIT_ABOUT]	Scikit-learn developers: About us [Online] https://scikit-learn.org/stable/about.html , Abruf: 16.04.2019
[SCIKIT_FAQs]	Scikit-learn: Frequently Asked Questions [Online] https://scikit-learn.org/stable/faq.html, , Abruf: 21.03.2019.
[SCIKIT_GPU]	Scikit-learn: Related Projects [Online] https://scikit-learn.org/stable/related_projects.html#statistical-learning-with-python , Abruf: 22.03.2019.
[SCIKIT_IMAGE]	Scikit-image development team: Module: io [Online] https://scikit-image.org/docs/dev/api/skimage.io.html , Abruf: 10.04.2019
[SCIPY]	SciPy developers: SciPy library [Online] https://www.scipy.org/scipylib/index.html , Abruf: 07.04.2019
[SCIPY_2019]	SciPy: SciPy2019 [Online] https://www.scipy2019.scipy.org/ , Abruf: 07.04.2019
[SCIPY_ABOUT]	SciPy: Scientific Computing Tools for Python [Online] https://scipy.org/about.html , Abruf: 10.05.2019
[SCIPY_INSTALL]	Oliphant, Travis E.; et al.: scipy 1.2.1 [Online] https://pypi.org/project/scipy/ , Abruf: 07.04.2019
[SIG_SPOOF]	Khandelwal, Swati: Over Dozen Popular Email Clients Found Vulnerable to Signature Spoofing Attackes [Online] https://thehackernews.com/2019/04/email-signature-spoofing.html , Abruf: 05.05.2019
[SK_BOSTON_LIN]	Agarwal, Animesh: Linear Regression on Boston Housing Dataset [Online] https://towardsdatascience.com/linear-

	regression-on-boston-housing-dataset-f409b7e4a155 , Abruf: 05.05.2019
[SK_GH_GUI]	GitHub: Conributions to Pykit-Learn Master [Online] https://github.com/bsuhagia/Pykit-Learn/graphs/contributors?from=2015-11-24&to=2015-12-16&type=c , Abruf: 23.03.2019.
[SK_H2O4GPU]	H2oai: H2Oai GPU Edition [Online] https://github.com/h2oai/h2o4gpu , Abruf: 22.03.2019.
[SK_IMPORT]	Scikit-learn: Dataset loading utilities [Online] https://scikit-learn.org/stable/datasets/index.html , Abruf: 29.03.2019
[SK_IMPORT_BOSTON]	Scikit-learn: sklearn.datasets.load_boston [Online] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html , Abruf: 29.03.2019
[SK_INSTALL]	Scikit-learn: Installing scikit-learn [Online] https://scikit-learn.org/stable/install.html , Abruf: 18.03.2019.
[SK_IRIS]	Scikit-learn: The Iris Dataset [Online] https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html , Abruf: 01.04.2019
[SK_MULTICPU]	Scikit-learn: How to optimize for speed [Online] https://scikit-learn.org/stable/developers/performance.html , Abruf: 22.03.2019.
[SK_NEWS_GIT]	Codewrestling: TextClassification.py [Online] https://github.com/codewrestling/TextClassification/blob/master/Text%20Classification.py , Abruf: 03.05.2019
[SK_SAMPLE_PICS]	Scikit-learn: sklearn.datasets.load_sample_images [Online] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_sample_images.html#sklearn.datasets.load_sample_images , Abruf: 31.01.2019
[SK_SP_PD]	Zaitlen, Ben: Python Setup Ipython Notebook [Online] https://quasiben.github.io/PyDataEMC/#/9 , Abruf: 17.04.2019
[SK_VULN]	Min RK: Open Redirect Vulnerability in Jupyter notebook, JupyterHub [Online] https://blog.jupyter.org/open-redirect-vulnerability-in-jupyter-jupyterhub-adf43583f1e4 , Abruf: 17.04.2019

[SKF_GIT]	TensorFlower Gardener: README.md [Online] https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/learn/README.md , Abruf: 28.04.2019
[SKF_REMAINS]	TensorFlow: PythonModule: tf.contrib.learn [Online], https://www.tensorflow.org/api_docs/python/tf/contrib/learn , Abruf: 28.04.2019
[SK-LEARN]	Scikit-Learn: scikit-learn user guide Release 0.21.dev0 [Online] https://scikit-learn.org/dev/_downloads/scikit-learn-docs.pdf , Abruf: 14.03.2019
[TENSOR]	Ganster, Maximilian: Definition eines Tensors [Online] https://www.math.tugraz.at/~ganster/vektoranalysis_ss_2012.html , Abruf: 15.09.2019
[TF_BACK]	TensorFlow: TensorFlow Version Compatibility [Online] https://www.tensorflow.org/guide/version_compat#what_is_covered , Abruf: 25.03.2019.
[TF_BOSTON_LIN]	Zaccone, Giancarlo: Linear Regression using TensorFlow [Online] https://www.tutorialkart.com/blog/linear-regression-using-tensorflow/ , Abruf: 05.05.2019
[TF_CLUSTER]	Yahs Kateriya: Distributed TensorFlow [Online] https://github.com/tensorflow/examples/blob/master/community/en/docs/deploy/distributed.md , Abruf: 14.04.2019
[TF_DATA]	TensorFlow: Module: tf.data [Online] https://www.tensorflow.org/guide/datasets , Abruf: 28.03.2019
[TF_DATASET]	Géron, Aurélien: TensorFlow Datasets [Online] https://github.com/tensorflow/datasets#usage , Abruf: 13.04.2019
[TF_DATASOURCE]	TensorFlow: TensorFlow>Datasets v1.0.1>Resources>Datasets [Online] https://www.tensorflow.org/datasets/datasets , Abruf: 22.03.2019
[TF_DOCKER]	TensorFlow: Docker [Online] https://www.tensorflow.org/install/docker , Abruf: 24.03.2019.
[TF_ESTIMATORS]	TensorFlow: TensorFlow>Learn>TensorFlow Core>Guide [Online] https://www.tensorflow.org/guide/estimators, Abruf: 21.04.2019
[TF_GPU]	TensorFlow: GPU support [Online] https://www.tensorflow.org/install/gpu , Abruf: 21.03.2019.

[TF_GUIDE]	TensorFlow: TensorFlow> Learn>TensorFlow Core> Guide [ONLINE] https://www.tensorflow.org/guide , Abruf: 16.05.2019
[TF_INSTALL]	Install TensorFlow with pip [Online] https://www.tensorflow.org/install/pip , Abruf: 18.09.2019.
[TF_INSTALL_BOARD]	TensorFlow: TensorBoard: Visualizing Learning [Online] https://www.tensorflow.org/tensorboard/r1/summaries , Abruf: 23.03.2019.
[TF_LINGOS]	TensorFlow: API r1.13 API Documentation [Online] https://www.tensorflow.org/api_docs , Abruf: 25.03.2019
[TF_LINGOS_2.0]	TensorFlow: API r2.0 API Documentation [Online] https://www.tensorflow.org/versions/r2.0/api_docs , Abruf: 25.03.2019
[TF_MNIST]	TensorFlow: Get Started with TensorFlow [Online] https://www.tensorflow.org/tutorials , Abruf: 02.05.2019
[TF_MULTI_GPU]	TensorFlow: Using GPU [Online] https://www.tensorflow.org/guide/using_gpu , Abruf: 21.09.2019.
[TF_NEWS_GIT]	Sastre, Juan Carlos: 20newsgroups_Classification_Pure_Tensorflow.ipynb [Online] https://github.com/jcsastre/20newsgroups_tensorflow/blob/master/20newsgroups_Classification_Pure_Tensorflow.ipynb , Abruf: 10.05.2019
[TF_NUMPY_MORE]	TensorFlow: Search results for numpy [Online] https://www.tensorflow.org/s/results/?q=numpy , Abruf: 07.04.2019
[TF_PY]	TensorFlow: TensorFlow>API r1.13>Python [Online] https://www.tensorflow.org/api_docs/python/tf , Abruf: 17.04.2019
[TF_SAMPLE_USE]	TensorFlow: TensorFlow>Resources>Datasets v1.0.1> Datasets Usage [Online] https://www.tensorflow.org/datasets/datasets#usage , Abruf: 09.04.2019
[TF_SCIPY]	TensorFlow: TensorFlow>API r1.13>Python>tf.contrib.opt.ScipyOptimizerInterface [Online]

https://www.tensorflow.org/api_docs/python/tf/contrib/opt/ScipyOptimizerInterface , Abruf: 11.04.2019

[TF_STRUCTURE] TensorFlow Team: Introduction to Tensorflow Datasets and Estimators [Online] https://developers.googleblog.com/2017/09/introducing-tensorflow-datasets.html , Abruf: 17.04.2019

[TF_USEBOARD] TensorFlow: tensorboard [ONLINE] https://github.com/tensorflow/tensorboard , Abruf: 23.03.2019.

[TF_VERSIONS] TensorFlow: TensorFlow API Versions [Online] https://www.tensorflow.org/versions/ , Abruf: 25.03.2019.

[TF_WHITEPAPER] TensorFlow (2015): Large-Scale Machine Learning on Heterogeneous Distributed Systems [Online] http://download.tensorflow.org/paper/whitepaper2015.pdf , Abruf: 10.04.2019

[TF_WINDOCK] TensorFlow: TensorFlow Docker Images [Online] https://hub.docker.com/r/tensorflow/tensorflow/ , Abruf: 24.03.2019.

[THEANO] Bastien, Frédéric: README.rst [ONLINE] https://github.com/Theano/Theano , Abruf: 18.04.2019

[THEANO_DL] Theano: Theano at a glance [Online] http://www.deeplearning.net/software/theano/introduction.html , Abruf: 18.04.2019

[THEANO_QUITS] Lamblin, Pascal: MILA and the future of Theano [Online] https://groups.google.com/forum/#!msg/theano-users/7Poq8BZutbY/rNCIfvAEAwAJ , Abruf: 18.04.2019

[TPU_ARTICLE] Teich, Paul: Tearing apart Google's TPU 3.0 Coprocessor [Online] https://www.nextplatform.com/2018/05/10/tearing-apart-googles-tpu-3-0-ai-coprocessor/ , Abruf: 23.03.2019

[TPU_PREEMPT] Google Cloud: Using Preemptible TPUs [Online] https://cloud.google.com/tpu/docs/preemptible , Abruf: 20.03.2019.

[TPU_PRICE] Google Cloud: Preise [Online] https://cloud.google.com/tpu/docs/pricing , Abruf: 20.03.2019.

[TPU_TFRC] TensorFlow: TensorFlow Research Cloud [Online] https://www.tensorflow.org/tfrc , Abruf: 20.03.2019.

[UBUNTU-MATE] Ubuntu MATE Team: Download [Online] https://ubuntu-mate.org/download/ , Abruf: 09.04.2019

[WI_MERTENS_ET] Mertens, Peter; et al.: Grundzüge der Wirtschaftsinformatik, Heidelberg: Springer Gabler, 12., grundlegend überarbeitete Auflage, 2017

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Es wurden keine anderen als die angegebenen Quellen und Hinweise verwandt.

Die vorliegende Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Unterhaching, 17.05.2019 _____

Andreas Eckl

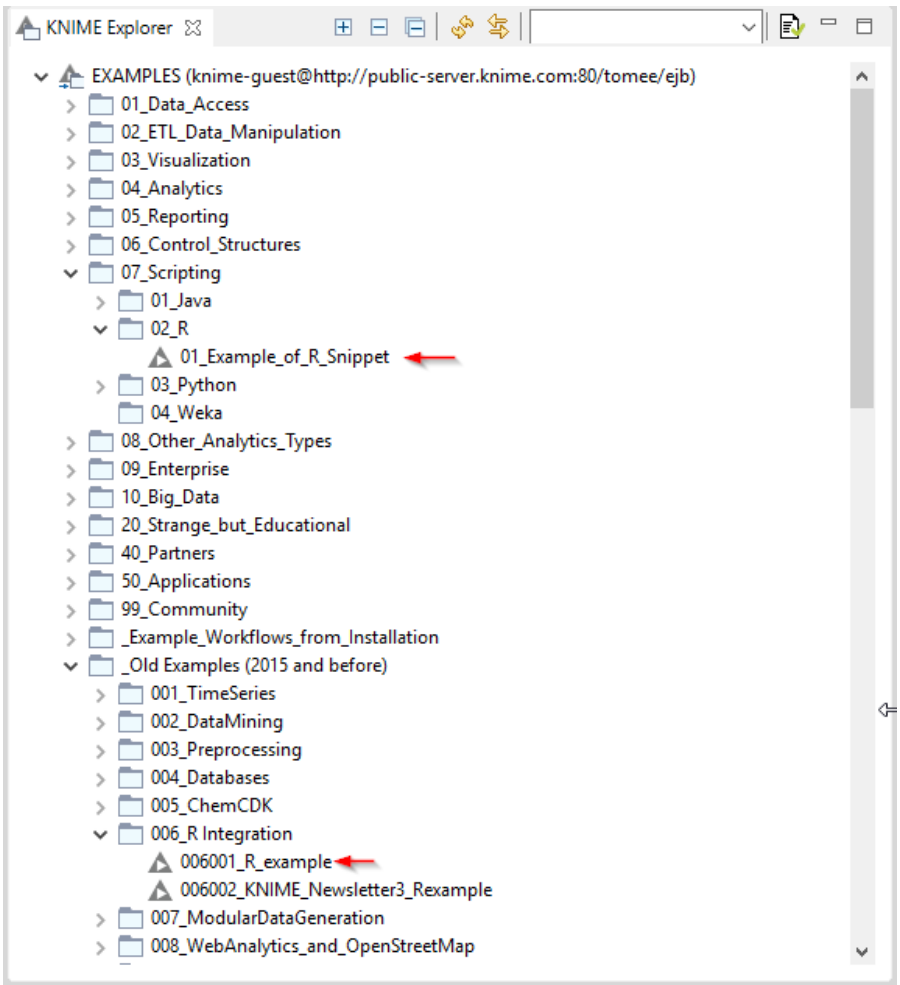
Anlagenverzeichnis

Jeweils ein USB Stick mit den innerhalb der Arbeit besprochenen Daten, sowie den gespeicherten Onlinequellen und den Verknüpfungen darauf.

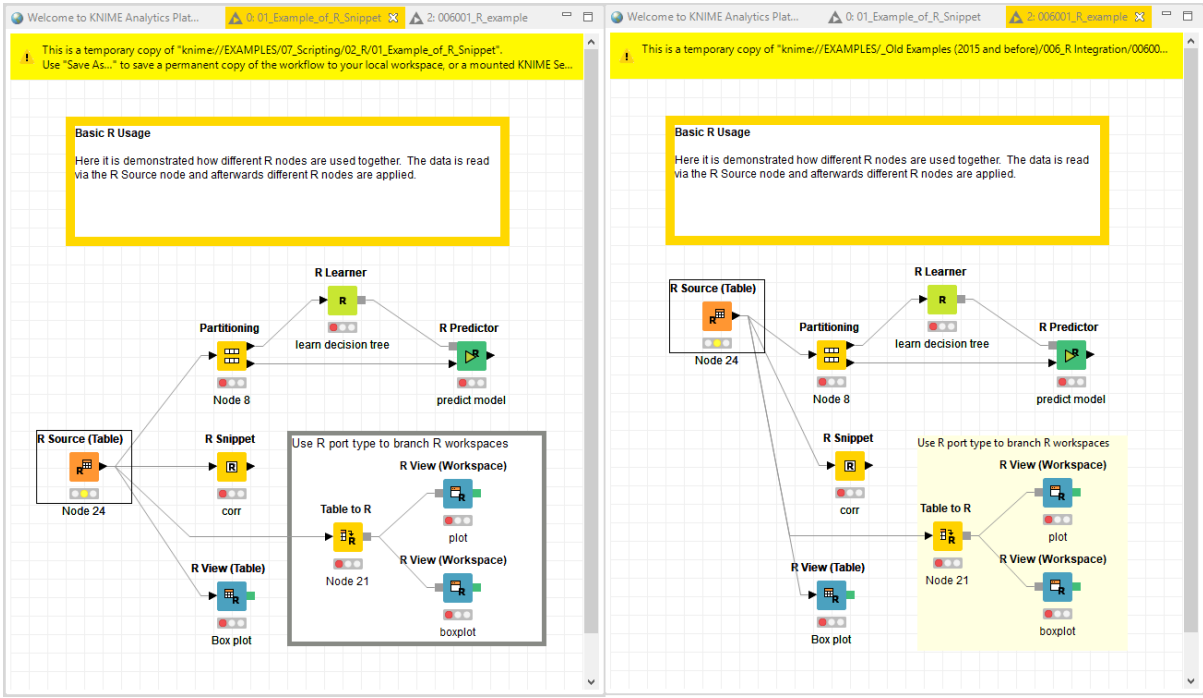
Anlage A

A1 Abbildung Beispiele KNIME

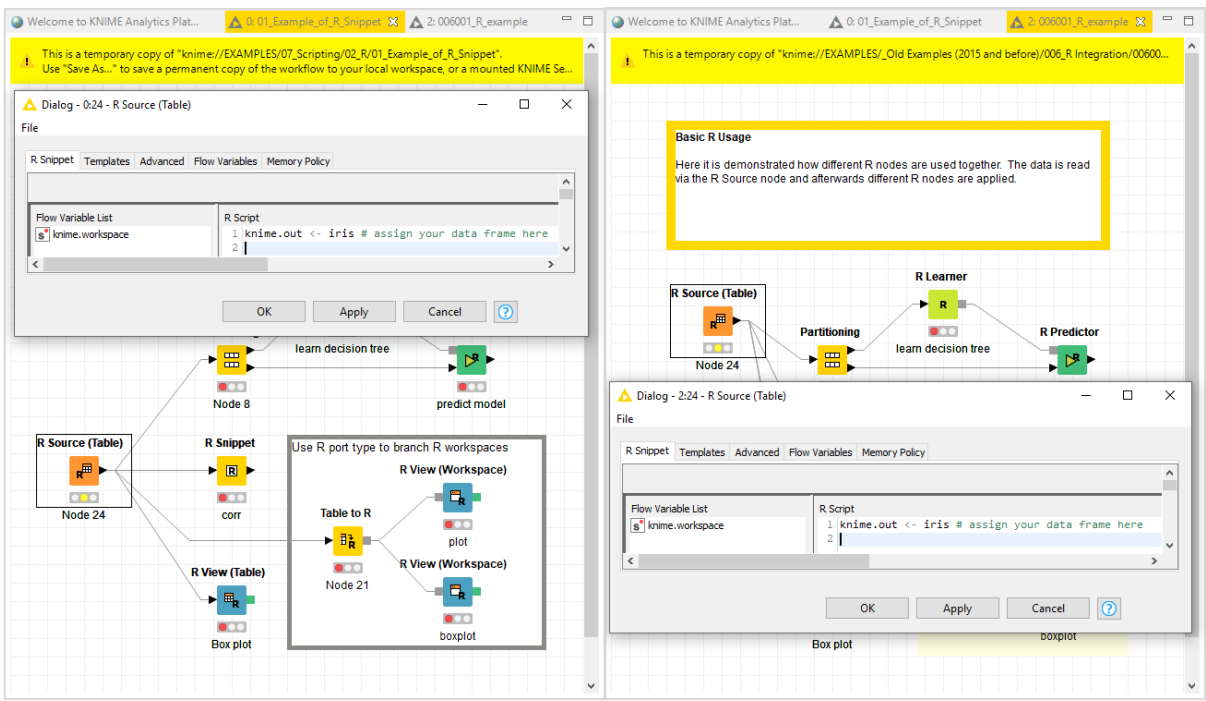
In den in A1 thematisierten Abbildungen wurde der Inhalt des Ordners „_Old Examples (2015) and before“ nicht abgebildet. Dieser wurde ausgelassen, da augenscheinlich ähnliche Inhalte hinterlegt sind. Die Inhalte finden sich nicht nur ähnlich, sondern auch fast identisch in den neueren Beispieldaten. Exemplarisch wird hier die Nutzung der Programmiersprache R als Beleg herangezogen. Verglichen wurde der Workflow „01_Example_of_R_Snippet“ des Unterordners „02_R“ aus dem Ordner „07_Scripting“ mit dem Workflow „006001_R_example“ des Unterordners „006_R Integration“ aus dem Ordner „_Old Examples (2015) and before“. Die Vergleichskandidaten sind in folgender Abbildung mit einem roten Pfeil markiert.



Wie sehr sich diese optisch ähneln zeigt die folgende Abbildung.

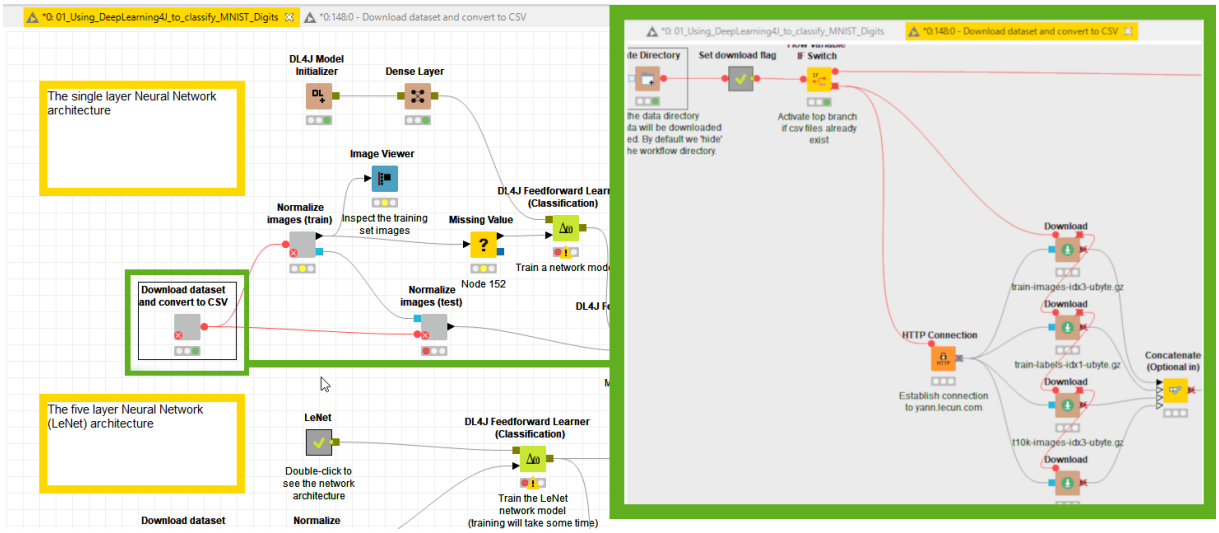


Zusätzlich dazu basieren beide Workflows auf der gleichen Zeile R-Code, dies bildet der nächste Screenshot-Vergleich ab.



A2 Import MNIST

Die folgende, selbst erstelle Abbildung gibt die am Import des MNIST Datensatzes beteiligten Nodes wieder, die sich innerhalb des Metanodes „Download Dataset and convert to CSV“ befinden.



Anlage B

B1 Mit TensorFlow installierte Pakete in Linux Distributionen

Basis der folgenden Tabelle ist die bildliche Dokumentation der Pakete im Zuge der Installation. Dieses findet sich auf den, zusätzlich zu dieser Arbeit ausgehändigten, Speichermedien im Ordner „Logs“ repräsentiert durch die Dateien „TF_install_routine1v2.png“ und „TF_install_routine2v2.png“.

Paketname	Version	Paketname	Version	Paketname	Version
absl-py	0.7.1	markdown	3.1	tensorboard	1.13.1
astor	0.7.1	mock	2.0.0	tensorflow	1.13.1
gast	0.2.2	numpy	1.16.2	tensorflow-estimator	1.13.0
grpcio	1.19.0	pbr	05.01.2003	termcolor	1.1.0
h5py	2.9.0	protobuf	3.7.1	werkzeug	0.15.1
keras-applications	1.0.7	setuptools	40.8.0	wheel	0.33.1
keras-preprocessing	1.0.9	six	1.12.0		

B2 Mit Anaconda3 installierte Pakete

Basis der folgenden Aufstellung sind zwei Quellen, [ANACONDA_PACKS] und [ANACONDA_NEW]. Die Gegenüberstellung bezieht sich auf die Version für das 64-bittige Windows. Ein Pluszeichen vor der Versionsnummer steht dabei für ein Update des Pakets in Anaconda3 „2019.03“ und zeigt die aktualisierte Versionsnummer an. Ein Minuszeichen repräsentiert das Fehlen in der neueren Version. Die Versionsnummer dort repräsentiert die in Anaconda3 „2018.12“ genutzte. Ein Sternchen steht für die Neuaufnahme des Pakets.

Paketname	Version	Paketname	Version	Paketname	Version	Paketname	Version
_ipyw_jlab_nb_ext_conf	0.1.0	cffi	+ 1.12.2	flask	1.0.2	jpeg	9b
alabaster	0.7.12	chardet	3.0.4	flask-cors	- 3.0.7	jsonschema	+ 3.0.1
anaconda-client	1.7.2	click	7.0	freetype	2.9.1	jupyter	1.0.0
anaconda-navigator	1.8.7	cloudpickle	+ 0.8.0	get_terminal_size	1.0.0	jupyter_client	5.2.4
anaconda-project	0.8.2	clyent	1.2.2	gevent	+ 1.4.0	jupyter_console	6.0.0
asn1crypto	0.24.0	colorama	0.4.1	glob2	0.6	jupyter_core	4.4.0
astroid	+ 2.2.5	comtypes	1.1.7	greenlet	0.4.15	jupyterlab	+ 0.35.4
astropy	+ 3.1.2	conda	- 4.5.12	h5py	+ 2.9.0	jupyterlab_server	0.2.0
atomicwrites	+ 1.3.0	conda-build	- 3.17.6	hdf5	+ 1.10.4	keyring	+ 18.0.0
attrs	+ 19.1.0	conda-env	- 2.6.0	heapdict	1.0.0	kiwisolver	1.0.1
automat	0.7.0	console_shortcut	0.1.1	html5lib	1.0.1	krb5	1.16.1
babel	2.6.0	constantly	15.1.0	hyperlink	18.0.0	lazy-object-proxy	1.3.1
backcall	0.1.0	contextlib2	0.5.5	icc_rt	2019.0.0	libarchive	3.3.3

Paketname	Version	Paketname	Version	Paketname	Version	Paketname	Version
backports	1.0	cryptography	+ 2.6.1	icu	58.2	libcurl	+ 7.64.0
backports.os	0.1.1	curl	+ 7.64.0	idna	2.8.19	libiconv	1.1.15
backports.shutil_get_terminal_size	1.0.0	cycler	0.10.0	imageio	+ 2.5.0	libpng	+ 1.6.36
beautifulsoup4	+ 4.7.1	cython	+ 0.29.6	imagesize	1.1.0	libsodium	1.0.16
bitarray	0.8.3	cytoolz	0.9.0.1	importlib_metadata	+ 0,8	libssh2	1.8.0
bkcharts	0.2	dask	+ 1.1.4	incremental	17.5.0	libtiff	+ 4.0.10
blas	1.0	dask-core	+ 1.1.4	intel-openmp	+ 2019,3	libxml2	+ 2.9.9
blaze	- 0.11.3	datashape	- 0.5.4	ipykernel	5.1.0	libxslt	+ 1.1.33
bleach	+ 3.1.0	decorator	+ 4.4.0	ipython	+ 7.4.0	llvmlite	+ 0.28.0
blosc	+ 1.15.0	defusedxml	0.5.0	ipython_genutils	0.2.0	locket	0.2.0
bokeh	+ 1.0.4	distributed	+ 1.26.0	ipywidgets	7.4.0	lxml	+ 4.3.2
boto	2.49.0	docutils	0.14	isort	+ 4.3.16	lz4-c	1.8.1.2
bottleneck	0.7.1	entrypoints	+ 0,3	itsdangerous	1.1.0	lzo	2.10.19
bzip2	1.0.6	et_xmlfile	1.0.1	jdcal	1.4.19	m2w64-gcc-libgfortran	5.3.0
ca-certificates	+ 2019.1.23	fastcache	1.0.2	jedi	+ 0.13.3	m2w64-gcc-libs	5.3.0
certifi	+ 2019.3.9	filelock	3.0.10	jinja2	2.10.19	m2w64-gcc-libs-core	5.3.0
m2w64-gmp	6.1.0	olefile	0.46	pycurl	7.43.0.2	requests	2.21.0
m2w64-libwinpthread-git	5.0.0.4634.697f757	openpyxl	+ 2.6.1	pyflakes	+ 2.1.1	rope	+ 0.12.0
m2w64-gmp	6.1.0	olefile	0.46	pycurl	7.43.0.2	requests	2.21.0
m2w64-libwinpthread-git	5.0.0.4634.697f757	openpyxl	+ 2.6.1	pyflakes	+ 2.1.1	rope	+ 0.12.0
markupsafe	+ 1.1.1	openssl	+ 1.1.1b0	pygments	2.3.1	ruamel_yaml	0.15.46
matplotlib	+ 3.0.3	packaging	+ 19	pylint	+ 2.3.1	scikit-image	+ 0.14.2
mccabe	0.6.1	pandas	+ 0.24.2	pyodbc	+ 4.0.26	scikit-learn	+ 0.20.3
menuinst	+ 1.4.16	pandoc	+ 2.2.3.2	pyopenssl	+ 19.0.0	scipy	+ 1.2.1
mistune	0.8.4	pandocfilters	1.4.2	pyparsing	+ 2.3.1	seaborn	0.9.0
mkl	+ 2019,3	parso	+ 0.3.4	pyqt	5.9.2	send2trash	1.5.0
mkl-service	1.1.2	partd	+ 0.3.10	pysocks	1.6.8	setuptools	+ 40.8.0
mkl_fft	+ 1.0.10	path.py	11.5.0	pytables	+ 3.5.1	simplegeneric	0.8.1
mkl_random	1.0.2	pathlib2	2.3.3	pytest	+ 4.3.1	singledispatch	3.4.0.3

Paketname	Version	Paketname	Version	Paketname	Version	Paketname	Version
more-itertools	+6.0.0	patsy	0.5.1	pytest-arraydiff	0.3	sip	4.19.8
mpmath	1.1.0	pep8	1.7.1	pytest-astropy	0.5.0	six	1.12.0
msgpack-python	+0.6.1	pickleshare	0.7.5	pytest-doctestplus	+0.3.0	snappy	1.1.7
msys2-conda-epoch	20160418	pillow	+5.4.1	pytest-openfiles	+0.3.2	snowballstemmer	1.2.1
multipledispatch	0.6.0	pip	+19.0.3	pytest-remotedata	0.3.1	sortedcollections	+1.1.2
navigator-updater	0.2.1	pkginfo	+1.5.0.1	python	+3.7.3	sortedcontainers	2.1.0
nbconvert	+5.4.1	pluggy	+0.9.0	python-dateutil	+2.8.0	sphinx	+1.8.5
nbformat	4.4.0	ply	3.11	python-libarchive-c	2.8	sphinxcontrib	1.0
networkx	2.2	progress	1.1.3	pytz	+2018.9	sphinxcontrib-websupport	1.1.0
nltk	3.4	prometheus_client	+0.6.0	pywavelets	+1.0.2	spyder	+3.3.3
nose	1.3.7	prompt_toolkit	+2.0.9	pywin32	223	spyder-kernels	+0.4.2
notebook	+5.7.8	psutil	+5.6.1	pywinpty	0.5.5	sqlalchemy	+1.3.1
numba	+0.43.1	py	+1.8.0	pyyaml	+5.1	sqlite	+3.27.2
numexpr	+2.6.9	pyasn1-modules	0.2.2	pyzmq	+18.0.0	statsmodels	0.9.0
numpy	+1.16.2	pycodestyle	+2.5.0	qt	5.9.7	sympy	1.3
numpy-base	+1.16.2	pycosat	0.6.3	qtawesome	+0.5.7	tblib	1.3.2
numpydoc	0.8.0	pycparser	43497	qtconsole	4.4.3	terminado	0.8.1
odo	-0.5.1	pycrypto	2.6.1	qtpy	+1.7.0	testpath	0.4.2
tk	8.6.8	wcwidth	0.1.7	xlrd	1.2.0	pyrsistent	*0.14.11
tk	8.6.8	wcwidth	0.1.7	xlrd	1.2.0	pyrsistent	*0.14.11
toolz	0.9.0	webencodings	0.5.1	xlsxwriter	+1.1.5	liblief	*0.9.0
tornado	+6.0.2	werkzeug	0.14.1	xlwings	0.15.4	powershell_shortcut	*0.0.1
tqdm	+4.31.1	wheel	+0.33.1	xlwt	1.3.0	py-lief	*0.9.0
traitlets	4.3.2	widgetsnbextension	3.4.2	xz	5.2.4	pyreadline	*2.1
typed-ast	+1.3.1	win_inet_pton	+1.1.0	yaml	0.1.7	soupsieve	*1.8
unicodecsv	0.14.1	win_unicode_console	0.5	zeromq	+4.3.1	zipp	*0.3.3
urllib3	1.24.1	wincertstore	0.2	zict	+0.1.4		
vc	14.1	winpty	0.4.3	zlib	1.2.11		
vs2015_runtime	14.15.26706	wrapt	+1.11.1	zstd	1.3.7		

B3 Inhalt der Variable „information“ bei Befüllen mit dem TensorFlow-Musterdatensatz „mnist“

Die abgebildete Grafik ist ein Bildschirmausschnitt eines Jupyter Notebooks nach Eingabe der folgenden Befehle:

```
import tensorflow as tf
import tensorflow_datasets as tfds
daten, informationen = tfds.load("mnist", with_info=True)
print(informationen)
```

```
tfds.core.DatasetInfo(
    name='mnist',
    version=1.0.0,
    description='The MNIST database of handwritten digits.',
    urls=['http://yann.lecun.com/exdb/mnist/'],
    features=FeaturesDict({
        'image': Image(shape=(28, 28, 1), dtype=tf.uint8),
        'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=10)
    }),
    total_num_examples=70000,
    splits={
        'test': <tfds.core.SplitInfo num_examples=10000>,
        'train': <tfds.core.SplitInfo num_examples=60000>
    },
    supervised_keys=('image', 'label'),
    citation='"""
        @article{lecun2010mnist,
          title={MNIST handwritten digit database},
          author={LeCun, Yann and Cortes, Corinna and Burges, CJ},
          journal={ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist},
          volume={2},
          year={2010}
        }

    """',
)
```

Anlage C

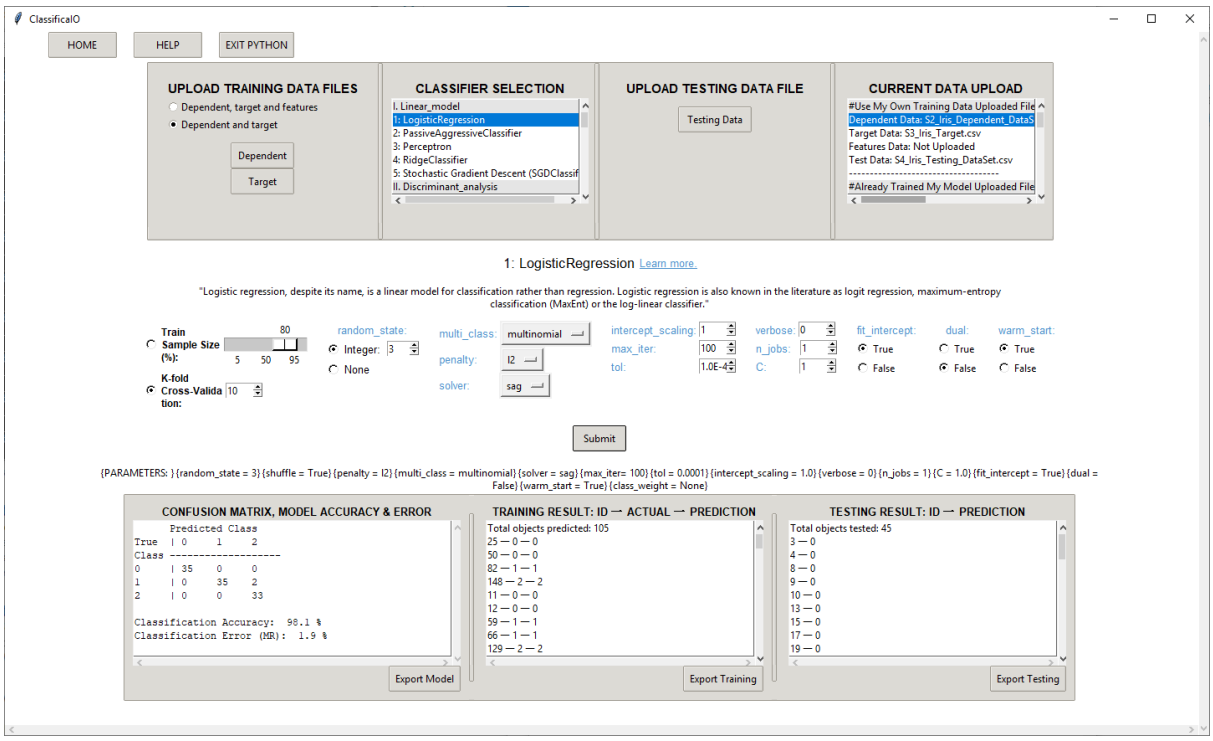
C1 Mit dem Upgrade von scikit-learn und anderer Komponenten installierte Pakete und deren Versionen

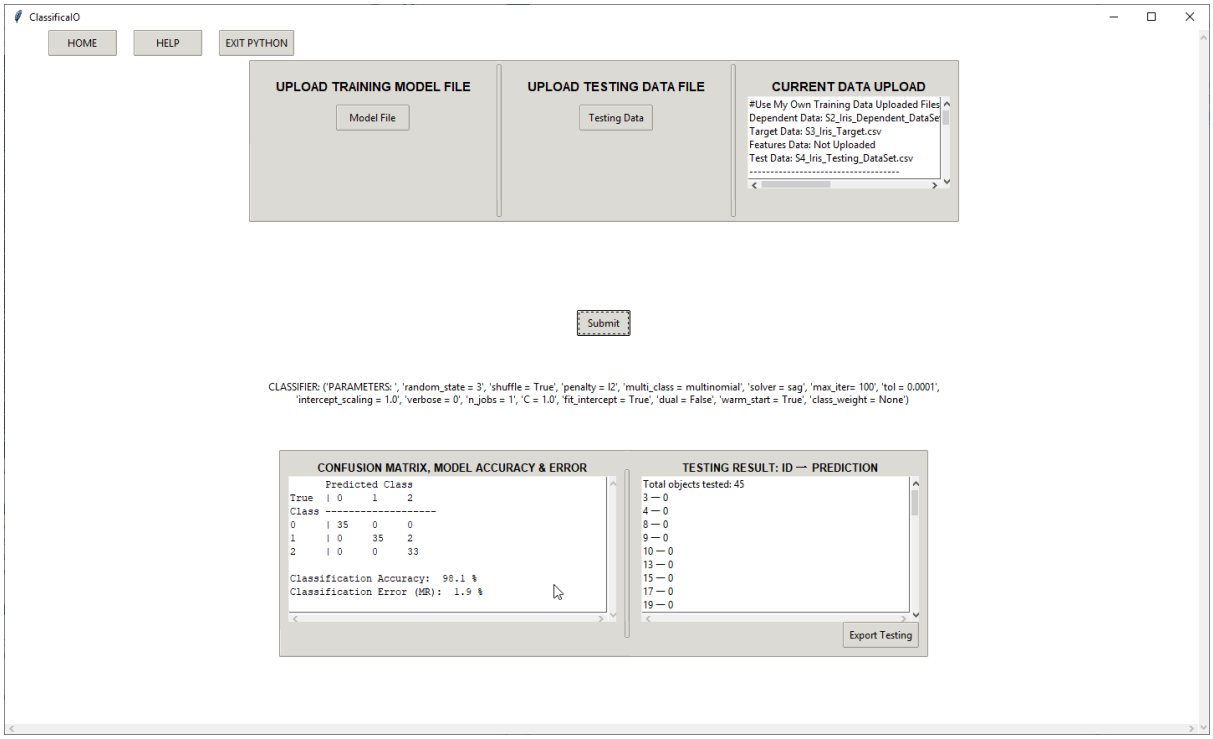
Basis dieser Tabelle ist ein selbst erstelltes Log während der Installation, dieses findet sich auf den, zusätzlich zu dieser Arbeit ausgehändigten, Speichermedien im Ordner „Logs“.

Paketname	Version	Paketname	Version	Paketname	Version
attrs	19.1.0	kiwisolver	1.0.1	pyrsistent	0.14.11
backcall	0.1.0	MarkupSafe	1.1.1	python-dateutil	2.8.0
bleach	3.1.0	matplotlib	3.0.3	pytz	2018,9
cycler	0.10.0	mistune	0.8.4	pyzmq	18.0.1
decorator	4.4.0	nbconvert	5.4.1	qtconsole	4.4.3
defusedxml	0.5.0	nbformat	4.4.0	scikit-learn	0.20.3
entrypoints	0.3	notebook	5.7.7	scipy	1.2.1
ipykernel	5.1.0	numpy	1.16.2	Send2Trash	1.5.0
ipython	7.4.0	pandas	0.24.2	setuptools	40.8.0
ipython-genutils	0.2.0	pandocfilters	1.4.2	six	1.12.0
ipywidgets	7.4.2	parso	0.3.4	terminado	0.8.2
jedi	0.13.3	pexpect	4.6.0	testpath	0.4.2
jinja2	2.10	pickleshare	0.7.5	tornado	6.0.2
jsonschema	3.0.1	prometheus-client	0.6.0	traitlets	4.3.2
jupyter	1.0.0	prompt-toolkit	2.0.9	wcwidth	0.1.7
jupyter-client	5.2.4	ptyprocess	0.6.0	webencodings	0.5.1
jupyter-console	6.0.0	pygments	2.3.1	widgetsnbextension	3.4.2
jupyter-core	4.4.0	pyparsing	2.3.1		

C2 Abbildungen zu ClassificaIO

Nachfolgend finden sich die beiden Auswahlbildschirme für ClassificaIO. Der erste zeigt die Abbildung nach der Entscheidung, die eigenen Trainingsdaten zu nutzen. Die zweite bildet die Darstellung ab, welche bei der Auswahl bereits ein Modell trainiert zu haben erscheint.





Anlage D

D1 Besonderheiten des Deep Learning Studios

Neben dem in der Arbeit thematisierten Onlinezwang gibt es noch weitere Aspekte die bei dieser Lösung auffallen. So nutzt das Deep Learning Studio neben Jupyter unter anderem Olympus – ein Werkzeug um Kommandozeilen-basiert jegliche Deep-Learning-Modelle in REST-APIs umzuwandeln – welches wiederum eine Abhängigkeitsliste von neunundsechzig quelloffenen Bibliotheken in der Datei „requirements“ im eigenen Hauptordner aufführt. Unter der Adresse „C:\Users\MachineLearner\AppData\Local\Programs\DeepLearningStudio\home\olympus“ findet sich diese im genutzten System wieder. Dabei stellt „MachineLearner“ den Nutzernamen dar. Eine Kopie des gesamten Verzeichnisses findet sich auf dem mitgelieferten Datenträger im Ordner „DeepLearningStudio“. Abbildungen zum Beleg der getätigten Feststellungen finden sich im Ordner „DLS_PROOF“.

Eine weitere Besonderheit ist das ausführliche Loggen der Nutzeraktivitäten auf dem System. In diesen Aufzeichnungen findet sich unter anderem der vom Autor, prophylaktisch schwierig de-anonymisierbar, gewählte Nutzername samt Fake-E-Mail-Adresse. Das Protokoll befindet sich unter „\var\logs\requests.log“ im Hauptordner der Software.

Die im Punkt 4.2.2 genannte Hochladepflicht bezüglich eigener Datensätze findet sich in einem Anleitungsvideo, welches nach der Registrierung beim Anbieter ersichtlich ist [DLS_VID]. Es findet sich auch im Ordner „DLS_PROOF“ als „DLS_VID.mp4“.

Anlage E

E1 JupyterLab Dateiformate

Diese Abbildung zeigt eine Vergrößerung der Abbildung 6 aus Punkt 2.1.2. Sie listet die von JupyterLab nativ unterstützten Dateiformate auf.