Python Tutorial

Projekt Collaborative Writing

Hochschule Kaiserslautern

Inhaltsverzeichnis

Kapitel 1

Grundlagen

Kapitel 2

Benutzeroberflächen

Kapitel 3

Python Bibliotheken

Kapitel 4

Weiterführende Themen

4.1 Maschinelles Lernen in Python

Das Themengebiet des maschinellen Lernens kann verschiedene Stufen von Komplexität erreichen. Grundlegend ist das mathematische Verständnis über die verschiedenen im maschinellen lernen eingesetzten Algorithmen. Diese werden in diesem Tutorial nicht beschrieben.
Sind die Algorithmen bekannt und sollen nun mittels Python angewendet werden, sollte zu Beginn mit einem kleinen Projekt begonnen werden. Hierbei ist es sinnvoll sich bereits zu Beginn eine Vorgehensweise zu überlegen, wie auch bei allen anderen Projekten. Python bietet im Bereich des maschinellen Lernens viele unterschiedliche Möglichkeiten an, sodass bereits frühzeitig der Aufbau des Projekts entschieden werden sollte. Grob kann ein Projekt in fünf Schritte aufgeteilt werden, anhand derer später das Ergebnis verifiziert werden kann.

1. Problem definieren
2. Daten vorbereiten
3. Algorithmen evaluieren
4. Ergebnisse verbessern
5. Ergebnisse darstellen

Eine weit verbreitete Möglichkeit ist das Einbinden von bereits existierenden Bibliotheken, die bereits gewissen Funktionalitäten von Haus auf anbieten. Eine eigene Nachbildung von verbreiteten Algorithmen aus dem Bereich Maschinelles Lernen ist daher meist nicht nötig.

Eine zweite Möglichkeit ist die Integration von R. Bei R handelt es sich um eine eigene Programmiersprache, welche den Schwerpunkt in mathematischen Problemlösungen hat. Python und R lassen sich beide sowohl eigenständig, also auch in Verbindung miteinander einsetzen.

Im Folgenden Abschnitt gibt es eine Übersicht, über wichtige und bekannte Bibliotheken aus dem Bereich maschinelles Lernen. Die Anzahl der Bibliotheken macht den Einstieg nicht ganz leicht. Die Stärken und Schwächen der einzelnen Bibliotheken sollten betrachtet werden.

4.1.1 Bekannte Bibliotheken

Im Bereich maschinelles Lernen sind schon viele Bibliotheken vorhanden, die unterschiedliche Schwerpunkte in dem Bereich bedienen. Aus diesem Grund erfolgt zuerst eine Übersicht über verbreitete Bibliotheken.

Download und Installation
Alle Bibliotheken die genutzt werden sollen müssen zuerst installiert werden. Hierfür gibt es je nach Bibliothek und teilweise je nach Betriebssystem mehrere Wege. Es wird empfohlen hier aus der jeweiligen Webseite die geeignetsten Variante zu wählen und auszuführen.

Versionen und Kompatibilität
Nach der Installation sollten alle Versionen ausgelesen und abgeglichen werden. Die Kompatibilität ist nicht durchgehend gewährleistet, das betrifft vor allem die unterschiedlichen Python-Versionen.

Der folgende Codeausschnitt zeigt ein Beispiel. Dieser kann entweder direkt in der Eingabeaufforderung bzw. Konsole nach Start von Python ausgeführt werden oder innerhalb einer geeigneten Entwicklungsumgebung.

```
# Python version
import sys
print("Pyton: {}".format(sys.version))

# numpy version
import numpy
print("numpy: {}".format(numpy.__version__))

# pandas version
```

```
import pandas
print("pandas: {}".format(pandas.__version__))

# rpy2 version
import rpy2
print("rpy2: {}".format(rpy2.__version__))
```

Die Ausgabe ist beispielsweise die folgende:

```
Pyton: 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52)
        [MSC v.1900 32 bit (Intel)]

numpy: 1.13.3
pandas: 0.21.0
rpy2: 2.8.6
```

Auf diese Art und Weise sollte alle Bibliotheken geprüft werden, da so eventuelle Kompatibilitätsprobleme oder fehlerhafte Installation erkannt werden können.

Bibliotheken

Grob kann man zwischen Datenanalyse und Visualisierung unterscheiden. Zwei bekannte Bibliotheken aus dem Bereich Datenanalyse sind *numpy* und *pandas*, mit denen beispielsweise Gleichungen und Optimierungsprobleme gelöst werden können. Außerdem lassen sich Integrale berechnen, statistische Berechnungen durchführen und auch simulieren. All das wird für maschinelles Lernen benötigt. Da die Berechnungen mit Routinen nah an der Hardware durchgeführt werden, lassen sich bei entsprechender Programmierung effiziente Programme schreiben. Ergebnisse solcher Berechnungen können mithilfe des Moduls *matplotlib* visualisiert werden. [Kas15]

numpy
In der Bibliothek *numpy* [num] wird ein flexibler Datentyp für mehrdimensionale Arrays zur Verfügung gestellt. Dies ermöglicht eine effiziente Durchführung von komplexen Rechnungen.

Die Arrays in *numpy* sind dreidimensional und können so eine Vielzahl von Anwendungsfällen abbilden. Der Fokus von *numpy* liegt in der Datenhaltung und Manipulation von Daten. Hier speziell die numerische Manipulation aus dem Bereich der linearen Algebra. Mit den Matrizen können bei-

spielsweise Multiplikationen und Dekompositionen durchgeführt werden. Aus diesem Grund sind *numpy*-Array oft die Datenstruktur, mit der weiterführende Bibliotheken arbeiten können.

pandas
Die Webseite zum *pandas*-Paket [pan] beschreibt dieses als gute Wahl zur schnellen und flexiblen Aufbereitung von Daten. *pandas* bietet verschiedene Möglichkeiten, um schnell auf Einträge zuzugreifen. Dies ist möglich, da mittels *pandas* Serien und Dataframes erzeugt werden können, die im Gegensatz zu einem Array in Python auch Spaltentitel und Indexes können.

Die *describe()*-Methode bietet hierbei einen ersten Überblick über die im Dataframe enthalten Daten. Ohne weitere Programmierung werden Informationen wie Maximalwert, Minimalwert und Durchschnitte für jede Spalte berechnet und angezeigt.

Spalten und Zeilen können mittels *pandas* gefiltert, erweitert und verändert werden, sodass Pandas oft im ersten Schritt genutzt wird, um die auszuwertenden Daten zu laden und genauer analysieren zu können.

scipy
Ergänzend bzw. aufbauen auf *numpy* werden durch *scipy* viele mathematische Operationen bereit gestellt. Das Modul *scipy* ist sehr mächtig und daher nochmal in Untermodule aufgeteilt. Innerhalb der Untermodule werden bestimmte Funktionalitäten gruppiert. Eine Übersicht hierzu gibt die Onlinedokumentation [scib].

scikit-learn
Viele DataMining bzw. Machine Learning Funktionalitäten werden bereits durch die *scikit-learn* Bibliothek (manchmal aus sklearn abgekürzt) zur Verfügung gestellt. Die klassischen Algorithmen, wie *k-Means* oder *knn* sind bereits integriert. Des Weiteren ist auch mit *scikit-learn* eine Aufbereitung der Daten möglich. Hier werden beispielsweise Normalierung und Skalierung unterstützt. Insgesamt handelt es sich um eine sehr mächtige Bibliothek, mit der es möglich ist unterschiedliche Algorithmen zu probieren und verschiedene Test- und Trainingsverfahren zu testen. Es können auch neue Daten anhand der gelernten Modelle klassifiziert und vorhergesagt werden. [scia]

rpy2
Eine weitere Möglichkeit ist das Einbinden des Pakets *rpy2* [rpy]. Hierbei handelt es sich um eine Bibliothek, welche Komponenten aus R zur Verfü-

gung stellt. *rpy2* ist zu sehen wie eine Schnittstelle zwischen Python und R. Es ist möglich R-Pakete mittels Python zu importieren und mit den darin enthaltenen Funktionalitäten zu interagieren.

matplotlib
Mit dem Modul *matplotlib* können Daten in einem Diagramm dargestellt werden. Hiermit kann ein erstes Verständnis der Daten oder Ergebnisse erreicht werden. Es werden unter anderem Liniendiagramme, Histogramme, Balkendiagramme aber auch Heatmaps unterstützt. Hier können sowohl Achsen, Farben und auch Beschriftungen nach Bedarf angepasst werden. *matplotlib* unterstützt sowohl *pandas* als auch *numpy* und ist oft daher bereits am Anfang von hoher Bedeutung, um einen Überblick über die Daten zu bekommen. [mat]

Literaturverzeichnis

[Kas15] KASIER, ERNESTI: *Python 3 Das umfassende Handbuch*. Rheinwerk VerlagGBmbH, 2015.

[mat] *matplotlib.org*. https://matplotlib.org/. zuletzt gesehen am 24.10.2018.

[num] *numpyreference*. https://docs.scipy.org/doc/numpy/reference/index.html. zuletzt gesehen am 24.10.2018.

[pan] *pandas*. http://pandas.pydata.org/pandas-docs/stable/. zuletzt gesehen am 24.10.2018.

[rpy] *rpy2*. https://pypi.org/project/rpy2/. zuletzt gesehen am 01.11.2018.

[scia] *scikit-learn*. http://scikit-learn.org/stable/index.html. zuletzt gesehen am 01.11.2018.

[scib] *scipyreference*. https://docs.scipy.org/doc/scipy/reference/. zuletzt gesehen am 24.10.2018.