



mitp

Sebastian  
Raschka

Vahid  
Mirjalili

3.,  
aktualisierte  
und erweiterte  
Auflage

# Machine Learning mit **Python** und Keras, TensorFlow 2 und Scikit-learn

Das umfassende **Praxis-Handbuch**  
für Data Science, Deep Learning und Predictive Analytics



## **Hinweis des Verlages zum Urheberrecht und Digitalen Rechtemanagement (DRM)**

Liebe Leserinnen und Leser,

dieses E-Book, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Mit dem Kauf räumen wir Ihnen das Recht ein, die Inhalte im Rahmen des geltenden Urheberrechts zu nutzen. Jede Verwertung außerhalb dieser Grenzen ist ohne unsere Zustimmung unzulässig und strafbar. Das gilt besonders für Vervielfältigungen, Übersetzungen sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Je nachdem wo Sie Ihr E-Book gekauft haben, kann dieser Shop das E-Book vor Missbrauch durch ein digitales Rechtemanagement schützen. Häufig erfolgt dies in Form eines nicht sichtbaren digitalen Wasserzeichens, das dann individuell pro Nutzer signiert ist. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

Beim Kauf des E-Books in unserem Verlagsshop ist Ihr E-Book DRM-frei.

Viele Grüße und viel Spaß beim Lesen,

*Ihr mitp-Verlagsteam*



Neuerscheinungen, Praxistipps, Gratiskapitel,  
Einblicke in den Verlagsalltag –  
gibt es alles bei uns auf Instagram und Facebook



[instagram.com/mitp\\_verlag](https://www.instagram.com/mitp_verlag)



[facebook.com/mitp.verlag](https://www.facebook.com/mitp.verlag)

# Inhaltsverzeichnis

**Impressum**

**Über die Autoren**

**Über die Korrektoren**

**Über den Fachkorrektor der deutschen Ausgabe**

**Einleitung**

Einstieg in Machine Learning

Zum Inhalt des Buches

Was Sie benötigen

Codebeispiele herunterladen

Konventionen im Buch

**Kapitel 1: Wie Computer aus Daten lernen können**

1.1 Intelligente Maschinen, die Daten in Wissen verwandeln

1.2 Die drei Arten des Machine Learnings

1.2.1 Mit überwachtem Lernen Vorhersagen treffen

1.2.2 Interaktive Aufgaben durch Reinforcement Learning lösen

1.2.3 Durch unüberwachtes Lernen verborgene Strukturen erkennen

1.3 Grundlegende Terminologie und Notation

1.3.1 Im Buch verwendete Notation und Konventionen

- 1.3.2 Terminologie
- 1.4 Entwicklung eines Systems für das Machine Learning
  - 1.4.1 Vorverarbeitung: Daten in Form bringen
  - 1.4.2 Trainieren und Auswählen eines Vorhersagemodells
  - 1.4.3 Bewertung von Modellen und Vorhersage anhand unbekannter Dateninstanzen
- 1.5 Machine Learning mit Python
  - 1.5.1 Python und Python-Pakete installieren
  - 1.5.2 Verwendung der Python-Distribution Anaconda
  - 1.5.3 Pakete für wissenschaftliches Rechnen, Data Science und Machine Learning
- 1.6 Zusammenfassung

## **Kapitel 2: Lernalgorithmen für die Klassifikation trainieren**

- 2.1 Künstliche Neuronen: Ein kurzer Blick auf die Anfänge des Machine Learnings
  - 2.1.1 Formale Definition eines künstlichen Neurons
  - 2.1.2 Die Perzeptron-Lernregel
- 2.2 Implementierung eines Perzeptron-Lernalgorithmus in Python
  - 2.2.1 Eine objektorientierte Perzeptron-API
  - 2.2.2 Trainieren eines Perzeptron-Modells mit der Iris-Datensammlung
- 2.3 Adaptive lineare Neuronen und die Konvergenz des Lernens
  - 2.3.1 Straffunktionen mit dem Gradientenabstiegsverfahren minimieren

2.3.2 Implementierung eines adaptiven linearen Neurons in Python

2.3.3 Verbesserung des Gradientenabstiegsverfahrens durch Merkmalstandardisierung

2.3.4 Large Scale Machine Learning und stochastisches Gradientenabstiegsverfahren

2.4 Zusammenfassung

## **Kapitel 3: Machine-Learning-Klassifikatoren mit scikit-learn verwenden**

3.1 Auswahl eines Klassifikationsalgorithmus

3.2 Erste Schritte mit scikit-learn: Trainieren eines Perzeptrons

3.3 Klassenwahrscheinlichkeiten durch logistische Regression modellieren

3.3.1 Logistische Regression und bedingte Wahrscheinlichkeiten

3.3.2 Gewichte der logistischen Straffunktion ermitteln

3.3.3 Konvertieren einer Adaline-Implementierung in einen Algorithmus für eine logistische Regression

3.3.4 Trainieren eines logistischen Regressionsmodells mit scikit-learn

3.3.5 Überanpassung durch Regularisierung verhindern

3.4 Maximum-Margin-Klassifikation mit Support Vector Machines

3.4.1 Maximierung des Randbereichs

3.4.2 Handhabung des nicht linear trennbaren Falls mit Schlupfvariablen

- 3.4.3 Alternative Implementierungen in scikit-learn
- 3.5 Nichtlineare Aufgaben mit einer Kernel-SVM lösen
  - 3.5.1 Kernel-Methoden für linear nicht trennbare Daten
  - 3.5.2 Mit dem Kernel-Trick Hyperebenen in höherdimensionalen Räumen finden
- 3.6 Lernen mit Entscheidungsbäumen
  - 3.6.1 Maximierung des Informationsgewinns: Daten ausreizen
  - 3.6.2 Konstruktion eines Entscheidungsbaums
  - 3.6.3 Mehrere Entscheidungsbäume zu einem Random Forest kombinieren
- 3.7 k-Nearest-Neighbors: Ein Lazy-Learning-Algorithmus
- 3.8 Zusammenfassung

## **Kapitel 4: Gut geeignete Trainingsdatensmengen: Datenvorverarbeitung**

- 4.1 Umgang mit fehlenden Daten
  - 4.1.1 Fehlende Werte in Tabellendaten
  - 4.1.2 Instanzen oder Merkmale mit fehlenden Daten entfernen
  - 4.1.3 Fehlende Werte ergänzen
  - 4.1.4 Die Schätzer-API von scikit-learn
- 4.2 Handhabung kategorialer Daten
  - 4.2.1 Codierung kategorialer Daten mit pandas
  - 4.2.2 Zuweisung von ordinalen Merkmalen
  - 4.2.3 Codierung der Klassenbezeichnungen
  - 4.2.4 One-hot-Codierung der nominalen Merkmale
- 4.3 Aufteilung einer Datensammlung in Trainings- und Testdaten

- 4.4 Anpassung der Merkmale
- 4.5 Auswahl aussagekräftiger Merkmale
  - 4.5.1 L1- und L2-Regularisierung als Straffunktionen
  - 4.5.2 Geometrische Interpretation der L2-Regularisierung
  - 4.5.3 Dünn besetzte Lösungen mit L1-Regularisierung
  - 4.5.4 Algorithmen zur sequenziellen Auswahl von Merkmalen
- 4.6 Beurteilung der Bedeutung von Merkmalen mit Random Forests
- 4.7 Zusammenfassung

## **Kapitel 5: Datenkomprimierung durch Dimensionsreduktion**

- 5.1 Unüberwachte Dimensionsreduktion durch Hauptkomponentenanalyse
  - 5.1.1 Schritte bei der Hauptkomponentenanalyse
  - 5.1.2 Schrittweise Extraktion der Hauptkomponenten
  - 5.1.3 Totale Varianz und erklärte Varianz
  - 5.1.4 Merkmalstransformation
  - 5.1.5 Hauptkomponentenanalyse mit scikit-learn
- 5.2 Überwachte Datenkomprimierung durch lineare Diskriminanzanalyse
  - 5.2.1 Hauptkomponentenanalyse kontra lineare Diskriminanzanalyse
  - 5.2.2 Die interne Funktionsweise der linearen Diskriminanzanalyse
  - 5.2.3 Berechnung der Streumatrizen

5.2.4 Auswahl linearer Diskriminanten für den neuen Merkmalsunterraum

5.2.5 Projektion in den neuen Merkmalsraum

5.2.6 LDA mit scikit-learn

5.3 Kernel-Hauptkomponentenanalyse für nichtlineare Zuordnungen verwenden

5.3.1 Kernel-Funktionen und der Kernel-Trick

5.3.2 Implementierung einer Kernel-Hauptkomponentenanalyse in Python

5.3.3 Projizieren neuer Datenpunkte

5.3.4 Kernel-Hauptkomponentenanalyse mit scikit-learn

5.4 Zusammenfassung

## **Kapitel 6: Bewährte Verfahren zur Modellbewertung und Hyperparameter-Optimierung**

6.1 Arbeitsabläufe mit Pipelines optimieren

6.1.1 Die Wisconsin-Brustkrebs-Datensammlung

6.1.2 Transformer und Schätzer in einer Pipeline kombinieren

6.2 Beurteilung des Modells durch k-fache Kreuzvalidierung

6.2.1 Holdout-Methode

6.2.2 k-fache Kreuzvalidierung

6.3 Algorithmen mit Lern- und Validierungskurven debuggen

6.3.1 Probleme mit Bias und Varianz anhand von Lernkurven erkennen

6.3.2 Überanpassung und Unteranpassung anhand von Validierungskurven erkennen

## 6.4 Feinabstimmung eines Lernmodells durch Grid Search

6.4.1 Optimierung der Hyperparameter durch Grid Search

6.4.2 Algorithmenauswahl durch verschachtelte Kreuzvalidierung

## 6.5 Verschiedene Kriterien zur Leistungsbewertung

6.5.1 Interpretation einer Verwechslungsmatrix

6.5.2 Optimierung der Genauigkeit und der Trefferquote eines Klassifikationsmodells

6.5.3 Receiver-Operating-Characteristic-Diagramme

6.5.4 Bewertungskriterien für Mehrklassen-Klassifikationen

## 6.6 Handhabung unausgewogener Klassenverteilung

## 6.7 Zusammenfassung

# **Kapitel 7: Kombination verschiedener Modelle für das Ensemble Learning**

## 7.1 Ensemble Learning

## 7.2 Klassifikatoren durch Mehrheitsentscheidung kombinieren

7.2.1 Implementierung eines einfachen Mehrheitsentscheidungs-Klassifikators

7.2.2 Vorhersagen nach dem Mehrheitsentscheidungsprinzip treffen

## 7.3 Bewertung und Abstimmung des Klassifikator-Ensembles

## 7.4 Bagging: Klassifikator-Ensembles anhand von Bootstrap-Stichproben entwickeln

7.4.1 Bagging kurz zusammengefasst

7.4.2 Klassifikation der Wein-Datensammlung durch Bagging

7.5 Schwache Klassifikatoren durch adaptives Boosting verbessern

7.5.1 Funktionsweise des Boostings

7.5.2 AdaBoost mit scikit-learn anwenden

7.6 Zusammenfassung

## **Kapitel 8: Machine Learning zur Analyse von Stimmungslagen nutzen**

8.1 Die IMDb-Filmdatenbank

8.1.1 Herunterladen der Datensammlung

8.1.2 Vorverarbeiten der Filmbewertungsdaten

8.2 Das Bag-of-words-Modell

8.2.1 Wörter in Merkmalsvektoren umwandeln

8.2.2 Beurteilung der Wortrelevanz durch das Tf-idf-Maß

8.2.3 Textdaten bereinigen

8.2.4 Dokumente in Tokens zerlegen

8.3 Ein logistisches Regressionsmodell für die Dokumentklassifikation trainieren

8.4 Verarbeitung großer Datenmengen: Online-Algorithmen und Out-of-Core Learning

8.5 Topic Modeling mit latenter Dirichlet-Allokation

8.5.1 Aufteilung von Texten mit der LDA

8.5.2 LDA mit scikit-learn

8.6 Zusammenfassung

## **Kapitel 9: Einbettung eines Machine-Learning-Modells in eine Webanwendung**

- 9.1 Serialisierung angepasster Schätzer mit scikit-learn
- 9.2 Einrichtung einer SQLite-Datenbank zum Speichern von Daten
- 9.3 Entwicklung einer Webanwendung mit Flask
  - 9.3.1 Die erste Webanwendung mit Flask
  - 9.3.2 Formularvalidierung und -ausgabe
- 9.4 Der Filmbewertungsklassifikator als Webanwendung
  - 9.4.1 Dateien und Ordner – die Verzeichnisstruktur
  - 9.4.2 Implementierung der Hauptanwendung app.py
  - 9.4.3 Einrichtung des Bewertungsformulars
  - 9.4.4 Eine Vorlage für die Ergebnisseite erstellen
- 9.5 Einrichtung der Webanwendung auf einem öffentlich zugänglichen Webserver
  - 9.5.1 Erstellen eines Benutzerkontos bei PythonAnywhere
  - 9.5.2 Hochladen der Filmbewertungsanwendung
  - 9.5.3 Updaten des Filmbewertungsklassifikators
- 9.6 Zusammenfassung

## **Kapitel 10: Vorhersage stetiger Zielvariablen durch Regressionsanalyse**

- 10.1 Lineare Regression
  - 10.1.1 Ein einfaches lineares Regressionsmodell
  - 10.1.2 Multiple lineare Regression
- 10.2 Die Boston-Housing-Datensammlung
  - 10.2.1 Einlesen der Datenmenge in einen DataFrame
  - 10.2.2 Visualisierung der wichtigen Eigenschaften einer Datenmenge

10.2.3 Zusammenhänge anhand der Korrelationsmatrix erkennen

10.3 Implementierung eines linearen Regressionsmodells mit der Methode der kleinsten Quadrate

10.3.1 Berechnung der Regressionsparameter mit dem Gradientenabstiegsverfahren

10.3.2 Schätzung der Koeffizienten eines Regressionsmodells mit scikit-learn

10.4 Anpassung eines robusten Regressionsmodells mit dem RANSAC-Algorithmus

10.5 Bewertung der Leistung linearer Regressionsmodelle

10.6 Regularisierungsverfahren für die Regression einsetzen

10.7 Polynomiale Regression: Umwandeln einer linearen Regression in eine Kurve

10.7.1 Hinzufügen polynomialer Terme mit scikit-learn

10.7.2 Modellierung nichtlinearer Zusammenhänge in der Boston-Housing-Datensammlung

10.8 Handhabung nichtlinearer Beziehungen mit Random Forests

10.8.1 Entscheidungsbaum-Regression

10.8.2 Random-Forest-Regression

10.9 Zusammenfassung

## **Kapitel 11: Verwendung von Daten ohne Label: Clusteranalyse**

11.1 Gruppierung von Objekten nach Ähnlichkeit mit dem k-Means-Algorithmus

- 11.1.1 k-Means-Clustering mit scikit-learn
- 11.1.2 Der k-Means++-Algorithmus
- 11.1.3 »Crisp« und »soft« Clustering
- 11.1.4 Die optimale Anzahl der Cluster mit dem Ellenbogenkriterium ermitteln
- 11.1.5 Quantifizierung der Clustering-Güte mit Silhouettendiagrammen
- 11.2 Cluster als hierarchischen Baum organisieren
  - 11.2.1 Gruppierung von Clustern
  - 11.2.2 Hierarchisches Clustering mittels einer Distanzmatrix
  - 11.2.3 Dendrogramme und Heatmaps verknüpfen
  - 11.2.4 Agglomeratives Clustering mit scikit-learn
- 11.3 Bereiche hoher Dichte mit DBSCAN ermitteln
- 11.4 Zusammenfassung

## **Kapitel 12: Implementierung eines künstlichen neuronalen Netzes**

- 12.1 Modellierung komplexer Funktionen mit künstlichen neuronalen Netzen
  - 12.1.1 Einschichtige neuronale Netze
  - 12.1.2 Mehrschichtige neuronale Netzarchitektur
  - 12.1.3 Aktivierung eines neuronalen Netzes durch Vorwärtspropagation
- 12.2 Klassifikation handgeschriebener Ziffern
  - 12.2.1 Die MNIST-Datensammlung
  - 12.2.2 Implementierung eines mehrschichtigen Perzeptrons
- 12.3 Trainieren eines künstlichen neuronalen Netzes
  - 12.3.1 Berechnung der logistischen Straffunktion

12.3.2 Ein Gespür für die Backpropagation entwickeln

12.3.3 Trainieren neuronaler Netze durch Backpropagation

12.4 Konvergenz in neuronalen Netzen

12.5 Abschließende Bemerkungen zur Implementierung neuronaler Netze

12.6 Zusammenfassung

## **Kapitel 13: Parallelisierung des Trainings neuronaler Netze mit TensorFlow**

13.1 TensorFlow und Trainingsleistung

13.1.1 Herausforderungen

13.1.2 Was genau ist TensorFlow?

13.1.3 TensorFlow erlernen

13.2 Erste Schritte mit TensorFlow

13.2.1 TensorFlow installieren

13.2.2 Tensoren in TensorFlow erstellen

13.2.3 Datentyp und Format eines Tensors ändern

13.2.4 Anwendung mathematischer Operationen auf Tensoren

13.2.5 Tensoren aufteilen, stapeln und verknüpfen

13.3 Eingabe-Pipelines mit tf.data erstellen – die Dataset-API von TensorFlow

13.3.1 Ein TensorFlow-Dataset anhand vorhandener Tensoren erstellen

13.3.2 Zwei Tensoren zu einer Datenmenge vereinen

13.3.3 Durchmischen, Batch erstellen und wiederholen

13.3.4 Erstellen einer Datenmenge anhand lokal gespeicherter Dateien

13.3.5 Zugriff auf die Datenmengen der tensorflow\_datasets-Bibliothek

13.4 Entwicklung eines NN-Modells mit TensorFlow

13.4.1 Die Keras-API (tf.keras) von TensorFlow

13.4.2 Entwicklung eines linearen Regressionsmodells

13.4.3 Trainieren des Modells mit den Methoden .compile() und .fit()

13.4.4 Entwicklung eines mehrschichtigen Perzeptrons zur Klassifikation der Iris-Datensammlung

13.4.5 Bewertung des trainierten Modells mit der Testdatenmenge

13.4.6 Das trainierte Modell speichern und einlesen

13.5 Auswahl der Aktivierungsfunktionen mehrschichtiger neuronaler Netze

13.5.1 Die logistische Funktion kurz zusammengefasst

13.5.2 Wahrscheinlichkeiten bei der Mehrklassen-Klassifikation mit der softmax-Funktion schätzen

13.5.3 Verbreiterung des Ausgabespektrums mittels Tangens hyperbolicus

13.5.4 Aktivierung mittels ReLU

13.6 Zusammenfassung

## **Kapitel 14: Die Funktionsweise von TensorFlow im Detail**

14.1 Grundlegende Merkmale von TensorFlow

## 14.2 TensorFlows Berechnungsgraphen: Migration nach TensorFlow v2

14.2.1 Funktionsweise von Berechnungsgraphen

14.2.2 Erstellen eines Graphen in TensorFlow v1.x

14.2.3 Migration eines Graphen nach TensorFlow v2

14.2.4 Eingabedaten einlesen mit TensorFlow v1.x

14.2.5 Eingabedaten einlesen mit TensorFlow v2

14.2.6 Beschleunigung von Berechnungen mit Funktionsdekoratoren

## 14.3 TensorFlows Variablenobjekte zum Speichern und Aktualisieren von Modellparametern

## 14.4 Gradientenberechnung durch automatisches Differenzieren und GradientTape

14.4.1 Berechnung der Gradienten der Verlustfunktion bezüglich trainierbarer Variablen

14.4.2 Berechnung der Gradienten bezüglich nicht trainierbarer Tensoren

14.4.3 Ressourcen für mehrfache Gradientenberechnung erhalten

## 14.5 Vereinfachung der Implementierung gebräuchlicher Architekturen mit der Keras-API

14.5.1 Lösen einer XOR-Klassifikationsaufgabe

14.5.2 Flexiblere Modellerstellung mit Keras' funktionaler API

14.5.3 Modelle mit Keras' »Model«-Klasse implementieren

14.5.4 Benutzerdefinierte Keras-Schichten

## 14.6 TensorFlows Schätzer

14.6.1 Merkmalsspalten

14.6.2 Machine Learning mit vorgefertigten Schätzern

14.6.3 Klassifikation handgeschriebener Ziffern mit Schätzern

14.6.4 Benutzerdefinierte Schätzer anhand eines Keras-Modells erstellen

14.7 Zusammenfassung

## **Kapitel 15: Bildklassifikation mit Deep Convolutional Neural Networks**

15.1 Bausteine von Convolutional Neural Networks

15.1.1 CNNs und Merkmalshierarchie

15.1.2 Diskrete Faltungen

15.1.3 Subsampling

15.2 Implementierung eines CNNs

15.2.1 Verwendung mehrerer Eingabe- oder Farbkanäle

15.2.2 Regularisierung eines neuronalen Netzes mit Dropout

15.2.3 Verlustfunktionen für Klassifikationen

15.3 Implementierung eines tiefen CNNs mit TensorFlow

15.3.1 Die mehrschichtige CNN-Architektur

15.3.2 Einlesen und Vorverarbeiten der Daten

15.3.3 Implementierung eines CNNs mit TensorFlows Keras-API

15.4 Klassifikation des Geschlechts anhand von Porträtfotos mit einem CNN

15.4.1 Einlesen der CelebA-Datenmenge

15.4.2 Bildtransformation und Datenaugmentation

15.4.3 Training eines CNN-Klassifikators

15.5 Zusammenfassung

## **Kapitel 16: Modellierung sequenzieller Daten durch rekurrente neuronale Netze**

### 16.1 Sequenzielle Daten

16.1.1 Modellierung sequenzieller Daten: Die Reihenfolge ist von Bedeutung

16.1.2 Repräsentierung von Sequenzen

16.1.3 Verschiedene Kategorien der Sequenzmodellierung

### 16.2 Sequenzmodellierung mit RNNs

16.2.1 Struktur und Ablauf eines RNNs

16.2.2 Aktivierungen eines RNNs berechnen

16.2.3 Rückkopplung mit der verdeckten Schicht oder der Ausgabeschicht

16.2.4 Probleme bei der Erkennung weitreichender Interaktionen

16.2.5 LSTM-Speicherzellen

### 16.3 Implementierung von RNNs zur Sequenzmodellierung mit TensorFlow

16.3.1 Projekt 1: Vorhersage der Stimmungslage von IMDb-Filmbewertungen

16.3.2 Projekt 2: Sprachmodellierung durch Zeichen mit TensorFlow

### 16.4 Sprache mit dem Transformer-Modell verstehen

16.4.1 Der Mechanismus der Selbst-Aufmerksamkeit

16.4.2 Multi-Head-Attention und Transformer-Block

### 16.5 Zusammenfassung

## **Kapitel 17: Synthetisieren neuer Daten mit Generative Adversarial Networks**

### 17.1 Einführung in GANs

- 17.1.1 Autoencoder
- 17.1.2 Generative Modelle zum Synthetisieren neuer Daten
- 17.1.3 Mit GANs neue Beispiele erzeugen
- 17.1.4 Die Verlustfunktion des Generator- und Diskriminator-Netzes in einem GAN-Modell
- 17.2 Ein GAN von Grund auf implementieren
  - 17.2.1 GAN-Modelle mit Google Colab trainieren
  - 17.2.2 Implementierung der Generator- und Diskriminator-Netze
  - 17.2.3 Definition der Trainingsdatenmenge
  - 17.2.4 Trainieren des GAN-Modells
- 17.3 Verbesserung der Qualität synthetisierter Bilder durch Convolutional GAN und Wasserstein-GAN
  - 17.3.1 Transponierte Faltung
  - 17.3.2 Batchnormierung
  - 17.3.3 Implementierung des Generators und des Diskriminators
  - 17.3.4 Maße für den Unterschied zwischen zwei Verteilungen
  - 17.3.5 Verwendung der EM-Distanz in der Praxis
  - 17.3.6 Strafterm
  - 17.3.7 Implementierung von WGAN-GP zum Trainieren des DCGAN-Modells
  - 17.3.8 Zusammenbrechen des Verfahrens
- 17.4 Weitere GAN-Anwendungen
- 17.5 Zusammenfassung

## **Kapitel 18: Entscheidungsfindung in komplexen Umgebungen per Reinforcement Learning**

- 18.1 Einführung: Aus Erfahrung lernen
  - 18.1.1 Reinforcement Learning
  - 18.1.2 Definition der Agent-Umgebung-Schnittstelle für ein Reinforcement-Learning-System
- 18.2 Theoretische Grundlagen des RLs
  - 18.2.1 Markov-Entscheidungsprozesse
  - 18.2.2 Mathematische Formulierung von Markov-Entscheidungsprozessen
  - 18.2.3 RL-Terminologie: Return, Policy und Wertfunktion
  - 18.2.4 Dynamische Programmierung und Bellman-Gleichung
- 18.3 Reinforcement-Learning-Algorithmen
  - 18.3.1 Dynamische Programmierung
  - 18.3.2 Reinforcement Learning mit Monte-Carlo-Algorithmen
  - 18.3.3 Temporal Difference Learning
- 18.4 Implementierung eines RL-Algorithmus
  - 18.4.1 OpenAI Gym
  - 18.4.2 Lösung der Grid-World-Aufgabe mit Q-Learning
  - 18.4.3 Ein Blick auf Deep Q-Learning
- 18.5 Zusammenfassung und Schlusswort

Sebastian Raschka  
Vahid Mirjalili

# **Machine Learning mit Python und Keras, TensorFlow 2 und Scikit- learn**

**Das umfassende Praxis-Handbuch für  
Data Science, Deep Learning und  
Predictive Analytics**

**3., aktualisierte und erweiterte Auflage**

Übersetzung aus dem Amerikanischen  
von Knut Lorenzen



**mitp**

# Impressum

## **Bibliografische Information der Deutschen Nationalbibliothek**

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

ISBN 978-3-7475-0215-0  
3. Auflage 2021

[www.mitp.de](http://www.mitp.de)

E-Mail: [mitp-verlag@sigloch.de](mailto:mitp-verlag@sigloch.de)

Telefon: +49 7953 / 7189 - 079

Telefax: +49 7953 / 7189 - 082

Copyright ©Packt Publishing 2019

First published in the English language under the title  
'Python Machine Learning - Third Edition -  
(9781789955750)'

© 2021 mitp Verlags GmbH & Co. KG

Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Lektorat: Janina Bahlmann

Fachkorrektur: Dr. Friedhelm Schwenker

Sprachkorrektur: Petra Heubach-Erdmann

Coverbild: Shantanu N. Zagade, © Packt Publishing

electronic **publication**: Ill-satz, Husby, [www.drei-satz.de](http://www.drei-satz.de)

Dieses Ebook verwendet das ePub-Format und ist optimiert für die Nutzung mit dem iBooks-reader auf dem iPad von Apple. Bei der Verwendung anderer Reader kann es zu Darstellungsproblemen kommen.

Der Verlag räumt Ihnen mit dem Kauf des ebooks das Recht ein, die Inhalte im Rahmen des geltenden Urheberrechts zu nutzen. Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und Einspeicherung und Verarbeitung in elektronischen Systemen.

Der Verlag schützt seine ebooks vor Missbrauch des Urheberrechts durch ein digitales Rechtemanagement. Bei Kauf im Webshop des Verlages werden die ebooks mit einem nicht sichtbaren digitalen Wasserzeichen individuell pro Nutzer signiert.

Bei Kauf in anderen ebook-Webshops erfolgt die Signatur durch die Shopbetreiber. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

# Über die Autoren

**Sebastian Raschka** erlangte seinen Dokortitel an der Michigan State University. Er befasst sich vornehmlich mit Fragen der Berechnung biologischer Phänomene und des Machine Learnings. Im Sommer 2018 wurde er Assistant Professor für Statistik an der University of Wisconsin-Madison. Bei seiner Forschungsarbeit geht es insbesondere um die Entwicklung neuer Deep-Learning-Architekturen zum Lösen von Aufgaben im Fachgebiet Biometrie.

Er verfügt über jahrelange Erfahrung in der Python-Programmierung und leitete mehrere Seminare über praktische Data-Science-Anwendungen, Machine Learning und Deep Learning, unter anderem eine Einführung in Machine Learning auf der SciPy-Konferenz, der maßgeblichen Veranstaltung für wissenschaftliche Anwendungen in Python.

Er ist Autor des viel verkauften Buches *Python Machine Learning*, das 2016 mit dem Preis ACM Computing Reviews Best of ausgezeichnet und in viele Sprachen übersetzt wurde, unter anderem Deutsch, Koreanisch, Chinesisch, Japanisch, Russisch, Polnisch und Italienisch.

In seiner Freizeit leistet Sebastian aktiv Beiträge zu Open-Source-Projekten und die von ihm implementierten Verfahren werden erfolgreich in Mustererkennungswettbewerben wie z.B. Kaggle eingesetzt.

---

Ich möchte diese Gelegenheit nutzen, der großartigen Python-Community und den Entwicklern der Open-Source-Pakete meinen Dank

auszusprechen, die mir dabei geholfen haben, die perfekte Umgebung für wissenschaftliche Forschung und Data Science einzurichten. Außerdem möchte ich meinen Eltern danken, die mich bei all meinen beruflichen Zielen, die ich so leidenschaftlich verfolgt habe, stets ermutigt und unterstützt haben.

Mein besonderer Dank gilt den Hauptentwicklern von scikit-learn und TensorFlow. Als jemand, der selbst aktiv an diesem Projekt beteiligt war, hatte ich das Vergnügen, mit tollen Leuten zusammenarbeiten zu dürfen, die sich nicht nur mit Machine Learning und Deep Learning auskennen, sondern auch hervorragende Programmierer sind.

---

**Vahid Mirjalili** erlangte seinen Dokortitel als Maschinenbauingenieur an der Michigan State University mit einer Arbeit über neue Verfahren für Computersimulationen molekularer Strukturen. Er interessiert sich leidenschaftlich für Machine Learning und trat dem iProBe-Lab der Michigan State University bei, wo er Anwendungen des Machine Learnings in verschiedenen Computer-Vision-Projekten («maschinelles Sehen») erforschte. Nach mehreren produktiven Jahren am iProBe-Lab und in der Forschung ist Vahid Mirjalili seit Kurzem beim Unternehmen 3M als Forscher tätig, wo er seine Kenntnisse einsetzen kann, um moderne Machine-Learning- und Deep-Learning-Verfahren auf Aufgabenstellungen aus der Praxis anzuwenden.

---

Ich möchte meiner Frau Taban Eslami danken, die mich während meiner Laufbahn stets unterstützt und ermutigt hat. Besonderer Dank gebührt meinen Mentoren Nikolai Priezjev, Michael Feig und Arun Ross, die mich während des Doktorats unterstützt haben, sowie meinen Professoren Vishnu Boddeti, Leslie Kuhn und Xiaoming Liu, die mich so vieles gelehrt haben und mich ermutigten, meiner Leidenschaft zu folgen.

---