Praktikumsbericht

Dominik Pegler (a01468373)

Machine Learning in der psychologischen Forschung

Zeitraum: 1.3.–30.4.2022

Betreuer: David Steyrl

Fakultät für Psychologie, Universität Wien

13. Mai 2022

Inhaltsverzeichnis

1 Kurzfassung

Im Praktikum wurde ein bestehender Datensatz aus einer psychologischen Untersuchung mit Methoden aus dem Machine-Learning (ML) analysiert, wozu eigens ein Script in der Programmiersprache Python angefertigt wurde. Weitere Applikationen von ML im psychologischen Kontext wurden diskutiert und Literaturrecherche betrieben.

2 Einstieg

Meine Interessen gehen in viele Richtungen, aber in den letzten zwei bis drei Jahren wurde mein Interesse für die Schnittstellen der Psychologie mit der Informatik stärker und damit auch der Wunsch Erfahrungen auf diesem Gebiet zu sammeln. Das hat auch damit zu tun, dass ich das Psychologiestudium berufsbedingt für eineinhalb Jahre unterbrechen musste und in dieser Zeit vermehrt mit der Programmiersprache Python und den Data-Science-Programmpaketen zu tun hatte. Als ich wieder ins Studium eingstiegen war und die Möglichkeit eines Pflichtpraktikums im Bereich Machine Learning auf der Fakultätsseite sah, war für mich der Wunsch da, dieses zu absolvieren.

Meine Erwartungen an das Praktikum waren hoch, was den Wissenserwerb betraf. Nicht so hoch waren die Erwartungen bezogen auf den persönlichen Austausch, da hier aufgrund der Pandemie Einschränkungen zu erwarten waren. David Steyrl lernte ich auch kurz zuvor in einer Vorlesung kennen, womit ich mir schon ein ungefähres Bild von seiner Herangehenswese machen konnte.

3 Praktikumsstelle

Der Forschungsbereich von Prof. Frank Scharnowski beschäftigt sich vor allem mit Neurofeedback und Computer-Brain-Interfaces. Ein Ziel seiner Forschunsgruppe ist es, ein besseres Verständnis zur Wirksamkeit von Konfrontationstherapie zu erlangen und diese Therapieform für den Einsatz im klinischen Bereich zu verbessern. Dabei werden neben psychologischen Merkmalen bildgebende Verfahren verwendet, um schlussendlich Closed-Loop-Computermodelle der Funktionsweise der Therapie zu erstellen. Weitere Schwerpunkte sind Neurofeedback mittels fMRT sowie Machine Learning für Datenanalysen. Mein Betreuer David Steyrl arbeitet als PostDoc im Team von Frank Scharnowski und ist insbesondere für die Durchführung von ML-Datenanalysen und das Erstellen von Computermodellen verantwortlich. Seine weiteren Forschungsinteressen umfassen Biosignalverarbeitung und simultane EEG-fMRT-Messungen. David Steyrl studierte Medizintechnik und Data Science an der TU Graz.

4 Ziele

Ziele des Praktikums waren für mich, eine Verbindung zwischen psychologischer Forschung und modernen Machine-Learning-Methoden herzustellen. Bisher kannte ich hier kaum

konkrete Anwendungsfälle. Mit dem Betreuer David Steyrl wurde daher vereinbart, einen bereits mit klassischen statistischen Methoden (Korrelationen, Moderationsanalyse) untersuchten Datensatz, der im Rahmen meiner Bachelorarbeit erhoben wurde und Zusammenhänge zwischen problematischem Alkoholkonsum und Mindfulness (sowie einer Reihe anderer Prädiktoren), mit ML-Methoden zu untersuchen. Dabei sollten nicht die Ergebnisse der früheren und der aktuellen Analyse gegenübergestellt werden, sondern ein Gefühl entwickelt werden, wie eine solche ML-Analyse in der Praxis funktioniert, da mir bisher die Anwendungsfälle fehlten. Weiteres Ziel war für mich eine Vertiefung des bisherigen Wissens in Form zusätzlicher Literaturrecherchen.

5 Ablauf

Die Dauer des Praktikums wurde mit mit 8 Wochen zu je 30 Arbeitsstunden pro Woche und aufgrund der Pandemiesituation auf einen Online-Modus festgelegt. Konkrekt bedeutete das für mich eine Arbeitswoche von Montag bis Samstag, in der mehrmalige Videocalls per Zoom mit meinem Betreuer stattfanden. Zusätzlich wurde immer wieder auch per Email oder Skype kommuniziert. Einmal wöchentlich fand ein Online-Meeting mit jenen MasterstudentInnen statt, die David Steyrl im Rahmen deren Masterarbeit betreute. Insgesamt schätze ich, dass sich die aufgewendete Zeit zu 50% auf Programmiertätigkeiten, etwa zu 10% auf die Kommunikation und zu weiteren 40% auf Literaturrecherche, Analyse und Ergebnissereflexion aufteilte.

6 Tätigkeit

6.1 Analyse des Datensatzes

Zu allererst ging es für mich darum, Vorkenntnisse aufzufrischen und die Methode der linearen Regression mit Python zu wiederhohlen, da diese als der Ausgangspunkt jeden maschinellen Lernens gesehen werden kann. Im nächsten Schritt wurden regularisierte lineare Regressionsmodelle und deren Hyperparameter hinzugenommen: „Ridge“-Regression, „Lasso“-Regression und deren Hybridversion: „Elastic Net“. Danach kamen non-lineare Modelle wie „Random Forests“, „Extra Trees“ und „Gradient Boosting“ ins Spiel. Aufgrund der Fragestellung mit einer kontinuierlichen Outcome-Variable wurden keine Klassifikationsalgorithmen in diesem Praktikum verwendet.

6.1.1 Der Datensatz

Der verwendete Datensatz stammte aus meiner eigenen Bachelorarbeit und wurde im Sommer 2021 über einen Online-Fragebogen erhoben. 500 Versuchspersonen machten darin u.a. Angaben zu ihrem Alkoholkonsum, Ausprägung von Impulsivität, Mindfulness und Bewältigungsmotive beim Alkoholkonsum. Zusammen mit den soziodemografischen Variablen wurden insgesamt 50 Variablen als Prädiktoren in den Analysen verwendet.

6.1.2 Erstellung des Scripts

Für die Analyse der Daten war es notwendig, ein Script zu erstellen, dass die folgenden Aufgaben erfüllt: (1) Einlesen der Daten, (2) Bereinigung der Daten, indem auf die relevanten Variablen reduziert wird, diese umbenannt und fehlende Werte imputiert werden, (3) Standardskalierung der metrischen Prädiktoren, (4) Aufteilung der Daten in Trainings- und Testsets, (5) äußere Kreuzvalidierung, um Overfitting zu vermeiden, (6) innere Kreuzvalidierung, um die optimalen Hyperparameter zu ermitteln, (7) Verfügung über ein Pool an Modellen (ElasticNet, ExtraTrees und GradientBoostRegressor) und dazu passende Hyperparameter-Räume, (8) jedes Modell einmal durchlaufen lassen, (9) Aufzeichnen der R^2-Scores, (10) Aufzeichnen der Shapely-Werte für die Wichtigkeit der einzelnen Prädiktoren.

Es dauerte dabei eine Weile, bis alle Fehler im Script beseitigt waren und es vernünftige Resulate lieferte. Um auch die letzte Skepsis zu beseitigen, wurden dieselben Daten zusätzlich mit einem fertigen Script von David Steyrl analysiert. Beide lieferten nahezu dieselben Ergebnisse.

6.1.3 Ergebnisdarstellung

Bis das Script fehlerfreie Resultate lieferte, wurde es dreimal komplett durchlaufen, wobei dies jeweils etwa 12 Stunden in Anspruch nahm. Die Resultate wurden anschließend für jeden Lauf in einem eigenen Jupyter-Notebook (verfügbar im Online-Repository) aufbereitet. Diese enthalten die Outputs des Scripts (R^2-Score jedes Modells pro Kreuzvalidierungs-Split) sowie die Plots zur Wichtigkeit der einzelnen Prädiktoren (Shapley-Werte). Abbildung 1 illustriert, welche Wichtigkeit die einzelnen Prädiktoren im jeweiligen Modell (im konkreten Beispiel der ExtraTreesRegressor aus scikit-learn) auf Basis der Shapley-Werte für die Variable Alkoholkonsum hatten. Abbildung 2 zeigt ein Beispiel, wie die Vorhersage einer einzelnen Versuchsperson mittels Shapley-Values erklärt werden kann. Die Höhe der Werte gibt die Wichtigkeit der einzelnen Prädiktoren für den Outcome an. Prädiktoren mit roten Balken drücken den Wert des Outcomes nach oben, bei blauen Balken ist es genau umgekehrt.

6.1.4 Online-Material

Zur besseren Nachvollziehbarkeit wurden alle Skripte, Notizen, Resultate sowie der verwendete Datensatz in einem eigens eingerichteten Online-Repository verfügbar gemacht (https://www.github.com/dominikpegler/internship_ml). Außerdem wurde eine einfache Webseite eingerichtet, die mir als Überblick über Fortschritt und anstehende Tasks diente (https://dominikpegler.github.io/internship_ml/).

6.2 Theorie: Künstliche neuronale Netzwerke

Im Zuge des Praktikums versuchte ich auch, die Grundprinzipien davon zu verstehen, wie künstliche neuronale Netzwerke lernen: Aktivierung und Aktualisierung der Gewichte

Abbildung 1

Zusammenfassung der Wichtigkeit der einzelnen Prädiktoren für die Variable Alkoholkonsum mittels „Shapley values“.

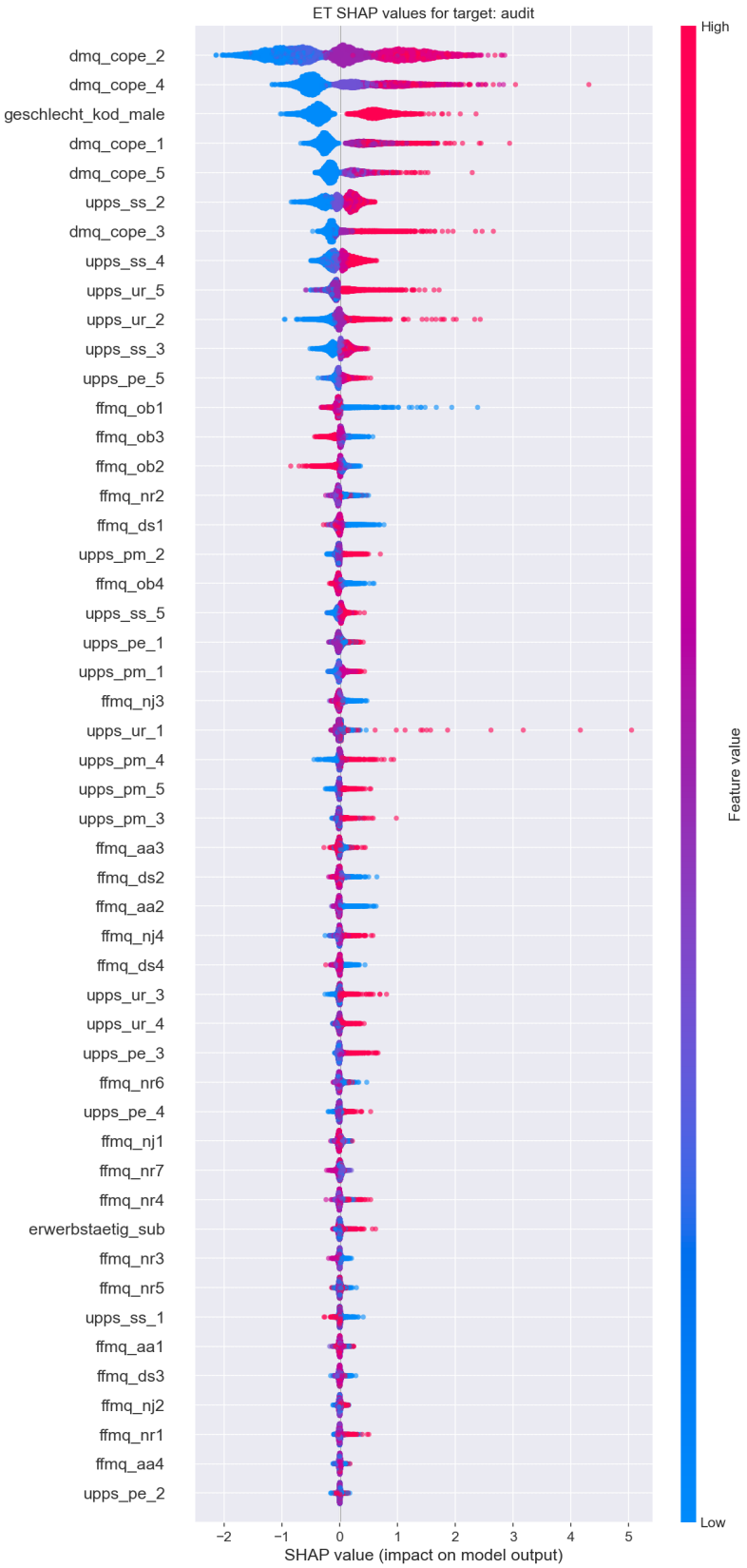
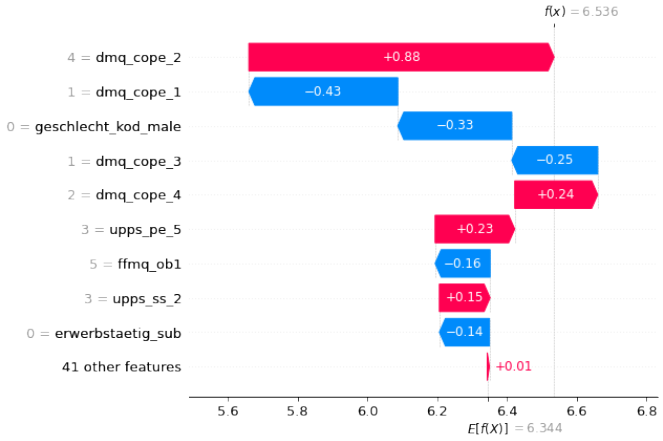


Abbildung 2

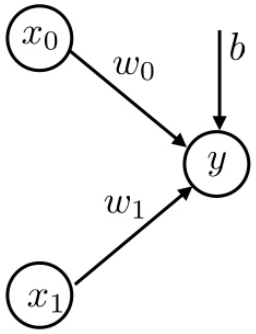
Erklärung der Vorhersage der Variable Alkoholkonsum mittels „Shapley values“



(Weights) und Biases. In Abbilung 3 ist die simpelste Form dargestellt: ein Neuron y mit zwei Eingängen x_0 und x_1, von deren Aktivierungsmuster (in diesem Fall die OR-Funktion) es abhängt, ob das Neuron feuert ($y = 1$) oder nicht ($y = 0$). Das Neuron wählt als Ausgangspunkt beim Lernen des OR-Zusammenhangs erst zufällige Werte für beide Gewichte w_0 und w_1) und den Bias (b). Über eine Aktivierungsfunktion (hier die Sigmoid-Funktion) wird aus den Eingangswerten, den Gewichten und dem Bias ein Wert berechnet, der entweder 0 oder 1 ergibt. Je nachdem, wie weit das Neuron mit seiner Berechnung vom tatsächlichen Wert abweicht („Error“), passt es seine Gewichte und den Bias an. Über mehrere Durchgänge („Epochs“) hinweg, wird die Schätzung des Neurons somit immer genauer (Abb. 4). Siehe hierzu die Gradientenberechnung in der nachfolgenden Implementation mit Programmpaket numpy:

Abbildung 3

„Netzwerk“ mit nur einem Neuron y lernt die OR-Funktion



logical OR

x_0	x_1	y
0	0	0
0	1	1
1	0	1
1	1	1

6.2.1 Beispiel: Training eines einzelnen Neurons in numpy

Input

```
import numpy as np
import matplotlib.pyplot as plt

# create input matrix X
X = np.zeros((4, 2), dtype = float)
X[0, :] = [0., 0.]
X[1, :] = [0., 1.]
X[2, :] = [1., 0.]
X[3, :] = [1., 1.]


# target vector
Y_or = np.array([0., 1., 1., 1.])

N = X.shape[0] # number of input patterns

# create arrays for weights and bias
w = np.random.randn(2)
b = np.random.randn(1)

alpha = 0.05 # learning rate
n_epochs = 5_000

def g_logistic(net):
    return 1. / (1. + np.exp(-net))

def loss(yhat, y):
    return (yhat - y)**2

def print_forward(x, yhat, y):
    print(f" input  = {x.astype(int)}")
    print(f" output = {yhat.item():.3f}")
    print(f" target = {y.item():.0g}")

def print_grad(grad_w, grad_b):
    print(f"  w_0 = {grad_w[0]: .3f}")
    print(f"  w_1 = {grad_w[1]: .3f}")
    print(f"  b   = {grad_b[0]: .3f}")

# Crucial part: Computing the gradients
# gradient is derivative of error (cost) divided by derivative of weights (or bias, respectively)
# x_i is not used, if calculated for bias is replaced by 1
def grad_func(y, yhat, net, g, x_i = 1):
    grad = 2 * (yhat - y) * g(net) * (1 - g(net)) * x_i
    return grad

track_error = []

for epoch in range(n_epochs):
    error_epoch = 0. # sum loss across the epoch
    perm = np.random.permutation(N)

    for p in perm: # visit data points in random order
        x = X[p, :] # input pattern

        # compute output of neuron
        net = np.dot(x, w) + b
        yhat = g_logistic(net)

        # compute loss
        y = Y_or[p]
        myloss = loss(yhat, y)
        error_epoch += myloss.item()

        # print output if this is the last epoch
        if (epoch == n_epochs - 1):
            print("\nFinal result:")
            print_forward(x, yhat, y)
            print("")
```

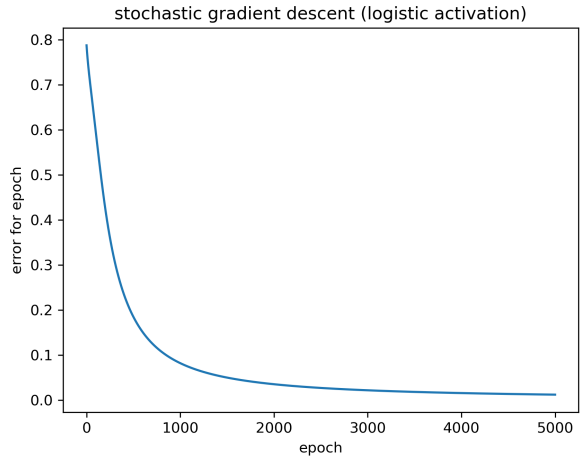
```
70
71          w_grad = grad_func(y, yhat, net, g_logistic, x)
72          b_grad = grad_func(y, yhat, net, g_logistic, 1)
73
74          # parameter update with gradient descent
75          w -= alpha * w_grad
76          b -= alpha * b_grad
77
78      track_error.append(error_epoch)
79      if epoch % 500 == 0:
80          print(f"epoch {epoch}")
81          print(f"  err = {error_epoch: .3f}")
82          print_grad(w_grad, b_grad)
83          print("")
84
85  fig, ax = plt.subplots()
86  ax.plot(track_error)
87  ax.set_title("stochastic gradient descent (logistic activation)")
88  ax.set_ylabel("error for epoch")
89  ax.set_xlabel("epoch")
90  fig.savefig("img/nn_OR_lr.png", dpi = 300)
```

```
epoch 0
  err =  1.081
  w_0 = -0.000
  w_1 = -0.029
  b   = -0.029

epoch 500
  err =  0.156
  w_0 = -0.001
  w_1 = -0.001
  b   = -0.001

epoch 1000
  err =  0.074
  w_0 =  0.000
  w_1 =  0.000
  b   =  0.068

epoch 1500
  err =  0.046
  w_0 = -0.000
  w_1 = -0.000
  b   = -0.000

epoch 2000
  err =  0.033
  w_0 = -0.000
  w_1 = -0.013
  b   = -0.013

epoch 2500
  err =  0.026
  w_0 = -0.010
  w_1 = -0.000
  b   = -0.010

epoch 3000
  err =  0.021
  w_0 = -0.000
  w_1 = -0.000
  b   = -0.000

epoch 3500
  err =  0.017
  w_0 = -0.000
  w_1 = -0.007
  b   = -0.007

epoch 4000
  err =  0.015
  w_0 = -0.000
  w_1 = -0.000
```

```
  b   = -0.000

epoch 4500
  err =  0.013
  w_0 =  0.000
  w_1 =  0.000
  b   =  0.014


Final result:
 input  = [0 1]
 output = 0.950
 target = 1


Final result:
 input  = [1 1]
 output = 1.000
 target = 1


Final result:
 input  = [1 0]
 output = 0.950
 target = 1


Final result:
 input  = [0 0]
 output = 0.081
 target = 0
```

Abbildung 4

Beispiel: Fehlerrate verringert sich mit zunehmender Anzahl an Durchgängen



7 Hilfreiche Literatur

Wichtige Literatur, die ich insbesondere in den ersten zwei Wochen immer wieder herangezogen habe, waren die offizielle Dokumentation des Softwarepakets Scikit-Learn (scikit-learn developers, 2022) und die erste Hälfte von Géron (2019). Diese behandelt klassische Machine-Learning-Algorithmen, wie sie in Scikit-Learn implementiert sind, die zweite Hälfte neuronale Netzwerke mit Tensorflow.

8 Reflexion

8.1 Bezug zu bisherigen Lehrinhalten

Im Verlauf des Praktikums war vor allem mein Vorwissen aus dem Bachelorstudium zu Statistik und Forschungsmethoden hilfreich.

Insgesamt seien hier folgende Lehrveranstaltungen genannt:

- Fachliteraturseminar und Bachelorarbeit bei Ulrich Tran
- Vorlesung Ausgewählte Methoden aus dem Bachelorstudium von Ulrich Tran
- Vorlesung Testtheorie aus dem Bachelorstudium von Michael Weber
- Vorlesung Differenzielle Psychologie von Georg Gittler
- Vorlesung Statistik für Fortgeschrittene aus dem Masterstudium von Ulrich Tran

Herausstreichen möchte ich hier die Themenblöcke zur multiplen linearen Regression aus der Vorlesungsreihe „Statistik für Fortgeschrittene“ von Ulrich Tran. Ein großer Teil der Tätigkeiten im Praktikum verlangt jedoch Wissen aus Bereichen, die wenig behandelt wurden (vor allem nicht im Bachelorstudium): Machine Learning und Programmierung. Das Praktikum erfordert daher entweder ein hohes Maß an Lernbereitschaft oder Vorwissen in diesen Bereichen.

8.2 Programmieren in Python

Aufgrund meiner Vorerfahrungen mit Programmierung, insbesondere mit Python, fiel es mir nicht schwer, den für die Analyse notwendigen Programmcode zu schreiben. Die Dokumentationen der Programmpakete sind frei zugänglich im Internet, sehr detailliert und einfach zu lesen. Es musste hauptsächlich der Zusammenhang zwischen der Theorie und der tatsächlichen Implementation in Form von Code hergestellt werden. Schwierigkeiten bereitete mir hier anfänglich das Konzept der Kreuzvalidierung, da ich diese in meinem Kopf als eine Schleife verinnerlicht hatte, im Programmpaket scikit-learn ist dies allerdings bereits auf eine einzelne Zeile Code reduziert (siehe nachfolgendes Beispiel). Das nimmt einem zwar viel Arbeit ab, jedoch dauerte es etwas festzustellen, dass sich hinter dieser einen Zeile tatsächlich eine Schleife verbirgt (nicht sichtbar für den Anwender, außer man wirft einen Blick in den Source-Code).

8.2.1 Beispiel: Innere Kreuzvalidierungsschleife

Input

```
# nested CV loop for parameter optimization
inner_cv = ShuffleSplit(n_splits = 50, test_size = 0.2)

# creating the regressor instance
```

```
# and passing it the previously created loop
reg_with_bayes_search_cv = BayesSearchCV(
    estimator = reg,
    search_spaces = hyper_space,
    n_iter = 200,
    cv=inner_cv,
    n_jobs = -2,
    random_state = 0)
```

8.3 Betreuung

Praktikumsbetreuer David Steyrl stand übers gesamte Praktikum hinweg zur Verfügung, wenn ich Fragen oder Schwierigkeiten mit den Aufgabestellungen hatte. Er hat mir im Erstgespräch und im späteren Verlauf immer wieder klar gesagt, was er von mir erwartet, und Feedback zu den von mir verrichteten Tätigkeiten gegeben. Sehr wertvoll war für mich, dass ich nie das Gefühl hatte, eine für mich wichtige Frage nicht stellen zu können. In den Gesprächen mit meinem Betreuer schien es mir auch kein Thema auf dem Gebiet von ML und AI zu geben, zu dem er keine Erfahrungen hatte. Ich fand das sehr bereichernd, da es zwar sehr viel Literatur zu ML-Themen gibt, aber jemanden zu finden, der persönlich aus der Praxis berichten kann, ist nicht immer so einfach.

Wie eingangs erwähnt, fand einmal wöchentlich ein Online-Meeting via Skype mit den MasterstudentInnen statt, deren Masterarbeit David Steyrl zu dem Zeitpunkt betreute. Das Kennenlernen anderer Studierender, deren Herangehensweisen und Fortschritte war insgesamt eine sehr interessante, aufschlussreiche und wichtige Erfahrung für mich im Hinblick auf die eigene bevorstehende Masterarbeit.

Aufgrund von Krankheit waren insgesamt zwei der acht Wochen leider weniger produktiv; in dieser Zeit fand deshalb kaum Austausch zwischen meinem Betreuer und mir statt.

Was im Praktikum vielleicht etwas fehlte, war der direkte Bezug zu aktuellen Forschungsthemen an der Praktikumsstelle. Das hätte z.B. in Form von Analysen an bestehenden Datensätzen aus Neurofeedback-Untersuchungen passieren können. Auf der anderen Seite hätte ein solches Vorgehen wahrscheinlich zur Folge gehabt, dass man viel Zeit mit dem Verstehen eines solchen Datensatzes verbracht hätte und für die tatsächliche Anwendung von ML-Methoden nicht ausreichend Zeit geblieben wäre. Mit dem mir bereits bekannten Datensatz aus meiner Bachelorarbeit fiel diese Problematik weg.

8.4 Ausblick

Ich nehme aus dem Praktikum viel neues Wissen und ein besseres Verständnis für die verschiedenen Methoden und deren Sinn mit. Nennen könnte man hier die Kreuzvalidierungsschleifen, etwas, das ich aus bisheriger Literatur nicht in der Form kannte. Auch für die verschiedenen Optimierungsmethoden zum Finden passender Hyperparameter habe ich nun ein tieferes Verständnis entwickelt. Was mir David Steyrl ebenfalls mitgegegeben hat, ist, dass man sich bei all dem aktuellen (zum Teil auch berechtigten) Optimismus und Hype rund um das Thema Machine Learning eine gute Portion Realismus beibehalten und sich vor übertriebenen Erwartungen in Acht nehmen sollte.

Es sind viele Dinge, auf denen ich jetzt aufbauen kann. Das im Praktikum erstellte Skript kann ich beispielsweise in Zukunft für weitere Anwendungen verwenden. Weiters könnte ich mich auch den Themen Bilderklassifikation mittels künstlicher neuronaler Netzwerke widmen. Dieses Thema wurde im Praktikum zwar nicht ausführlich behandelt, jedoch habe ich hier ebenfalls einige praktische Erfahrungen gesammelt und Informationen von meinem Betreuer erhalten. Die praktische Relevanz solcher Methoden in der psychologischen Forschung gilt es dabei noch zu ermitteln. Für mich haben sich durch das Praktikum nun einige gute Ausgangspunkte ergeben, nicht zuletzt auch für die eigene Masterarbeit, die ich im nächsten Semester beginne.

9 Literatur

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition* (2. Aufl.). O'Reilly. Zugriff 7. März 2022 unter https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/

scikit-learn developers. (2022). *Scikit-learn user guide*. scikit-learn User Guide. https://scikit-learn.org/stable/user_guide.html