OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

INFORMATIK UND
MATHEMATIK

Studienarbeit im Fach Data Mining

Analyse eines Affenpocken Datensatzes

Studienarbeit

an der Fakultät Informatik und Mathematik
im Studiengang Wirtschaftsinformatik

eingereicht
im Januar 2023

Linus Schlepp
Regensburg

Professor Dr. Edwin Schicker

Inhaltsverzeichnis

1. Vorwort

Ziel dieser Arbeit ist es, eine Analyse zu diesem Datensatz: `https://www.kaggle.com/datasets/muhammad4hmed/monkeypox-patients-dataset` zu liefern.

Der Datensatz behandelt das Thema Affenpocken. Dabei sind unterschiedliche Symptomatiken als Features gegeben. Die Ergebnis-Spalte stellt dar, ob der Patient positiv oder negativ auf Affenpocken getestet wurde.

Es findet eine Analyse des vorliegenden Datensatzes statt. Die Daten werden visuell in Form von Diagrammen aufbereitet und die grobe Werteverteilungen der Symptome beziehungsweise Features vorgestellt. Zudem findet eine Korrelationsanalyse, in welcher die Daten auf gegenseitige Zusammenhänge geprüft werden, statt. Danach werden die Daten getestet sowie trainiert und die verwendeten Modelle mit den entsprechenden Genauigkeiten dargestellt. Der letzte Schritt der Datenanalyse besteht daraus, dass ein neuronales Netz auf die Daten angewendet wird und die resultierenden Genauigkeiten berechnet werden. Am Schluss werden die gewonnenen Ergebnisse dieser Studienarbeit aufbereitet.

Binär-Klassifikation wird in dieser Studienarbeit als Anwendung verwendet. Diese Studienarbeit sowie der dazugehörige Python-Code ist diesen GitHub-Repositories zu entnehmen:

- Studienarbeit: `https://github.com/linusschlepp/Studienarbeit_Data_Mining`
- Python-Code: `https://github.com/linusschlepp/Data-Mining-Project`

2. Analyse der Daten

In diesem Kapitel wird der Datensatz untersucht. Zunächst findet eine oberflächliche Analyse des Datensatzes statt, in dieser wird festgestellt, wie viele Features beziehungsweise Zeilen und Spalten der Datensatz enthält. Der nächste Schritt der Analyse stellt die Korrelationsanalyse dar. Dabei ist das Ziel, mögliche Zusammenhänge und Korrelationen zwischen den Features zu erkennen.

2.1. Oberflächliche Datenanalyse

Der Datensatz weißt insgesamt 25000 Zeilen und 11 Spalten auf. Davon sind neun Features. Eine Spalte ist der Index und eine stellt die Ergebnis-Spalte dar. Damit Funktio-

nen, wie *pandas.DataFrame.replace* und *pandas.DataFrame.query* problemlos verwendet werden können, ist es notwendig, die Leerzeichen in den Spaltenbenennungen mit Unterstrichen zu ersetzen. So wird beispielsweise *Systemic Illness* zu *Systemic_Illness*. *Sexually_Transmitted_Infection* wird aufgrund der Länge, auf *STI* gekürzt. Die Spalte *Patient_ID* wird als Index verwendet. *MonkeyPox* stellt die Ergebnis-Spalte dar. Das folgende Bild zeigt die Aufteilung der Spalten und die korrespondierenden Werte:

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
1	Patient_ID	Systemic Illness	Rectal Pain	Sore Throat	Penile Oedema	Oral Lesions	Solitary Lesion	Swollen Tonsils	HIV Infection	Sexually Transmitted Infection	MonkeyPox
2	P0	None	False	True	True	True	False	True	False	False	Negative
3	P1	Fever	True	False	True	True	False	False	True	False	Positive
4	P2	Fever	False	True	True	False	False	False	True	False	Positive
5	P3	None	True	False	False	False	True	True	True	False	Positive
6	P4	Swollen Lymph Nodes	True	True	True	False	False	True	True	False	Positive
7	P5	Swollen Lymph Nodes	False	True	False	False	False	False	False	False	Negative
8	P6	Fever	False	True	False	False	False	False	True	False	Positive
9	P7	Fever	True	True	False	True	True	True	False	False	Positive
10	P8	Muscle Aches and Pain	False	True	True	True	False	False	False	False	Positive
11	P9	Fever	False	False	True	True	True	False	True	False	Negative

Abbildung 1: Ausschnitt aus dem vorliegendem Datensatz

Bei den Werten der Features handelt es sich zu einem großen Teil um *bool*-Werte. Die Spalte *Systemic_Illness* stellt dabei eine Ausnahme dar. In dieser werden vier *str*-Werte verwendet: *None*, *Fever*, *Swollen Lymph Nodes* und *Muscle Aches and Pain*. Es gilt diese Werte auf *int*-Werte zu mappen. So wird *None* zu 1, *Fever* zu 2, *Swollen Lymph Nodes* zu 3 und *Muscle Aches and Pain* zu 4. Die Spalte *MonkeyPox* enthält die Werte *Positive* und *Negative* und gibt an, ob der entsprechende Patient an Affenpocken erkrankt ist.

Die Werteverteilung der Features, die *bool*-Werte enthalten, ist in dieser Tabelle aufgelistet:

	Rectal_Pain	Sore_Throat	Penile_Oedema	Oral_Lesion	Solitary_Lesion	Swollen_Tonsils	HIV_Infection	STI
True	12345	12554	12612	12486	12527	12533	12584	12446
False	12655	12446	12388	12514	12473	12467	12416	12554

Tabelle 1: Werteverteilung der *bool*-Features

Es lässt sich feststellen, dass die Werteverteilung innerhalb der *bool*-Features sehr gleichmäßig ist. Zusätzlich wäre es interessant zu wissen, wie die Verteilung von *MonkeyPox* aussieht, wenn die Werte der *bool*-Features *True* sind. Man könnte damit erkennen, ob das Auftreten eines Symptoms eine beziehungsweise keine Affenpocken-Erkrankung zur Folge hat.

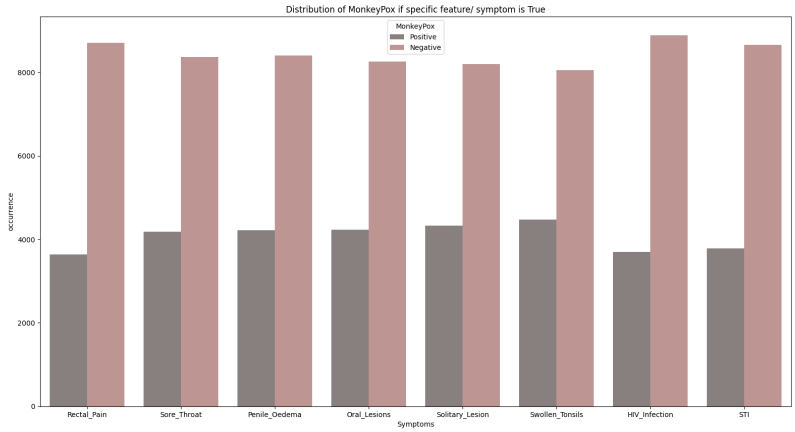

Abbildung 2: Verteilung von *MonkeyPox* wenn bool-Features *True* sind

Anhand von Abbildung 2 kann man also erkennen, dass die Features analog zu Tabelle 1 sehr gleichmäßig ist. Folglich lässt kein Symptom auf eine Affenpocken-Erkrankung schließen.

In Bezug auf das Feature *Systemic_Illness*, welches ausschließlich *str*-Werte enthält, lässt sich die Werteverteilung, durch dieses Diagramm visualisieren:

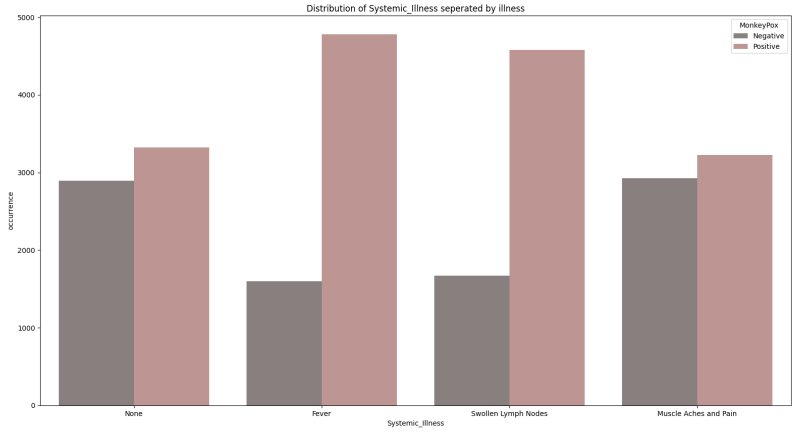

Abbildung 3: Werteverteilung des Features *Systemic_Illness*

Man stellt also fest, dass Patienten, die an *Fever* oder *Swollen_Lymph_Nodes* leiden, zu einer höheren Wahrscheinlichkeit an Affenpocken erkranken, als Patienten, die an keiner Symptomatik (*None*) oder an *Muscle Aches and Pain* leiden.

In Betracht auf die Spalte *MonkeyPox* lässt sich sagen, dass 15909 Patienten an Affenpocken erkrankt sind, während 9091 *Negative* sind. Diese Werteverteilung würde sich visuell folgendermaßen äußern:

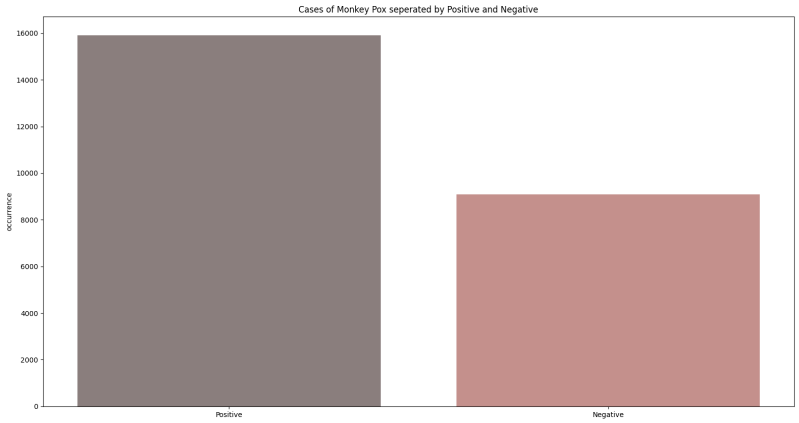

Abbildung 4: Werteverteilung der Ergebnis-Spalte *MonkeyPox*

2.2. Korrelationsanalyse

Zur Prüfung, ob Zusammenhänge zwischen den einzelnen Features bestehen wird eine Heatmap verwendet. Eine Heatmap ist eine Zusammenstellung von Rechtecken. Sie kann genutzt werden, um Muster und Veränderungen im Zeitverlauf zu betrachten. Die x-Achse wird oft mit einem Zeitmaß bezeichnet. Die y-Achse stellt eine Variable dar, die die Kategorie in den Daten definiert. In Betracht auf den vorliegenden Datensatz ergibt sich die folgende Heatmap:

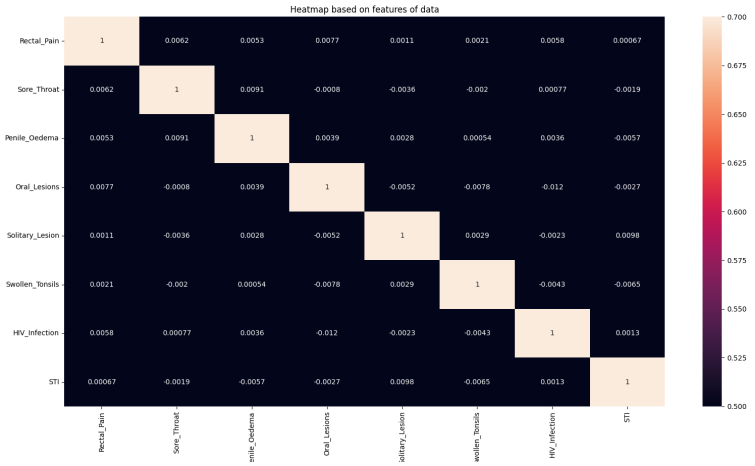

Abbildung 5: Aus den Features resultierende Heatmap

Anhand von Abbildung 5 lässt sich also feststellen, dass sich die berechneten Wahrscheinlichkeiten in einem sehr kleinem beziehungsweise negativen Bereich befinden. Es bestehen somit keine Zusammenhänge und Korrelationen zwischen den verschiedenen Features.

Ein weiterer Schritt der Korrelationsanalyse stellt die Betrachtung der Lift und Konfidenz Werte dar. Dabei ist es notwendig, den Datensatz in so weit zu bearbeiten, dass die verschiedenen Werte von *Systemic_Illness* als eigene Spalten betrachtet werden. Dies ist gefordert, da die verschiedenen Krankheisbilder der Spalte *Systemic_Illness* als eigene Symptome (Features) verstanden werden sollen. Somit können mehr Feature-Kombinationen getestet werden und es gelingt, sich einen größeren Überblick über die vorhandenen Assoziationen im Datensatz zu schaffen. Demnach sieht der überarbeitete Datensatz folgendermaßen aus:

	Fever	None	Swollen Lymph Nodes	Muscle Aches and Pain	Rectal_Pain	Sore_Throat	Penile_Oedema	Oral_Lesions	Solitary_Lesion	Swollen_Tonsils	HIV_Infection	STI
P1	True	False	False	False	True	False	True	True	False	False	True	False
P2	True	False	False	False	False	True	True	False	False	False	True	False
P3	False	True	False	False	True	False	False	False	True	True	True	False
P4	False	False	True	False	True	True	True	False	False	True	True	False
P6	True	False	False	False	False	True	False	False	False	False	True	False
P7	True	False	False	False	True	True	False	True	True	True	False	False
P8	False	False	False	True	False	True	True	True	False	False	False	False
P12	True	False	False	False	True	False	True	False	True	True	True	True
P13	False	False	True	False	True	True	False	True	True	True	False	False
P15	False	False	True	False	False	True	False	True	True	True	True	False

Abbildung 6: Ausschnitt des erweiterten Datensatzes

Die Analyse dieser Werte stützt die gesammelten Erkenntnisse der Heatmap. Die maximale Konfidenz ist 0.604167, der entsprechende Lift 1.104538. Dementsprechend ist

festzustellen, dass die Werte sich gegenseitig wenig bis gar nicht beeinflussen beziehungsweise nicht zusammenhängen.

3. Genauigkeitsanalyse

In diesem Kapitel werden verschiedene Genauigkeiten mit mehreren Modellen und Feature-Kombinationen ermittelt. Dabei wird versucht, die best mögliche Genauigkeit herauszufinden. Danach wird das ausgewählte Modell grafisch auf Overfitting überprüft.

3.1. Wahl des Modells

Zum Einen wird eine Untersuchung mit dem Modell *GaussianNB()* getätigt. Im Laufe der Untersuchung wird bei der Ermittlung der Trainings- und Testdaten der Zufallsstrom nicht aktiv über den Parameter *random_state* vorgegeben. Dieser wird also von der Funktion *train_test_split* selbst gewählt. 68.53 ist bei *GaussianNB* die berechnete Genauigkeit. Diese Genauigkeit stellt sich als wenig zufriedenstellen heraus. Es ergibt sich somit die Frage, ob sich diese optimieren lässt.

In der Datei *utils.py* wird eine Funktion *create_feature_accuracy_dict()* entwickelt, bei welcher jede mögliche Feature-Kombination für drei Modelle angewendet wird. Bei den drei Modellen handelt es sich um *DecisionTreeClasifier()*, *RandomForestClassifier()* und *GaussianNB()*.

Die Feature-Kombinationen, das verwendete Modell (Key) und die korrespondierenden Genauigkeiten (Value) werden in einer Dictionary gespeichert. Diese Dictionary lässt sich anhand der Genauigkeit absteigend sortieren. Man kommt zu dem Ergebnis, dass für das Modell *RandomForestClassifier()* und den Features: *Systemic_Illness*, *Rectal_Pain*, *Sore_Throat*, *Solitary_Lesion*, *HIV_Infection* und *STI* eine Genauigkeit von 69.71 berechnet wird.

```
1 [    (    (    'RandomForestClassifier()',
2              "['Systemic_Illness', 'Rectal_Pain', 'Penile_Oedema',
                   "
3              "'Solitary_Lesion', 'HIV_Infection', 'STI']"),
4          69.71199999999999),
5 (    (    "DecisionTreeClassifier(criterion='entropy',
      min_samples_split=5)",
6              "['Systemic_Illness', 'Rectal_Pain', 'Penile_Oedema',
                   "
7              "'Solitary_Lesion', 'HIV_Infection', 'STI']"),
```

```
8           69.71199999999999),
9       (   (   "DecisionTreeClassifier(criterion='entropy',
            min_samples_split=5)",
10              "['Systemic_Illness', 'Rectal_Pain', 'Solitary_Lesion
                    ', "
11              "'HIV_Infection', 'STI']"),
12          69.568),
13      (   (   'RandomForestClassifier()',
14              "['Systemic_Illness', 'Rectal_Pain', 'Solitary_Lesion
                    ', "
15              "'HIV_Infection', 'STI']"),
```

Listing 1: Verwendetes Modell, Feature-Kombination und die korrespondierende Genauigkeit

Man stellt fest, dass die oben gennante Feature-Kombination von jedem Modell in Listing 1 verwendet wird. Somit kann man annehmen, dass diese Feature-Kombination die Beste für den vorliegenden Datensatz ist. Diese Untersuchung ermöglicht es, die Genauigkeit um 1% zu optimieren und *RandomForestClassifier()* als das beste Modell für diesen Datensatz zu erkennen.

3.2. Prüfung auf Overfitting

Aufbauend auf Kapitel 3.1 ist es wichtig festzustellen, ob beziehungsweise wann Overfitting in *RandomForestclassifier()* auftritt. Overfitting beschreibt das Phänomen, dass sich Daten zu sehr den Trainingsdaten anpassen. Hierbei besteht die Gefahr, dass der Algorithmus den Datensatz auswendig, aber nicht die zugrundeliegenden Muster und Zusammenhänge erkennt.

Um Overfitting in dem vorliegenden Datensatz zu erkennen wird in der *utils.py* eine weitere Funktion *check_over_fitting* hinzugefügt. In welcher die Genauigkeit anhand der berechneten Test- und Trainingsdaten und dem ausgewählten Modell (*RandomForestClassifier()*) aus Kapitel 3.1 für die Baumtiefen 2 bis 21 berechnet wird. Die Test- und Trainingsdaten werden dabei im Verhältnis 50:50 geteilt. Das Ergebnis dieser Untersuchung wird grafisch in Form eines Plots ausgewertet:

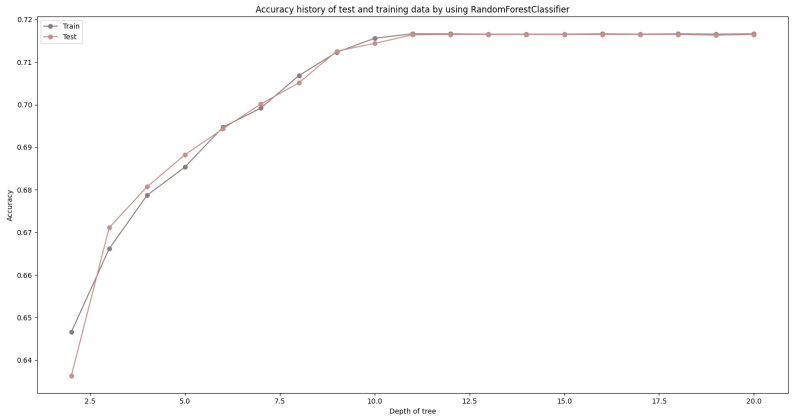

Abbildung 7: Genauigkeits-Verlauf der Test- und Trainingsdaten bei steigender Baumtiefe

Bei Betrachtung von Abbildung 7 stellt man fest, dass ab einer Baumtiefe von 11, die Test- und Trainingsdaten parallel verlaufen. Dieses Verhalten ist kein Indiz für Overfitting. Es ist eher ein Zeichen dafür, dass sich die Test- und Trainingsdaten ab einer Baumtiefe von 11 stabilisieren. Demnach sollte die maximale Tiefe des Baumes in *RandomForestClassifier()* mithilfe der Variable *max_depth* auf 11 festgelegt werden.

3.3. Darstellung als Entscheidungsbaum

Das Aufzeigen eines Entscheidungsbaums wird gewählt, um anhand von verschiedenen visualisierten Antwortoptionen auf eine Affenpocken-Erkrankung zu schließen. Da es sich in dieser Studienarbeit, um einen sehr großen Datensatz handelt, ist es notwendig den Entscheidungsbaum durch den Parameter *min_sample_split* etwas zu kürzen. Dieser wird in der folgenden Abbildung auf 0.2 gesetzt.

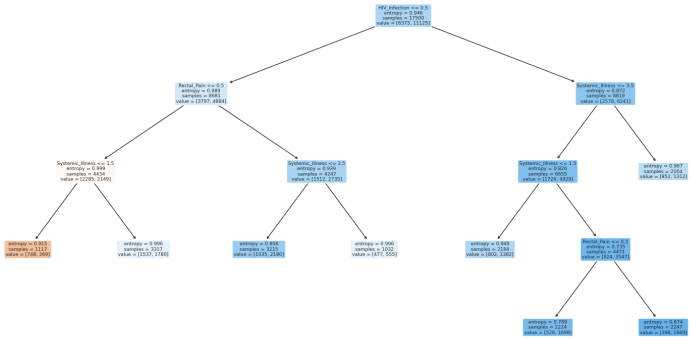

Abbildung 8: Gekürzter Entscheidungsbaum

Es lässt sich also sagen, dass die True- und False-Werte, der Endknoten zu nahe beieinander liegen, um eine klare Aussage zu treffen. Demnach lässt sich anhand des Baumes schwer ein Muster ablesen beziehungsweise nicht auf eine Affenpocken-Erkrankung schließen.

4. Erstellung eines neuronalen Netzes

Am Ende der Datenanalyse wird ein neuronales Netz erstellt. Der erste Schritt besteht darin, die Daten für die Arbeit mit *tensorflow* vorzubereiten. Hierbei werden die enthaltenen *bool*- und *int*-Werte in *float*-Werte umgewandelt. Dies wird durch die Funktion *prepare_for_tensor* in der *utils.py* Datei bewärkstelligt. Auf Funktionen der Bibliothek *sklearn.preprocessing* wie beispielsweise *MinMaxScaler()* konnte aufgrund des einheitlichen sowie kleinen Wertebereichs verzichtet werden.

Der zweite Schritt, besteht in der Definition des neuronalen Netzes, dieses ist folgendermaßen aufgebaut:

```
1 model = Sequential()
2
3 X, y = utils.prepare_for_tensor(X_data, target)
4
5 # 12 input nodes
6 model.add(Dense(12, input_dim=X_data.shape[1], activation='sigmoid
      ', name='input'))
7
```

```
8  # 20 nodes for the hidden layers
9  model.add(Dense(20, activation='relu', name='layer1'))
10 model.add(Dense(20, activation='relu', name='layer2'))
11 model.add(Dense(20, activation='relu', name='layer3'))
12 model.add(Dense(1, activation='sigmoid', name='output'))
13
14 model.compile(optimizer='adam', loss='mse', metrics=['accuracy'])
```

Listing 2: Code des neuronalen Netzes

In Listing 2 werden für den Aufbau des Netzes insgesamt 5 Schichten verwendet. Je eine *input*- sowie *output*-Schicht. Aufgrund der Größe des Datensatzes werden 3 versteckte Schichten verwendet. Die Anzahl der Neuronen innerhalb der versteckten Schichten ist auf 20 festgelegt, bei der *input*- Schicht liegt sie bei 12. Die verwendeten Aktivierungsfunktionen beschränken sich auf *relu* und *sigmoid*. Die Verlustfunktion *mse* hat sich im Laufe dieser Untersuchung als die beste Wahl herausgestellt. Darüber hinaus wird die *batch_size* auf 64 gesetzt und es werden 100 Epochen festgelegt.

Für die Test- und Trainingsdaten ergibt sich somit eine berechnete Genauigkeit von 0.70 und ein Verlust von 0.20. Graphisch sieht der Epochenverlauf der Verlust- und Genauigkeitswerte folgendermaßen aus:

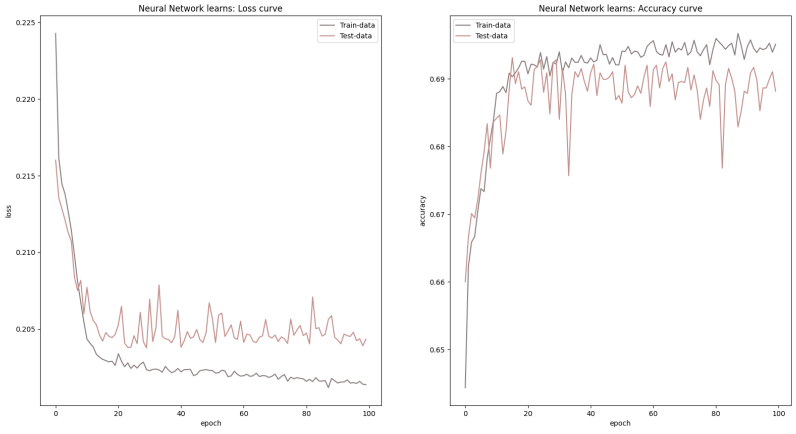

Abbildung 9: Verlust- und Genauigkeitskurven des neuronalen Netzes

Es ist ein Abfall im Verlust der Test- und Trainingsdaten zu verzeichnen. Der Verlust der Trainingsdaten ist jedoch stärker. In Bezug auf die Testdaten ist anstelle eines Abfalls, eher eine Schwankung erkennbar. Der ermittelte Genauigkeitsverlauf der Trainings- und

Testdaten schwankt zwar, ist aber insgesamt stabil. Somit ist in Betracht der Verlust- und Genauigkeitskurven kein Overfitting zu verzeichnen.

5. Bewertung der Ergebnisse

Nach einer ausgiebigen Analyse des vorliegenden Datensatzes können keine eindeutigen Schlüsse aus diesem Datensatz gezogen werden. Man kann weder anhand eines bestimmten Features auf eine Affenpocken-Erkrankung schließen, noch besteht ein Zusammenhang/ Korrelationen zwischen den Features. Die Datenanalyse mithilfe verschiedener Modelle blieb ebenfalls erfolglos. So konnte anhand verschiedener Feature-Kombinationen und dem Modell *RandomForestClassifier()* nur eine Genauigkeit von ≈ 0.70 ermittelt werden. Die ermittelte Genauigkeit ist zu gering, um medizinische Untersuchungen auf diesem Datensatz durchzuführen. Overfitting konnte im verwendeten Modell *RandomForestClassifier()* als auch im neuronalen Netz nicht nachgewiesen werden.

Der Grund für diese eher erfolglose Datenanalyse liegt vermutlich an der Homogenität der Daten. In Kapitel 3 wurde bereits bewiesen, dass kein Feature ein wirkliches Indiz für eine Affenpocken-Erkrankung darstellt. Es wäre sinnvoller gewesen, den Features *int*-Werte beispielsweise in einem Intervall [0, 5] zu geben, um die Daten vielseitiger/ heterogener zu machen. Dabei beschreiben die Werte des Intervalls die Heftigkeit der jeweiligen Symptomatik.

Verzeichnisse

I. Abbildungsverzeichnis

II. Tabellenverzeichnis

III. Listings