

Evolutionäre Optimierung von sequenz- und strukturbasierten Klassifizierer-Ensembles unter Berücksichtigung der Diversität

Julian Hesse

Matrikelnummer: 3077423

Betreuer Prof. Dr. Dominik Heider

Fachbereich 12 Mathematik und Informatik

16.11.2020

Inhaltsverzeichnis

Abkürzungsverzeichnis.....	i
1. Einleitung.....	1
2. Grundlagen.....	1
2.1. Peptide.....	1
2.2. Kodierung von Peptiden	1
2.3. Evolutionäre Algorithmen.....	1
2.4. Klassifizierer.....	2
2.5. Evaluierung	5
2.6. Overfitting	6
2.7. (k-fache) Kreuzvalidierung	7
2.8. Ensemble Learning	8
2.9. Stacked Generalization.....	8
2.10. Pareto Optimierung.....	13
3. Methoden.....	13
3.1. Software.....	13
3.2. Datensätze.....	14
3.3. Architektur	14
4. Ergebnisse.....	17
4.1. Finale Individuen.....	17
4.2. Einzelne Durchläufe	19
5. Diskussion.....	24
5.1. Anzahl der Generationen und Größe des Datensatzes	24
5.2. Genauigkeit.....	25
5.3. Genauigkeit und Diversität.....	25
5.4. Sequenz und strukturbasierte Kodierungen	26
5.5. Häufigkeiten der Kodierungen	26
5.6. Stacked Generalization.....	26
5.7. Wahl von n und der Populationsgröße	27

5.8. Fitness als Linearkombination aus -Score und Diversität	27
6. Ausblick.....	27
7. Fazit	28
Literaturverzeichnis	30

Abkürzungsverzeichnis

TP	-	True Positive/ Richtig Positiv
FP	-	False Positive/ Falsch Positiv
TN	-	True Negative/ Richtig Negativ
FN	-	False Negative/ Falsch Negativ
P	-	Precision
R	-	Recall
MCC	-	Matthews Correlation Coefficient/ Matthews Korrelationskoeffizient
PEA	-	Pearson Korrelationskoeffizient
MK	-	Metaklassifizierer
ML	-	Maschinelles Lernen
stk	-	strukturbasierte Kodierungen
seq	-	sequenzbasierte Kodierungen
PG	-	Populationsgröße
EG	-	Ensemblegröße (Anzahl der Klassifizierer/ Kodierungen im Ensemble)
Gen	-	Anzahl Generationen
MB	-	Megabyte
EA	-	Evolutionärer Algorithmus

Abstrakt

Diese Arbeit beschäftigt sich mit der Verbesserung eines evolutionären Algorithmus zur Klassifikation von Peptiden und der Analyse ihrer Kodierungen. Der zugrundeliegende Algorithmus basiert auf der Arbeit von Markus Flicke (2019) [1]. Zur Klassifikation der Peptide wird ein Klassifizierer Ensemble genutzt. Die Verbesserungen des Algorithmus wurden hauptsächlich an diesem Ensemble vorgenommen. Der Algorithmus bestimmt nun die Diversität des Ensembles und unterscheidet zwischen zwei verschiedenen Kodierungsansätzen. Dies sollte die Frage klären, ob ein Ensemble mit sequenz- und strukturbasierten Kodierungen eine bessere Genauigkeit erreicht als Ensembles, die nur einen der Ansätze nutzen und ob dies auf die Diversität zurückzuführen ist. Die Ergebnisse zeigen, dass ein Ensemble mit beiden Ansätzen zwar bessere Ergebnisse erzielt, dies jedoch nicht auf die Diversität zurück-zuführen ist.

Abstract

This work deals with the improvement of an evolutionary algorithm to classify peptides and analyze it's encodings. The main algorithm was build in a previous work from Markus Flicke (2019) [1]. The algorithm uses a classifier ensemble. The improvement primarily focuses on the ensemble. It is now possible to measure the ensemble's diversity and distinguish between two different encoding approaches. This tried to clear up the hypothesis that an ensemble, which uses sequence and structure based encodings with a high diversity, improves the overall performance of the algorithm. The results show that an ensemble which uses both encoding approaches performs better than one that uses just a single approach. Although, the better performance cannot be fully credited to a higher diversity between the encodings.

1. Einleitung

Die folgende Arbeit beschäftigt sich mit der Erweiterung und Auswertung eines evolutionären Algorithmus zur Bestimmung einer antimikrobiellen Eigenschaft eines Peptids. Dieser wurde in einer Bachelorarbeit aus dem vergangenen Jahr (2019) von Markus Flicke [1] implementiert. Um die Peptide numerisch darstellen zu können, wurden verschiedene Kodierungen verwendet.

Ansatzpunkt der Erweiterungen für diese Arbeit war insbesondere das Klassifizierer-Ensemble, welches zur Evaluierung der einzelnen Individuen im evolutionären Algorithmus genutzt wird. Dabei wurde untersucht, ob eine höhere Diversität zwischen den Klassifizierern zu einer besseren Genauigkeit des Ensembles führt.

Die Diversität ist deshalb interessant, da die Trainingsdaten der Klassifizierer im Ensemble zwei verschiedene Konzepte der Kodierung von Peptiden nutzen: den sequenz- und den strukturbasierten Ansatz. Ziel der Untersuchungen ist herauszufinden, ob eine Kombination dieser Ansätze zu besseren Ergebnissen führt als die Nutzung nur eines einzelnen Kodierungsansatzes und ob dies auf die Diversität zurückzuführen ist. Zwei weitere Modifikationen zum ursprünglichen Algorithmus sind die optionale Verwendung des Stacking Verfahrens als Methode zur Zusammenführung der einzelnen Klassifikationen und die Nutzung von verschachtelter Kreuzvalidierung.

2. Grundlagen

Dieses Kapitel beschäftigt sich mit den theoretischen Grundlagen, die für die Bearbeitung und für das Verständnis der Arbeit von Bedeutung sind.

2.1. Peptide

Organische Verbindungen, welche Peptidbindungen (eine Carbonsäurebindung) zwischen ihren Aminosäuren enthalten, werden Peptide genannt. Peptide können sich immunmodulatorisch und/oder antimikrobiell auf einen Organismus auswirken. Erstere regulieren dessen Immunsystem. Es kann dabei sowohl angeregt als auch gehemmt werden [23]. Antimikrobielle Peptide unterstützen das Immunsystem bei Infektionen mit Mikroorganismen, indem sie die bakterielle Zelle auf zwei Wegen angreifen. Zum Einen können sie dessen Membran zerstören und zum anderen in die Zelle eindringen und im Cytoplasma die Produktion von Nukleinsäuren und Proteinen hemmen. Gerade in der heutigen Zeit von antibiotikaresistenten Mikroorganismen sind sie von großem Interesse [2].

2.2. Kodierung von Peptiden

Peptide müssen für die maschinelle Verarbeitung numerisch erfasst werden können. Dazu gibt es verschiedene Kodierungsansätze. Zwei Arten der Kodierung sind der strukturbasierte und der sequenzbasierte Ansatz. Der sequenzbasierte Ansatz basiert auf der Primärstruktur, welche die Anordnung der Aminosäuren modelliert. Strukturbasierte Kodierungen basieren auf der Tertiärstruktur, die die dreidimensionale, funktionsbestimmende Zusammensetzung des Peptids abbildet. Primär- und Tertiärstruktur können folglich unterschiedliche Eigenschaften eines Peptids darstellen [2].

2.3. Evolutionäre Algorithmen

Ein evolutionärer Algorithmus ist ein Machine Learning Algorithmus, der auf der biologischen Evolution basiert. Wie bei der Evolution aus der Biologie gibt es eine Population von Individuen, die ihre Grundeigenschaften, den Genotyp, von Generation zu Generation

weitergeben. Ein Individuum ergibt sich aus (meist) zwei Individuen der vorherigen Generation im Rekombinationsschritt. Teile der Ausprägungen des neuen Individuums kommen also von Elternteil A und die anderen von Elternteil B. Bei der Zusammensetzung kann es zu Mutationen kommen, sodass auch die expliziten Ausprägungen der Eigenschaften, der Phänotyp, variieren können.

In jeder Generation wird jedem Individuum der Population durch eine Fitnessfunktion ein Fitnesswert zugeordnet. Anschließend wird die Population nach diesem Wert geordnet. Die i Individuen mit dem höchsten Fitnesswert werden Eltern der Individuen in der folgenden Generation. i ist dabei die Anzahl der Elternindividuen und orientiert sich an der Größe der Population. Der Algorithmus terminiert, wenn ein festgelegtes Kriterium erfüllt ist. Dies kann die Anzahl der abgelaufenen Generationen, ein bestimmter Fitnesswert und/oder die Anzahl der Generationen ohne Verbesserung des Fitnesswerts sein. Abb. 1 stellt diesen Ablauf graphisch dar. Sie beginnt mit der Initialisierung der ersten Generation. Dies kann zufällig oder deterministisch geschehen. Anschließend beginnt der Zyklus mit der Evaluation der Individuen [3].

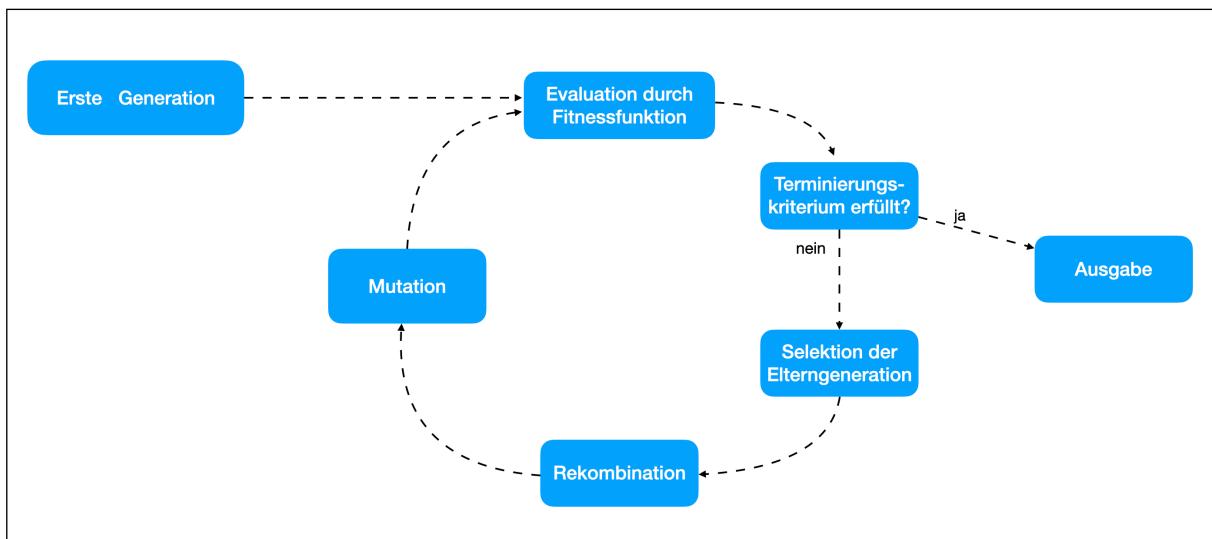


Abb.1: Ablauf eines evolutionären Algorithmus

2.4. Klassifizierer

Ein Klassifizierer im Maschinellen Lernen (ML) ist ein Modell, welches aus bekannten Daten (Trainingsdaten) Strukturen ableiten soll. Dabei bestehen die Daten aus Objekten, welche bekannten Kategorien zugeordnet werden können. Die erlernten Strukturen sollen dazu dienen, ein unbekanntes Objekt einer der Kategorien zuzuordnen. Die Zuordnung erfolgt, indem das Modell jeder Klasse eine Wahrscheinlichkeit zuweist, mit der es das Objekt in diese Klasse einordnen würde. Für binäre Klassifizierungsprobleme würden alle Wahrscheinlichkeiten unter 0,5 als Klasse 0 gewertet werden und alle über 0,5 für Klasse 1 [4, S. 128ff].

2.4.1. Support Vector Machine

Eine Support Vector Machine ist ein überwachter ML Algorithmus, welcher Daten, repräsentiert durch entsprechende Vektoren, durch eine oder mehrere Hyperebenen separiert. Diese Trennung ermöglicht die Einteilung von Datenpunkten mit unbekannter Klasse. In der grundlegenden Form erfolgt eine Separation in zwei Klassen mit einer linearen Hyperebene. Gesucht wird eine Hyperebene, die die Distanz zwischen den Punkten maximiert, die ihr am nächsten liegen. Diese Punkte nennt man Stützvektoren. Die

Hyperebene ist also durch die Stützvektoren definiert. Die Klasse eines Vektors ergibt sich dann durch die Seite der Hyperebene auf der er liegt. Im Beispiel in Abb. 2 wird diese (hier 1-dimensionale) Hyperebene durch die Linie h dargestellt. Die Distanz d wird durch eine gewählte Norm berechnet. d ist die Distanz der Stützvektoren zur Hyperebene. Ein neuer Datenpunkt wird je nach Lage über oder unter h in die Klassen eingeteilt.

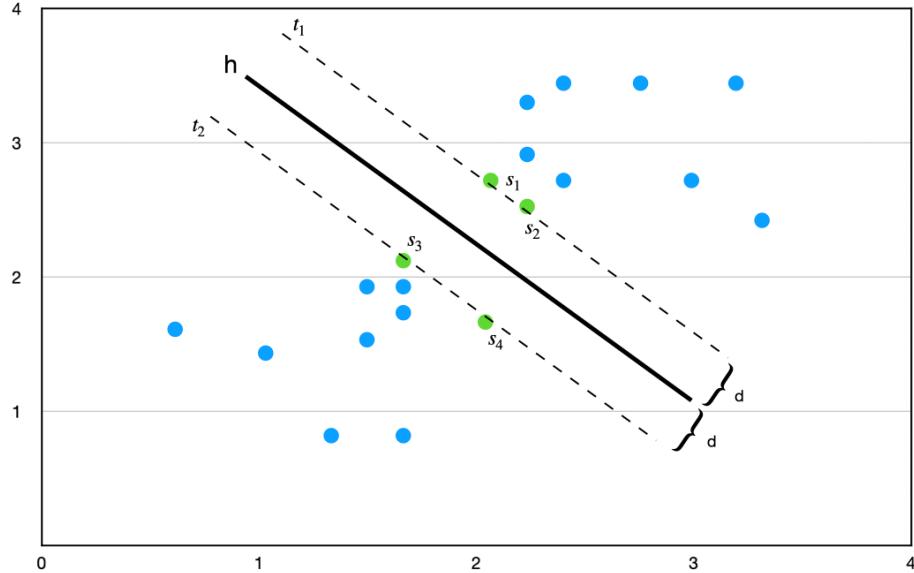


Abb. 2: 2-dimensionales Beispiel einer Hyperebene h und Stützvektoren s_i einer Support Vector Machine

Sei $n \leq m$, $\vec{x} = (x_1, \dots, x_n)' \in D \subseteq \mathbb{R}^m$ ein Eingabepunkt, $f : D \rightarrow \mathbb{R}$ eine Funktion, für die gilt $f(x) \geq 0$, wenn x der Klasse 1 angehört und $f(x) < 0$, wenn x der Klasse -1 angehört und $(\vec{w}, b) \in \mathbb{R}^n \times \mathbb{R}$, mit w dem Normalenvektor der Hyperebene. Ist $f(\vec{x})$ linear, gilt

$$f(x) = \langle \vec{w} \cdot \vec{x} \rangle - b \quad (1)$$

Eine Hyperebene ist dann definiert durch die Punkte \vec{x} für die gilt

$$\vec{w} \cdot \vec{x} - b = 0. \quad (2)$$

Es gilt $d = \frac{b}{\|\vec{w}\|}$.

Sei nun $i \in 1 \leq m$, t_i die Klasse des Punktes x_i , dann liegen durch die Bedingungen

$$\begin{aligned} \vec{w} \cdot \vec{x} - b &\geq 1 \quad \forall t_i = 1 \\ \wedge \quad \vec{w} \cdot \vec{x} - b &\leq 1 \quad \forall t_i = -1 \end{aligned} \quad (3)$$

$$\iff t_i(\vec{w} \cdot \vec{x} - b) \geq 1 \quad \forall i \quad (4)$$

jeder Punkt \vec{x} hinsichtlich der Hyperebene auf der Seite seiner jeweiligen Klasse. Unter Einhaltung dieser Bedingungen ergibt sich die Hyperebene aus (\vec{w}, b) durch

$$\min(\|\vec{w}\|) \quad (5)$$

Sollte es keine *lineare* (Hyper)ebene geben, die die Daten aufteilt, werden die Daten mithilfe des Kernel Tricks solange in einen höherdimensionalen Raum überführt, bis es diese Hyperebene gibt. Anschließend kann die Hyperebene wieder in einen zweidimensionalen Raum überführt werden, um dort die Daten (nicht-linear) in zwei Klassen einzuteilen. Es ist zudem möglich, Fehler bei der Klassifizierung zuzulassen. Wie hoch der Anteil der Fehlklassifizierungen ist, wird durch den Regularisierungsparameter $C \in (0, \infty)$ festgelegt. Je höher C , desto mehr Fehler werden erlaubt [4, S. 359ff].

2.4.2. Random Forest

Ein Random Forest ist ein Ensemble Machine Learning Algorithmus. Er besteht aus Entscheidungsbäumen und dient somit auch zur Klassifizierung oder Regression. Die Entscheidungsbäume werden mit unterschiedlichen Hyperparametern und unterschiedlichen Teildatensätzen trainiert. Bei der Klassifizierung entspricht die meist gewählte Klasse durch die Entscheidungsbäume der Klasse, die endgültig durch den Random Forest Algorithmus als Vorhersage ausgegeben wird. Für die Regression wird dafür der Durchschnitt der Vorhersagen gebildet [4, S. 319f].

2.4.3. Deep Learning

Deep Learning ist eine Art des Maschinellen Lernens, bei denen künstliche neuronale Netze (KNN) zum Einsatz kommen. Den Namen und auch ihren grundlegenden Aufbau haben sie von neuronalen Netzen des Gehirns.

Ein KNN besteht aus verschiedenen Schichten. Die Eingabeschicht repräsentiert die eingehenden Daten bzw. ein Datenfeld in einer numerischen Form. Die Ausgabeschicht gibt die Wahrscheinlichkeit an, mit der der Algorithmus die Eingabe einer entsprechenden Ausgabe, wie bspw. verschiedenen Klassen, zuordnet. Dazwischen befindet sich mindestens eine verdeckte Schicht, welche dazu beiträgt, dass der Algorithmus Regelmäßigkeiten in den Daten erkennt. Im Allgemeinen wird mit jeder Schicht eine weitere Abstraktionsebene für die Erkennung der Regeln hinzugefügt, um somit einen höheren Informationsgrad extrahieren zu können.

Das besondere an Deep Learning Algorithmen besteht darin, dass sie mehrere verdeckte Schichten haben. Mit den verdeckten Schichten steigt jedoch nicht zwangsläufig auch die Genauigkeit. Für eine detaillierte Erklärung der Funktionsweise von Deep Learning Algorithmen, die über den Rahmen dieser Arbeit hinausgehen würde, siehe auch [5].

2.4.4. Logistische Regression

Die Logistische Regression wird in der Klassifikation verwendet, indem sie Wahrscheinlichkeiten für die Klassen 0 und 1 ausgibt. Liegt die Wahrscheinlichkeit für einen Punkt über dem Schwellenwert von 0,5 wird er zur Klasse 1 zugeordnet werden und vice versa. Abb. 3 zeigt eine logistische Regression mit einem Attribut. Punkt x_1 wird hier der Klasse 1 zugeordnet.

Sei $p(x)$ die Wahrscheinlichkeit, dass der Punkt x in Klasse 1 liegt. Dann ist mit der Logistischen Regression

$$p(x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}} \quad (6)$$

Die Parameter β_0 und β_1 werden anschließend mit üblichen Verfahren wie der Methode der kleinsten Quadrate oder der Maximum-Likelihood-Schätzung approximiert.

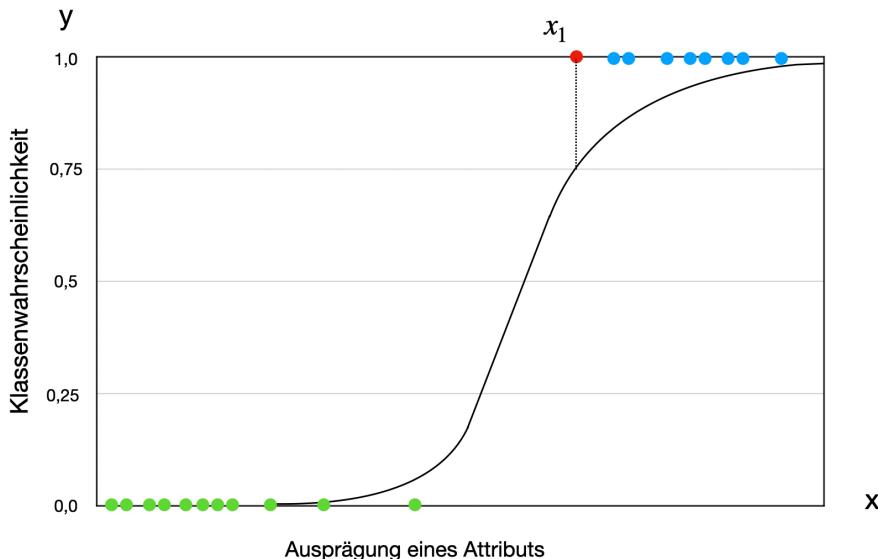


Abb. 3: Beispiel einer eindimensionalen (binären) Logistischen Regression. Grün: bekannte Datenpunkte aus Klasse 0, blau: bekannte Datenpunkte aus Klasse 1, rot: zu klassifizierender Datenpunkt

Für die Klassifikationen von Daten mit mehreren Attributen wird die multiple lineare Regression verwendet. Es gilt also

$$p(x) = \frac{e^{\beta_0 + \beta_1 x + \dots + \beta_p}}{1 + e^{\beta_0 + \beta_1 + \dots + \beta_p}} \quad (7)$$

[4, S. 131ff].

2.5. Evaluierung

Um Modelle vergleichen zu können, gibt es verschiedene Maße. In dieser Arbeit wird das F-Maß (F_1 -Score), der Matthews Korrelationskoeffizient, die Accuracy, Precision und Recall genutzt, da sie speziell für die binäre Klassifizierung geeignet sind.

Die Berechnung dieser Maße basiert auf der Beobachtung der Richtig Positiven (engl. *true positive*, TP), Falsch Positiven (FP), Richtig Negativen (engl. *true negative*, TN) und Falsch Negativen (FN) Werte nach der Berechnung der Vorhersagen. “Positiv” und “Negativ” beziehen sich auf die Ausgabe des Algorithmus. Sie stehen für eine der beiden Klassen, die der Algorithmus dem untersuchten Objekt zugeordnet hat. “Richtig” und “Falsch” beziehen sich auf den Vergleich der ausgegebenen Klasse mit der wahren Klasse.

Die *Precision* gibt den Anteil der richtig klassifizierten Positiven an den insgesamt positiv klassifizierten Objekten an. Recall gibt den Anteil der richtig klassifizierten Positiven an den wirklich positiven Objekten an. Die Accuracy (A) spiegelt den Anteil der richtig klassifizierten Objekte an allen vorhandenen Objekten wieder. Der F_1 -Score (F_1) vereint Precision und Recall. Auch der Matthews Korrelationskoeffizient (engl. *Mathews Correlation Coefficient*, MCC) bezieht die Maße mit ein. Bis auf den MCC , der zwischen -1 und 1 liegen kann, haben alle Maße einen maximalen Wert von 1 und einen minimalen Wert von 0. Ein höherer Wert des jeweiligen Maßes, spricht für eine höhere Genauigkeit.

Für ein Objekt i aus n Objekten, mit o_i als vorhergesagte Klasse und t_i als wahre Klasse und \mathbb{I} als Indikatorfunktion, gilt also:

$$TP = \sum_{i=1}^n \mathbb{I}_{o_i=1 \wedge t_i=1} \quad FP = \sum_{i=1}^n \mathbb{I}_{o_i=1 \wedge t_i=0} \quad (1),(2)$$

$$TN = \sum_{i=1}^n \mathbb{I}_{o_i=0 \wedge t_i=0} \quad FN = \sum_{i=1}^n \mathbb{I}_{o_i=0 \wedge t_i=1} \quad (3),(4)$$

Die oben genannten Maße berechnen sich damit wie folgt:

$$P = \frac{TP}{TP + FP} R = \frac{TP}{TP + FN} \quad (5),(6)$$

$$A = \frac{TP + TN}{TP + FP + TN + FN} \quad (7)$$

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \quad (8)$$

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (9)$$

[6]

2.6. Overfitting

Overfitting bedeutet, dass für die Trainingsdaten die Messwerte des Modells zwar auf eine äußerst gute Genauigkeit hindeuten, dies allerdings ausschließlich für exakt diese Daten gilt. Wendet man das Modell auf unbekannte Daten an, schneidet es deutlich schlechter ab. Dies spiegelt sich im Training- und Testfehler wieder. In Abb. 4 ist ein typisches Zeichen für Overfitting abgebildet. Je länger der Algorithmus trainiert wird, desto geringer wird der TrainingSError und desto höher wird der Testerror [4, S. 22 & 144].

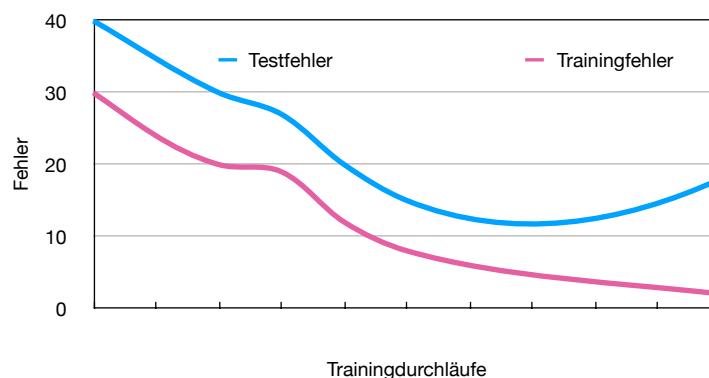


Abb. 4: Zeichen von Overfitting

2.7. (k-fache) Kreuzvalidierung

Für das Trainieren von Modellen und das anschließende Testen der Modellgüte wird der ursprüngliche Datensatz meist in zwei Teile geteilt. Auf dem Trainingsdatensatz wird das Modell trainiert. Anschließend wird auf dem Testdatensatz überprüft, wie sich das trainierte Modell, angewandt auf unbekannte Daten, verhält.

Die Kreuzvalidierung ist eine Methode, um während der Trainingsphase eine bessere Generalisierbarkeit zu erreichen und zusätzlich Overfitting zu vermeiden. Dazu wird der Trainingsdatensatz noch weiter unterteilt. Eine k -fache Kreuzvalidierung teilt den Datensatz in k Teile. Mit dem Teildatensatz, der aus $k-1$ Teilen besteht, wird dann das Modell trainiert. Der übrig gebliebene Teil des Datensatzes wird "Validierungsdatensatz" genannt. Mit diesem wird die Güte des Modells bestimmt. Anschließend wird ein anderer der $k-1$ Teile Validierungsdatensatz. Die übrig gebliebenen Teile werden zusammen der Trainingsdatensatz, auf dem ein weiteres Modell trainiert und die Güte bestimmt wird. Wenn jeder der k Teile einmal Validierungsdatensatz war, wird der Durchschnitt der Gütwerte als Gesamtgüte bestimmt. Nach einem Durchlauf der Kreuzvalidierung wird der Durchschnitt der Messwerte für jeden Teil bestimmt. Sollten die Ergebnisse zufriedenstellend sein, kann die Modellgüte auf dem Testdatensatz bestimmt werden. Ist dies nicht der Fall, können die Hyperparameter justiert werden und nochmals eine Kreuzvalidierung durchgeführt werden. In Abb. 5 ist eine 4-fache Kreuzvalidierung dargestellt. Schritt 1 zeigt die Teilung des Datensatzes in Trainings- und Testdatensatz. In Schritt 2.1 bis 2.4 ist jeweils ein anderer Teil der vier Teile Validierungsdatensatz [4, S. 181ff].

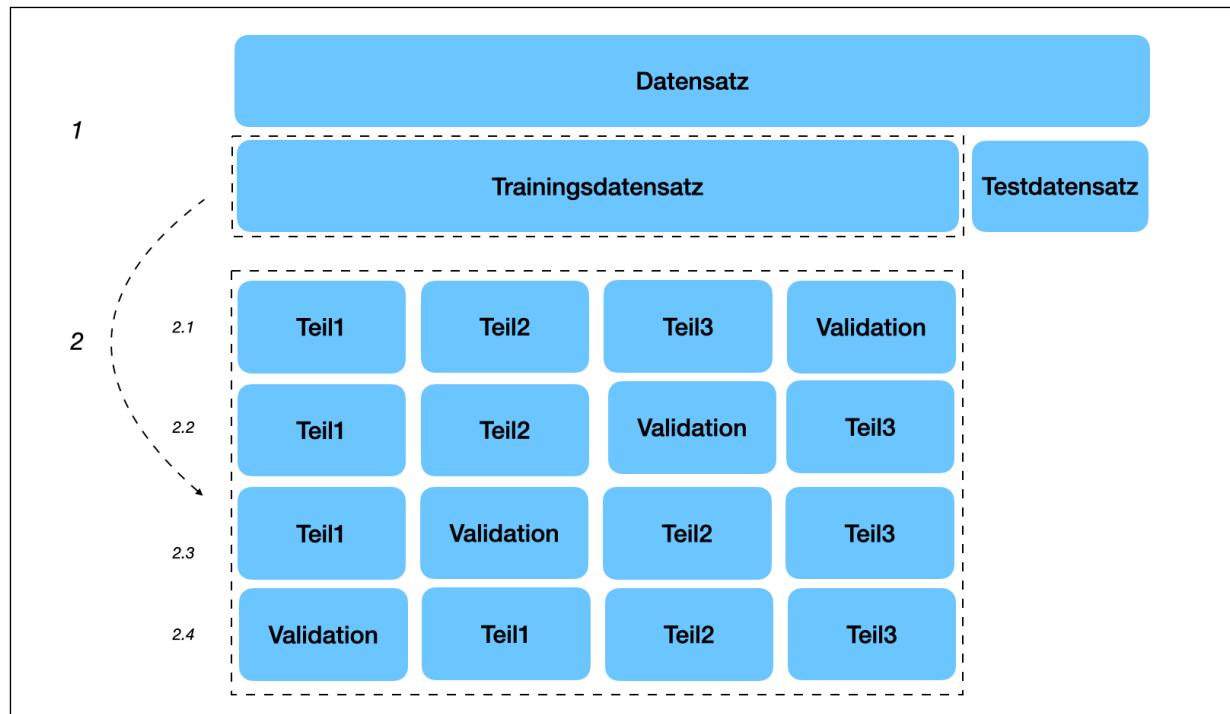


Abb. 5: k-fache Kreuzvalidierung mit $k=4$

Verschachtelte Kreuzvalidierung

Abb. 2 zeigt die verschachtelte Kreuzvalidierung. Dabei wird der gesamte Datensatz wie bei einer Kreuzvalidierung geteilt. Es gibt damit mehrere Testdatensätze, die den Validierungsdatensätzen entsprechen. Für jeden Durchlauf des Modells werden die $k-1$ Teile, die nicht dem Testdatensatz entsprechen, nochmals in i Teile aufgeteilt. Davon sind $i-1$ Teile für das Trainieren und ein Teil als Validierungsdatensatz vorgesehen. Der Validierungs-

datensatz rotiert wieder in jeden der Teile, wobei jeder Durchlauf mit unterschiedlichen, vorher festgelegten Hyperparametern läuft. Die Hyperparameter, die zum besten Ergebnis führen, werden für das Modell in der äußeren Kreuzvalidierung verwendet [19]. In Abb. 6 ist $k = 4$ und $i = 3$. Im Schritt 1 wird der Datensatz in vier Teile geteilt. Jeder Teil entspricht einmal dem Testdatensatz. Die Aufteilung im gestrichelten Kasten wird dann für jede Aufteilung durchgeführt. In Schritt 2 wird das Modell mit den festgelegten Hyperparametern trainiert und auf dem Testdatensatz evaluiert [7].

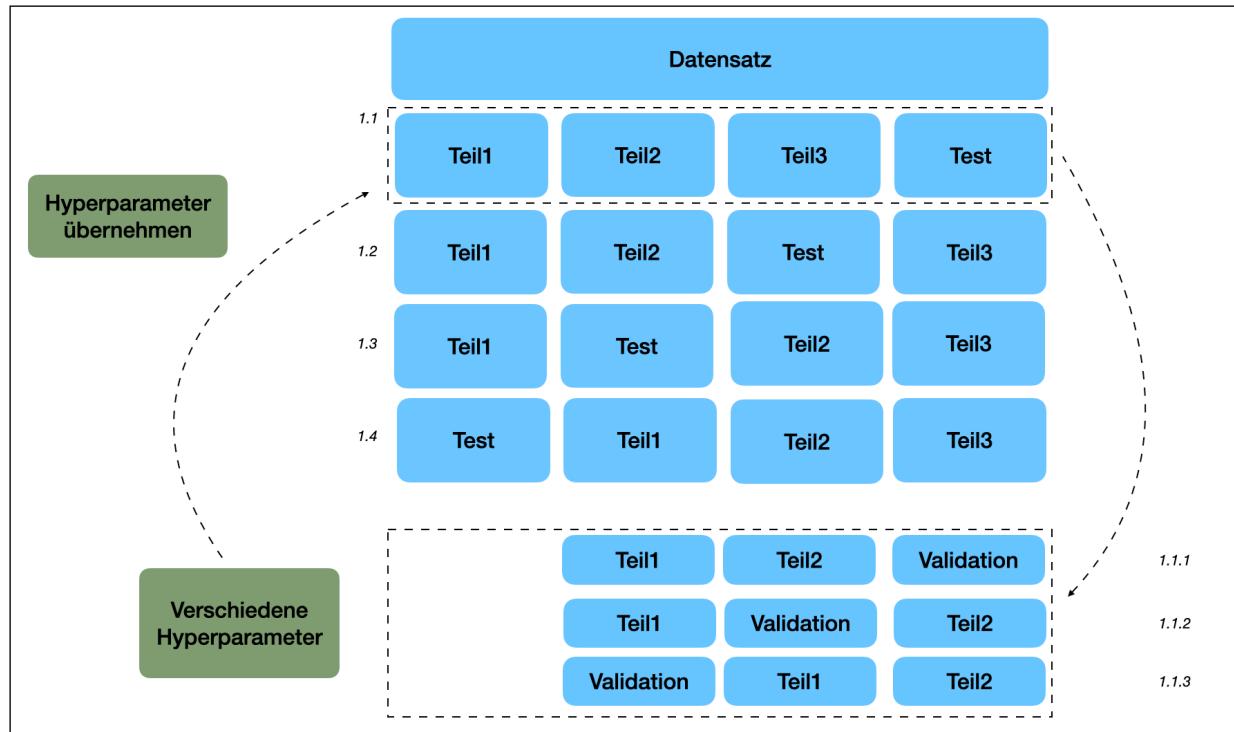


Abb. 6: verschachtelte Kreuzvalidierung

2.8. Ensemble Learning

Bei der Nutzung einzelner, sog. schwachen Lerner, kann es zu zwei sehr verschiedenen und folgenschweren Problemen kommen. Neben *Overfitting* können einzelne schwache Lerner in ihrem Ergebnis stark streuen, sodass verschiedene Durchläufe zum Teil zu sehr unterschiedlichen Ergebnissen führen.

Ensembles im Maschinellen Lernen versuchen diese Probleme zu lösen. Ensemble Learner bestehen aus mehreren schwachen Lernern. Dies sind Modelle, die Vorhersagen mit einer höheren Genauigkeit als 50% treffen können. Sie werden jeweils einzeln trainiert und anschließend zusammengesetzt. Eine Art der Zusammensetzung ist die *Stacked Generalization* (2.9) [8, S. 94f].

Ein Klassifizierer-Ensemble ist ein Ensemble mit Klassifizierern als schwache Lerner.

2.9. Stacked Generalization

Stacked Generalization oder Stacking ist ein Verfahren, um Ergebnisse der einzelnen Klassifizierer (Basisklassifizierer) eines Klassifizierer-Ensembles zusammenzufassen. Hierbei wird ein Metaklassifizierer trainiert. Dieser soll anhand eines zu klassifizierenden Objekts erkennen, welcher Basisklassifizierer zu welchem Anteil genutzt werden sollte [8, S. 105].

Sei T die Anzahl der Basisklassifizierer im Ensemble. Um den Metaklassifizierer zu trainieren, werden zunächst die Basisklassifizierer $h_t, t = 1, \dots, T$ basierend auf dem gegebenen Datensatz D trainiert. D besteht dabei aus den Tupeln $\{x_i, y_i\}_{i=1}^m$ ($x_i \in \mathbb{R}^n, y_i \in \gamma$). x_i entspricht den Ausprägungen der Attribute an der Stelle i und m stellt die Anzahl der zu klassifizierenden Objekte in der Datenmenge dar. γ entspricht dem Zielvektor, der die Klassenzugehörigkeiten enthält.

Anschließend wird für den Metaklassifizierer ein modifizierter Datensatz D' erstellt. Die Ausgaben der Basisklassifizierer x'_i werden die Attribute dieses neuen Datensatzes. Der Zielvektor y bleibt derselbe. D' setzt sich dabei aus $\{x'_i, y_i\}$ mit $x'_i = \{h_1(x_i), h_2(x_i), \dots, h_T(x_i)\}$ zusammen. Der Metaklassifizierer erhält also die Ausgaben der T Basisklassifizierer als Input und lernt daraus, welcher Klasse das zu klassifizierende Objekt angehört. Das Ergebnis ist ein Ensemble Klassifizierer $E(x)$, der für einen unbekannten Punkt x durch den Metaklassifizierer, angewandt auf die Ausgaben der Basisklassifizierer, eine Klassenwahrscheinlichkeit ausgibt. Abb. 7 stellt diesen Vorgang grafisch dar. Algorithmus 1 zeigt den Ablauf in Pseudocode.

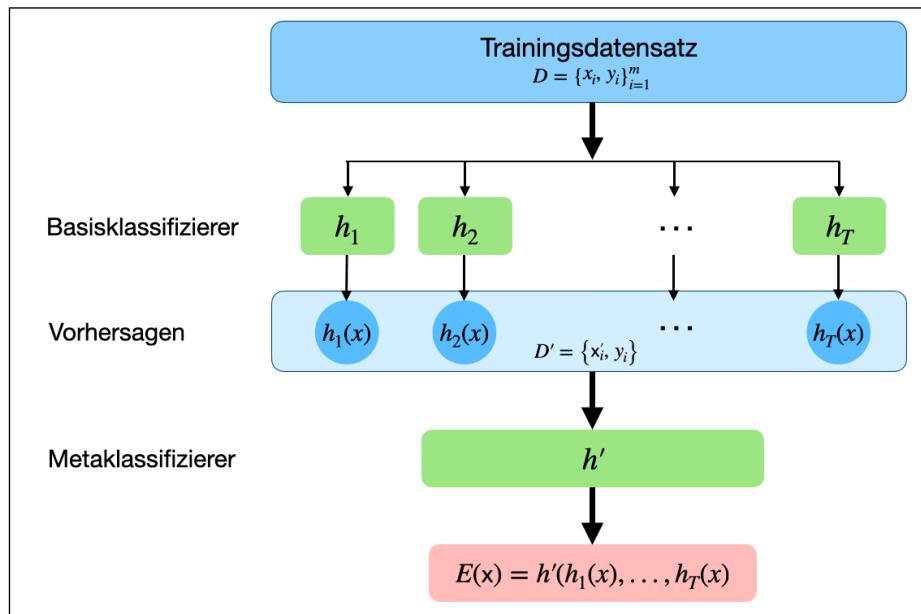


Abb. 7 : Stacked Generalization [9, 11]

Algorithmus 1: Stacked Generalization

Eingabe: Trainingsdaten $D = \{\mathbf{x}_i, y_i\}_{i=1}^m, (\mathbf{x}_i \in \mathbb{R}^n, y_i \in \gamma)$

Ausgabe: Ein Ensemble Klassifizierer $E(x)$

```

for  $t \leftarrow 1$  to  $T$  do
    | trainiere Basisklassifizierer  $h_t$  basierend auf  $D$ 
end
for  $i \leftarrow 1$  do
    | Konstruiere einen neuen Datensatz  $D'$  aus  $\{\mathbf{x}'_i, y_i\}$ ,
    | mit  $\mathbf{x}'_i = \{h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \dots, h_T(\mathbf{x}_i)\}$ 
end

return  $E(x) = h'(h_1(x), h_2(x), \dots, h_T(x))$  mit  $h'$  als Metaklassifizierer

```

[12]

Stacked Generalization mit Kreuzvalidierung

Der beschriebene Stacking Algorithmus nutzt für das Trainieren der Basisklassifizierer und die Konstruktion des Datensatzes D' denselben Datensatz. Um daraus resultierendes Overfitting zu vermeiden, kann zusätzlich Kreuzvalidierung in den Algorithmus integriert werden.

Kreuzvalidierung wird beim Trainieren der Basisklassifizierer für die Konstruktion des Datensatzes D' angewandt. Mit einer k -fachen Kreuzvalidierung werden für jeden der $k-1$ Trainingsteile die Basisklassifizierer trainiert. D' ergibt sich dann aus $\{\mathbf{x}'_i, y_i\}$ mit $\mathbf{x}'_i = \{h_{K1}(\mathbf{x}_i), h_{K2}(\mathbf{x}_i), \dots, h_{KT}(\mathbf{x}_i)\}$. Anschließend wird der Metaklassifizierer mit D' trainiert. Unabhängig davon, werden die Basisklassifizierer auch basierend auf dem gesamten Datensatz D trainiert. $E(x)$ ergibt sich aus dem Metaklassifizierer, der auf D' basiert, angewandt auf die Basisklassifizierer, die auf D basieren. Dieses Vorgehen ist in Abb. 8 und Algorithmus 2 verdeutlicht [12].

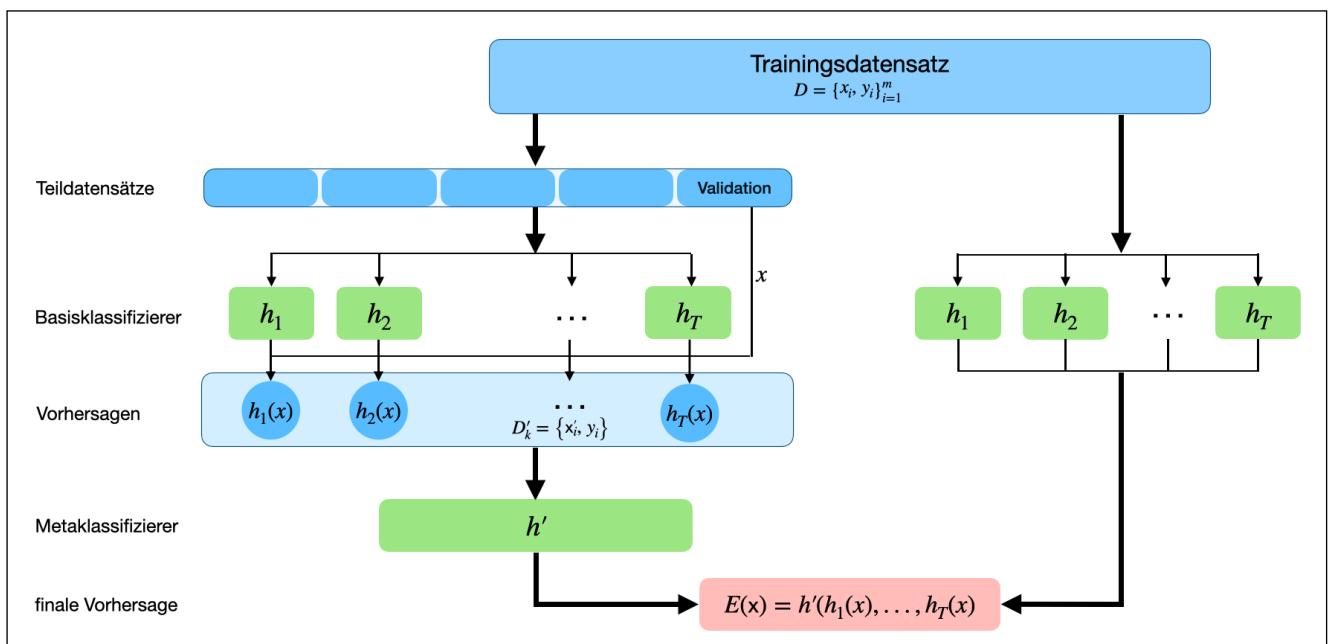


Abb.: 8: Stacked Generalization mit Cross Validation [10,11]

Algorithmus 2: Stacked Generalization mit Kreuzvalidierung

Eingabe: Trainingsdaten $D = \{\mathbf{x}_i, y_i\}_{i=1}^m, (\mathbf{x}_i \in \mathbb{R}^n, y_i \in \gamma)$

Ausgabe: Ein Ensemble Klassifizierer $E(x)$

Splitte Datensatz D in K gleich große Teile: $D = \{D_1, D_2, \dots, D_K\}$

for $k \leftarrow 1$ to K **do**

for $t \leftarrow 1$ to T **do**

 | trainiere Basisklassifizierer h_t basierend auf $D \setminus D_K$

end

for $x_i \in D_k$ **do**

 | Konstruiere einen neuen Datensatz D'_k aus $\{x'_i, y_i\}$,

 | mit $x'_i = \{h_{k1}(x_i), h_{k2}(x_i), \dots, h_{kT}(x_i)\}$

end

end

trainiere Metaklassifizierer h' basierend auf $D' = \bigcap_{k=1}^K D'_k$

for $t \leftarrow 1$ to T **do**

 | trainiere Klassifizierer h_t basierend auf D

end

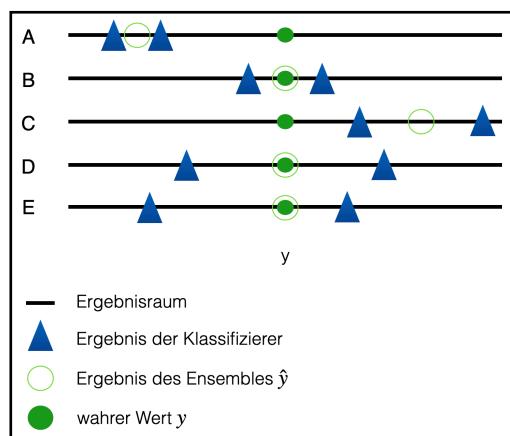
return $E(x) = h'(h_1(x), h_2(x), \dots, h_T(x))$ mit h' als Metaklassifizierer

[12]

2.9.1. Diversität eines Ensembles

Die Ensemble Diversität gibt an, wie weit sich die enthaltenen schwachen Lerner in ihren Vorhersagen unterscheiden. Eine hohe Diversität bedeutet also, dass sich die Lerner bei vielen Ergebnissen unterscheiden und vice versa [8, S. 247].

Ein schwacher Lerner, der keine Fehler machen würde, könnte alleine mindestens genauso exakte Ergebnisse erzielen, wie mit mehreren in einem Ensemble. Man kann also annehmen, dass die Lerner in einem Ensemble nicht perfekt sind und Fehler machen. Unter dieser Annahme scheint es erstrebenswert, dass die Klassifizierer oder Regressoren in unterschiedlichen Bereichen voneinander abweichen und so ihre Schwächen ausgleichen. Es ist sogar üblich die Genauigkeit der einzelnen Modelle absichtlich zu mindern, um so eine größere Streuung der Ergebnisse zu erreichen [13].



In Abb. 9 sind fünf unterschiedliche Szenarien eines Ensemble Outputs visualisiert. Der Ergebnisraum wäre hierbei beispielsweise die Wahrscheinlichkeit, dass ein betrachtetes Objekt einer bestimmten Klasse zugehört.

Wie zu erkennen ist, folgt aus einer höheren Diversität unter den Klassifizierern nicht zwangsläufig eine höhere Genauigkeit des Ensembles. Sind alle Modelle im Ensemble ungenau, so kann auch im Endergebnis kein gutes Ergebnis erzielt werden (A & B). Andererseits ist zu erkennen, dass auch ein Ensemble mit geringer Diversität eine hohe Genauigkeit erreichen kann (C).

Abb. 9: Ensembles und Diversität

Die ersten vier Fälle würden als Gesamtergebnis den Mittelwert der einzelnen Modelle als Ensembleleergebnis ausgeben.

In Fall E erkennt man, dass obwohl ein Klassifizierer deutlich vom wahren Wert y abweicht, das Ensemble diesen immer noch vorhersagt. Dieser Fall könnte durch Stacked Generalization generiert werden.

2.9.2. Diversitätsmaße

Es gibt verschiedene Maße um die Diversität vergleichen und ggf. optimieren zu können. Im Folgenden wird besonders auf Maße zur Messung der Diversität in Klassifizierer-Ensembles eingegangen, da diese in der Arbeit verwendet werden. Die Konzepte können jedoch meist auch für andere Arten von Ensembles übernommen werden.

Die Diversität wird immer über einen gegebenen Datensatz berechnet. Dabei gibt es zwei übliche Ansätze die gewählt werden können. Maße die paarweise vergleichen, betrachten ausschließlich zwei Modelle. Um die Diversität des gesamten Ensembles zu messen, wird diese zunächst zwischen allen enthaltenen Klassifizierern berechnet. Anschließend wird üblicherweise und in dieser Arbeit der Durchschnitt der einzelnen Werte berechnet. Eine andere Fusion wie beispielsweise eine Gewichtung einzelner Modelle bzw. Diversitätswerte wäre auch denkbar. Für ein Ensemble mit $m \in \mathbb{N}$ Klassifizierern müssen $\frac{m(m - 1)}{2}$ einzelne Diversitätswerte berechnet werden.

Ein paarweises Maß der Diversität ist beispielsweise der Spearman'sche Rangkorrelationskoeffizient ρ , der sich vom Korrelationskoeffizienten von Pearson ableitet. Er gibt an, ob ein positiver oder negativer monotoner Zusammenhang zwischen zwei Vektoren besteht und wie stark dieser ausgeprägt ist. Bei der Klassifikation wird ρ für alle Objekte zwischen den zwei Klassifiziererausgaben berechnet. Seien $i, j \in \mathbb{N}$ mit $i, j \leq m$ die Indizes der zu untersuchenden Klassifizierer, dann ist er durch die Kovarianz cov und die Standardabweichung σ , wie folgt definiert:

$$\rho_{rg_i, rg_j} = \frac{cov(rg_i, rg_j)}{\sigma_{rg_i} \sigma_{rg_j}} \quad (10)$$

$rg_z, z \in \{i, j\}$ ergibt sich aus den sortierten Ausgaben der Wahrscheinlichkeiten von z für alle Objekte. ρ liegt in $[-1, 1]$. Für die Messung der Diversität eines Ensembles wird der Absolutwert von ρ benutzt.

Ein weiteres paarweises Maß ist das Disagreement-Maß, welches das Verhältnis zwischen der Anzahl an identischen Outputs der Klassifizierer und der Gesamtzahl an Objekten im Datensatz misst. Dies entspricht der Wahrscheinlichkeit, dass die beiden Klassifizierer bei einem bestimmten Objekt nicht dasselbe Ergebnis erzeugen. Sei n die Anzahl der Objekte im gegebenen Datensatz und $o_k^z \in \{0, 1\}$ mit $z \in \{i, j\}$ die Ausgabe des Klassifizierers z für das Objekt $k \in \mathbb{N}, k \leq n$, dann ergibt sich die Diversität von i und j $D_{i,j}$ wie folgt:

$$D_{i,j} = \frac{1}{n} \cdot \sum_{k=1}^n |o_k^i - o_k^j|. \quad (11)$$

$D_{i,j}$ liegt in $[0, 1]$. Ein höherer Wert bedeutet eine höhere Diversität [8, S. 250].

Die Diversität $\Psi_F \in [0, 1]$ eines Ensembles F mit $z \in \mathbb{N}$ Klassifizierern ergibt sich dann aus

$$\Psi_F = \frac{1}{z} \cdot \sum_{i \neq j} D_{i,j} \quad \text{mit } i, j \in \mathbb{N} \text{ und } i, j \leq z \quad [3,6]. \quad (12)$$

Ein weiteres Maß der Diversität ist das Entropie-Maß E , welches auf einem nicht-paarweisen Ansatz basiert und somit direkt alle Klassifizierer im Ensemble betrachtet. Hinter diesem Maß steht die Überlegung, dass im Falle, dass die Hälfte der Klassifikationswerte für ein Objekt 0 (und der Rest 1) entsprechen, die Diversität am größten ist. Andersherum gäbe es keine Diversität, wenn alle Klassifikationen identisch wären.

Seien z, o_k^z, n und m wie oben, dann ist das Entropie-Maß wie folgt definiert:

$$E = \frac{1}{n} \cdot \frac{2}{m-1} \cdot \sum_{k=1}^m \min \left\{ \left(\sum_{i=1}^m o_k^i \right), \left(m - \sum_{i=1}^m o_k^i \right) \right\} \quad (13)$$

[8, S. 251].

Es existieren noch andere Maße zur Messung der Diversität [8, S. 249ff.]. In dieser Arbeit wird jedoch hauptsächlich der Spearman'sche Korrelationskoeffizient und das Disagreement-Maß verwendet.

2.10. Pareto Optimierung

Die Optimierung eines Problems nach mehr als einer Variable wird Pareto Optimierung genannt. Im Pareto Optimum ist es nicht mehr möglich eine Variable zu verbessern, ohne die andere(n) schlechter zu stellen.

Formell bedeutet dies

$$\min(\{f_1(x), f_2(x), \dots, f_k(x)\}) \text{ bzw. } \max(\{f_1(x), f_2(x), \dots, f_k(x)\}) \quad (14)$$

u.d.B. $x \in S$

mit $k \geq 2$, $f_i : \mathbb{R}^n \Rightarrow \mathbb{R}$ und dem zulässigen Raum $S \subseteq \mathbb{R}^n$ [14].

3. Methoden

Dieses Kapitel beschäftigt sich zunächst mit den in dieser Arbeit verwendeten Elementen, wie Softwarepakete und Datensätzen. Es werden der verwendete Algorithmus bzw. die Modifikationen am ursprünglichen Algorithmus [1] erläutert. Außerdem werden die Rahmenbedingungen für die in dieser Arbeit durchgeführten Untersuchungen beschrieben.

3.1. Software

Da die vorliegende Arbeit auf der Vorarbeit von Markus Flicke beruht, wurden die dort genutzte Software und importierten Pakete weitestgehend übernommen. Als Programmiersprache wurde weiterhin Python v3.8.2 verwendet.

Tabelle 1: Verwendete Software		
Paket	Versionsnummer	Einsatzbereich
Python	3.8.2	Programmiersprache
DEAP	1.3	Evolutionäre Algorithmen
Pandas	1.1.0	Datenanalyse & Datenbearbeitung
Numpy	1.19.1	Datenanalyse & Datenbearbeitung
scikit-learn	0.23.1	Maschinelles Lernen
mlxtend	0.17.3	Maschinelles Lernen insb. Stacked Generalization

Ein Überblick über die verwendeten Pakete bietet Tabelle 1. Das Paket *DEAP* ist ein Framework, welches den Aufbau und das Training von evolutionären Algorithmen erleichtert. *Pandas* und *Numpy* sind Pakete zur Datenanalyse und -manipulation, wobei *Numpy* besonders auf die Nutzung von paketeigenen Arrays setzt. Für das Trainieren der Klassifizierer und deren Evaluation wurden hauptsächlich *scikit-learn* und *mlxtend* genutzt. *mlxtend* wurde verwendet, da es einen Stacking Klassifizierer beinhaltet, der die Basisklassifizierer auf unterschiedlichen Attributen des Trainingsdatensatzes trainieren lässt [14,15,16,17,18,19].

Für die Verwendung von Pandas 1.1.0 waren einige kleinere Anpassungen am vorhandenen Code notwendig.

3.2. Datensätze

Es wurde mit zwei verschiedenen Datensätzen mit kodierten Peptiden getestet. Im VaxinPad Datensatz [20] besitzen einige Peptide immunmodulatorische Eigenschaften. Der Datensatz setzt sich aus 195 sequenzbasierten und 17 strukturbasierten Kodierungen zusammen, die alle dieselben 648 Peptide kodieren. 289 der 648 Peptide besitzen eine immunmodulatorische Eigenschaft und 359 besitzen diese nicht. Der AntiBP2 Datensatz [21] enthält 1993 Peptide. Von diesen besitzen 966 antimikrobielle Eigenschaften und 1027 haben diese nicht. AntiBP2 enthält 249 sequenz- und 18 strukturbasierte Kodierungen. Für beide Datensätze ist gegeben, ob es sich beim entsprechenden Peptid um ein antimikrobielles bzw. immunmodulatorisches Peptid handelt oder ob es diese Eigenschaft nicht besitzt.

Zusätzlich ist für beide Datensätze jeweils eine Matrix der Diversitäten zwischen allen Kodierungen gegeben. Die Diversität wurde mithilfe des Disagreement Maßes gemessen.

3.3. Architektur

Die grundsätzliche Architektur und Modellierung des Algorithmus wurden ebenfalls aus [1] übernommen. Die Modifikationen beziehen sich im Wesentlichen auf folgende Bereiche:

- Ensemble bestehend aus Paaren von struktur- und sequenzbasierten Kodierungen
- (Pareto) Maximierung der Diversität des Ensembles als weitere Optimierungsvariable (optional)
- Evaluation des finalen Ensembles auf unabhängigem Testdatensatz
- Stacked Generalization [mit Kreuzvalidierung] (optional)
- verschachtelte Kreuzvalidierung (optional)

Die optionalen Neuerungen können in der Kommandozeile beim Starten des Algorithmus eingestellt werden.

3.3.1. Datenaufbereitung

Die Peptidkodierungen wurden zunächst mit der gegebenen Diversitätstabelle abgeglichen, um für jedes betrachtete Encodingpaar (sequenzbasiert/strukturbasiert) auch die entsprechende Diversität vorzuweisen.

Das Verhältnis von Trainings- zu Testdatensatz liegt bei 8 zu 1, kann jedoch über die Kommandozeile angepasst werden ('-train_size').

3.3.2. Individuen

Ein Individuum stellt ein Klassifizierer Ensemble einschließlich einer Kodierung pro Klassifizierer dar. Sie wurden durch

$$(\{cls_1, params_1, struk_kodier_1\}, \{cls_2, params_2, seq_kodier_1\}, \dots, \{cls_n, params_n, struk_kodier_{n/2}\}, \{cls_n, params_n, seq_kodier_{n/2}\})$$

(14)

mit $n \in \mathbb{N}$ eine gerade Zahl, $cls \in \{\text{Random Forest, Support Vector Machine}\}$ modelliert. Die Parameter $params$ bestehen aus einem Dictionary und hängen vom jeweiligen Klassifizierer ab. Für den Random Forest Klassifizierer RF ist dies die Anzahl an Entscheidungsbäumen. Für die Support Vector Machine SVM ist es der Regularisierungsparameter C . Als Kernel wird in dieser Arbeit die radiale Basisfunktion genutzt, die sich in [1] bewährt hat.

Die Anzahl der Klassifizierer im Individuum (und im Ensemble) n muss bei der Untersuchung, von sequenz- und strukturbasierten Kodierungen gerade sein. Wird Algorithmus dabei eine ungerade Zahl übergeben, so wird automatisch abgerundet.

Mit dem Befehl `-enco_method` kann der Kodierungsansatz der Kodierungen in den Individuen ausgewählt werden. Entfällt der Befehl, werden struktur- und sequenzbasierte Kodierungen genutzt.

3.3.3. Mutation

Im Mutationsschritt wurden die Parameter der Klassifizierer des Individuums nach der Normalverteilung verändert, um einen Wert zu erreichen, der nicht zu sehr vom ursprünglichen Wert abweicht. Mit einer 4%igen Wahrscheinlichkeit wurde der Klassifizierer komplett ausgetauscht. Die geringe Wahrscheinlichkeit wurde gewählt, da sich der Klassifizierer nach einem Austausch in einem noch nicht optimierten Zustand befindet [1, S. 16]. Mit einer Wahrscheinlichkeit von 10% wurden die Kodierungen des Individuums mit anderen Kodierung im Datensatz ausgetauscht. Die neue Kodierung basiert auf demselben Kodierungsansatz wie die von ihr ersetzte und ist nach einem vorher berechneten Diversitätsmaß ähnlich. Da Kodierungen mit einer geringen Diversität eine ähnliche Genauigkeit aufweisen, ist die Wahl einer Kodierung mit geringer Diversität wichtig. Da nur die selektierte Subpopulation mutiert, könnte eine zufällige Auswahl der neuen Kodierungen den Algorithmus weit vom Optimum entfernen, anstatt langsam in seine Richtung zu konvergieren.

3.3.4. Rekombination und Selektion

Die Rekombination und Selektion entsprechen, bis auf die Zusammensetzung der Fitness, dem Vorgehen aus [1, S. 17]. Hierbei wurden die Individuen nach der Fitness (F_1 -Maß und Diversität) sortiert und die oberen $s = 50\%$ ('-selection s ') für die Elterngeneration selektiert. Mit der DEAP-internen Cross Over Funktion wurden diese nach dem Mutationsschritt rekombiniert.

3.3.5. Diversität

Die Diversität eines Ensembles wird durch den Durchschnitt der Diversitätswerte aller Peptidkodierungspaare erfasst. Ein Kodierungspaar besteht dabei aus einer sequenzbasierten und einer strukturbasierten Kodierung. Zur Messung der Diversität kann sowohl eine vorher berechnete Matrix mit den entsprechenden Diversitätswerten aller Paare genutzt werden als auch eine Berechnung der Diversität während der Laufzeit erfolgen. Dies kann beim Starten des Algorithmus in der Kommandozeile eingestellt werden ('-pre_div'). Über ('-pre_div_path') kann der Pfad zur Diversitätsmatrix angegeben werden.

Um die Diversität als weitere Optimierungsvariable in den evolutionären Algorithmus zu integrieren wurden zwei Ansätze gewählt. Zunächst wurde im ersten Ansatz der Fitnesswert

Φ eines Individuums aus einer Konvexitätskombination aus F_1 -Score des Ensembles F_1 und Diversität Θ gebildet:

$$\Phi = (1 - \lambda) \cdot F_1 + \lambda \cdot \Theta \quad | \text{ mit } \lambda \in [0,1] \quad (15)$$

F_1 ergibt sich aus dem Durchschnitt des F_1 -Scores aller Klassifizierer. Mit dem Argument ‘-div_frac λ ’ lässt sich der Parameter λ konfigurieren. Eine Funktion, die diesen Parameter während des Durchlaufs optimiert, wurde integriert (‘-train_fit_frac’).

Der zweite Ansatz der Integration der Diversität erfolgte mithilfe des Frameworks DEAP. Mit DEAP ist es möglich, die zu optimierenden Fitnesskriterien direkt in das Individuum einzufügen. Statt nur den F_1 Wert oder den zusammengesetzten Wert Φ zu optimieren, werden F_1 -Maß und Diversität pareto-optimiert.

Mit der Option ‘-div_excl’ und einem λ von 0 (Default) wird die Diversität als Optimierungskriterium entfernt. Unter dieser Einstellung wird nur nach dem F_1 -Score optimiert.

3.3.6. Kombination der schwachen Lerner

In der Standardeinstellung entspricht der Durchschnitt der Klassenwahrscheinlichkeiten der einzelnen Klassifizierer der Vorhersage des gesamten Ensembles.

Mit dem Argument ‘-stacking’ wurde die Option einer Stacked Generalization hinzugefügt. Im Evaluationsschritt des Ensembles für die Berechnung des Fitnesswerts und für die finale Evaluation wurden dabei die Klassen mit einem Metaklassifizierer vorhergesagt. Der Metaklassifizierer wurde als Hyperparameter mit einer 10%-igen Chance in jeder Generation verändert. Die möglichen Klassifizierer sind dabei die Logistische Regression, welche initial verwendet wird, Random Forest und ein Deep Learning Klassifizierer (Multi Layer Perceptron [22]) mit 100 Neuronen pro Schicht.

Da die Basisklassifizierer zwar dieselben Peptide vorhersagen sollen, aber auf unterschiedlichen Kodierungen basieren, nutzen sie unterschiedliche Attributteilmengen der Trainingsdaten bei gleicher Zielvariable. Hierzu wurde das Paket *mlxtend* genutzt, das diese Anforderungen erfüllt. Mit der Option ‘-stacking_cv’ kann zusätzlich Stacked Generalization mit Cross Validation genutzt werden.

3.3.7. Evaluation

Für die Evaluation des finalen Modells wurde der F_1 -Score, Matthews Korrelationskoeffizient, die Precision, die Accuracy und der Recall berechnet und erfasst.

In der Standardeinstellung wird der Algorithmus mit 80% des gegebenen Datensatzes trainiert und auf den verbleibenden 20% evaluiert. Für eine verschachtelte Kreuzvalidierung kann mit der Option ‘-outer_cv cv’ die Anzahl der Teile cv angegeben werden, in die der Datensatz unterteilt werden soll.

3.3.8. Argumente

Beim Starten des Algorithmus durch die Kommandozeile können folgende Argumente verwendet werden:

Tabelle 2: Optionale Argumente des Algorithmus

Befehl	Beschreibung	Default
-runtime	Maximale Laufzeit in Stunden	20
-gens	Maximale Anzahl an Generationen	-1 (kein limit)
-n_classifiers	Anzahl der Klassifizierer	4
-savepath	Pfad zum Ordner in dem die Auswertungen gespeichert werden	Results/
-data_path	Pfad zum Datensatz	data/
-dataset	Unterordner in <i>data_path/</i> an.	amp/
-popsize	Populationsgröße	40
-selection	Anteil der Individuen der Population, die in die folgende Generation als Eltern übernommen werden	0,5
-div_excl	Diversität wird aus Fitness ausgeschlossen	FALSE
-div_frac	Anteil Diversität am Fitnesswert $((1-\text{div_frac}) * F_1\text{-Score} + \text{div_frac} * \text{Diversität})$	0
-train_fit_compo	Anteil Diversität am Fitnesswert als Hyperparameter	FALSE
-pre_div	Diversitätswerte aus Matrix entnehmen, statt Berechnung während der Laufzeit	TRUE
-pre_div_path	Pfad zur Diversitätsmatrix	seq_vs_struc_div.csv
-struc_seq_pairs	Hinweis ob Daten in struktur-/sequenzpaar Kodierungen (für schnellere Berechnung der Diversität)	TRUE
-enco_method	Gibt an, welcher Kodierungsansatz in den Individuen verwendet werden soll.	None (struk. und seq.)
-stacking	Stacked Generalization als Ensemblekombinierer	FALSE
-stacking_cv	Stacked CV Generalization als Ensemblekombinierer	FALSE
-fixed_meta_clf	Index eines festen Metaklassifizierers aus [[LogisticRegression(), RandomForestClassifier(), SVC(), MLPClassifier()]] (für Stacking)	None (MK als Hyperparameter)
-inner_train_size	Anteil Trainingsdatensatz an Gesamtdatensatz	0,8
-outer_cv	Anzahl Folds für Nested CV	1 (keine NCV)
-import_ind	Import von Indizes des Datensatzes für Parallelisierung der NCV	None (kein Import)

4. Ergebnisse

Bei den Durchläufen des Algorithmus standen verschiedene Fragen im Fokus. Zunächst wurde der Zusammenhang zwischen Diversität und Genauigkeit des Ensembles betrachtet. Außerdem wurde die Hauptfrage untersucht: Erzielt ein Ensemble, bestehend aus sequenz- und strukturbasierten Kodierungen, genauere Ergebnisse und tut es dies aufgrund einer höheren Diversität? Des Weiteren wurde untersucht, welchen Einfluss Stacked Generalization auf die Genauigkeit hat und welcher Metaklassifizierer bevorzugt wird. Zudem kann die Anzahl der am häufigsten auftretenden Kodierungen aus den Ergebnissen entnommen werden.

Der Trainingsdatensatz enthielt 80% und der unabhängige Testdatensatz 20% des ursprünglichen Datensatzes.

4.1. Finale Individuen

Tabelle 3 zeigt sowohl Genauigkeitsmaße der besten Individuen als auch allgemeine Informationen über den jeweiligen Durchlauf. Alle Genauigkeitsmaße wurden mit einem unabhängigen Testdatensatz, der aus 20% des gesamten Datensatzes bestand, berechnet. Der Pearson'sche Korrelationskoeffizient PEA wurde für die F_1 -Scores und Diversitätswerte aller Ensembles im gesamten Durchlauf berechnet. Die Tabelle ist nach dem Datensatz und F_1 -Score geordnet.

Tabelle 3: Übersicht Finale Individuen

DS	Nr.	Div	F_1	MCC	ACC	P	R	PEA	seq	strk	Gen	PG	EG	Stack
Vaxin Pad	#1	0,3459	0,9679	0,9708	0,9718	0,9651	0,9428	-0,4819	x	x	913	30	2	Nein
	#2	0,0937	0,9649	0,9380	0,9692	0,9483	0,9821	-0,0465	x	x	1000	40	4	Ja
	#3	0,2425	0,9611	0,9301	0,9641	0,9944	0,9297	-0,2796	x	x	950	40	4	Nein
	#4	0,1348	0,9547	0,9140	0,9590	1,0000	0,9210	0,1906	x		1000	40	4	Nein
	#5	0,1254	0,9532	0,9444	0,9615	0,9621	0,9207	-0,0065	x	x	1500	30	2	Nein
	#6	0,0853	0,9423	0,9082	0,9538	0,8909	1,0000	-0,0016	x		1500	40	4	Ja
	#7	0,2989	0,9193	0,9136	0,9333	0,9254	0,8628	0,3085	x	x	1500	40	4	Nein
	#8	0,1130	0,9086	0,9231	0,9256	0,8951	0,8466	-0,2002	x		1500	40	4	Nein
	#9	0,1798	0,8666	0,8821	0,9037	0,7639	0,8333	-0,0138		x	822	40	4	Nein
	#10	0,3113	0,8598	0,7628	0,8846	0,8846	0,8364	-0,0121		x	1000	40	4	Ja
Anti BP2	[23]			0,9000	0,9500									
	#1	0,2278	0,9497	0,9330	0,9492	0,9671	0,8991	0,0304	x		53	40	4	Nein
	#2	0,2332	0,9278	0,8577	0,9276	0,9626	0,8955	-0,2523	x		267	40	4	Ja
	#3	0,3362	0,9231	0,8395	0,9199	0,9208	0,9254	-0,0322	x	x	206	40	4	Ja
	#4	0,3860	0,9179	0,9043	0,9156	0,9321	0,8316	0,3717		x	269	40	4	Nein
	#5	0,0344	0,8022	0,6310	0,8140	0,8488	0,7604	-0,0340		x	253	40	4	Ja
	#6	0,1190	0,8053	0,7784	0,8114	0,8342	0,6243	-0,0113		x	232	40	4	Nein
	[24]			0,8430	0,9214									

Tabelle geordnet nach Datensatz und F_1 -Score, inkl. Benchmark [28,29]

Die Anzahl der Generationen zwischen den beiden Datensätzen unterscheidet sich deutlich. Im Schnitt durchlief der Algorithmus mit dem VaxinPad Datensatz 1207 Generationen. Der Modus liegt bei 1500 Generationen. Beim AntiBP2 Datensatz sind es durchschnittlich 213 Generationen mit einem Maximalwert von 269.

Für den VaxinPad Datensatz liegt der höchste F_1 -Score mit 96,36% und deutlich höchste MCC (97,08%) bei Durchlauf 1 mit einer Populationsgröße von 30 und zwei Klassifizierern. Für diesen Datensatz erreichen drei Individuen mit sequenz- und strukturbasierten Kodierungen die höchste F_1 -Scores. Betrachtet man den MCC, setzen sich sogar die Individuen bei vier der neun Durchläufe aus beiden Kodierungsansätzen zusammen. Des Weiteren liegt der Durchlauf mit sequenzbasierten Kodierungen mit einem F_1 -Wert von 95,47% (ohne Stacking) und 94,23% (mit Stacking) deutlich über dem besten mit ausschließlich strukturbasierten (86,98% bzw. 85,98%). Im Durchlauf VaxinPad #6 wird ein Recall Wert von 1,0 erreicht und in #6 ein Precision Wert von 1,0.

Beim AntiBP2 Datensatz erreichen Individuen mit ausschließlich sequenzbasierten Kodierungen die besten F_1 Werte. Der Durchlauf mit beiden Kodierungsansätzen liegt knapp dahinter und vor den strukturbasierten Kodierungen. Für den MCC liegt ein Individuum mit ausschließlich strukturbasierten Kodierungen an zweiter Stelle, vor einem mit sequenzbasierten.

Der Pearson'sche Korrelationskoeffizient (F_1 -Score/Diversität) reicht von -0,4819 bis +0,3085, ist für die Mehrzahl der Durchläufe beider Datensätze jedoch negativ.

Hinsichtlich der Nutzung von SVM oder RF ließen die Daten keinen Vorteil des einen oder anderen Klassifizierers erkennen.

Es fiel außerdem auf, dass Durchläufe mit Stacking eine höhere Laufzeit pro Generation benötigten.

4.2. Einzelne Durchläufe

Um den Zusammenhang zwischen Diversität und Genauigkeit zu untersuchen, werden hier einige Durchgänge, die vermehrt auftretende Eigenschaften und Besonderheiten aufzeigen, genauer betrachtet. In den folgenden Graphen stellen die dunkelblauen Punkte die Individuen, die für die Rekombination in der folgenden Generation genutzt werden, dar. Die hellblauen Punkte modellieren die aussortierten unteren 50% der jeweiligen Generation. Auf diese beiden Gruppen wird sich im folgenden bezogen, falls von "oberen 50%" bzw. "unteren 50%" die Rede ist. Lila, orange, grüne und rote Linien stehen für den Anteil der am häufigsten verwendeten Kodierungen in der jeweiligen Generation. Wenn nicht anders angegeben, befinden sich auf der X-Achse die Generationen, auf der linken Y-Achse der F_1 -Score und auf der rechten der Anteil der am häufigsten genutzten Kodierungen. Sind zwei Graphen in einer Abbildung zu sehen, stellt die obere die Diversität und die Häufigkeit der Kodierungen und die untere die Diversität dar.

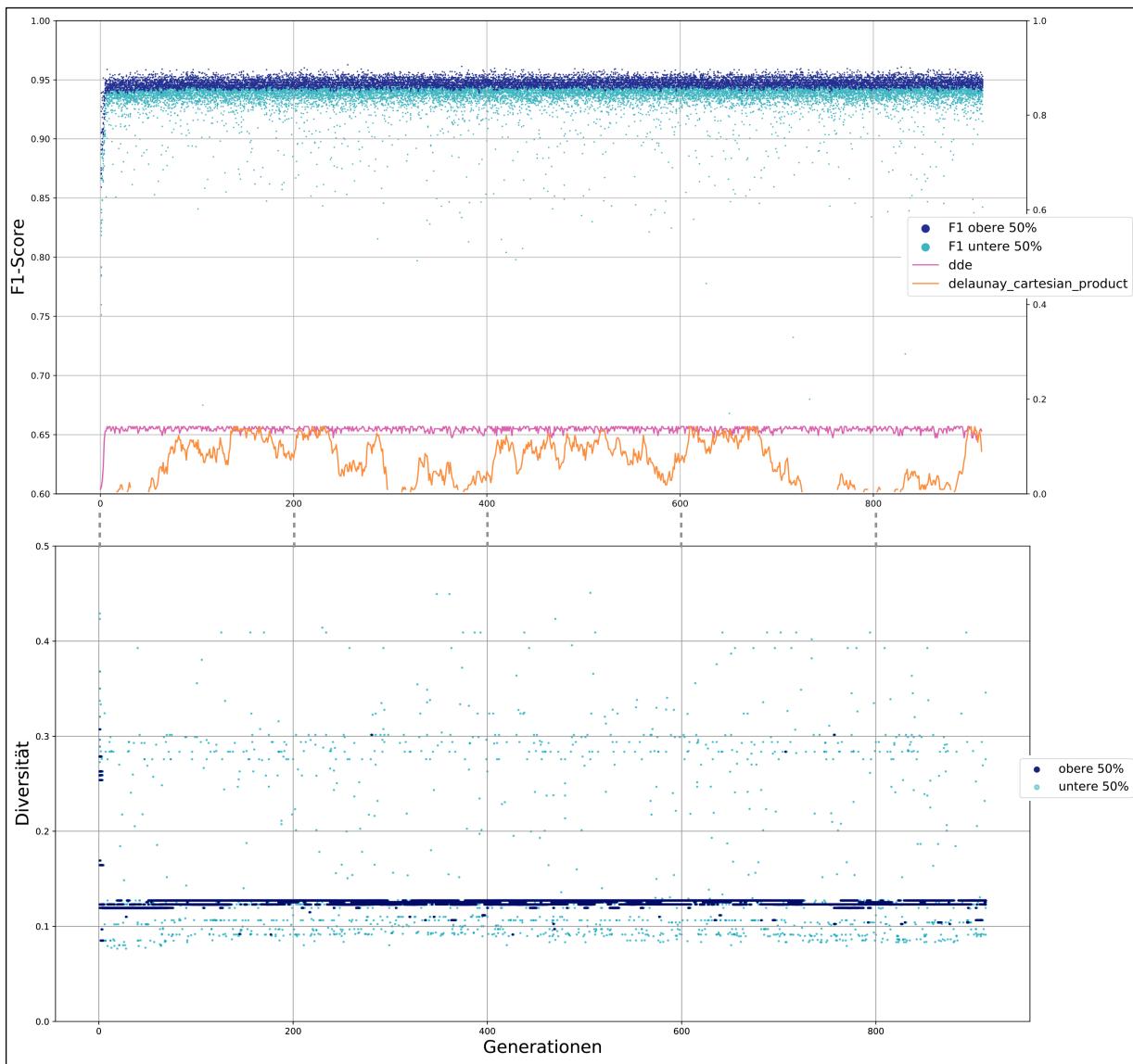


Abb. 11: F_1 -Score, Diversität und Anteil der häufigsten Kodierungen des besten Durchlaufs (VaxinPad #1, Generationen: 913, Populationsgröße: 30, Ensemblegröße 2, ohne Stacking)

Abb. 11 zeigt zwei Graphen desselben Durchlaufs VaxinPad #1. VaxinPad #1 war der Durchlauf mit dem höchsten F_1 -Score und MCC aller Durchläufe. Außerdem war hier der Pearson'sche Korrelationskoeffizient mit -0,48 der extremste Wert. Der F_1 -Score steigt in den ersten Generationen stark an. Danach sind keine nennenswerten Veränderungen zu erkennen. Der F_1 -Score des Großteils der Individuen bleibt innerhalb von 0,92 und 0,96. Die Diversität liegt überwiegend bei 0,15 mit einigen Ausreißern um 0,4. Die Diversität der ausgewählten Individuen für die Rekombination bleibt überwiegend konstant, wohingegen die Diversität der unteren 50% stärker streut. Die hier am häufigsten verwendete sequenzbasierte Kodierung *dde* wird meist in 15% der Individuen in einer Generation genutzt. Dieser Wert wird schon in den ersten Generationen erreicht. Die meist verwendete strukturbasierte Kodierung wird auch maximal in 15% der Individuen genutzt. Der Anteil schwankt jedoch im gesamten Verlauf des Trainings.

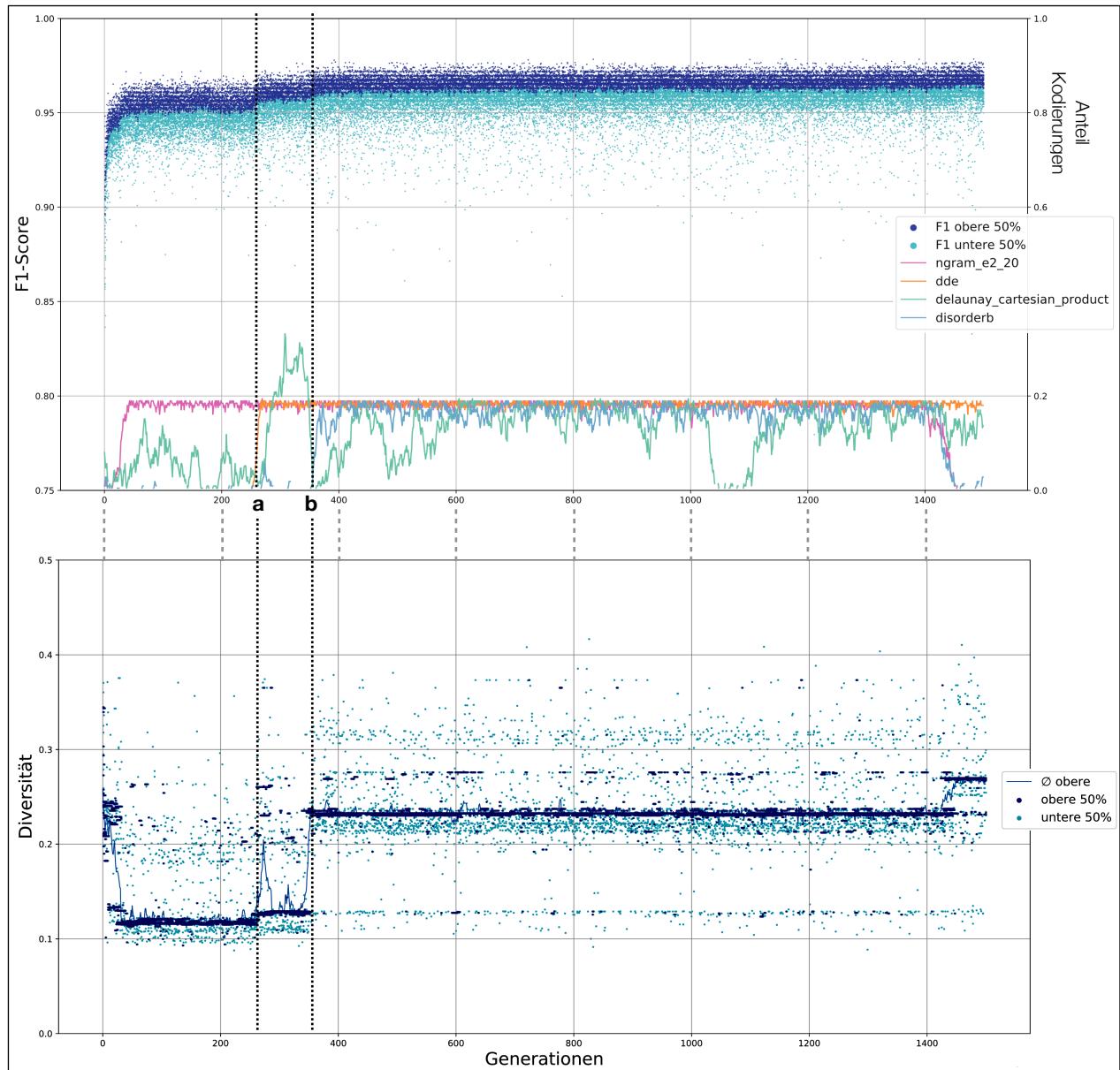


Abb. 12: F_1 -Score, Diversität und Anteil der häufigsten Kodierungen (VaxinPad #7)

Abb. 12 zeigt den Durchlauf 7 des VaxinPad Datensatzes in Tabelle 3. Der Durchschnitt der Diversität ist für eine bessere Übersicht nur für die obere Hälfte (bzgl. der F_1 -Scores) der Population angegeben. Wie beim Durchlauf zuvor steigt der F_1 -Score zu Beginn stark an. Auffällig ist jedoch der Sprung des F_1 -Scores um Generation 250 (**a**) und dessen weiterer Anstieg bis zur 400. Generation. Parallel zum ersten Anstieg verzeichnet auch die Diversität bzw. der Durchschnitt aller Diversitäten einer Generation einen starken Anstieg. Dieser sinkt jedoch wieder, noch während das F_1 -Maß weiter steigt. Zum eingetragenen Zeitpunkt **b** verzeichnet die Diversität nochmals einen starken Anstieg. Diesmal bleibt sie auf diesem Niveau bis zum Ende des Durchlaufs. Anzumerken ist, dass der F_1 -Score in **b** nicht nochmal einen starken Zuwachs erfährt, sondern leicht steigt und dann konvergiert. Während des Anstiegs des F_1 -Scores zu Beginn und auch ab **a** bis Generation 400 steigt ebenfalls mindestens eins der am häufigsten genutzten Kodierungen.

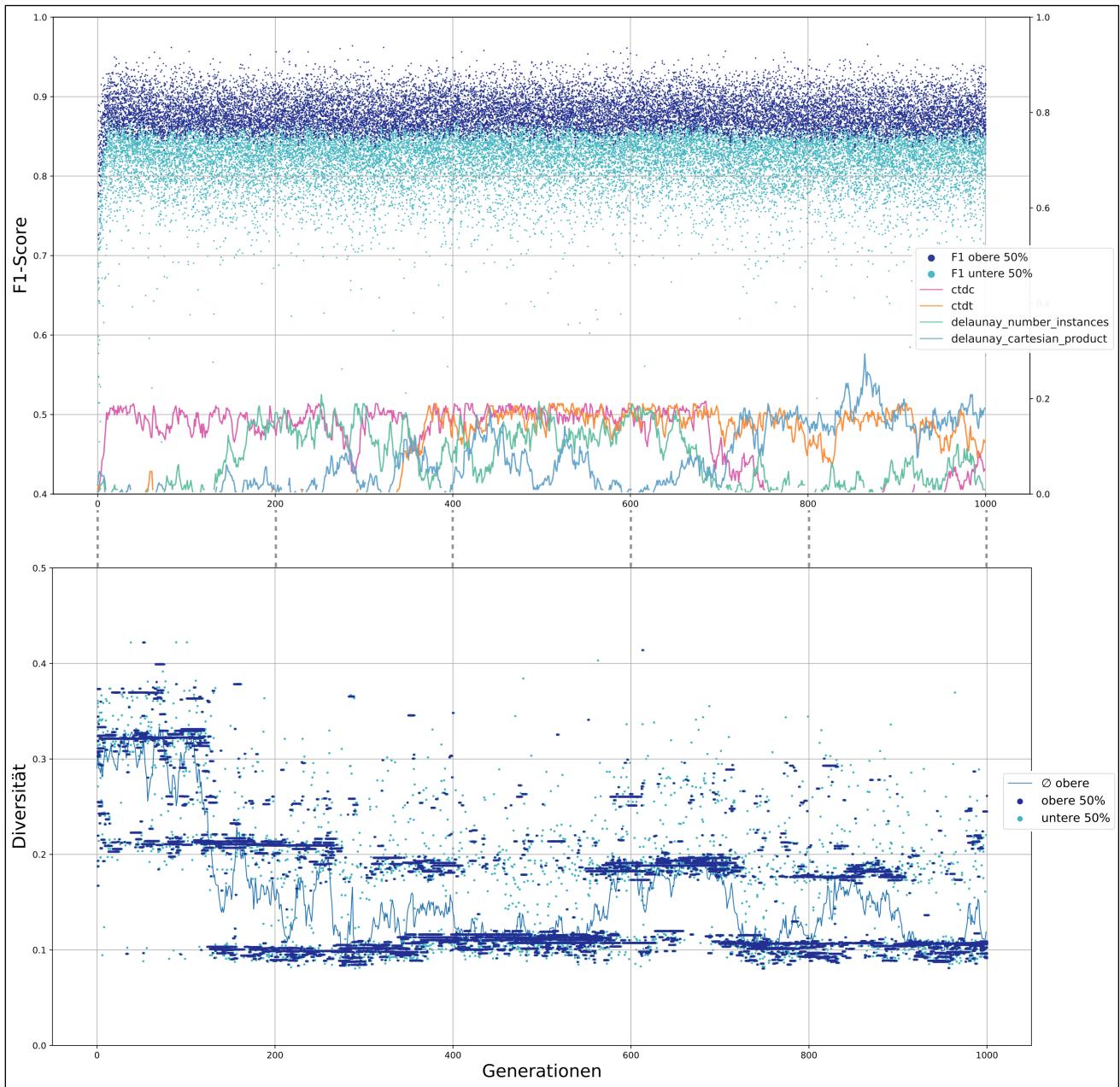


Abb. 13: F_1 -Score und Diversität (VaxinPad #2)

Abb. 13 zeigt den besten VaxinPad Durchlauf mit Stacking (VaxinPad #2). Hier fallen starke Sprünge der Diversitätswerte auf. Langfristig sinkt die Diversität schrittweise während das F_1 -Maß sich nicht merklich verändert. Außerdem streuen die F_1 -Scores deutlich mehr.

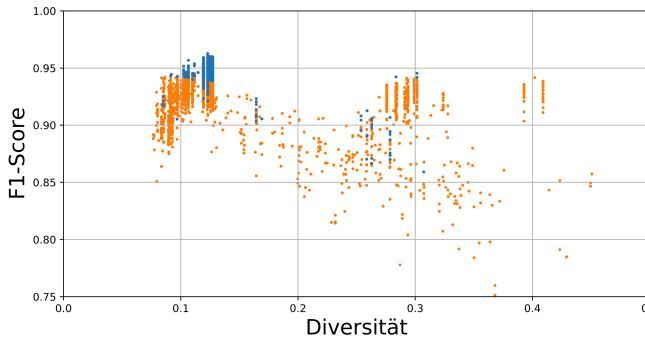


Abb. 14: F_1 -Score gegen Diversität (VaxinPad #1)

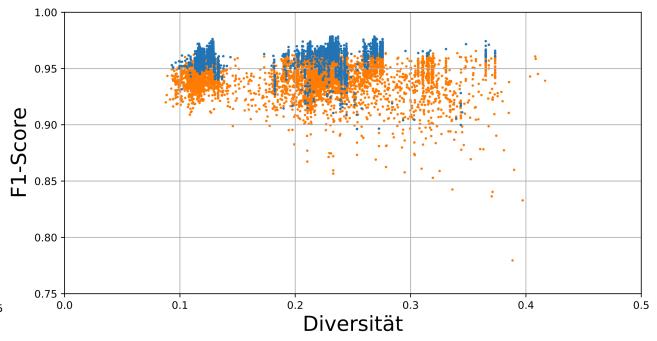


Abb. 15: F_1 -Score gegen Diversität (VaxinPad #7)

Die Abbildungen 14 und 15 zeigen F_1 -Score in Abhängigkeit von der Diversität zu den oben behandelten Durchläufen VaxinPad #1 und VaxinPad #7. Für ersteren ist der negative Trend, der bereits durch den Korrelationskoeffizient von Pearson angedeutet wurde, zu erkennen.

Die Ergebnisdaten der anderen Durchläufe erzeugen ähnliche Graphen wie die der beiden hier näher untersuchten. Mit der Verwendung von Stacked Generalization gibt es seltener Sprünge des F_1 -Scores wie in Abb. 12 zum Zeitpunkt **a** zu sehen.

Abb. 16 zeigt die Häufigkeit der sequenzbasierten Kodierungen in den finalen Individuen von zwölf Durchläufen (AntiBP2 und VaxinPad Datensatz) mit jeweils mehr als 500 Generationen. Die Individuen der Durchläufe setzten sich aus sequenz- und strukturbasierten und aus nur sequenzbasierten Kodierungen zusammen. Die am häufigsten vorkommenden Kodierungen sind mit einer Anzahl von vier *ngram_a2_20* und *ssec*. 29 der 35 Kodierungen kommen nur einmal vor.

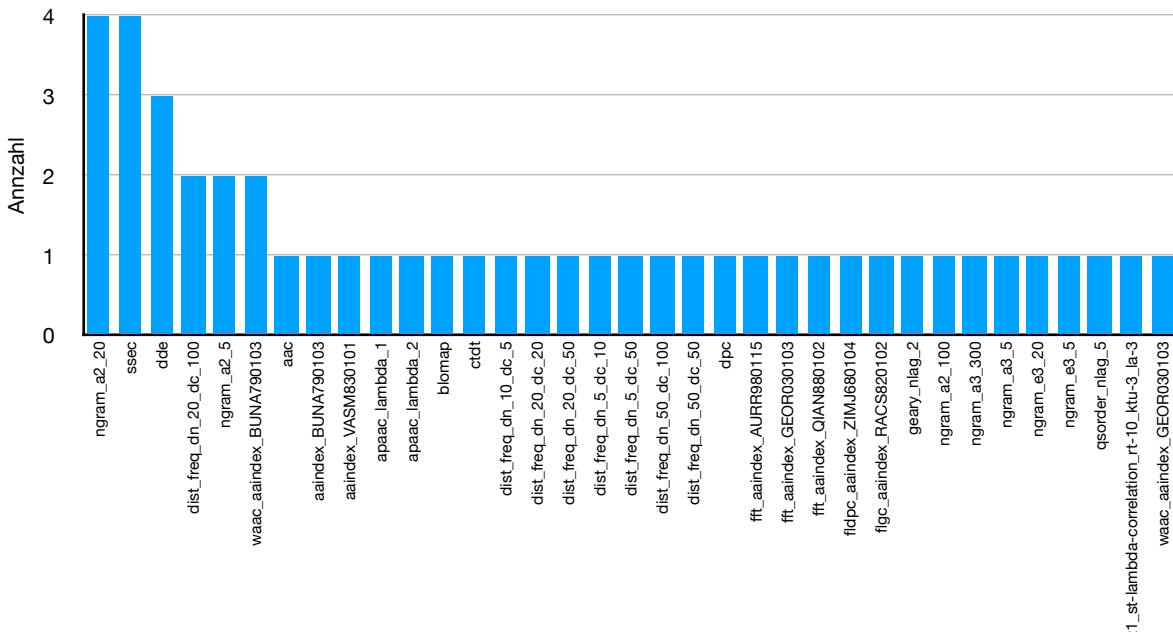


Abb. 16: Häufigkeit der sequenzbasierten Kodierungen in den finalen Individuen (aus 12 Durchläufen mit mehr als 500 Generationen pro Durchlauf)

Abb. 17 zeigt die Häufigkeit der strukturbasierten Kodierungen in den finalen Individuen von zehn Durchläufen mit ebenfalls über 500 Generationen pro Durchlauf. Die häufigste Kodierung war die *delaunay_average_distance*. Sie liegt mit acht Vorkommnissen deutlich vor der zweithäufigsten Kodierung *delaunay_cartesian_product* (fünf).

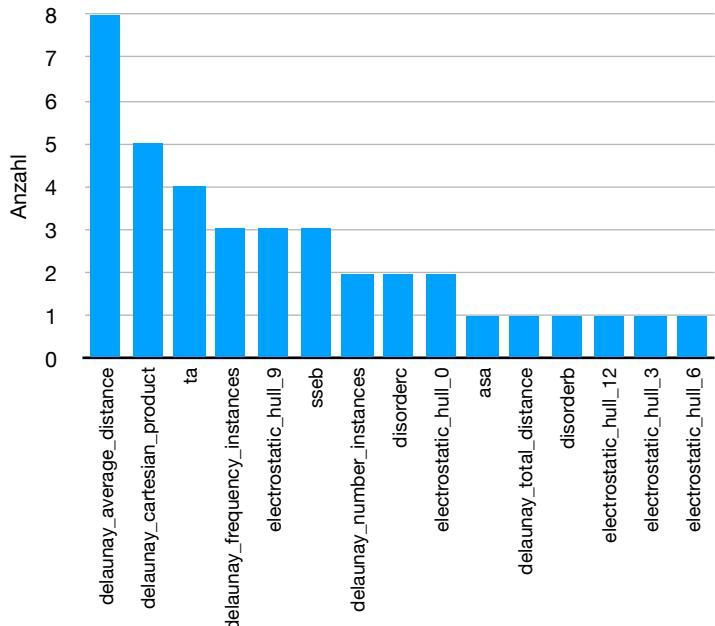


Abb. 17: Häufigkeit der strukturbasierten Kodierungen in den finalen Individuen (aus 10 Durchläufen mit mehr als 500 Generationen pro Durchlauf)

5. Diskussion

In diesem Kapitel werden die obigen Ergebnisse analysiert.

5.1. Anzahl der Generationen und Größe des Datensatzes

Der AntiBP2 Datensatz enthält mit 1993 deutlich mehr Peptide als der VaxinPad Datensatz (648 Peptide). Beim Training benötigt ein Klassifizierer schon dadurch ca. doppelt so viel Zeit mit diesem Datensatz.

Hinzu kommt, dass der AntiBP2 Datensatz insgesamt 267 Kodierungen enthält. Der VaxinPad beinhaltet hingegen nur 202. Für die Berechnung der Diversität ist eine Suche in der Diversitätsmatrix deshalb noch einmal langsamer.

Für diesen Datensatz ist die zeitliche Abbruchbedingung schneller erreicht. Auch wenn teilweise gute Ergebnisse nach unter 300 Generationen erreicht werden, ist die Aussagekraft bspw. über die besten Kodierungen und damit auch über die Diversität, eingeschränkt.

Da die Kodierungen *cgr_res_200_sf_0.5* und *cgr_res_200_sf_0.8632713* nochmal um ein Vielfaches größer sind als alle anderen Kodierungen, wurden diese beiden Kodierungen zu Gunsten einer besseren Laufzeit aus dem Datensatz entfernt. Zum Vergleich: die Dateien dieser beiden Kodierungen sind 320 MB groß, wobei die drittgrößte Kodierungsdatei 80MB groß ist und die Mehrheit der Dateien kleiner als 10 MB sind.

Für den Algorithmus bedeutet dies, dass der gewählte Ansatz nicht für jeden beliebig großen Datensatz anwendbar ist, da eine entsprechend hohe Laufzeit oder Rechenleistung nicht immer gegeben ist.

5.2. Genauigkeit

In Nagpal *et al.* wird für den VaxinPad Datensatz im besten Fall eine MCC von $0,9 \pm 0,02$ und eine Accuracy von $0,95 \pm 0,125$ erreicht [23]. Der beste Durchlauf mit dem vorgestellten Modell übertrifft dies mit einem Wert von 0,97 für beide Maße auf dem unabhängigen Testdatensatz deutlich (siehe Tabelle 3).

Ein Recall Wert 1 (in VaxinPad #6) bedeutet, dass alle als immunmodulatorisch klassifizierten Peptide auch diese Eigenschaft besaßen. Einige nicht immunmodulatorische Peptide wurden als immunmodulatorisch eingestuft, da sonst der F_1 -Score auch bei 1 liegen würde. Ein Recall Wert von 1 (in VaxinPad #4) bedeutet, dass alle immunmodulatorischen Peptide im Testdatensatz korrekt als solche erkannt wurden. Einige Peptide ohne diese Eigenschaft wurden jedoch auch als immunmodulatorisch klassifiziert, da auch hier der F_1 -Score nicht bei 100% liegt.

Mit dem AntiBP2 Datensatz erzielten Lata *et al.* eine Accuracy von 0,9214 und einen MCC von 0,843 [24]. Vier der sechs vorgestellten Durchläufe übertreffen den MCC Wert mit einem Bestwert von 0,933. Die Accuracy wird in zwei Durchläufen knapp übertroffen. Der beste Wert liegt hier bei 0,9497.

Ein bedeutender Faktor für die Überlegenheit des hier vorgestellten Modells dürfte auch die Nutzung des EA sein. In [23] und [24] wird nicht näher erläutert, wie genau die Hyperparameter optimiert wurden. Außerdem wurden nur eine geringe Anzahl an Kodierungen manuell untersucht.

5.3. Genauigkeit und Diversität

Abb. 10 stellt den F_1 -Score in Abhängigkeit der Diversität dar. Die hellblauen Punkte repräsentieren die Ergebnisse aus dem AntiBP2 Datensatz und die dunkelblauen Punkte die aus dem VaxinPad Datensatz. Es sind außerdem drei logistische Regressionslinien eingezeichnet. Die graue, gestrichelte Linie ist die Regressionslinie der Ergebnisse beider Datensätze. **a** stellt den Durchlauf VaxinPad #2 und **b** VaxinPad #8 dar. Für den AntiBP2 Datensatz ist ein positiver Trend zu erkennen, für den VaxinPad Datensatz ein leicht negativer.

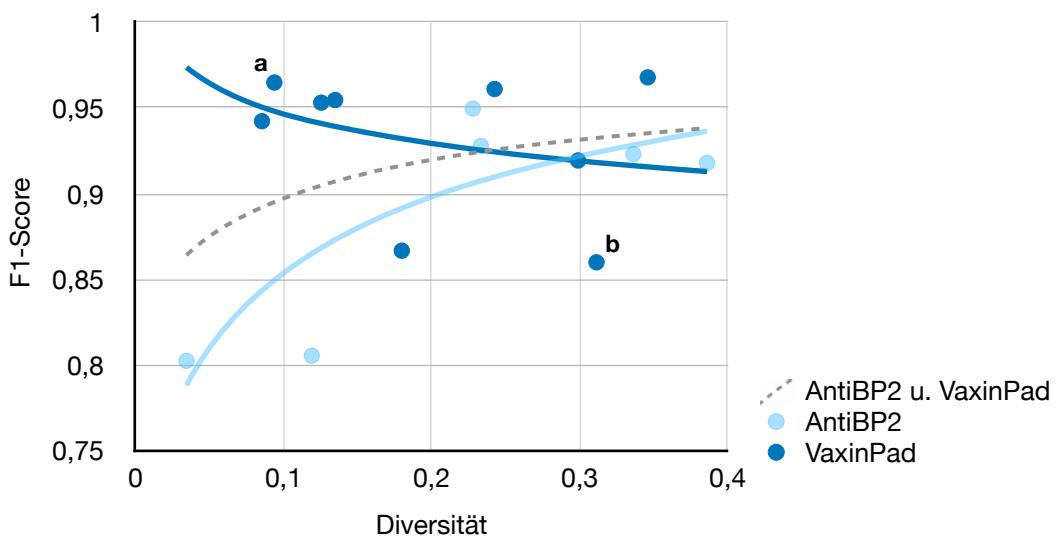


Abb. 10: F_1 gegen Diversität inkl. log. Regressionslinien, der finalen Individuen mit dem AntiBP2 und dem VaxinPad Datensatz

Die Auswertung der finalen Individuen zeigt, dass sowohl mit geringer Diversität eine hohe Genauigkeit (VaxinPad #2) als auch mit hoher Diversität eine geringe Genauigkeit erreicht werden kann (VaxinPad #8). Auch für weniger extreme Werte lässt sich kein Trend erkennen.

Wie Abb. 14 und 15 zeigen, gibt es auch innerhalb der Durchläufe keinen erkennbaren (linearen) Zusammenhang. Die dort auffällige Häufung der Punkte um einen Diversitätswert kommt daher, dass Kodierungen nur mit einer Wahrscheinlichkeit von 10% mutierten und somit in vielen Ensembles dieselben Kodierungen auftraten. Sie zeigen aber auch, dass für einen Diversitätswert die F1-Scores in einem relativ großen Intervall liegen (bis zu 20 Prozentpunkte).

Mit den generierten Daten kann die These eines existierenden positiven Zusammenhangs zwischen Diversität, gemessen mit dem Disagreement Maß, und Genauigkeit des Modells also nicht unterstützt werden. Kuncheva und Whitaker finden in ihrer Studie ähnliche Resultate. Für einen Datensatz wird mit unterschiedlichen Diversitätsmaßen zwar ein positiver Zusammenhang festgestellt, für einen anderen jedoch nicht [25]. Eine grundsätzliche Aussage über die Genauigkeit eines Modells mithilfe eines Diversitätsmaßes lässt sich offensichtlich nicht treffen. Die Diversität als Optimierungsparameter eines evolutionären Algorithmus ist folglich nicht immer sinnvoll.

5.4. Sequenz und strukturbasierte Kodierungen

Hinsichtlich der acht aufgeführten Durchläufe mit dem VaxinPad Datensatz setzen sich die Individuen der drei besten Ergebnisse (bzgl. F_1 -Score) aus beiden Kodierungsansätzen zusammen. Betrachtet man den MCC oder die Accuracy enthalten sogar die Individuen der vier besten Durchläufe sequenz- und strukturbasierte Kodierungen.

Die Nutzung eines Ensembles mit Klassifizierern, die auf sequenz- und strukturbasierten Kodierungen aufbauen, könnte auch dazu beigetragen haben, dass die Referenzwerte [23,24] übertroffen wurden.

Da die beiden Durchläufe mit ausschließlich strukturbasierten Kodierungen deutlich schlechter abschneiden, ist eine Nutzung von Kodierungen mit nur diesem Ansatz nicht zu empfehlen.

5.5. Häufigkeiten der Kodierungen

Für beide Kodierungsansätze gibt es Kodierungen, die häufiger im finalen Individuum auftreten als andere, was darauf schließen lässt, dass diese besser für die Vorhersage von antimikrobiellen und immunmodulatorischen Peptiden geeignet sind. Da in den Datensätzen nur 17 (VaxinPad) bzw. 18 (AntiBP2) sequenzbasierte und 196 bzw. 249 strukturbasierte Kodierungen ist die Wahrscheinlichkeit, dass sich sequenzbasierte Kodierungen wiederholen trivialerweise deutlich höher.

5.6. Stacked Generalization

Für Durchgänge mit sequenz- und strukturbasierten Kodierungen scheint Stacked Generalization eine ähnliche gute Leistung zu erzielen wie ohne das Verfahren. Allerdings war nur einer von fünf dieser Durchläufe mit Stacking. Dieser erzielte den zweitbesten Wert und sogar den besten mit einer Populationsgröße von 40 und einer Ensemblegröße von 4. Für die Durchläufe mit Kodierungen mit nur einem Kodierungsansatz schnitten Individuen mit Stacking schlechter ab.

Für eine genaue Aussage müssen noch weitere Versuche durchgeführt werden.

5.6.1. Metaklassifizierer

Mit der Einstellung des Metaklassifizierers als Hyperparameter wurde keiner der vier Modelle (RF, SVM, Logistische Regression, MLP) favorisiert. Für die Durchläufe waren alle der MK sowohl häufigster MK innerhalb eines Durchlaufs als auch im besten Individuum enthalten.

5.7. Wahl von n und der Populationsgröße

In [1] wurde $n = 5$ gewählt, da es sich als guter Kompromiss zwischen Laufzeit und Genauigkeit erwies. Eine Wahl von 5 war für Ensembles mit sequenz- und strukturbasierten Kodierungen nicht möglich, da das Ensemble aus einem Paar der beiden bestehen sollte. So wurde in den meisten Durchläufen $n = 4$ gewählt.

Die Populationsgröße von 40 Individuen wurde ebenfalls durch die Versuche aus [1] übernommen.

Zum Ende des Versuchszeitraums wurde für eine bessere Laufzeit in zwei Durchgängen $n = 2$ bei einer Populationsgröße von 30 gewählt. Einer der beiden Durchgänge lieferte sogar die höchsten Genauigkeitswerte. Die besten vier Ergebnisse unterscheiden sich jedoch nur marginal und ein weiterer Durchlauf mit diesen Hyperparametern schnitt etwas schlechter ab als zwei Durchläufe mit $n = 4$ und PG = 40. Dies lässt darauf schließen, dass mit beiden Einstellungen dieser Hyperparameter ähnliche Ergebnisse erreicht werden können. Für eine deutlich bessere Laufzeit ist somit für zukünftige Versuche durchaus eine kleinere Ensemble- und Populationsgröße denkbar.

5.8. Fitness als Linearkombination aus F_1 -Score und Diversität

Die Zusammensetzung der Fitness aus F_1 -Score und Diversität als Linearkombination, wie in Formel (15) beschrieben, erwies sich, zumindest für die gegebenen Datensätze, als nicht sinnvoll. Der Wert der Diversitäten lag hier meist unter 0,5; der F_1 -Score erreichte hauptsächlich Werte über 0,9. Optimierte man schlicht nach einem maximalen Fitnesswert, der sich aus diesen beiden Faktoren zusammensetzt, reduziert jeder Anteil der Diversität diesen Wert. Mit festem λ wird also eine geringere Diversität präferiert. In Durchläufen mit λ aus Formel (15) als Hyperparameter machte sich dies insofern bemerkbar, dass im Verlaufe des Durchlaufs λ zu Gunsten des F_1 -Scores sukzessive weniger wurde.

6. Ausblick

Eine denkbare Erweiterung dieser Arbeit ist die Zusammensetzung des Ensembles hinsichtlich ihres Kodierungsansatzes als Hyperparameter aufzunehmen. So würde der Algorithmus selbst auswählen, wie viele Kodierungen eines Ensembles strukturiert und wie viele sequenzbasiert sind. Es könnten dabei auch noch weitere Kodierungsansätze untersucht werden.

Derzeit ergibt sich die Diversität aus dem Durchschnitt der Diversitäten der Kodierungen im Individuum. In einer weiterführenden Arbeit kann untersucht werden, ob das Maximum der einzelnen Diversitäten als Ensemblediversität zu besseren Ergebnissen führt oder einen Zusammenhang zur Genauigkeit erkennen lässt.

Es wäre interessant herauszufinden, ob die Verwendung anderer Diversitätsmaße, wie dem Entropiemaß, zu ähnlichen Ergebnissen führen würde oder ob damit ein Zusammenhang zwischen Diversität und Genauigkeit nachweisbar wäre.

Außerdem wäre es möglich die Ergebnisse der einzelnen Klassifizierer inklusive der Kodierungen noch während der Laufzeit zu analysieren. So könnten Kodierungen, die eine hohe Genauigkeit erreichen, bei der Rekombination und Mutation favorisiert werden. Die Kodierungen werden derzeit insofern favorisiert, dass in der Selektion nur die besten 50% ausgewählt werden und bei der anschließenden Mutation der Kodierungen eher ähnliche Kodierungen ausgewählt werden. Die durchgeführten Versuche zeigten aber, dass die Ähnlichkeit nicht zwangsläufig eine ähnlich gute Performance bedeutet. Eine Ordnung der Kodierungen nach Genauigkeit im Mutationsschritt könnte schneller zum Optimum führen.

Bei der Mutation der Kodierungen ließe sich auch mit einer geringen Wahrscheinlichkeit eine komplett zufällige Kodierung auswählen. Anschließend sollte untersucht werden, ob der Algorithmus schneller oder langsamer konvergiert.

Um die Laufzeit des Algorithmus zu verbessern und damit dessen Praktikabilität für größere Datensätze zu erhöhen, sollte Multi-Processing und/oder Multi-Threading eingeführt werden. Zwar müssen die Ensembles bzw. Individuen einer Population komplett evaluiert worden sein, bevor sie für die folgende Generation rekombiniert und mutiert werden können, jedoch kann die Evaluation der Ensembles innerhalb einer Generation parallel stattfinden.

Die Anzahl der Klassifizierer eines Ensembles n könnte auch als Hyperparameter gesetzt werden. Um zu verhindern, dass n zu groß wird und es zu einer hohen Gesamlaufzeit kommt, sollte sich eine längere Evaluationslaufzeit des Ensembles negativ auf dessen Fitness auswirken.

Eine nähere Untersuchung der meistgenutzten Kodierungen könnte weitere Erkenntnisse über Kodierungen von Peptiden zur Vorhersage von antimikrobiellen und immunmodulatorischen Eigenschaften bringen.

Für die Evaluation könnte zusätzlich der Testfehler berechnet werden, um Zeichen von Overfitting zu erkennen.

7. Fazit

Der Algorithmus übertrifft mit den beiden Datensätzen VaxinPad und AntiBP2 bei der Klassifikation von antimikrobiellen und auch immunmodulatorischen Peptiden die Benchmarks. Für den VaxinPad Datensatz wurde ein F_1 -Score von bis zu 96,79%, eine MCC von 97,08% und eine Accuracy von 97,18% erreicht. Mit dem AntiBP2 Datensatz wurde ein F_1 -Score von 96,79%, ein MCC von 93,3% und eine Accuracy von 96,71% erreicht. Ob diese auch signifikant besser als der Referenzwert sind, gilt noch zu überprüfen.

Wie vermutet, deuten auch die generierten Daten darauf hin, dass ein Ensemble bestehend aus sequenz- und strukturbasierten Kodierungen Schwächen des jeweils anderen Ansatzes kompensieren kann. Hinsichtlich des MCC nutzen vier der insgesamt zehn Durchläufe des VaxinPad Datensatzes beide Kodierungsansätze. Durchläufe mit ausschließlich sequenzbasierten Kodierungen schneiden überwiegend schlechter ab, jedoch besser als jene, in denen nur strukturbasierte Kodierungen genutzt wurden. Trotzdem ließ sich der Zusammenhang zwischen Diversität und Genauigkeit mit dem verwendeten Disagreement-Maß, nicht nachweisen. Für die Vorhersage von Peptiden eignet sich folglich ein Ensemble aus Klassifizierern, die auf sequenz- und strukturbasierten Kodierungen aufbauen sehr gut. Das Disagreement Maß als Optimierungsparameter wirkte sich für die betrachteten Datensätze nicht positiv auf die Genauigkeit des Modells aus.

Ob Stacked Generalization hier zu einer höheren Genauigkeit führt, konnte aufgrund von mangelnden Daten noch nicht endgültig geprüft werden. Hierfür sollten noch weitere Durchläufe durchgeführt werden.

Der Algorithmus ist auf verschiedenen weiteren Datensätzen anwendbar. Deutliche Verbesserungen hinsichtlich der Laufzeit sind durch Parallelisierung der Ensembleevaluierung möglich.

Literaturverzeichnis

1. M. A. Flicke, *Evolutionary optimisation of ensemble classifiers for predicting antimicrobial peptides*, 2019
2. S. Spänig and D. Heider, *Encodings and models for antimicrobial peptide classification for multi-resistant pathogens*, BioData Mining, 12.1, S. 2-4 & 14
3. K. Weicker, *Evolutionäre Igoithmen*. Springer-Verlag, 2015, S.39
4. G. James, D. Witten, T. Hastie, R. Tibshirani, *An introduction to statistical learning*, 112 New York: springer, 2013, S. 130-135
5. M. H. Hassoun, *Fundamentals of artificial neural networks*, MIT press, 1995, S. 1-2
6. C. Cao, D. Chicco, M. M. Hoffman, *The MCC-F1 curve: a performance evaluation technique for binary classification*, arxiv, 2020, S. 2-3
7. S. Parvandeh, H. W. Yeh, M. P. Paulus, B. A. McKinney, *Consensus features nested cross-validation*, Bioinformatics, 36.10, 2020, S. 2
8. L. Kuncheva, 2, *Combining Pattern Classifiers - Methods and Algorithms*, 2014
9. S. Raschka, Abrufbar: http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/ (Abrufdatum: 13.11.2020)
10. Abrufbar: http://rasbt.github.io/mlxtend/user_guide/classifier/StackingCVClassifier/ (Abrufdatum: 13.11.2020)
11. S. Raschka Lizenz: <http://rasbt.github.io/mlxtend/#license> (Abgeru
12. C. C. Aggarwal, *Data Clustering - Algorithms and Applications*, 2014, S. 498-500
13. Freund, R. Schapire, N. Abe, *A short introduction to boosting*, Journal-Japanese Society For Artificial Intelligence 14.771-780, 1999
14. K. Miettinen, *Nonlinear Multiobjective Optimization*, 1998, S. 5-6
15. Deap. deap.readthedocs.io.
16. SKlearn. <https://scikit-learn.org>
17. Numpy <https://numpy.org/>
18. Pandas <https://pandas.pydata.org/>
19. mlxtend <http://rasbt.github.io/mlxtend/>
20. VaxinPad Datensatz <https://webs.iiitd.edu.in/raghava/vaxinpad/sequences.php>
21. AntiBP2 Datensatz <http://crdd.osdd.net/raghava/antibp2/>
22. S. K. Pal, S. Mitra, *Multilayer perceptron, fuzzy sets, classifiaction*, 1992
23. G. Nagpal, K. Chaudhary, P. Agrawal, G. Raghava, *Computer-aided prediction of antigen presenting cell modulators for designing peptide-based vaccine adjuvants*, Journal of translational medicine, 16.1, 2018
24. S. Lata, N. K. Mishra, G. P. Raghava, *AntiBP2: improved version of antibacterial peptide prediction*, BMC Bioinformatics, 11, 2010, S. 4
25. L. I. Kuncheva, and C. J. Whitaker, *Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy*, Machine learning 51.2, 2003