h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Fachbereich Mathematik und Naturwissenschaften

Angewandte Mathematik B.Sc.

Projektarbeit:
Regression with an Abalone Dataset

vorgelegt am: 13. Oktober 2024
von: David van Weyck, Max Uhl
Matrikel Nr.: 772682, 772773

Modulverantwortlicher: Dr. Inna Mikhailova
Hochschule Darmstadt

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Im Zuge des rasanten technologischen Fortschritts und der exponentiell wachsenden Datenmengen hat sich das maschinelle Lernen als unverzichtbares Werkzeug in zahlreichen wissenschaftlichen und industriellen Anwendungsbereichen etabliert. Diese Arbeit beschäftigt sich mit der Evaluation verschiedener maschineller Lernmethoden zur Lösung einer anspruchsvollen Regressionsaufgabe. Grundlage der Untersuchung bildet der Kaggle-Wettbewerb „Regression with an Abalone Dataset“ [11], bei dem das Ziel darin besteht, das Alter von Abalonen anhand der Anzahl der Ringe auf deren Schale vorherzusagen.

Aufgrund des begrenzten Umfangs dieser Arbeit werden ausschließlich drei der wettbewerbsentscheidenden Methoden untersucht, die durch ihre erfolgreiche Anwendung in den Top-3-Platzierungen [10] [18] [16] des Wettbewerbs hervorgehoben wurden. Um eine Vergleichsbasis zu schaffen, wird zusätzlich die in der universitären Lehre oft behandelte Methode der Linearen Regression herangezogen. Diese dient als Benchmark zur Bewertung der fortgeschritteneren Ansätze. Darüber hinaus werden die für die Datenverarbeitung und Validierung eingesetzten Verfahren ausführlich erläutert.

Eine zentrale Methode ist die Gradient Boosting Machine (GBM) [8], eine leistungsstarke Technik des überwachten Lernens. GBM-Modelle sind bekannt für ihre Fähigkeit, durch iterative Kombination von Entscheidungsbäumen robuste Vorhersagen zu treffen und Vorhersagefehler zu minimieren.

Ein weiterer Schwerpunkt liegt auf dem Einsatz von AutoML-Techniken, insbesondere der Bibliothek AutoGluon [3]. Diese ermöglicht die automatische Erstellung von Machine-Learning-Modellen und verwendet fortgeschrittene Methoden wie das Multi-Layer Stack Ensembling, um die Genauigkeit der Vorhersagen weiter zu steigern. Dabei werden mehrere Basismodelle kombiniert und durch ein Meta-Modell in verschiedenen Schichten optimiert.

Zusätzlich wird das Konzept des Feature Engineering durch das Framework OpenFE [19] betrachtet. OpenFE ermöglicht die automatische Generierung neuer Features, wodurch die Vorhersagegenauigkeit der Modelle verbessert werden kann. Dieser Prozess, der üblicherweise manuelle Eingriffe und fundiertes Fachwissen erfordert, wird durch OpenFE effizient und systematisch automatisiert.

Ziel dieser Arbeit ist es die Effektivität moderner Machine Learning Methodiken aufzuzeigen und diese verständlich zu erklären. Durch die Anwendung der vorgestellten Methodiken ist es in dem untersuchten Wettbewerb mit gerigem Aufwand möglich unter die besten 20% aller Teilnehmer des Untersuchten Wettbewerbs zu kommen. Anwendungbeispiele und Evaluationsskripts findet man unter https://github.com/MaxUhl98/Proseminar

1.1 Kaggle

In dieser Arbeit werden Methodiken vorgestellt, die sich in einem Kaggle-Wettbewerb[11] als besonders erfolgreich erwiesen haben. Kaggle ist eine Online-Plattform, welche regelmäßig Wettbewerbe im Bereich Data Science veranstaltet. Die Themen umfassen Big Data, maschinelles Lernen und Data Mining. Die Aufgabe besteht darin, für eine bestimmte Problemstellung und einen bereitgestellten Datensatz ein Modell zu entwickeln, das die Problemstellung bestmöglich löst. Die eingereichten Modelle werden anhand eines vorenthaltenen Datensatzes getestet und in einem Leaderboard verglichen. Dieser Vergleich erfolgt anhand einer von den Wettbewerbsveranstaltern festgelegten Metrik.

1.2 Regression with an Abalone Dataset

Der betrachtete Wettbewerb trägt den Titel „Regression with an Abalone Dataset“[11]. Ziel ist es, anhand physischer Merkmale einer Abalone deren Alter oder die Anzahl der Ringe auf der Schale zu bestimmen. Der bereitgestellte Datensatz[12] umfasst insgesamt 90.615 Einträge. Jeder Eintrag enthält verschiedene physische Merkmale der Abalone, die in der nachfolgenden Tabelle dargestellt sind:

Bezeichnung	Rolle	Typ	Beschreibung	Einheit	Fehlende Werte
Sex	Merkmal	Kategorie	M, F oder I	-	nein
Length	Merkmal	Stetig	längste Seite	mm	nein
Diameter	Merkmal	Stetig	senkrecht zur Länge	mm	nein
Height	Merkmal	Stetig	Höhe des Weichkörpers	mm	nein
Whole_weight	Merkmal	Stetig	Gewicht der ganzen Abalone	g	nein
Shucked_weight	Merkmal	Stetig	Gewicht des Weichkörpers	g	nein
Viscera_weight	Merkmal	Stetig	Gewicht der Eingeweide	g	nein
Shell_weight	Merkmal	Stetig	Gewicht der trockenen Schale	g	nein
Rings	Zielvariable	Ganzzahl	Alter in Jahren $\pm 1,5$	-	nein

Der Datensatz einer Abalone umfasst acht Merkmale und eine Zielvariable. Als Merkmale (Features) bezeichnen wir die Eigenschaften, die zur Prognose genutzt werden, während das Ziel (Target) die Variable ist, die vorhergesagt werden soll. Der erste Schritt besteht darin, den Datensatz zu untersuchen. In unserem Fall sind im Datensatz keine fehlenden Werte vorhanden. Für den Umgang mit dem kategorischen Merkmal „Sex“ verwenden wir das One-Hot-Encoding, auf das in Kapitel 2 zu den Grundlagen näher eingegangen wird.

Die im Trainingsprozess und zur Bewertung der Modelle verwendete Verlustfunktion ist der Root Mean Squared Logarithmic Error (RMSLE). Der RMSLE wird durch folgende Formel berechnet[11]:

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\log(1+\hat{y}_i) - \log(1+y_i)\right)^2}$$

n : Anzahl der Vorhersagen
$\hat{y}_i$: Vorhergesagter Wert der Zielvariable
y_i : Wahrer Wert der Zielvariable

Der RMSLE misst die Differenz zwischen den tatsächlichen und den vorhergesagten Werten unter Verwendung des natürlichen Logarithmus. Diese Metrik ist besonders nützlich, wenn die Zielwerte und Vorhersagen über einen großen Wertebereich variieren. Durch die Logarithmierung werden relative Unterschiede anstelle absoluter Unterschiede gemessen, was zu einer Robustheit der Metrik gegenüber Ausreißern führt.

Für die Altersbestimmung haben wir verschiedene Machine-Learning-Modelle eingesetzt. Darüber hinaus wurde eine multiple lineare Regression[2, S. 450] als Benchmark durchgeführt, um die Leistung der komplexeren Machine-Learning-Modelle zu evaluieren.

1.3 Machine Learning

Machine Learning befasst sich mit der Analyse großer Datenmengen mithilfe statistischer Algorithmen. Dabei werden Muster in diesen Daten identifiziert, um komplexe Problemstellungen zu lösen, die sich nicht durch einfache Regeln beschreiben lassen. Ein entscheidender Faktor ist, dass die zur Verfügung stehende Datenmenge ausreichend groß ist.

Ein Machine-Learning-Modell stellt eine mathematische Funktion dar, die eine Vielzahl von lernbaren Parametern umfasst.

$$f(a_1, a_2, ..., a_n | x_i) = y_i$$

$a_{i \in \{1,2,...,n\}}$: Lernbare Parameter des Modells
x_i : Eingabedaten zur Vorhersage
y_i : Vorhersage

Die lernbaren Parameter werden im Trainingsprozess an die Trainingsdaten angepasst. Das Modell berechnet die Vorhersagen $\hat{y}_i$ für die Eingabedaten x_i. Mithilfe der Verlustfunktion wird die Qualität der Vorhersagen bewertet, indem die Diskrepanz zwischen den vorhergesagten Werten $\hat{y}_i$ und den tatsächlichen Werten y_i ermittelt wird. Durch die Anwendung eines Optimierungsverfahrens werden die lernbaren Parameter so angepasst, dass die Verlustfunktion minimiert wird. Dieser iterative Prozess wird so lange wiederholt, bis ein zufriedenstellendes Ergebnis erreicht ist.

2 Grundlagen

2.1 One-Hot-Encoding

Da Machine Learning Modelle ausschließlich mit Zahlen arbeiten können wird ein Weg benötigt um Kategorische Features wie das Geschlecht in eine sinnvolle numerische Repräsentation zu bringen. In dem konkreten Anwendungsfall

des Wettbewerbs sollen die Ausprägungen 'M', 'F' und 'I' in numerische Werte überführt werden. Jedoch würde eine einfache Übersetzung wie:

$$M \mapsto 0$$
$$F \mapsto 1$$
$$I \mapsto 2$$

eine ordinale Beziehung zwischen den Ausprägungen suggerieren. In dem gegebenen Beispiel würden die Werte einen Infant als doppelten Female darstellen, was in dem Kontext zweifelhaft erscheint. Die beschriebene Problematik wird von One-Hot-Encoding[9, S. 68] gelöst. Die Idee besteht darin, jede Kategorie in einen binären Vektor umzuwandeln, wobei für jede Kategorie genau eine Position im Vektor den Wert 1 hat, und alle anderen Positionen den Wert 0.

$$M \mapsto [0, 1]$$
$$F \mapsto [1, 0]$$
$$I \mapsto [0, 0]$$

2.2 Overfitting

Ein Problem, welches beim Trainern auftreten kann, ist das Overfitting. Dabei haben sich die lernbaren Parameter so sehr an die Trainingsdaten angepasst, dass das Modell nicht in der Lage ist, auf zuvor unbekannte Daten zu generalisieren. Das Modell hat nur zu einem geringen Maß Muster in den Daten erkannt und hat die Trainingsdaten eher auswendig gelernt. Ein Beispiel hierzu sieht man in 2.2. Um dem entgegenzuwirken, wird nach jeder Trainingsiteration ein Evaluierungsschritt durchgeführt. Dazu wird ein Teil der Trainingsdaten dem Modell vorenthalten und zur Kontrolle verwendet, wie das Modell auf unbekannten Daten abschneidet.

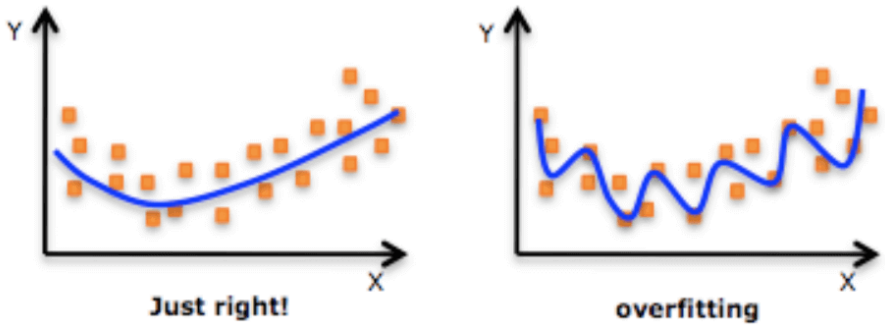


Abbildung 1: Overfitting [7]

Eine Problematik, die dadurch entsteht, ist allerdings, dass wir während des Trainingsprozesses auf wertvolle Daten verzichten müssen. Außerdem ist es

möglich, dass das Modell bei bestimmten Aufteilungen verschieden performt. Um diese Nachteile zu vermeiden verwendet man die sogenannte K-Fold Kreuzvalidierung[9, S. 76], welche den Datensatz in K gleich große Teildatensätze aufspaltet. Dann wird das Modell auf allen bis auf dem k-ten Teildatensätzen trainiert, während der k-te Datensatz, wie in 2.2 gezeigt benutzt wird, um die Vorgersagegüte zu berechnen. Dieses Vorgehen wird so lange wiederholt, bis jeder Teildatensatz zur Berechnung der Vorhersagegüte verwendet wurde. Dann berechnet man den Mittelwert aller Vorhersagegüten als Annäherung für die Modellperformanz. Da hierbei alle Datenpunkte für Training und Validierung verwendet wurden löst diese Methodik alle vorherig aufgezeigten Problematiken.

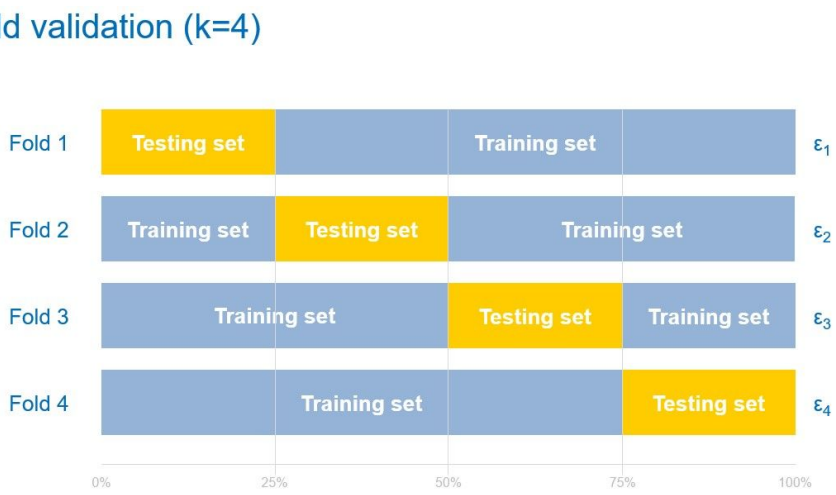


Abbildung 2: Kreuzvalidierung mit 4 Folds[19]

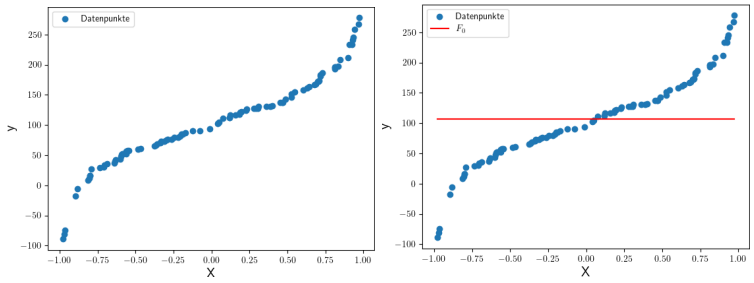
3 Gradient Boosting Machines

3.1 Einführung

Obwohl das ursprüngliche Paper über Gradient Boosting Machines vor über 25 Jahren veröffentlicht wurde [8] waren Gradient Boosting Machines innerhalb der letzten 6 Monaten maßgeblich an Geldgewinnen von über 265000$ auf der Data Science Plattform Kaggle beteiligt. Zusätzlich dazu sind Gradient Boosting Machines teilweise interpretierbar und laufzeitarm genug, um mittels Erklärbarkeitsalgorithmen wie SHAP [15] analysiert zu werden. Folgend wird die mathematische Formulierung der Gradient Boosting Machine anhand eines Beispiels erläutert.

3.2 Funktionsweise

Als Beispieldatensatz wird ein mittels scikit-learn [17] zufällig generierter Regressionsdatensatz mittels $make\_regression(random\_state = 1, noise = 10, n\_features = 1)$ generiert und mittels des Tangens Hyperbolicus zu dem Datensatz aus Plot (a) transformiert. Für das Beispiel wurde $L(y_i, y) = |y_i - y|$ als Verlustfunktion gewählt.



(a) (b)

Nach der Generierung des Datensatzes (a) wird das Modell F_0 durch eine Konstante initialisiert, welche die Verlustfunktion L auf dem gegebenen Datensatz minimiert (b).

$$F_0(x) = \arg\min_y \sum_{i=1}^{n} L(y_i, y) \tag{1}$$

Anschließend werden die sogenannten Residuen r_{i1} des derzeitigen Modells berechnet.(c).

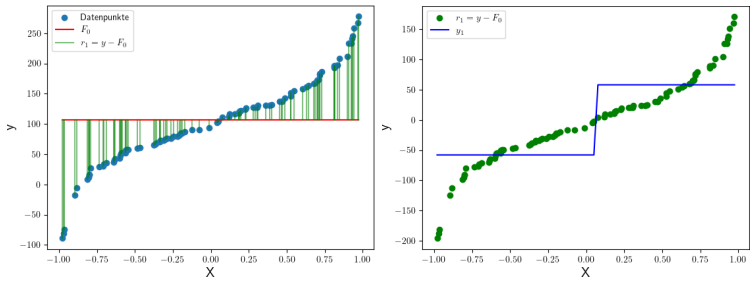
$$r_{i1} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F=F_0} \tag{2}$$

Mittels der berechneten Residuen wird ein Entscheidungsbaum trainiert, welcher diese minimiert (d).

$$h_1(x) = \arg\min_h \sum_{i=1}^{N} (r_{i1} - h(x_i))^2 \tag{3}$$

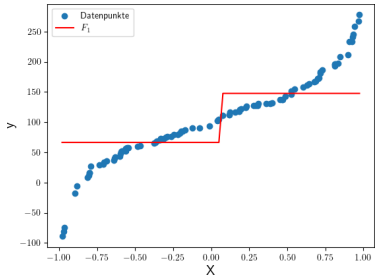
Abschließend wird der neu trainierte Entscheidungsbaum mit der sogenannten Learning Rate η multipliziert und dem aktualisierten Modell additiv hinzugefügt (e).

$$F_1(x) = F_0(x) + \eta \cdot h_1(x) \tag{4}$$

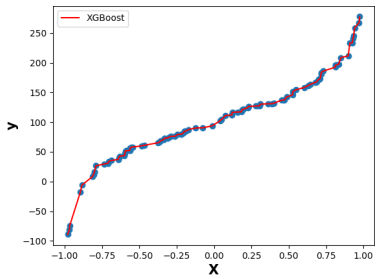


(c)

(d)



(e)



(f)

In den meisten Fällen wird ein $\eta \in (0, 1)$ gewählt, um das Risiko des overfittings zu reduzieren. Klassischerweise werden die Schritte 2 - 4 iterativ wiederholt, bis eine maximale Iterationszahl oder ein vorher festgelegtes Abbruchkriterium erreicht wird. Der vollständige Algorithmus wird folgend formuliert:

Algorithm 1 Gradient Boosting

Input: Features: x, Zielvariable: y, Verlustfunktion: L
Output: Neue Merkmalsmenge

$F_0(x) = \arg\min_y \sum_{i=1}^{n} L(y_i, y)$

for $m = 1$ to M **do**

$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F=F_{m-1}}$

$h_m(x) = \arg\min_h \sum_{i=1}^{N} (r_{im} - h(x_i))^2$

$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x)$

end for

return F_M

Aufgrund des Erfolges und des Alters der Idee des Gradient Boostings gibt es Bibliotheken wie XGBoost[5] oder LightGBM[13], in denen der Algorithmus bereits implementiert und optimiert wurde. Wenn man XGBoost auf den Beispieldatensatz anwendet, erhält man das Ergebnis in (f).

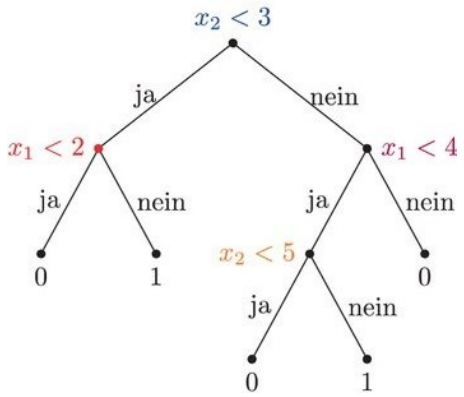


Abbildung 3: Entscheidungsbaum mit Features x_1 und x_2 [14]

4 Interpretierbarkeit

Wie in der Einführung erwähnt, sind Gradient Boosting Machines nicht nur sehr performant, sondern auch interpretierbar. Man kann zum Beispiel die Anzahl der Vorkommnisse aller Features in den Entscheidungsbäumen zählen, um so eine Abschätzung für deren Einfluss auf die Modellvorhersage zu erhalten, ein Beispiel anhand des Abalone Datensatzes findet man in Figure 3. Alternativ

kann man den durchschnittlichen Informationsgewinn oder die durchschnittliche Anzahl der von den Featureknoten betroffenen Datenpunkten berechnen [5]. Zusätzlich dazu existieren Erklärbarkeitsverfahren wie SHAP [15], mit deren Hilfe man den Einfluss der Features auf eine Vorhersage berechnen kann.

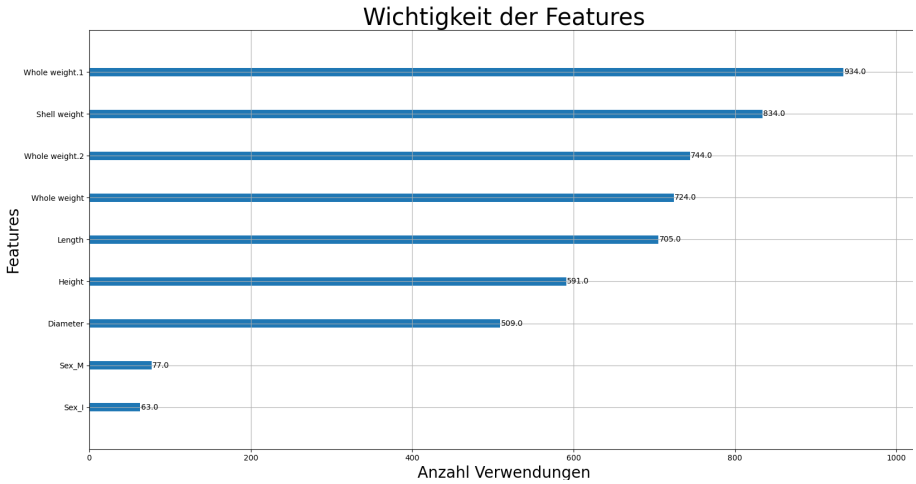


Abbildung 4: Feature Wichtigkeit nach Verwendungsanzahl in XGBoost[5]

5 OpenFE

5.1 Einleitung

Trotz des Erfolges von State-of-the-Art Algorithmen wie Gradient Boosting sind Machine-Learning Methodiken immer noch stark abhängig von den Features mit denen sie trainiert werden. Meistens kann man die Vorhersagegenauigkeit eines Modells verbessern, indem man durch geschicktes Bearbeiten von existierenden Features dem Datensatz neue Features hinzufügt. Diese weit verbreitete Praktik bezeichnet man als Feature Engineering. Da Feature Engineering oftmals intensives Testen und Domänenkentnisse benötigt, ist dieses eine schwierige und zeitintensive Aufgabe. Allerdings gibt es mit dem Framework OpenFE [19] eine moderne und automatisierte Methodik, um diesen Prozess zu automatisieren. Die Verfasser des OpenFE Papers [19] konnten anhand mehrerer Kaggle Wettbewerbe demonstrieren, dass von OpenFE [19] generierte Features auf Expertenniveau sind. In dem folgenden Abschnitt wird genauer auf die Funktionsweise von OpenFE [19] eingegangen.

5.2 Funktionsweise

Die Idee von OpenFE ist es, eine Vielzahl an Operationen, die Features kombinieren oder einzeln nicht linear verändern, auf den Datensatz anzuwenden.

Durch geschickte Featureselektion können laufzeitarm nützliche Features ermittelt werden, welche dem Datensatz hinzugefügt werden. Der Prozess wird anschließend mit dem neu angereicherten Datensatz wiederholt, bis eine vorgegebene Anzahl an Durchläufen erreicht wurde [19]. Eine grafische Darstellung des Algorithmus findet man in: Figure 2

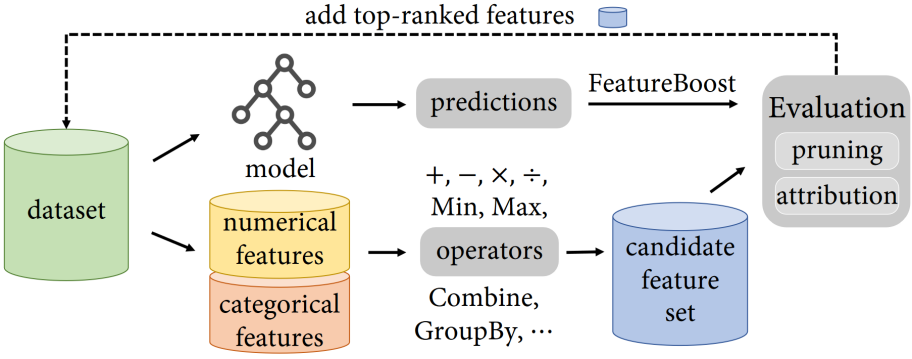


Abbildung 5: OpenFE Algorithmus [19]

6 AutoGluon

6.1 Einleitung

AutoGluon ist eine Bibliothek zur Umsetzung des Automated machine learning (AutoML). Dabei handelt es sich um die Automatisierung des Machine Learning-Prozesses. Machine-Learning-Modelle können auf dieser Weise schneller und einfacherer kreiert werden und sind nicht länger den Experten vorenthalten. An Qualität büßen die automatischen Modelle auch nicht ein. Häufig können diese sogar die händisch von Experten erstellten Modelle übertreffen[3].

Der Anspruch von AutoGluon ist es, komplexe Modelle mit nur wenigen Zeilen an Code zu erstellen. Für die Analyse tabellarischer Daten werden im Modul die Klassen `TabularDataset` und `TabularPredictor` zur Verfügung gestellt. Die Anwendung dieser Klassen wird in [1] genauer erläutert.

6.2 Multi-Layer Stack Ensembling

AutoGluon[3] verfolgt eine Multi-Layer Stack Ensembling Strategie. Modell-Ensembles sind eine leistungsstarke Technik im maschinellen Lernen, die darauf abzielt, die Vorhersagegenauigkeit zu verbessern, indem mehrere Modelle kombiniert werden. Anstatt sich auf die Vorhersage eines einzigen Modells zu verlassen, nutzt ein Ensemble die Stärken mehrerer Modelle, um stabilere und

genauere Vorhersagen zu treffen. Diese Methode kann die Schwächen einzelner Modelle kompensieren und zu besseren Ergebnissen führen. Dietterich hat gezeigt, dass eine Kombination von Modellen häufig deutlich besser abschneidet als die einzelnen Modelle[6].
Jedes maschinelle Lernmodell hat seine eigenen Stärken und Schwächen, abhängig von der Art des verwendeten Algorithmus und der Struktur der Daten. Ein einzelnes Modell kann anfällig für Überanpassung (Overfitting) sein, besonders wenn es stark an den Trainingsdaten ausgerichtet ist, oder es kann die zugrunde liegenden Muster in den Daten nicht ausreichend erfassen (Underfitting). Modell-Ensembles bieten eine Möglichkeit, diese Probleme zu minimieren, indem sie die Diversität mehrerer Modelle nutzen, um eine robuste Gesamtvorhersage zu erstellen.

Der Aufbau des von AutoGluon[3] verwendeten Multi-Layer Stack Ensembling ist in der nachfolgenden Abbildung dargestellt:

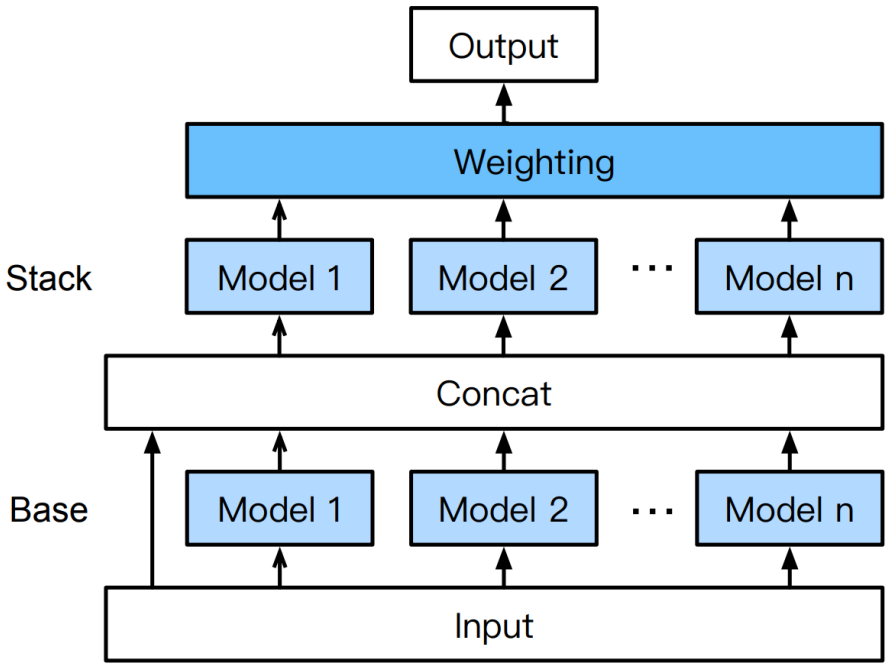


Abbildung 6: AutoGluon: Multi-Layer Stack Ensembling Strategie, hier mit n Basismodellen und 2 Schichten[3]

In diesem mehrstufigen Modell sind mehrere Modelle in Schichten angeordnet. Die Ausgangsvorhersagen der ersten Schicht (Base) von Modellen dienen als Eingangsdaten für eine nachfolgende Schicht (Stack). Das Ziel dieser Technik ist es, die Vorhersagekraft durch das Lernen auf mehreren Ebenen zu maximieren. Der Ablauf des Multi-Layer Stack Ensembling läuft in mehreren Stufen ab.

1. **Training der Basismodelle**: Zunächst werden unterschiedliche Modelle auf den Trainingsdaten individuell trainiert. Diese Modelle können unterschiedliche Algorithmen verwenden (z. B. Random Forests[4], Gradient Boosting[8], neuronale Netze). Jedes dieser Modelle erstellt eine Vorhersage für die Zielvariable.

2. **Stack-Modell-Ebene**: In der nächsten Stufe werden die Vorhersagen der Basismodelle als neue Eingabedaten für ein weiteres Modell, dem sogenannten Meta-Modell, verwendet. Dieses Modell lernt, wie es die Vorhersagen der Basismodelle optimal kombiniert, um die endgültige Vorhersage zu treffen. Der Vorteil dieser Methode liegt darin, dass das Meta-Modell die Stärken der Basismodelle nutzen kann, um deren Schwächen auszugleichen.

3. **Cross-Validation**: Um Overfitting zu vermeiden, verwendet AutoGluon während des Trainings die in Kapitel 2 vorgestellte Cross-Validation. Dabei werden die Trainingsdaten in mehrere Teildatensätze aufgeteilt, und die Modelle werden auf verschiedenen Datenaufteilungen trainiert. Dies gewährleistet, dass die Vorhersagen der Basismodelle so generalisierbar wie möglich sind, bevor sie in das Meta-Modell einfließen.

4. **Automatisches Modell-Tuning**: Zusätzlich zum Stacked Ensembling bietet AutoGluon eine automatisierte Hyperparameter-Optimierung an, die die Leistung jedes einzelnen Modells im Ensemble weiter verbessert. Dies geschieht ohne manuelles Eingreifen, was AutoGluon besonders nützlich für Benutzer macht, die wenig Fachwissen im Bereich des maschinellen Lernens haben.

In der Baseschicht befinden sich die einzelnen Basismodelle. Diese werden zunächst auf üblicher Art individuell trainiert. Die Prognosen dieser Modelle gehen anschließend als Input in die darüberliegende Stackschicht ein[3].

7 Ergebnisse

Nachdem die von uns verwendeten Methodiken vorgestellt wurden, ist es an der Zeit, anhand des Abalone Regression Wettbewerbs[11] die Effektivität der einzelnen Methodiken oder deren Kombinationen zu prüfen. Die Ausführung der vorgestellten Methodiken auf dem Abalone Datensatz liefert die Ergebnisse aus Table 1.

Modell	Score Leaderboard	MAE Score	Platzierung
Lineare Regression (LR)	0.16372	1.37410	2243 / 2606 (Bottom 14%)
LR + OpenFE	0.15278	1.31188	1908 / 2606 (Bottom 27%)
XGBoost	0.14740	1.24489	876 / 2606 (Top 34%)
XGBoost + OpenFE	0.14625	1.23801	525 / 2606 (Top 20.5%)
Autogluon	0.14616	1.18539	498 / 2606 (Top 19.2%)
Autogluon + OpenFE	0.14601	1.17730	441 / 2606 (Top 17%)
Erstplatziertes Modell	0.14374	Unbekannt	1 / 2606 (Top 0,04%)

Tabelle 1: Wettbewerbsergebnisse

Während die lineare Regression wenig kompetitive Ergebnisse liefert, untermauert die Platzierung des einfachen XGBoost[5] (Gradient Boosting) Modells, welche knapp nicht für das obere $\frac{1}{3}$ reicht die Behauptung, dass Gradient Boosting ein leistungsstarker Algorithmus ist. Autogluon erweist sich als sehr kompetetiv, mit einer Platierung innerhalb des besten $\frac{1}{5}$, was die Effektivität des Gradient Boosting Algorithmus weiter untermauert, da das Autogluon Ensemble größtenteils aus Gradient Boosting Modellen besteht. Außerdem fällt es schwer, die Effektivität von OpenFE anzuzweifeln, da OpenFE in allen von uns durchgeführten Experimenten die Ergebnisqualität der getesteten Modelle deutlich verbessert.

8 Fazit

In dieser Arbeit haben wir verschiedene Techniken des maschinellen Lernens und deren Anwendung auf den Kaggle-Wettbewerb „Regression with an Abalone Dataset“ vorgestellt. Dabei wurde der gesamte Prozess, von der Datenvorbereitung über die Wahl geeigneter Modelle bis hin zur Modellbewertung anhand der RMSLE-Metrik, erläutert.

Ein besonderes Augenmerk lag auf den Gradient Boosting Machines, die aufgrund ihrer Effizienz und interpretierbaren Struktur in vielen Anwendungen herausragende Ergebnisse erzielen.

Des Weiteren wurde die Bedeutung von Feature Engineering hervorgehoben. Durch den Einsatz von OpenFE, einem Framework, welches die manuellen und zeitaufwändigen Aspekte des Feature Engineerings automatisiert, kann die Leistungsfähigkeit von Modellen deutlich gesteigert werden.

Abschließend wurde das Konzept Automated machine learning anhand des Frameworks AutoGluon dargestellt. Mit AutoGluon-Tabular kann der ganze machine learning Prozess für tabellarische Daten automatisiert werden. AutoGluon vereinigt eine robuste Datenverarbeitung, eine moderne Architektur von neuronalen Netzen sowie ein leistungsstarkes Modell Ensembling. Das integrierte

Multi-Layer Stack Ensembling zeigt, dass die Kombination mehrerer Modelle zu einer signifikanten Verbesserung der Vorhersagequalität führen kann.

Die durch die Arbeit gewonnenen Erkenntnisse zeigen, dass maschinelles Lernen, wenn es korrekt angewendet wird, leistungsstarke Lösungen für komplexe Probleme, wie unter anderem die Altersbestimmung von Abalonen bieten kann.

Literaturverzeichnis

[1] Erickson et al. *Autogluon Cheat Sheet*. https://raw.githubusercontent.com/Innixma/autogluon-doc-utils/main/docs/cheatsheets/stable/autogluon-cheat-sheet.jpeg. Zugriff: 25.09.2024. 2020.

[2] Ludwig Fahrmeir et al. *Statistik: Der Weg zur Datenanalyse*. Springer-Lehrbuch, 2022.

[3] Nick Erickson et al. "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data". In: *arXiv preprint arXiv:2003.06505* (2020).

[4] Leo Breiman. "Random forests". In: *Machine Learning* 45.1 (2001), S. 5–32.

[5] Tianqi Chen und Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, S. 785–794. arXiv: 1603.02754 [cs.LG].

[6] Thomas G. Dietterich. "Ensemble Methods in Machine Learning". In: *International Workshop on Multiple Classifier Systems*. Springer, 2000, S. 1–15.

[7] Steve Finger. *Regularisierung (Shrinkage Methods)*. http://www.mi.uni-koeln.de/wp-znikolic/wp-content/uploads/2018/11/20181116_Regularisierung_Finger.pdf. Zugriff: 25.09.2024. 2018.

[8] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *IMS Reitz Lecture*. 1999.

[9] Aurélien Géron. *Praxiseinstieg Machine Learning*. 2. Auflage. O'Reilly Media, 2020.

[10] Johannes Heller. *1st Place Solution for the Regression with an Abalone Dataset Competition*. https://www.kaggle.com/competitions/playground-series-s4e4/discussion/499174. Zugriff: 25.09.2024. 2024.

[11] kaggle. *Regression with an Abalone Dataset*. https://www.kaggle.com/competitions/playground-series-s4e4/overview. Zugriff: 25.09.2024. 2024.

[12] kaggle. *Regression with an Abalone Dataset*. https://www.kaggle.com/competitions/playground-series-s4e4/data. Zugriff: 25.09.2024. 2024.

[13] Guolin Ke u. a. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Neural Information Processing Systems*. 2017. URL: https://api.semanticscholar.org/CorpusID:3815895.

[14] Donald E. Knuth. "Lernende Entscheidungsbäume: Überholtes Verfahren oder vielseitige KI-Methode?" In: *Informatik Spektrum* 44 (2021), S. 223–229.

[15] Stan Lipovetsky und Michael Conklin. "Analysis of Regression in Game Theory Approach". In: *Applied Stochastic Models in Business and Industry* 17.4 (2001), S. 319–330.

[16] LuminousC. *3rd Place Solution for the Regression with an Abalone Dataset Competition*. `https://www.kaggle.com/competitions/playground-series-s4e4/discussion/499747`. Zugriff: 25.09.2024. 2024.

[17] Fabian Pedregosa u. a. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.

[18] Lennart Purucker. *2nd Place Solution for the Regression with an Abalone Dataset Competition*. `https://www.kaggle.com/competitions/playground-series-s4e4/discussion/499698`. Zugriff: 25.09.2024. 2024.

[19] Hao Zhang u. a. "OpenFE: Automated Feature Generation with Expert-level Performance". In: *arXiv preprint* (2022). arXiv: `2202.08009 [cs.AI]`.