

Technische Hochschule Ingolstadt
Studiengang: Data Science in Technik und Wirtschaft

Palmer Penguins Klassifikation

Von EDA zur Modellauswahl

Seminararbeit Machine Learning Lab

Autor: Mirza Muhammad Omer Baig
Matrikelnummer: 00
Abgabedatum: 21.12.2025

Abstract

Diese Arbeit verfolgt kein konkretes Anwendungsziel, sondern dient als **didaktischer Referenzfall** für einen sauberen, übertragbaren Workflow in Klassifikationsprojekten. Die eigentliche Vorhersageaufgabe steht nicht im Vordergrund, sondern die Frage, wie man eine Modellierung so strukturiert, dass Entscheidungen transparent begründet werden können und die Ergebnisse fair bewertet bleiben. Als Beispiel wird der Palmer Penguins Datensatz verwendet, weil er reale Feldmessungen, eine überschaubare Featuremenge und gleichzeitig überlappende Klassen enthält und damit typische Entscheidungssituationen (Featureauswahl, Modellvergleich, Fehleranalyse) realistisch abbildet. Das Ergebnis ist ein kompakter **Blueprint** von der Datenanalyse bis zur finalen Evaluation für vergleichbare Fragestellungen, der sich auf andere Anwendungsfälle übertragen und dort je nach Kontext erweitern lässt. Aus Platzgründen können nicht alle Abbildungen, Zwischenschritte und Implementierungsdetails in dieser Arbeit dargestellt werden; sie sind im begleitenden Notebook [3], im Repository [4] und auf der interaktiven Webseite [5] vollständig dokumentiert und frei zugänglich.

Inhaltsverzeichnis

1 Einleitung	2
1.1 Daten	3
1.2 Überblick	3
1.3 Featurebeschreibung	3
1.3.1 Zielvariable	3
1.3.2 Numerische Merkmale	3
1.3.3 Kategoriale Merkmale	3
1.4 Train-Test Split	3
2 Explorative Datenanalyse	4
2.1 Klassenverteilung	4
2.2 Fehlende Werte	4
2.3 Visualisierungen	5
2.3.1 Numerische Features	5
2.3.2 Kategoriale Features	5
2.3.3 Numerisch x Kategorisch (multivariat)	6
2.4 Korrelationsmatrix	6
2.5 Feature Engineering	7
2.6 Outliers	7
3 Modellierung und Modellauswahl	7
3.1 Evaluation Setup	7
3.2 Gaussian Naive Bayes (GNB)	7
3.3 k-Nearest Neighbors (kNN)	9
3.4 Entscheidungsbaum	11
3.5 Manueller Entscheidungsbaum	11
3.6 Modellvergleich und Auswahl	13
3.7 Bootstrapping	14
4 Auswertung	14
4.1 Evaluation auf dem Testset	14
4.2 Interpretation von Fehlern	15
5 Fazit	16
6 Reflexion	16

Langschwanzpinguine (Pygoscelis)



(a) Adelie

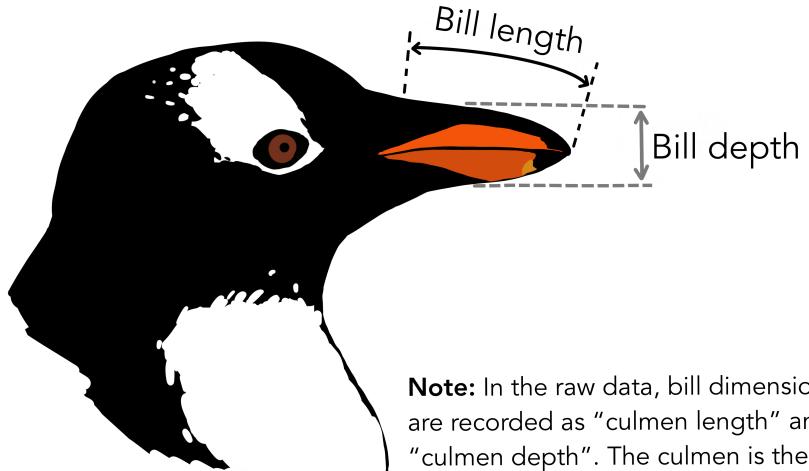


(b) Chinstrap



(c) Gentoo

Abbildung 1: Beispielbilder der drei Klassen (generiert mit ChatGPT)



Note: In the raw data, bill dimensions are recorded as "culmen length" and "culmen depth". The culmen is the dorsal ridge atop the bill.

Abbildung 2: Illustration der Schnabelmessung [9]

1 Einleitung

Ziel dieser Arbeit ist es, die Pinguinart vorherzusagen. Dabei liegt der Schwerpunkt ganz klar auf einer transparenten und gut nachvollziehbaren Modellierung. **Interpretierbarkeit** hat Vorrang vor reiner Vorhersageleistung. Machine Learning wird nur dann eingesetzt, wenn klassische, leicht nachvollziehbare statistische Verfahren an ihre Grenzen stoßen.

1.1 Daten

Der Datensatz `palmerpenguins` [10] basiert auf realen Feldmessungen, die ursprünglich von Gorman [8] im Rahmen des Palmer Station Long Term Ecological Research (LTER) Programms in der Antarktis [13] erhoben und anschließend öffentlich zur Verfügung gestellt wurden.

1.2 Überblick

species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
Adelie	Torgersen	39.1	18.7	181	3750	Male
Adelie	Torgersen	39.5	17.4	186	3800	Female
Adelie	Torgersen	40.3	18.0	195	3250	Female
Adelie	Torgersen	nan	nan	nan	nan	nan
Adelie	Torgersen	36.7	19.3	193	3450	Female

Tabelle 1: Ausschnitt aus `df.head()`

1.3 Featurebeschreibung

Der Palmer Penguins Datensatz enthält morphologische Messungen von 344 Pinguinen aus der Antarktis [10].

1.3.1 Zielvariable

species Kategoriale Zielvariable mit drei Klassen: `Adelie`, `Chinstrap`, `Gentoo`

1.3.2 Numerische Merkmale

bill_length_mm Länge des Schnabels in Millimetern

bill_depth_mm Tiefe (Höhe) des Schnabels in Millimetern

flipper_length_mm Länge der Flosse in Millimetern

body_mass_g Körpergewicht in Gramm

1.3.3 Kategoriale Merkmale

sex Geschlecht des Pinguins (Werte: `male`, `female`)

island Insel (Werte: `Biscoe`, `Dream`, `Torgersen`)

1.4 Train-Test Split

Ganz am Anfang trennen wir das Testset (20%) stratifiziert vollständig ab, um jegliche Form von Data Leakage zu vermeiden. Im gesamten Notebook arbeiten wir ausschließlich mit den Trainingsdaten. Das Testset wird erst am Ende verwendet, um die finale Modellperformance zu evaluieren.

2 Explorative Datenanalyse

2.1 Klassenverteilung

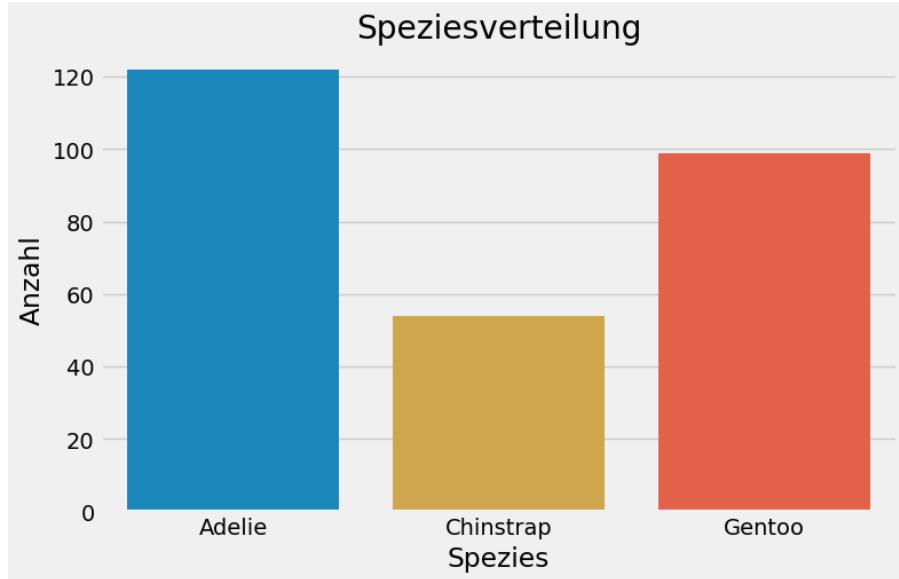


Abbildung 3: Klassenverteilung

Die Klassen sind leicht ungleich verteilt. Oversampling Verfahren (z.B. SMOTE [7] von `imbalanced-learn` [11]) wären möglich, werden hier aber erst dann eingesetzt, wenn das Ungleichgewicht tatsächlich ein relevantes Problem in der Modellperformance erzeugt.

2.2 Fehlende Werte

S. No.	Column Name	Min	Max	n Unique	Nan count	Nan percentage	dtype
1	island	nan	nan	3	0	0.0%	object
2	bill_length_mm	33.1	59.6	147	2	0.727%	float64
3	bill_depth_mm	13.1	21.5	78	2	0.727%	float64
4	flipper_length_mm	172.0	231.0	54	2	0.727%	float64
5	body_mass_g	2700.0	6300.0	91	2	0.727%	float64
6	sex	nan	nan	2	11	4.0%	object
7	species	nan	nan	3	0	0.0%	object

Abbildung 4: Fehlende Werte (Übersicht)

Abbildung 4 zeigt, dass für alle vier numerischen Messungen jeweils zwei Werte fehlen. Zusätzlich fehlen elf Werte für das Feature `sex`. Man sieht in der MSNO Matrix (5), dass bei denselben zwei Datenpunkten nicht nur alle vier numerischen Messungen fehlen, sondern auch das Geschlecht. Daher würde ich diese Datenpunkte einfach droppen. Für Produktions-/Testfälle wäre alternativ eine transparente Fallback-Logik möglich (z.B. “keine Vorhersage” oder Majoritätsklasse), ohne dem Modell unrealistische Muster beizubringen. Für eine detaillierte Diskussion, siehe [3].

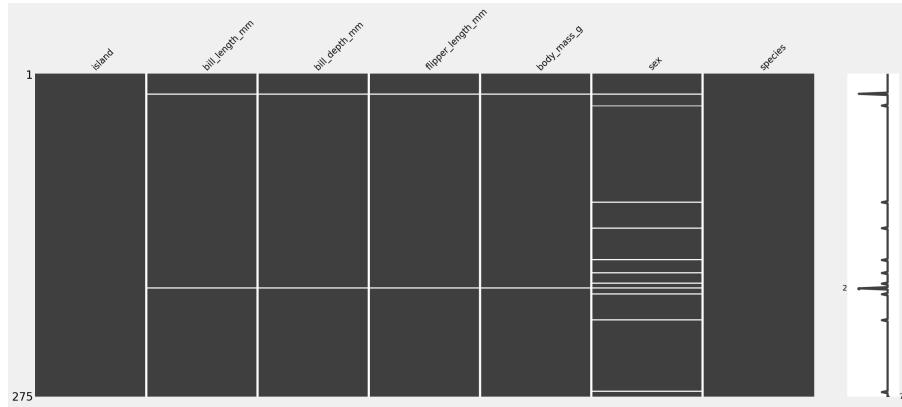


Abbildung 5: MSNO Matrix (missingno)

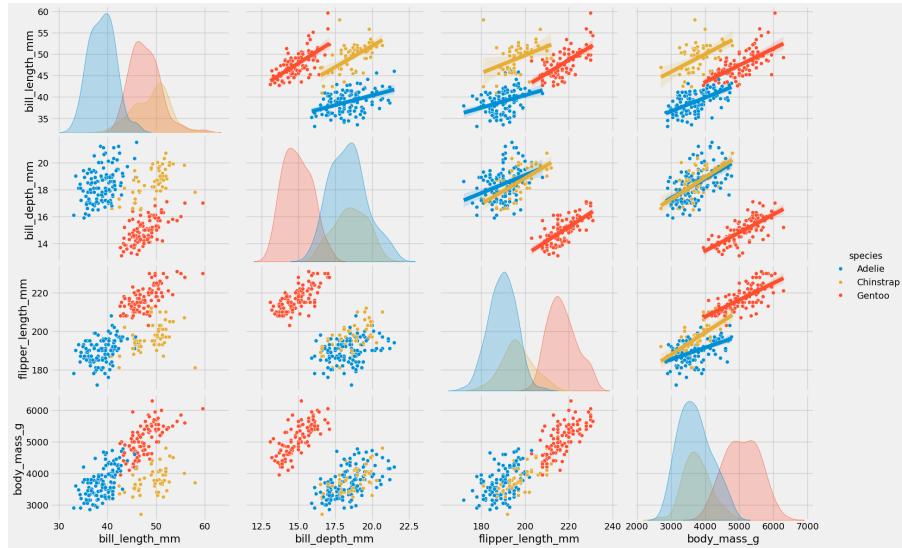


Abbildung 6: Pair-Plot numerischer Features

2.3 Visualisierungen

2.3.1 Numerische Features

Abbildung 6 zeigt, dass `bill_length_mm` bereits eine relativ gute Trennung zwischen allen drei Arten ermöglicht. Bei Kombinationen ohne `bill_length_mm` lässt sich Gentoo weiterhin gut unterscheiden, während Adelie und Chinstrap stärker überlappen. Auf der Hauptdiagonalen wirken die Feature-Verteilungen innerhalb der Klassen näherungsweise normalverteilt.

2.3.2 Kategorische Features

In Abbildung 7 fällt auf, dass für `island` ein klarer Zusammenhang erkennbar ist: Wenn `island = Torgersen`, ist die Art in der Stichprobe immer `Adelie`. Damit wirkt `island` grundsätzlich als potenziell hilfreiches Feature, wobei offen bleibt, wie stark dieser Effekt in Kombination mit den morphologischen Messungen tatsächlich trägt. Dieses Feature wirkt auf den ersten Blick trivial, erfordert jedoch eine sorgfältige Einordnung; eine ausführliche Diskussion findet sich in [3].

Für `sex` ergibt sich dagegen kein nennenswerter Zusammenhang mit `species`: Innerhalb jeder Art sind männliche und weibliche Tiere ungefähr gleich häufig vertreten. **Hinweis:** Das schließt

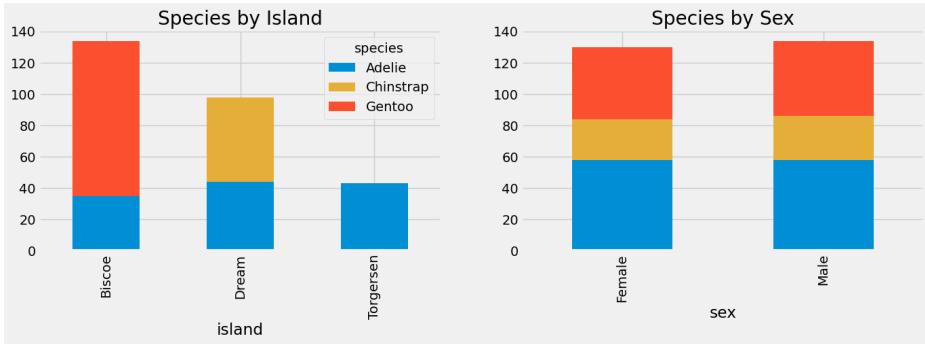


Abbildung 7: Kategorische Features

nicht aus, dass `sex` in Kombination mit anderen Merkmalen indirekt nützlich sein könnte. Daher bleibt das Feature an dieser Stelle zunächst im Datensatz.

2.3.3 Numerisch x Kategorisch (multivariat)

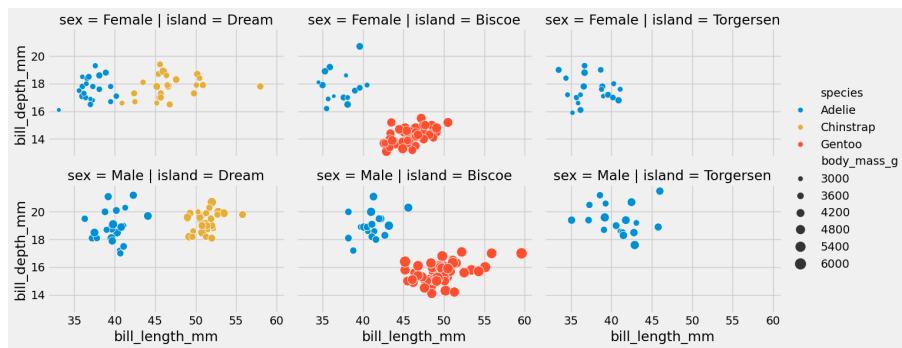


Abbildung 8: Multivariate Darstellung (6 Dimensionen)

In dieser Visualisierung lassen sich insgesamt sechs Dimensionen gleichzeitig darstellen:

- `sex` in den Zeilen
- `island` in den Spalten
- `bill_length_mm` auf der x-Achse
- `bill_depth_mm` auf der y-Achse
- `body_mass_g` über die Punktgröße
- `species` über die Farbe

Bereits auf den ersten Blick wird deutlich, dass `sex` praktisch keine zusätzliche Aussagekraft besitzt, selbst in Kombination mit mehreren anderen Variablen. Auch ein klarer Einfluss von `body_mass_g` ist kaum erkennbar. Wobei der Einfluss von `island` hier schwieriger zu beurteilen ist; der 3D-Plot [2] zeigt jedoch deutlich, dass `island` auch in Anwesenheit von `bill_length_mm` und `bill_depth_mm` kaum noch zur Trennung beiträgt. Als potenzielle Kernfeatures bleiben daher `bill_length_mm`, `bill_depth_mm` und `flipper_length_mm`.

2.4 Korrelationsmatrix

Die Pearson-Korrelation in Abbildung 9 zeigt, dass die lineare Korrelation zwischen `flipper_length_mm` und `bill_length_mm` moderat bis stark ausgeprägt ist, was auf teilweise redundante Information hindeutet. Für eine Auswahl von nur zwei Merkmalen spricht dies dafür, `bill_length_mm` mit

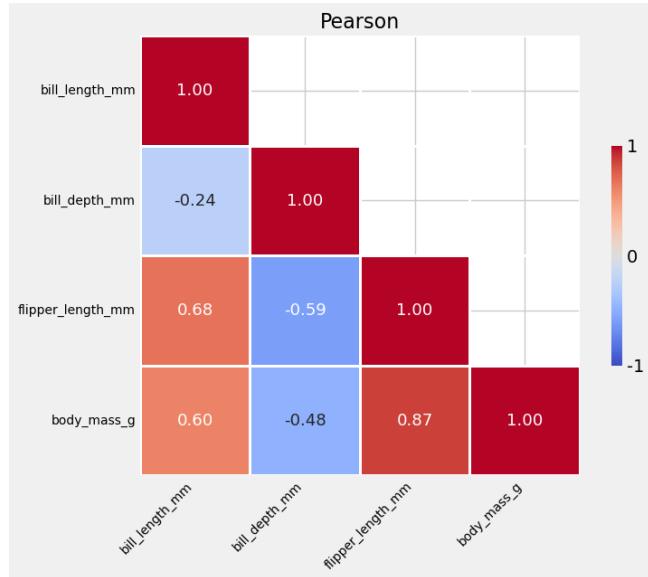


Abbildung 9: Pearson Korrelation (numerische Features)

`bill_depth_mm` zu kombinieren, anstatt mit `flipper_length_mm`.

2.5 Feature Engineering

Es wurden zwei neue Features getestet: `bill_prop` (Verhältnis Schnabellänge zu Schnabeltiefe) und `length_ratio` (Verhältnis Schnabellänge zu Flipperlänge). Eine klare Verbesserung der Trennschärfe ist dabei nicht zu erkennen. Für Details siehe [3].

2.6 Outliers

Eine Analyse nach der 1,5 IQR Regel zeigt keine auffälligen Ausreißer in den betrachteten Features [3].

3 Modellierung und Modellauswahl

3.1 Evaluation Setup

Alle Modellentscheidungen basieren ausschließlich auf Trainingsdaten. Zur Einordnung werden genutzt:

- LOOCV als sehr genauer Schätzer für `accuracy` bei kleinem Datensatz
- Stratifizierte 10 Fold CV für stabile klassenweise Metriken (Macro F1, Precision/Recall) und Konfusionsmatrix

Die Wahl der Evaluationsmetriken, insbesondere die Verwendung von `accuracy` im Rahmen der LOOCV, wird im begleitenden Notebook begründet [3].

3.2 Gaussian Naive Bayes (GNB)

GNB ist in diesem Kontext attraktiv, weil die verwendeten Features stetig sind und die Feature Verteilungen innerhalb der Klassen näherungsweise normalverteilt wirken [1]. Gleichzeitig trifft

das Modell eine zentrale Annahme: bedingte Unabhängigkeit der Features gegeben der Klasse.

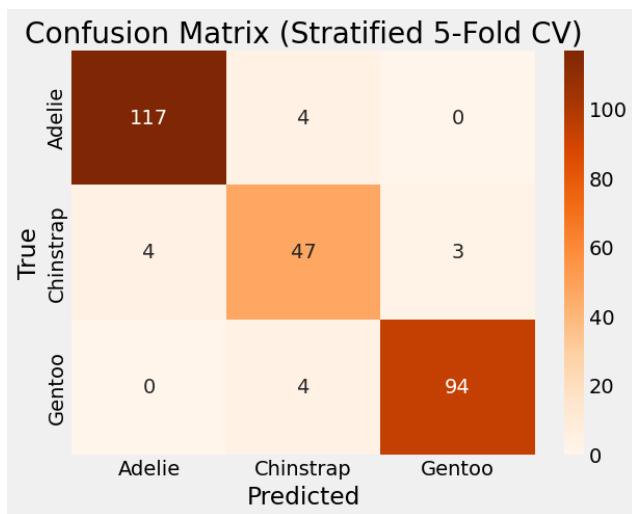
- Vorteile: sehr transparent, schnell, probabilistisches Modell, gute Baseline
- Nachteile: Unabhängigkeitsannahme, sensitiv gegenüber Verteilungsannahmen

Wir starten mit `bill_length_mm` und `bill_depth_mm`. Als Prior wird $1/3$ pro Klasse gesetzt. Das heißt, dass alle Klassen unabhängig von ihrer Häufigkeit im Datensatz gleich wahrscheinlich sind. Die zugrunde liegende Annahme ist dabei, dass eine größere Anzahl an Messungen für eine bestimmte Art im Datensatz nicht zwangsläufig bedeutet, dass diese Art auch in der Realität häufiger vorkommt.

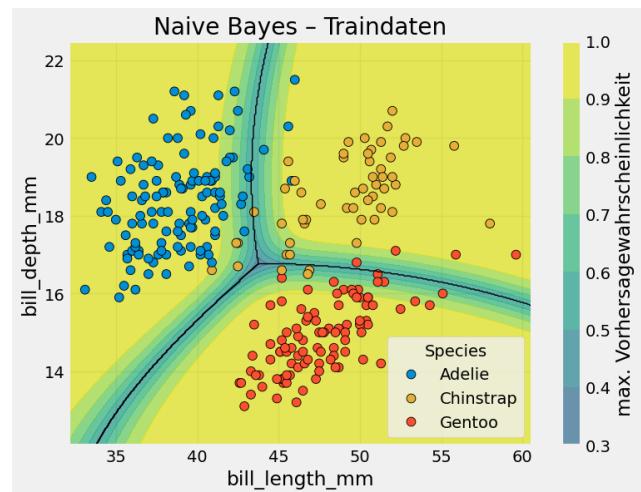
LOOCV

- `train_accuracy`: 0.9450 ± 0.0007
- `validation_accuracy`: 0.9414 ± 0.2349

CV und Konfusionsmatrix



(a) Konfusionsmatrix (CV)



(b) Entscheidungsgrenze

Abbildung 10: GNB: Konfusionsmatrix und Entscheidungsgrenze

- `validation_accuracy`: 0.9454 ± 0.0371
- `validation_f1_macro`: 0.9292 ± 0.0499
- `validation_precision_macro`: 0.9373 ± 0.0452
- `validation_recall_macro`: 0.9304 ± 0.0550

Abbildung 10a zeigt, dass die Klasse Chinstrap mit anderen Klassen verwechselt wird.

Entscheidungsgrenze

Abbildung 10b zeigt die Entscheidungsgrenze des GNB-Modells. Aufgrund der zugrunde liegenden Normalverteilungsannahme werden Beobachtungen, die weit vom jeweiligen Klassenmittelpunkt entfernt liegen, systematisch geringer gewichtet und werden komplett ignoriert. Die Klasse Chinstrap ist dadurch strukturell schwerer zu trennen und wird entsprechend häufiger verwechselt.

3.3 k-Nearest Neighbors (kNN)

kNN setzt voraus, dass Abstände im Feature-Raum sinnvoll sind. Daher ist Skalierung essenziell.

- Vorteile: flexible Entscheidungsgrenze, oft sehr gute Performance, wenig Modellannahmen
- Nachteile: langsamere Inferenz, schwer interpretierbar, *Curse of Dimensionality* in hohen Dimensionen [12]

Auswahl von k

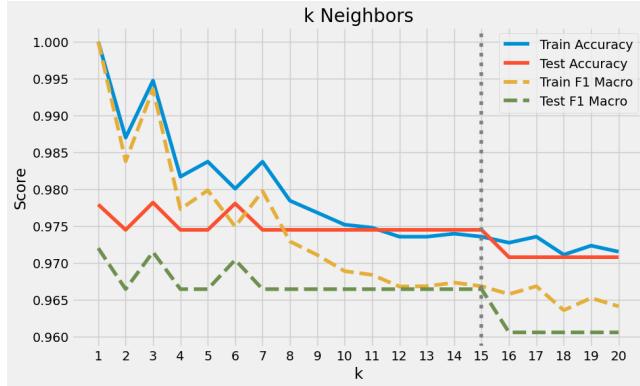


Abbildung 11: kNN: Auswahl von k per LOOCV

Abbildung 11 zeigt, dass die Validation Accuracy im Bereich [7 : 15] relativ stabil bleibt. Man würde hier eher $k = 15$ wählen, da ein etwas größeres k in der Regel besser generalisiert.

LOOCV Feature-Permutation

- Features: [bill_length_mm, bill_depth_mm, flipper_length_mm]
 - train_accuracy: 0.9744 ± 0.0006
 - validation_accuracy: 0.9744 ± 0.1581
- Features: [bill_length_mm, bill_depth_mm]
 - train_accuracy: 0.9669 ± 0.0009
 - validation_accuracy: 0.9634 ± 0.1879
- Features: [bill_length_mm, flipper_length_mm]
 - train_accuracy: 0.9526 ± 0.0011
 - validation_accuracy: 0.9524 ± 0.2130

Obwohl die beste Performance mit allen drei Features erreicht wird, ist die Kombination von bill_length_mm und bill_depth_mm auch sehr gut und bietet den Vorteil der Interpretierbarkeit. bill_length_mm und flipper_length_mm schneiden etwas schlechter ab.

Anmerkung: Da ich k auf denselben Trainingsdaten auswähle, auf denen ich danach LOOCV berechne, können die Ergebnisse leicht optimistisch sein. Für die ideale Vorgehensweise siehe [4].

SHAP (modellanagnostisch)

Abbildung 12 zeigt, dass die wichtigsten Features in allen drei Klassen entweder bill_length_mm oder bill_depth_mm sind, wenn wir das Modell mit allen drei Features verwenden. Das bestätigt

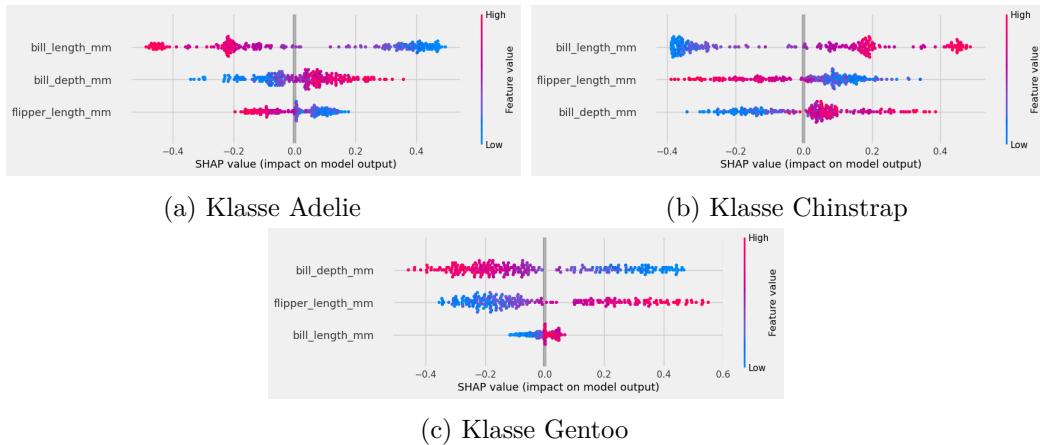


Abbildung 12: SHAP: kNN

unseren Eindruck aus der vorherigen Analyse, dass diese beiden Merkmale wichtiger sind als `flipper_length_mm`.

CV und Konfusionsmatrix

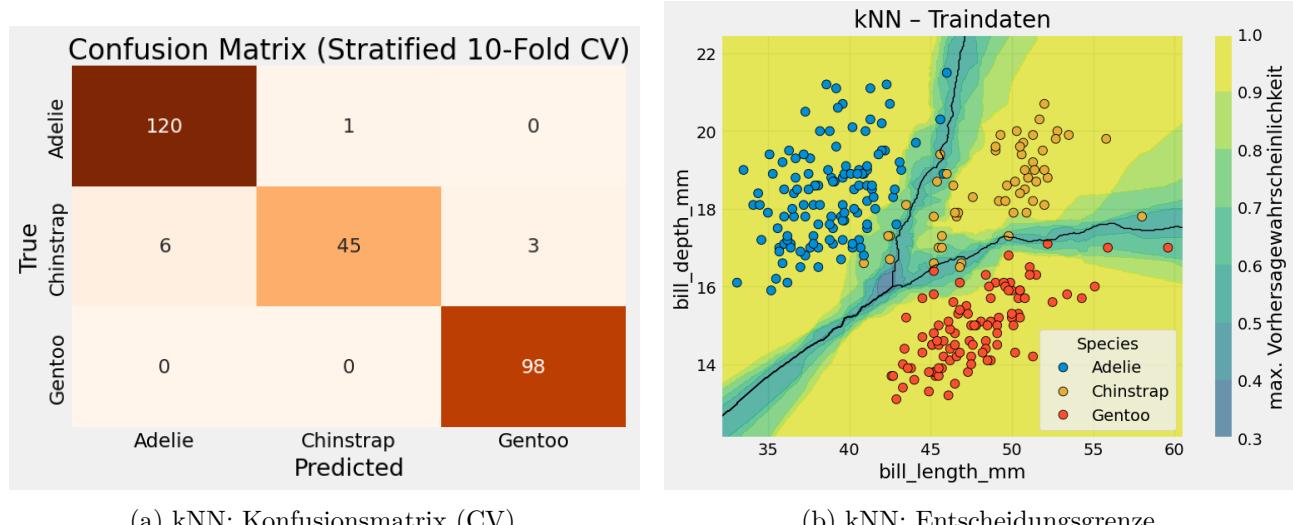


Abbildung 13: kNN: Konfusionsmatrix und Entscheidungsgrenze

- validation_accuracy: 0.9636 ± 0.0360
 - validation_f1_macro: 0.9506 ± 0.0478
 - validation_precision_macro: 0.9689 ± 0.0369
 - validation_recall_macro: 0.9417 ± 0.0536

Abbildung 13a zeigt, dass das Modell insgesamt gut generalisiert. Die Validation Accuracy und die Precision sind sehr hoch, während der Recall etwas schlechter ausfällt. Das bedeutet, dass das Modell mehr False Negatives produziert.

Dieses Verhalten lässt sich sehr wahrscheinlich auf die Klasse Chinstrap zurückführen. Das Modell übersieht diese Klasse häufiger, als dass es sie fälschlicherweise erkennt. Konkret wurde Chinstrap neun Mal übersehen und nur einmal falsch vorhergesagt. Aus der Analyse mit Naive Bayes wissen wir bereits, dass diese Klasse strukturell schwieriger zu unterscheiden ist, was

unter anderem auch mit ihrem geringeren Vorkommen im Datensatz zusammenhängen kann (Klassenungleichgewicht).

Entscheidungsgrenze

Die Entscheidungsgrenze in Abbildung 13b sieht insgesamt sehr sinnvoll aus. Die Bereiche wo GNB Probleme hatte, sind hier deutlich besser getrennt. Das bereits erwähnte Recall-Problem bei Chinstrap ist auch hier erkennbar.

3.4 Entscheidungsbaum

Entscheidungsbäume sind sehr gut interpretierbar, neigen aber ohne Regularisierung zu Overfitting.

- Vorteile: White-Box, sehr gut erklärbar, schnelle Inferenz
- Nachteile: hohe Varianz ohne Pruning, achsenparallele Splits, *gierige* lokale Optimierung

Hyperparameter

ccp_alpha	0.0	max_leaf_nodes	10
class_weight	null	min_impurity_decrease	0.0
criterion	gini	min_samples_leaf	1
max_depth	4	min_samples_split	2
max_features	null	random_state	42
splitter	best		

LOOCV Feature-Permutation

- Features: [bill_length_mm, bill_depth_mm, flipper_length_mm]
 - train_accuracy: 0.9890 ± 0.0005
 - validation_accuracy: 0.9634 ± 0.1879
- Features: [bill_length_mm, bill_depth_mm]
 - train_accuracy: 0.9816 ± 0.0026
 - validation_accuracy: 0.9707 ± 0.1687
- Features: [bill_length_mm, flipper_length_mm]
 - train_accuracy: 0.9780 ± 0.0009
 - validation_accuracy: 0.9414 ± 0.2349

Die Validation Accuracy der Featurekombination bill_length_mm und bill_depth_mm ist am besten und sogar besser als die anderen Modelle, die wir bisher probiert haben, und das trotz der Einschränkung von Tiefe und Anzahl der Blätter.

3.5 Manueller Entscheidungsbaum

Pruning und LOOCV

Mithilfe vom interaktiven SuperTree [6] wird der Baum vereinfacht, um eine klare Entscheidungslogik zu erhalten. Nach der Vereinfachung wird der Baum in 14 dargestellt. Die finalen

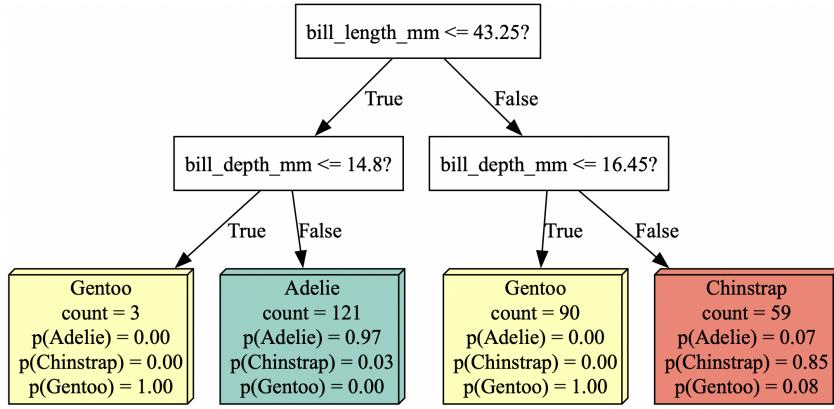


Abbildung 14: Entscheidungsbaum (manueller Baum)

Regeln sehen wie folgt aus:

- Wenn $\text{bill_length_mm} \leq 43.25$ und $\text{bill_depth_mm} > 14.8$, dann `Adelie`
- Wenn $\text{bill_length_mm} > 43.25$ und $\text{bill_depth_mm} > 16.45$, dann `Chinstrap`
- Sonst `Gentoo`

Und damit bekommt man die folgenden LOOCV-Ergebnisse:

- `train_accuracy`: 0.9524 ± 0.0008
- `validation_accuracy`: 0.9524 ± 0.2130

Das vereinfachte Modell ist zwar etwas weniger genau, hat aber deutlich bessere Chancen, sauber zu generalisieren. Zusätzlich liegt der Fokus dieses Projekts klar auf Interpretierbarkeit. Der vereinfachte Baum ist wesentlich leichter und direkter zu verstehen als der vorherige, komplexere Baum.

Anmerkung: Für die ausführlichen Schritte zur Baumvereinfachung sowie eine Erläuterung, warum das Feature `flipper_length_mm` hier keine weitere Verbesserung liefert, siehe [5].

CV und Entscheidungsgrenze

- `validation_accuracy`: 0.9526 ± 0.0327
- `validation_f1_macro`: 0.9404 ± 0.0429
- `validation_precision_macro`: 0.9427 ± 0.0388
- `validation_recall_macro`: 0.9459 ± 0.0464

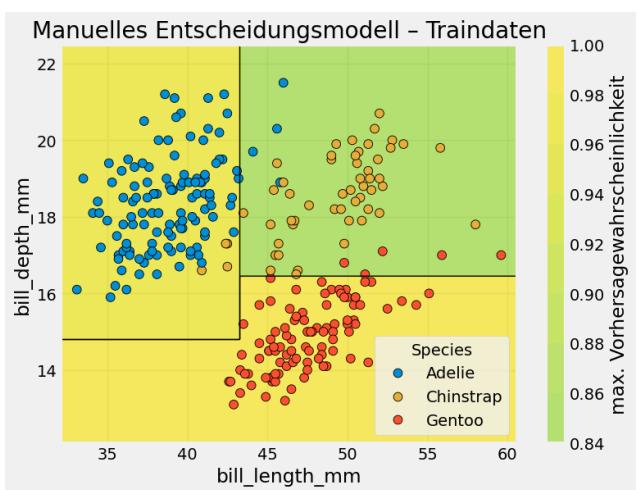
Das Modell zeigt kein generelles Precision Recall Problem. Es ist lediglich, ähnlich wie bei GNB, also insgesamt schwächer in der Vorhersage der Klasse `Chinstrap`.

Entscheidungsgrenze

In Abbildung 15b sehen wir im Grunde dasselbe Problem wie bei GNB in 10b, allerdings aus einem anderen Grund. Dort lag das Problem in der Annahme einer Normalverteilung. Hier liegt es an der Annahme achsenparalleler Entscheidungsgrenzen.

Confusion Matrix (Stratified 10-Fold CV)		
		Predicted
True	Adelie	117
	Chinstrap	4
Gentoo	Adelie	0
	Chinstrap	5
	Gentoo	93

(a) Manueller Baum: Konfusionsmatrix (CV)



(b) Manueller Baum: Entscheidungsgrenze

Abbildung 15: Manueller Entscheidungsbaum: Konfusionsmatrix und Entscheidungsgrenze

3.6 Modellvergleich und Auswahl

Abbildungen 10a, 13a und 15a zeigen die Konfusionsmatrizen und 10b, 13b und 15b die Entscheidungsgrenzen der drei Modelle im Vergleich. Die Abbildung 16 zeigt, kNN schneidet insgesamt am besten ab, gefolgt vom manuellen Entscheidungsbaum, während GNB die niedrigste Performance zeigt. GNB ist dabei relativ stark biased.

Der Entscheidungsbaum 15b hat hauptsächlich einen anderen Nachteil. Innerhalb einer Region weist er allen Punkten die gleiche Wahrscheinlichkeit zu. Dadurch bekommen auch Punkte, die sehr weit von der Entscheidungsgrenze entfernt liegen und eigentlich eindeutig zu einer Klasse gehören, keine besonders hohe Wahrscheinlichkeit, wenn die gesamte Region als weniger zuverlässig eingestuft wird.

kNN hat diese beiden Nachteile nicht. Es findet eine sehr flexible Entscheidungsgrenze, ohne dabei stark zu overfitten, und ordnet jedem Punkt eine eigene Wahrscheinlichkeit zu, abhängig davon, wie nah er an der Entscheidungsgrenze liegt. Dadurch funktioniert kNN in diesem Fall sehr gut. Wegen nur zwei Features kann kNN hier sogar als eine Art Quasi White Box Modell betrachtet und relativ einfach interpretiert werden. Der eigentliche Nachteil von kNN ist die langsame Inferenz, da für jede Vorhersage alle Trainingspunkte betrachtet werden müssen.

Man muss allerdings beachten, dass der Vergleich zwischen kNN und dem Entscheidungsbaum nicht ganz fair ist. kNN erhält hier maximale Freiheit, während der Entscheidungsbaum absichtlich manuell auf maximal drei Splits beschränkt wurde. Wir haben zuvor gesehen, dass ein Entscheidungsbaum mit vier Ebenen deutlich besser abschneidet als kNN. Wenn reine Performance das wichtigste Kriterium wäre, sollte daher eher ein tieferer Entscheidungsbaum verwendet werden.

Da ich mich in diesem Projekt jedoch bewusst für Einfachheit und Interpretierbarkeit entschieden habe, ist die Performance des manuellen Entscheidungsbaums dennoch sehr solide. Aus diesem Grund entscheide ich mich, weiterhin mit dem vereinfachten Entscheidungsbaum zu arbeiten.

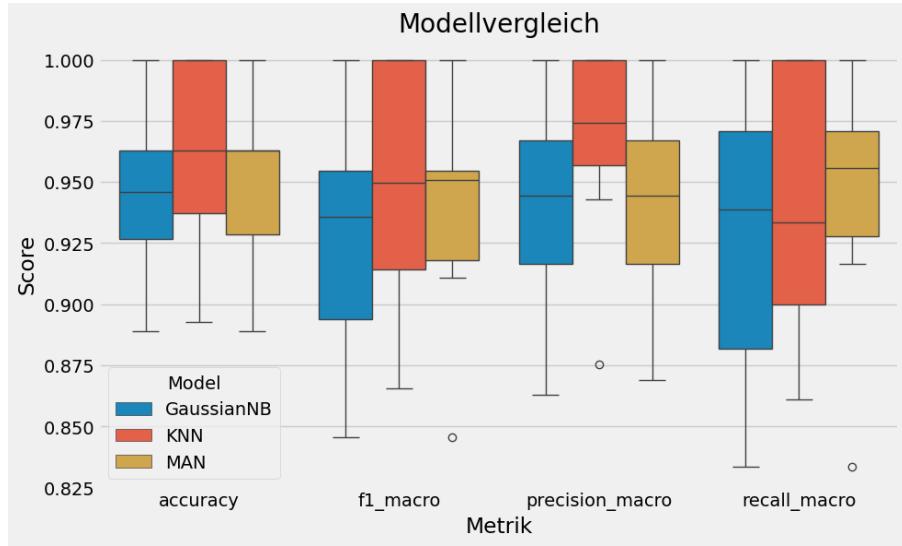


Abbildung 16: Benchmarking der Modelle

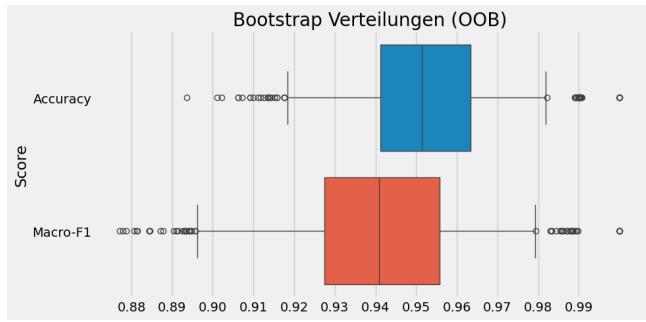


Abbildung 17: Bootstrap Verfahren (finales Modell)

3.7 Bootstrapping

Das Bootstrap Ergebnis dient ausschließlich zur Abschätzung der erwarteten Schwankungsbreite auf Basis der Trainingsdaten. Es wird erst nach Abschluss der Modellentscheidungen durchgeführt und nicht zur Modellwahl verwendet. Aus Abbildung 17 ergibt sich das 95%-Konfidenzintervall mit 1000 Bootstrap Samples:

- `oob_accuracy`: 0.952 [0.918 – 0.982]
- `oob_f1_macro`: 0.941 [0.896 – 0.979]

4 Auswertung

4.1 Evaluation auf dem Testset

Jetzt ist der Moment der Wahrheit: Das finale Modell (manueller Entscheidungsbaum) wird auf dem zuvor vollständig ungenutzten Testset evaluiert. Die Ergebnisse sind in Tabelle 2 und Abbildung 18 dargestellt.

Die Testdaten in Abbildung 18b zeigen Probleme an denselben Stellen wie bereits bei den Trainingsdaten, was auf Underfitting hindeutet. Diese Einschränkung wurde bewusst im Kauf genommen, da Interpretierbarkeit und Stabilität im Fokus stehen.

Klasse	Precision	Recall	F1-Score	Support
Adelie	0.9667	0.9667	0.9667	30
Chinstrap	0.7500	0.8571	0.8000	14
Gentoo	0.9565	0.8800	0.9167	25
Accuracy			0.9130	69
Macro Avg	0.8911	0.9013	0.8944	69
Weighted Avg	0.9190	0.9130	0.9147	69

Tabelle 2: Testset Ergebnisse

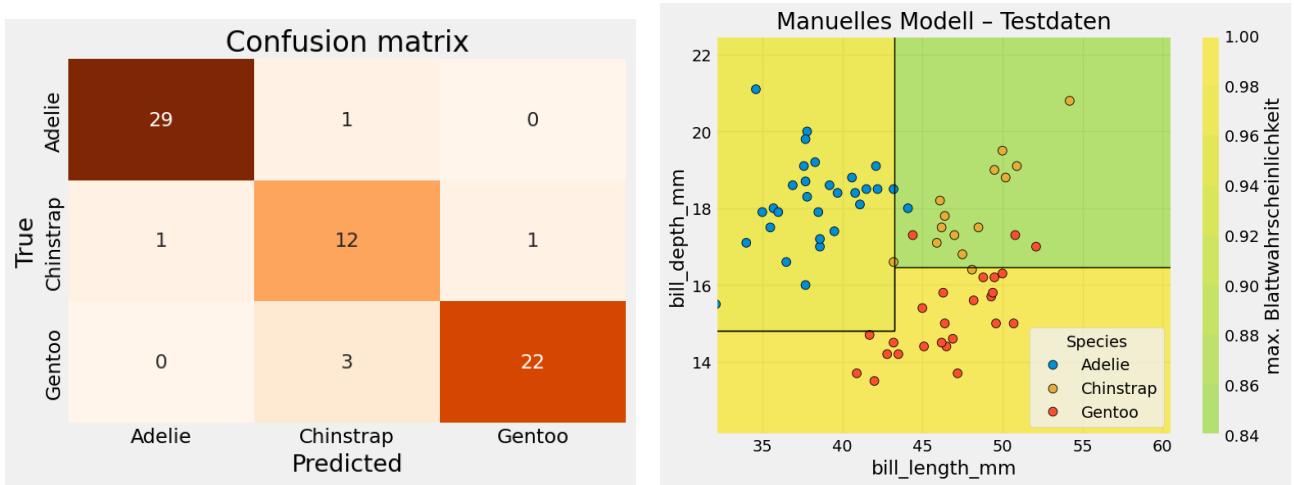


Abbildung 18: Finale Evaluierung auf dem Testset

4.2 Interpretation von Fehlern

Wir schauen uns jetzt genauer an, warum das Modell bestimmte Fehler macht. Zum Beispiel: Warum sagt es Chinstrap, obwohl die tatsächliche Klasse Adelie ist?

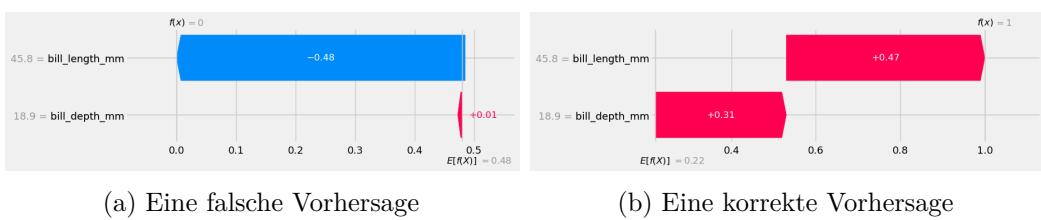


Abbildung 19: SHAP: eine Beispielvorhersage

Abbildung 19 zeigt, dass `bill_length_mm` eindeutig für diese falsche Vorhersage verantwortlich ist. Konkret reduziert der Wert dieses Features im Modell die Wahrscheinlichkeit für die korrekte Klasse Adelie von etwa 48% auf nahezu 0%.

Abbildung 20 zeigt, dass der Wert von `bill_length_mm` deutlich näher am Chinstrap Cluster liegt als am Adelie Cluster. In diesem Fall hat das Modell also tatsächlich sinnvoll entschieden, da wir intuitiv vermutlich zur gleichen Einschätzung gekommen wären. Man könnte überlegen, ob ein zusätzliches Feature wie `flipper_length_mm` dieses Problem lösen könnte. Rechter Plot (20) zeigt jedoch, dass auch dieses Feature näher am Chinstrap Cluster liegt als am Adelie Cluster. Also hätte auch mit diesem Feature das Modell vermutlich dieselbe falsche Vorhersage

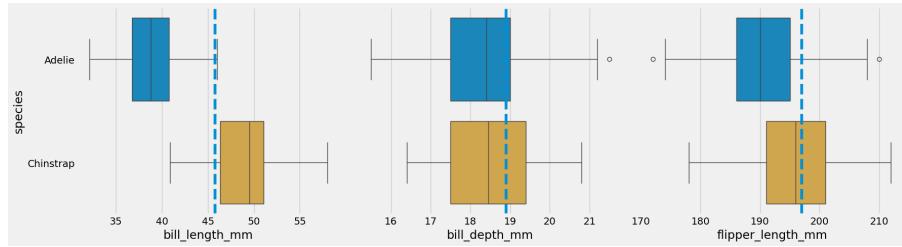


Abbildung 20: Einordnung der Featurewerte (Boxplots)

gemacht.

5 Fazit

Ziel dieser Arbeit war es, ein Klassifikationsmodell zu entwickeln, das nicht nur gute Vorhersagen liefert, sondern vor allem gut interpretierbar ist. Die explorative Datenanalyse zeigte früh, dass `bill_length_mm` und `bill_depth_mm` den größten Beitrag zur Trennung der Klassen leisten. Im Modellvergleich schnitt kNN insgesamt am besten ab, wurde jedoch aufgrund der gewünschten Transparenz nicht als finales Modell gewählt. Stattdessen wurde ein manuell vereinfachter Entscheidungsbaum verwendet, der aus wenigen klaren Regeln besteht und damit direkt erklärbar bleibt. Die finale Evaluation auf dem Testset bestätigt die erwartete Performance und zeigt eine konsistente Fehlerstruktur.

6 Reflexion

Rückblickend gäbe es mehrere Strategien, die theoretisch zu besseren Ergebnissen führen könnten:

- Oversampling zur besseren Erkennung der seltenen Klasse `Chinstrap`
- Robusterer Umgang mit fehlenden Werten (Fallback Logik / separate Inferenzpfade)
- Weniger starke Einschränkung des Baums, um Underfitting zu reduzieren

Diese Optionen wurden bewusst zugunsten einer einfachen, stabilen und transparenten Modelllogik nicht priorisiert.

Literatur

- [1] Mirza Muhammad Omer Baig. Multivariate 3d kde oberfläche von `bill_length_mm` und `bill_depth_mm` (interaktiv). https://mib1213.github.io/MachineLearningLab/figs/kde_3d_surface.html, 2025. Zugriff am 22.12.2025.
- [2] Mirza Muhammad Omer Baig. 3d streudiagramm von `bill_length_mm`, `bill_depth_mm` und `flipper_length_mm` (interaktiv). https://mib1213.github.io/MachineLearningLab/figs/scatter_3d_morphology.html, 2025. Zugriff am 22.12.2025.
- [3] Mirza Muhammad Omer Baig. Machinelearninglab: Quellcode `notebook.ipynb`. <https://github.com/mib1213/MachineLearningLab/blob/main/notebook.ipynb>, 2025. Zugriff am 21.12.2025.

- [4] Mirza Muhammad Omer Baig. Machinelearninglab: Begleitendes projekt-repository. <https://github.com/mib1213/MachineLearningLab>, 2025. Zugriff am 21.12.2025.
- [5] Mirza Muhammad Omer Baig. Machinelearninglab: Begleitende interaktive projekt-webseite. <https://mib1213.github.io/MachineLearningLab/>, 2025. Zugriff am 21.12.2025.
- [6] Mirza Muhammad Omer Baig. Interaktiver entscheidungsbaum (supertree). https://mib1213.github.io/MachineLearningLab/figs/interactive_tree/, 2025. Zugriff am 22.12.2025.
- [7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
- [8] Kristen B. Gorman. Faculty profile (university of alaska fairbanks), 2025. URL <https://www.uaf.edu/cfos/people/faculty/detail/kristen-gorman.php>. Zugriff am 20.12.2025.
- [9] Allison Horst. Artwork for the palmer penguins dataset. <https://allisonhorst.github.io/palmerpenguins/>, 2025. Zugriff am 21.12.2025.
- [10] Allison Horst, Alison Hill, and Kristen Gorman. palmerpenguins: Palmer archipelago (antarctica) penguin data, 2020. URL <https://github.com/allisonhorst/palmerpenguins>. Zugriff am 21.12.2025.
- [11] imbalanced-learn developers. imbalanced-learn documentation. <https://imbalanced-learn.org/>, 2025. Zugriff am 21.12.2025.
- [12] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *An Introduction to Statistical Learning: with Applications in Python*. Springer Texts in Statistics. Springer Nature, Cham, 2023. ISBN 978-3-031-38747-0. URL <https://link.springer.com/book/10.1007/978-3-031-38747-0>. Zugriff am 17.12.2025.
- [13] Palmer Station LTER. Palmer station long term ecological research (lter), 2025. URL <https://pallter.marine.rutgers.edu/>. Zugriff am 20.12.2025.

Hinweis zur Nutzung von KI

Für die vorliegende Arbeit wurden KI-gestützte Werkzeuge unterstützend eingesetzt. Die KI wurde ausschließlich zur sprachlichen Überarbeitung, zum Umformulieren einzelner Textpassagen sowie zur Übersetzung verwendet. Zusätzlich wurde KI punktuell zur Unterstützung bei der Erstellung einzelner Code-Snippets sowie bei der Generierung von Bildern eingesetzt, die jeweils im Quellcode explizit gekennzeichnet sind. Die inhaltliche Ausarbeitung, Analyse und Bewertung der Ergebnisse erfolgten vollständig eigenständig durch den Autor.