Digital Business Management (M.Sc.)

Modul: Artifical Intelligence

Dozent: Herr Prof. Dr. Alexander Rossmann

Fallstudie zur Vorhersage von Hotel Stornierungen durch Machine Learning am Beispiel von StayAwhile Hotels

- Hausarbeit -

von

Felix Zentowski

Github Repository: https://github.com/Flitschi7/stornierungsvorhersage-im-hotelgewerbe
Vorhersage WebTool: https://stayawhile.flitschi7.repl.co/

Böblingen, 16.07.2023

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ML- Machine Learning

KNN- K Nearest Neighbor

1. Unternehmensbeschreibung

Im Fokus dieser Fallstudie steht das fiktive Unternehmen "StayAwhile Hotels". StayAwhile Hotels ist eine international agierende Hotelkette mit Hauptsitz in Bonn, Deutschland.

StayAwhile hat sich einen Namen gemacht durch seinen Fokus auf Kundenzufriedenheit und Qualitätsservice. Das Unternehmen hat eine starke Marke aufgebaut, die auf den Prinzipien von Gastfreundschaft, Integrität und Exzellenz basiert. Trotz ihres Erfolgs in der Branche sieht sich StayAwhile mit einer Reihe von Herausforderungen konfrontiert, insbesondere im Zusammenhang mit Stornierungen.

Im Rahmen ihrer digitalen Transformation hat StayAwhile erkannt, dass Technologie und Datenanalyse Schlüsselkomponenten zur Bewältigung dieser Herausforderungen sind. Das Unternehmen hat daher in die Entwicklung von Machine-Learning-Modellen investiert, um seine Betriebsabläufe zu optimieren und eine datengesteuerte Entscheidungsfindung zu ermöglichen.

2. Problem und Fragestellung

Stornierungen sind in der Hotelbranche ein weit verbreitetes Problem, das erhebliche Auswirkungen auf den Betrieb und die Rentabilität eines Unternehmens haben kann. (Verot, 2023) Für StayAwhile ist das Problem der Stornierungen besonders akut (32% aller Buchungen werden storniert). Unvorhergesehene Stornierungen führen zu unbesetzten Zimmern, die wiederum zu verlorenen Einnahmen führen. Darüber hinaus können sie die Planung und Ressourcenallokation des Unternehmens erheblich beeinträchtigen.

Die zentrale Fragestellung dieser Fallstudie ist daher: Wie kann StayAwhile mithilfe von Machine Learning Stornierungen vorhersagen und so die Betriebsplanung und Ressourcenallokation verbessern?

Um diese Frage zu beantworten, wurde ein Machine-Learning-Modell entwickelt, das auf historischen Buchungsdaten basiert und in der Lage ist, zukünftige Stornierungen vorherzusagen. Durch die Verbesserung der Vorhersagegenauigkeit von Stornierungen kann StayAwhile seine Betriebsplanung optimieren, seine Ressourcen effizienter zuweisen und letztendlich seine Rentabilität steigern.

3. Einordnung in den Bereich Machine Learning

Machine Learning (ML) ist ein Teilbereich der künstlichen Intelligenz, der sich darauf konzentriert, Computern das "Lernen" aus Daten zu ermöglichen, um Muster zu erkennen und Vorhersagen zu treffen. (Alotaibi, 2020)

In der Hotellerie kann ML dazu beitragen, eine Vielzahl von Herausforderungen zu bewältigen, von der Verbesserung der Kundenzufriedenheit bis hin zur Optimierung der Betriebsabläufe. (Devisme, 2019)

In dieser Fallstudie wird das Klassifikationsverfahren verwendet, um das Problem der Stornierungen bei StayAwhile anzugehen. Im ML wird diese Art von Analyse als Klassifikation bezeichnet und entscheidet, ob ein Binäres Ereignis eintritt. Das entwickelte Modell ist ein Beispiel für überwachtes Lernen, eine Art von ML, bei der ein Modell aus einem Satz von Eingabe- und Ausgabedaten "lernt". In diesem Fall sind die Eingabedaten die historischen Buchungsdaten und die Ausgabedaten sind, ob eine Buchung storniert wurde oder nicht.

Das Modell verwendet verschiedene Merkmale aus den Buchungsdaten, wie z.B. den Buchungszeitpunkt, den Aufenthaltszeitraum und den Buchungstyp, um Muster zu erkennen und Vorhersagen über zukünftige Stornierungen zu treffen. Durch die Verbesserung der Vorhersagegenauigkeit von Stornierungen kann StayAwhile Datenbasierte Entscheidungen treffen.

Zum Angehen des Problems wird sich auf das CRISP- DM Framework bezogen. (Abbildung 1: CRISP DM Cycle)
Hierbei werden Datenanalyse Probleme, strukturiert angegangen und können einer Logik folgen. Umso das Businessproblem zu lösen. Das Business Understanding wurde in den ersten beiden Kapiteln behandelt. Zum Data Understanding und den folgenden Schritten wurde Python mit seiner Vielzahl an Daten Analyse Libraries hinzugezogen.

4. Data Understanding

Der vorliegende Abschnitt konzentriert sich auf die Analyse und Interpretation des Datensatzes mit dem Ziel, ein tieferes Verständnis der Daten zu erlangen und wertvolle Erkenntnisse zu gewinnen. Die Daten wurden in einem Jupyter Notebook mit Python analysiert, wobei verschiedene Bibliotheken für Datenanalyse und maschinelles Lernen verwendet wurden.

Der analysierte Datensatz enthält Informationen über verschiedene Aspekte von Hotelreservierungen, darunter die Anzahl der Erwachsenen und Kinder pro Buchung, die Anzahl der Wochenend- und Wochentage pro Buchung, der Typ des gebuchten Mahlzeitplans, ob ein Parkplatz benötigt wird oder nicht, der Typ des reservierten Zimmers und die Zeitspanne zwischen der Reservierung und dem Ankunftsdatum. Jede Reservierung ist durch eine eindeutige Buchungs-ID gekennzeichnet. Unsere Zielvariable ist booking_status mit den Ausprägungen: Canceled und Not_Canceled.

Eine erste Untersuchung des Datensatzes ergab, dass es keine Duplikate oder Nullwerte gibt, was auf eine hohe Datenqualität hindeutet. Eine detaillierte statistische Analyse zeigte, dass die meisten Buchungen von ein oder zwei Erwachsenen gemacht werden und dass die meisten Buchungen keine Kinder beinhalten. Darüber hinaus beinhalten viele Buchungen sowohl Wochenendnächte als auch Wochentage, wobei die Anzahl variiert.

Die Verteilung der Zielvariable in einem Datensatz, insbesondere in einem Klassifikationsproblem, hat erhebliche Auswirkungen auf die Modellierung und die Interpretation der Ergebnisse. In diesem Fall ist die Zielvariable der Buchungsstatus, welcher zwei Klassen hat: "storniert" und "nicht storniert". Die Verteilung dieser Klassen ist etwa 30% zu 70%. Diese Verteilung bedeutet, dass unser Datensatz leicht unausgewogen ist, da die "nicht stornierten" Buchungen etwa 70% der Daten ausmachen, während die "stornierten" Buchungen nur etwa 30% ausmachen. Dies ist jedoch nicht unausgewogen; in der Praxis wird eine Verteilung, die näher an 50/50 liegt, oft als ideal angesehen, aber eine 70/30-Verteilung ist immer noch handhabbar. Die Auswirkungen dieser Verteilung auf die Modellierung können vielfältig sein. Ein Modell könnte dazu neigen, die Mehrheitsklasse ("nicht storniert") zu bevorzugen, da es durch die Vorhersage der Mehrheitsklasse eine höhere Genauigkeit erzielen kann. Dies könnte dazu führen, dass das Modell weniger effektiv bei der Vorhersage der Minderheitsklasse ("storniert") ist. (Abbildung 2: Verteilung Zielvariable)

4.1 Untersuchen der Numerischen Features

Der Datensatz enthält eine Vielzahl von numerischen Merkmalen, die wichtige Informationen über die Hotelreservierungen liefern. Diese Merkmale umfassen:

- no_of_adults: Die Anzahl der Erwachsenen pro Buchung.

- no_of_children: Die Anzahl der Kinder pro Buchung.

- no_of_weekend_nights: Die Anzahl der Wochenendnächte pro Buchung.

- no_of_week_nights: Die Anzahl der Wochentage pro Buchung.

- lead_time: Die Zeitspanne zwischen der Reservierung und dem Ankunftsdatum.

-arrival_year/ month/ date: Jahr/ Monat und Tag der Buchung

- no_of_previous_cancellations: Die Anzahl der vorherigen Stornierungen des Gastes.

- no_of_previous_bookings_not_canceled: Die Anzahl der vorherigen Buchungen des Gastes, die nicht storniert wurden.

- avg_price_per_room: Der durchschnittliche Preis pro Zimmer.

- no_of_special_requests: Die Anzahl der speziellen Anfragen des Gastes.

Darüber hinaus gibt es zwei Merkmale, die numerische Werte annehmen, aber tatsächlich kategoriale Daten darstellen:

- required_car_parking_space: Gibt an, ob ein Parkplatz benötigt wird oder nicht.

- repeated_guest: Gibt an, ob der Gast schonmal eine Buchung vorgenommen hat oder nicht.

Obwohl diese Merkmale numerische Werte annehmen (0 und 1), sind sie kategorial, weil sie eine bestimmte Kategorie oder Gruppe darstellen, nicht eine quantitative Messung. Sie sind binär, weil sie nur zwei mögliche Werte haben.
Für alle numerischen Features wurde ein Histogramm erstellt, um diese besser bewerten zu können und eventuelle Auffälligkeiten zu erkennen.

Das Histogramm für lead_time zeigt einen positiven Skew, was darauf hindeutet, dass die meisten Buchungen spontan getätigt werden. Ob hier Ausreiser entfernt werden sollten, wird im nächsten Schritt geklärt, wenn ein Vergleich der Werte für storniert und nicht storniert untersucht wird. (Abbildung 3: Histogramm für lead_time)

Das Histogramm für avg_price_per_room zeigt auch einen leichten positiven Skew, dieses ist aber deutlich normalverteilter. Auffällig ist, dass es auch Zimmer umsonst gibt. Hier ist die Frage, ob es fehlerhafte Daten sind oder gegebenenfalls per Gutschein gebuchte Zimmer sind. Ob hier eine Anpassung erfolgen soll, wird auch im nächsten Schritt geklärt. (Abbildung 4: Histogramm für avg_price_per_room) Das Histogramm für arrival_month zeigt, dass die Anzahl der Buchungen je nach Monat variiert. Es gibt einige Monate mit einer höheren Anzahl von Buchungen, was auf die Hochsaison hinweisen könnte, und andere Monate mit einer niedrigeren Anzahl von Buchungen, was auf die Nebensaison hinweisen könnte. Generell ist auffällig, dass sich die Buchungen in den Winter Monaten häufen. (Abbildung 5: Histogramm für arrival_month)

Es ist zusätzlich noch zu erkennen, dass es Datensätze gibt, die eine Anzahl von 0 Erwachsenen Person haben, im Weiteren wird untersucht, ob diese Erkenntnis besagt, dass die Datensätze falsch sind oder eventuell nur Kinder gebucht haben, das wird im Schritt des Feature Engineerings geklärt. Weitere Verteilungen sind im Notebook zu finden.

4.2 Untersuchung der Numerischen Variablen zu der Zielvariablen

Im Folgenden werden die Verteilungen in Bezug auf die Zielvariable geplottet.

Die interessantesten Beobachtungen, sind, dass je kürzer die lead time, desto eher werden die Buchungen nicht storniert, weshalb auch zunächst keine Ausreißer entfernt werden. (Abbildung 6: Histogramm für lead_time nach Zielvariable)

Zudem fällt beim Preis auf, dass die Verteilung für storniert etwas nach rechts verschoben ist. Inklusive der wichtigsten Erkenntnis, dass Buchungen für 0 € kaum bzw. gar nicht storniert werden. Diese Information ist wichtig und soll dem Modell auch mitgegeben werden, also werden die Daten nicht entfernt. (Abbildung 7: Histogramm für avg_price_per_room nach Zielvariable)

Um Korrelationen zu untersuchen, wird eine Korrelationsmatrix erstellt. (Abbildung 8: Korrelationsmatrix)
Es ist zu erkennen dass Korrelationen zwischen repeated_guest, no_of_previous_cancellations und no_of_previous_bookings_not_canceled vorliegt. Das macht auch Sinn, da diese alle in Abhängigkeit dazu sind ob schonmal vorher gebucht worden ist. Der avg_price_per_room korreliert zu dem mit der Anzahl der Personen, was auch Sinn ist da größere Räume teurer sein müssten.
Ansonsten fällt noch auf, wie schon vorher entdeckt, dass die lead_time negativ mit dem booking_status korreliert und somit der wichtigste Wert für unser Model ist.
Zunächst wird Multikollinearität nicht untersucht und der VIF nicht berechnet. Da dies nur für eine logistische Regression relevant ist. (Strike, 2020) Jedoch ist es trotzdem relevant Feature Engineering zu betreiben. Und eventuell das Feature „repeated_guest“ rauszuschmeißen, da es durch die Features: „no_of_previous_cancellations“ und „no_of_previous_bookings_not_canceled“ beschrieben wird. Das wird aber weiter in der Evaluation geklärt.

4.3 Untersuchung der kategorialen Features

In dem Abschnitt wurden alle Features untersucht welche Kategorial sind. Wichtige Erkenntnisse, die hier aufgefallen sind, sind das Wiederkehrende Gäste, Gäste die einen Parkplatz buchen und geschäftliche Buchungen tendenziell weniger storniert werden.

5. Data Preparation

5.1 Ausreiser entfernen

Anfangs sollen keine Ausreiser entfernt werden, dies kann bei Bedarf nach der Evaluation erfolgen. Dazu entschieden wurde sich, da es keine eindeutigen Ausreißer gab und auch die Ausreißer einen Einfluss auf das Buchungsverhalten haben.

5.2 Feature Engineering

Als Feature wurde, das Jahr und der Tag zunächst entfernt. Diese Informationen sind für das Model nicht aussagekräftig. Das Feature Monat wurde behalten, da dieses Saisonalität mit einbehaltet.

Es wurden zwei neue Features erstellt und einige gelöscht. Aus den Features no_of_weekend_nights und no_of_week_nights wurde das Feature total_stay. Und aus no_of_adults und no_of_children wurde total_guests.

Total_guests weist in diesem Zuge dann auch keine 0 Werte auf, was das Problem mit no_of_adults behebt. Denn vermutlich können auch Zimmer für Kinder gebucht werden und nicht nur für Erwachsene. Man könnte in dieser Hinsicht ein weiteres Kategoriales Feature etablieren: Adult/ Children/ Both. Wenn das Model ein nicht hinreichendes Ergebnis erzielt, wird dieses Feature zusätzlich eingebaut.

5.3 Kategoriale Variablen auswertbar machen

Um das Problem der kategorialen Variablen anzugehen, wird One-Hot-Encoding verwendet.

One-Hot-Encoding ist ein weit verbreitetes Verfahren zur Transformation kategorialer Variablen in eine Form, die für maschinelles Lernen und statistische Modelle geeignet ist. Kategoriale Variablen sind typischerweise diskret und repräsentieren verschiedene Klassen oder Kategorien, die keine numerische Beziehung zueinander haben. Daher ist es oft nicht sinnvoll, diese Variablen direkt in numerischer Form zu verwenden, da dies zu irreführenden oder falschen Interpretationen führen kann. One-Hot-Encoding adressiert dieses Problem, indem es jede Kategorie einer kategorialen Variable in eine separate binäre Variable umwandelt. Jede dieser binären Variablen nimmt den Wert 1 an, wenn die ursprüngliche Variable die entsprechende Kategorie aufweist und 0 wenn nicht. Auf diese Weise wird jede Kategorie durch eine eindeutige binäre Variable repräsentiert, und es wird keine irreführende numerische Beziehung zwischen den Kategorien impliziert. (Trotta, 2022)

One-Hot-Encoding wird für die Variablen: type_of_meal_plan, room_type_reserved, market_segment_type angewendet.

5.4 Trainings und Testdatensätze erstellen und Standardisieren

Der Train- Test Split, ist ein grundlegender Schritt in der Vorbereitung von Daten für maschinelles Lernen und statistische Modellierung. Der Hauptzweck dieser Aufteilung besteht darin, die Fähigkeit des Modells zur Generalisierung auf neue, unbekannte Daten zu bewerten.

Der Trainingsdatensatz wird verwendet, um das Modell zu trainieren, d.h., die Modellparameter so anzupassen, dass sie die Beziehungen in den Trainingsdaten so gut wie möglich abbilden. Der Testdatensatz wird dann verwendet, um die Leistung des trainierten Modells zu bewerten. Da das Modell während des Trainingsprozesses keinen Zugang zu den Testdaten hat, bietet der Testdatensatz eine unvoreingenommene Messung der Fähigkeit des Modells, Vorhersagen für neue Daten zu treffen.

In dem Hotelbuchungsdaten von StayAwhile, wird ein Modell trainiert, um den booking_status auf der Grundlage der anderen Variablen im Datensatz

vorherzusagen. Der Trainingsdatensatz wird verwendet, um dieses Modell zu trainieren, und der Testdatensatz wird verwendet, um zu bewerten, wie gut das Modell in der Lage ist, den booking_status für neue Buchungen vorherzusagen.

Die Standardisierung der Daten erfolgt nach dem Train Test Split. Viele maschinelle Lernalgorithmen arbeiten besser, wenn die Eingabevariablen auf einer ähnlichen Skala liegen und eine ähnliche Verteilung aufweisen. Die Standardisierung erreicht dies, indem sie jede Variable so transformiert, dass sie einen Mittelwert von 0 und eine Standardabweichung von 1 hat. Es wird die Standardisierung der Normalisierung vorgezogen, da einige der Variablen eine Skew nach rechts aufweisen und Ausreißer enthalten können. Zu diesem Zweck verwenden wir den „StandardScaler“ von Scikit-learn. Er standardisiert Merkmale durch Entfernen des Mittelwerts und Skalierung auf eine Einheitsvarianz. (Pedregosa & al., 2011)

Die Standardisierung kann besonders wichtig sein, wenn die Eingabevariablen sehr unterschiedliche Skalen oder Einheiten haben. Im Hotelbuchungsdatensatz haben Variablen wie lead_time (gemessen in Tagen) und no_of_adults (gemessen in der Anzahl der Personen) sehr unterschiedliche Skalen und Einheiten. Die Standardisierung dieser Variablen wird sicherstellen, dass sie auf einer ähnlichen Skala liegen und dass keine Variable aufgrund ihrer Skala oder Einheiten einen unverhältnismäßig großen Einfluss auf das Modell hat.

6. Modeling

In der vorliegenden Arbeit werden unterschiedliche maschinelle Lernmodelle implementiert. Die Leistung der Modelle wird dann durch verschiedene Metriken bewertet. Der Klassifikationsbericht enthält Metriken wie Genauigkeit, F1-Score, Recall und Precision. Diese Metriken bieten eine umfassende Bewertung der Leistung des Modells. Die Konfusionsmatrix bietet eine visuelle Darstellung der Leistung des Modells und zeigt die Anzahl der wahren positiven, wahren negativen, falsch positiven und falsch negativen Vorhersagen. Genauso die Learning Curves, welche beim Bewerten der Trainings und Testdaten helfen können.

6.1 Logistische Regression

Die logistische Regression ist ein statistisches Modell, das in der Regel zur Lösung von binären Klassifikationsproblemen eingesetzt wird. Es handelt sich um ein überwachtes Lernverfahren, das die logistische Funktion verwendet, um eine Wahrscheinlichkeit zwischen 0 und 1 zu erzeugen. Diese Wahrscheinlichkeit wird dann zur Vorhersage der Zielklasse verwendet. Die logistische Regression ist ein parametrisches Modell, das die Beziehung zwischen den Merkmalen und der logarithmischen Wahrscheinlichkeit des Ausgangs durch eine lineare Gleichung beschreibt. Die Parameter dieser Gleichung werden durch den Trainingsprozess bestimmt, der darauf abzielt, die Diskrepanz zwischen den vorhergesagten und den tatsächlichen Ausgangswerten zu minimieren. Dies wird in der Regel durch eine Methode erreicht, die als maximale Likelihood-Schätzung bekannt ist. (Bonthu, 2021)

6.2 Decision Tree

Im weiteren Verlauf der Analyse wird ein weiteres maschinelles Lernmodell implementiert, der Entscheidungsbaum-Klassifikator. Entscheidungsbäume sind eine Art überwachtes Lernmodell, das sowohl für Klassifikations- als auch für Regressionsprobleme verwendet werden kann. Sie sind besonders nützlich, wenn

die Daten nichtlineare Beziehungen aufweisen, die von Modellen wie der logistischen Regression nicht gut erfasst werden können. Ein Entscheidungsbaum teilt die Daten rekursiv auf der Grundlage von Merkmalswerten auf, um homogene Untergruppen in Bezug auf die Zielvariable zu erstellen. Jede Aufteilung wird durch eine Entscheidungsregel repräsentiert, die auf einem Merkmal basiert. Diese Regeln bilden zusammen die Struktur des Baums. (Chauhan, 2022)

6.3 Random Forest

Im Anschluss an die logistische Regression und dem Entscheidungsbaum-Klassifikator wird in der Analyse ein weiteres maschinelles Lernmodell, der Random Forest Klassifikator, eingeführt. Der Random Forest Klassifikator ist ein Ensemble-Lernverfahren, das auf einer Sammlung von Entscheidungsbaumklassifikatoren basiert. Diese Entscheidungsbäume werden unabhängig voneinander auf verschiedenen Teilmengen der Daten trainiert und ihre Vorhersagen werden dann gemittelt, um eine endgültige Vorhersage zu erzeugen.Die Stärke des Random Forest Klassifikators liegt in seiner Fähigkeit, Overfitting zu vermeiden, ein Problem, das häufig bei einzelnen Entscheidungsbaumklassifikatoren auftritt. Overfitting tritt auf, wenn ein Modell zu komplex ist und die Trainingsdaten zu genau "lernt", was dazu führt, dass es bei neuen, unbekannten Daten schlecht abschneidet. Durch die Kombination von Vorhersagen aus mehreren Modellen kann der Random Forest Klassifikator dieses Problem umgehen und in der Regel eine bessere allgemeine Vorhersageleistung erzielen. (Donges, 2023)

6.4 K- Nearest Neighbor

Zum Abschluss der Modellübersicht wird ein K-Nearest Neighbors (KNN) Klassifikator implementiert. Der KNN ist ein instanzbasiertes Lernverfahren, das auf der Idee basiert, dass Datenpunkte, die in dem Merkmalsraum nahe beieinander liegen, wahrscheinlich ähnliche Ausgangswerte haben. Bei einer gegebenen Anfrage sucht der KNN die k nächstgelegenen Datenpunkte in den Trainingsdaten und gibt die am häufigsten vorkommende Klasse unter diesen Punkten als Vorhersage aus. (Raschka, 2018) In der spezifischen Implementierung, wird ein KNN Klassifikator mit k gleich 5 verwendet. Das Modell wird mit den skalierten Trainingsdaten und den entsprechenden Zielwerten trainiert.

6.5 Einordnen der Ergebnisse

6.5.1 Accuracy

Die Genauigkeit (Accuracy) ist eine Metrik zur Bewertung von Klassifikationsmodellen und wird berechnet als das Verhältnis der Anzahl der korrekten Vorhersagen zur Gesamtzahl der Vorhersagen. In Bezug auf die Konfusionsmatrix, die die wahren positiven (TP), wahren negativen (TN), falsch positiven (FP) und falsch negativen (FN) Vorhersagen darstellt, kann die Genauigkeit auch wie folgt berechnet werden:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Modell	Genauigkeit
Logistische Regression	0.80
Decision Tree	0.87
Random Forest	0.90
K-Nearest Neighbors (KNN)	0.86

6.5.2 Precision

Die Precision (Präzision) ist eine Metrik, die das Verhältnis der wahren positiven Vorhersagen zur Summe der wahren positiven und falsch positiven Vorhersagen darstellt. Es ist ein Maß dafür, wie viele der als positiv klassifizierten Proben tatsächlich positiv sind. Die Formel zur Berechnung der Precision ist:

Precision = TP / (TP + FP)

Modell	Klasse 0	Klasse 1
Logistische Regression	0.75	0.83
Decision Tree	0.80	0.91
Random Forest	0.87	0.91
K-Nearest Neighbors (KNN)	0.81	0.89

6.5.3 Recall

Der Recall, auch als Sensitivität oder Trefferrate bekannt, ist eine Metrik, die das Verhältnis der wahren positiven Vorhersagen zur Summe der wahren positiven und falsch negativen Vorhersagen darstellt. Es ist ein Maß dafür, wie viele der tatsächlich positiven Proben korrekt als positiv klassifiziert wurden. Die Formel zur Berechnung des Recall ist:

Recall = TP / (TP + FN)

Modell	Klasse 0	Klasse 1
Logistische Regression	0.62	0.89
Decision Tree	0.81	0.90
Random Forest	0.81	0.94
K-Nearest Neighbors (KNN)	0.77	0.91

6.5.4 F1 Score

Der F1-Score ist eine Metrik, die das harmonische Mittel von Precision und Recall darstellt. Es ist ein Maß dafür, wie gut ein Modell sowohl Precision als auch Recall ausbalanciert. Die Formel zur Berechnung des F1-Scores ist:

F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

Modell	Klasse 0	Klasse 1
Logistische Regression	0.68	0.86
Decision Tree	0.80	0.90
Random Forest	0.84	0.92
K-Nearest Neighbors (KNN)	0.79	0.90

Zusammenfassend lässt sich sagen, dass der Random Forest Klassifikator insgesamt die beste Leistung unter den vier Modellen zeigte, mit der höchsten Accuracy, der höchsten Precision, dem höchsten Recall und dem höchsten F1-Score. Dies deutet darauf hin, dass der Random Forest Klassifikator ein effektives Modell für dieses spezifische Klassifikationsproblem ist. Beim Random Forest, werden auch beide Klassen gut bewertet und haben die geringste Differenz.
Eine zusätzlich wichtige Erkenntnis ist, dass auf die höher balancierte Klasse meist auch ein besserer Score erreicht wird. Diese Information deutet darauf hin, dass die Klassen angeglichen werden sollten, zum Beispiel durch Oversamplen.

7. Evaluation

7.1 Vergleich von Trainings und Test Accuracy

Die Überprüfung der Trainings- und Testgenauigkeit ist ein grundlegender Schritt in der Evaluierung von maschinellen Lernmodellen. Dieser Prozess hilft dabei, die Fähigkeit des Modells zu beurteilen, auf neue, unbekannte Daten zu generalisieren, die es während des Trainings nicht gesehen hat.

In dem vorliegenden Notebook wird der Datensatz in zwei Teile aufgeteilt: ein Trainingssatz und ein Testsatz. Der Trainingssatz wird verwendet, um das Modell zu trainieren, d.h. die Modellparameter so anzupassen, dass sie die Beziehungen in den Daten so gut wie möglich abbilden. Der Testsatz wird dann verwendet, um die Leistung des trainierten Modells zu bewerten. Da das Modell die Testdaten während des Trainings nicht gesehen hat, gibt die Leistung auf dem Testsatz einen guten Hinweis darauf, wie das Modell auf neue, unbekannte Daten reagieren wird.

Modell	Training Score	Test Score
Logistische Regression	0.805	0.804
Entscheidungsbaum	0.992	0.869
Random Forest	0.992	0.896

Modell	Training Score	Test Score
K-Nearest Neighbors (KNN)	0.899	0.864

Die logistische Regression zeigt eine nahezu identische Genauigkeit sowohl auf den Trainings- als auch auf den Testdaten (ca. 0.80). Dies deutet auf eine gute Generalisierungsfähigkeit hin, da das Modell weder overfitted noch underfitted ist.

Der Entscheidungsbaum und der Random Forest zeigen eine nahezu perfekte Genauigkeit auf den Trainingsdaten, was auf eine mögliches Overfitting hindeuten könnte. Dennoch bleibt die Genauigkeit auf den Testdaten mit 0.87 bzw. 0.90 relativ hoch, was darauf hindeutet, dass diese Modelle trotz ihrer Komplexität eine gute Generalisierungsfähigkeit aufweisen.

Der KNN Klassifikator zeigt eine Genauigkeit von ca. 0.90 auf den Trainingsdaten und 0.86 auf den Testdaten. Dies deutet darauf hin, dass das Modell eine gute Balance zwischen Bias und Varianz erreicht und auf neue Daten gut generalisiert.

Zusammenfassend lässt sich sagen, dass alle vier Modelle eine gute Leistung zeigen, wobei der Random Forest Klassifikator die höchste Genauigkeit auf den Testdaten aufweist. Da das Random Forest Model die besten Ergebnisse erzielt jedoch vermutlich overfitted, werden Anpassungen vorgenommen und dann verglichen.

7.2 Random Forest Model verfeinern

Im folgenden Abschnitt wird das Modell basierend auf den Ergebnissen der ersten Iteration angepasst und durchläuft eine Schleife im CRISP- DM Cycle.
Für das Business ist es wichtig, dass so wenig Zimmer wie möglich unbelegt bleiben, aber es so selten wie möglich vorkommt, dass ein Zimmer welches wirklich gebucht ist und gebucht bleibt als storniert vorhergesagt wird. Also sollte der False Negativ Wert möglichst niedrig sein aber generell auch die unterrepräsentierte Klasse gut bewerten.

7.2.1 Hyperparameter Tuning

Unter Hyperparameter-Tuning versteht man den Prozess der Suche nach den optimalen Hyperparametern. Die Wahl der Hyperparameter kann die Leistung des Modells stark beeinflussen. Zur Abstimmung der Hyperparameter wird eine Methode namens "grid search" verwendet. Der "grid search" ist eine Brute-Force-Methode, bei der ein Modell mit jeder Kombination von Hyperparametern in einem vordefinierten Raster trainiert wird und die Hyperparameter ausgewählt werden, die die beste Leistung erbringen. (Pedregosa & al., 2011)

Zu den wichtigen Hyperparametern für das Random-Forest-Modell gehören: n_estimators: Die Anzahl der Bäume im Wald. max_depth: Die maximale Tiefe der Bäume. min_samples_split: Die Mindestanzahl von Stichproben, die erforderlich ist, um einen internen Knoten zu teilen. min_samples_leaf: Die Mindestanzahl von Stichproben, die erforderlich ist, um einen Blattknoten zu erreichen. (Donges, 2023)
Es wird ein Raster mit möglichen Werten für diese Hyperparameter definiert und eine Rastersuche durchgeführt. (Abbildung 9: GridSearchCV Code)
Die Ergebnisse des Hyperparameter Tuning sind wie folgt:

Random Forest: {'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 500}

Decision Tree: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 10}

7.2.2 Parameter descaling um Overfitting zu verhindern

Durch die Anpassung der Modellparameter, wie die maximale Tiefe der Bäume und die minimale Anzahl von Proben, die benötigt werden, um einen Knoten zu teilen, wurde versucht, die Modellkomplexität zu kontrollieren und Overfitting zu verhindern.

Es wurden folgende Parameter für den Random Forest benutzt: n_estimators=100; max_depth=10; min_samples_split=10

und folgende für den Decision Tree: max_depth=10; min_samples_split=10.

7.2.3 Oversampling von Trainingsdaten mit bestem Parameter

Die Methode, die hier verwendet wird, ist als Random Oversampling bekannt und zielt darauf ab, die Anzahl der Beispiele in der Minderheitsklasse zu erhöhen. Dies wird erreicht, indem zufällig Beispiele aus der Minderheitsklasse ausgewählt und in den Trainingsdaten repliziert werden. Die resultierenden "Oversampled" Trainingsdaten weisen eine ausgeglichene Klassenverteilung auf, was dazu beitragen kann, die Leistung der Modelle zu verbessern. (Lemaître, Nogueira, & Aridas, 2017)

Nachdem die Oversampling-Methode angewendet wurde, wurden die Modelle (Random Forest und Decision Tree) auf den oversampleten Trainingsdaten trainiert. Es ist wichtig zu beachten, dass Oversampling nur auf den Trainingsdaten durchgeführt wird und nicht auf den Testdaten, um eine korrekte Evaluierung der Modellleistung zu gewährleisten.

7.3 Vergleichen der 3 Modelle

Im folgenden Abschnitt wird eine umfassende Vergleichsanalyse der drei Varianten des Random Forest Klassifikationsmodells durchgeführt.

Die drei Varianten unterscheiden sich in der Art und Weise, wie sie trainiert und eingestellt wurden:

Random Forest mit Hyperparameter-Tuning: Dieses Modell wurde mit einer Reihe von Hyperparametern trainiert, die durch ein systematisches Verfahren zur Optimierung der Modellleistung ausgewählt wurden.

Random Forest mit Parameteranpassung zur Vermeidung von Overfitting: Dieses Modell wurde mit spezifischen Parametereinstellungen trainiert, um Overfitting zu verhindern.

Random Forest trainiert auf oversampleten Daten mit optimalen Parametern: Dieses Modell wurde auf einem Datensatz trainiert, der durch Anwendung einer Oversampling-Technik modifiziert wurde, um das Ungleichgewicht der Klassen in den Daten zu beheben. Dabei wurden die "besten" Parameter verwendet, die durch vorherige Analyse als optimal ermittelt wurden.

In der Analyse werden verschiedene Metriken zur Bewertung der Leistung der Modelle herangezogen.

7.3.1 Learning Curves

Für die drei beschriebenen Modelle werden Funktionen definiert, um Lernkurven für die gegebenen Modelle zu zeichnen. Lernkurven sind ein nützliches Tool zur Visualisierung und Analyse des Lernprozesses von Modellen während des Trainings.

Der Trainingsscore gibt an, wie gut das Modell auf den Trainingsdaten abschneidet. Der Cross-Validation-Score gibt an, wie gut das Modell auf einem unabhängigen Datensatz abschneidet, der nicht zum Trainieren des Modells verwendet wurde. Ein hoher Cross-Validation-Score, deutet darauf hin, dass das Modell gut generalisiert und voraussichtlich gut auf neue, unbekannte Daten abschneiden wird.

Ein underfittets Modell hat sowohl auf den Trainingsdaten als auch auf den Validierungsdaten eine schlechte Leistung. Dies zeigt sich in Lernkurven durch eine hohe Fehlerrate (oder einen niedrigen Score) für beide Datensätze, unabhängig von der Größe des Trainingssets.

Ein overfittetes Modell hat auf den Trainingsdaten eine sehr gute Leistung, aber auf den Validierungsdaten eine schlechtere Leistung. Dies zeigt sich in Lernkurven durch eine große Lücke zwischen den Scores (oder Fehlern) für die Trainings- und Validierungsdaten. Mit zunehmender Größe des Trainingssets kann die Leistung auf den Validierungsdaten verbessert werden, aber die Lücke bleibt bestehen.

Modell 1 – Random Forest- Best Params:
Man erkennt durch die Learning Curve, dass das Modell overfittet. Die Lücke zwischen Trainings und Cross Validation Score ist zu hoch. (Abbildung 10: Learning Curve RF Best Params)

Modell 2 – Random Forest- Best Params with Oversampling:
Es ist zu erkennen, dass am Anfang das Modell overfittet. Aber mit zunehmenden Daten wird die Lücke kleiner. Das ist normalerweise ein gutes Zeichen. Es bedeutet, dass das Modell von mehr Trainingsdaten profitieren kann. (Abbildung 11: Learning Curve Best Params with Oversampling)

Modell 3 – Random Forest- Best Params to prevent Overfitting:
Hier ist ein ähnliches Verhalten wie in Modell 2 zu erkennen, jedoch nimmt auch der Trainingsscore ab und nähert sich dem Cross-validation Score an. (Abbildung 12: Learning Curve Best Params to Prevent Overfitting)

7.3.2 Confusion Matrix

Im Folgenden werden Konfusionsmatrizen für die drei Varianten des Random Forest-Modells erstellt und vergliche. Eine Konfusionsmatrix ist eine spezielle Tabelle, die die Leistung eines Klassifikationsmodells darstellt. (Lemaître, Nogueira, & Aridas, 2017) Es zeigt die Anzahl der wahren positiven, wahren negativen, falsch positiven und falsch negativen Vorhersagen.

Zuerst werden die Vorhersagen der Modelle für den Testdatensatz berechnet. Anschließend werden die Konfusionsmatrizen erstellt, indem die tatsächlichen und vorhergesagten Klassen miteinander verglichen werden. (Abbildung 13: Konfusionsmatrix)

Modell 1 – Random Forest- Best Params:
Das Modell scheint eine gute Balance zwischen der korrekten Vorhersage positiver und negativer Instanzen zu haben. Es hat die geringste Anzahl an falsch negativen (FN) und eine relativ geringe Anzahl an falsch positiven (FP) Vorhersagen.

Modell 2 – Random Forest- Best Params with Oversampling:
Hat im Vergleich mehr wahre negative (TN) und weniger falsch positive (FP) Vorhersagen, aber auch mehr falsch negative (FN) und weniger wahre positive (TP) Vorhersagen.

Modell 3 – Random Forest- Best Params to prevent Overfitting:
Ergibt die höchste Anzahl an falsch positiven (FP) Vorhersagen und die geringste Anzahl an wahren negativen (TN) Vorhersagen, obwohl es eine ähnliche Anzahl an wahren positiven (TP) Vorhersagen hat wie die anderen Modelle.

7.3.3 Classification Reports

Der Klassifikationsbericht liefert eine Zusammenfassung der Leistungen der Klassifikationsmodelle. (Abbildung 14: Classification Reports)

Modell 1 – Random Forest- Best Params:
Das Modell hat eine hohe Genauigkeit (90%), eine hohe Precision (0.88 für Klasse 0 und 0.90 für Klasse 1), einen hohen Recall (0.80 für Klasse 0 und 0.94 für Klasse 1) und einen hohen F1-Score (0.84 für Klasse 0 und 0.92 für Klasse 1). Dies deutet darauf hin, dass das Modell gut bei der Vorhersage sowohl der Klasse 0 (keine Stornierung) als auch der Klasse 1 (Stornierung) abschneidet.

Modell 2 – Random Forest- Best Params with Oversampling:
Das Modell hat eine etwas geringere Genauigkeit (89%) und etwas geringere Werte für Precision, Recall und F1-Score im Vergleich zum Modell mit den besten Parametern. Dies könnte darauf hindeuten, dass das Oversampling zu einigen Fehlern in der Klassifikation geführt hat, obwohl die Unterschiede gering sind.

Modell 3 – Random Forest- Best Params to prevent Overfitting:
Dieses Modell hat die niedrigste Genauigkeit (86%) und die niedrigsten Werte für Precision, Recall und F1-Score. Insbesondere der Recall für Klasse 0 ist deutlich niedriger (0.70) im Vergleich zu den anderen Modellen, was darauf hindeuten könnte, dass dieses Modell Schwierigkeiten hat, die Klasse 0 korrekt zu identifizieren. Dies könnte darauf hinweisen, dass das Modell unter Umständen overfitting vermeidet, aber dabei Leistungseinbußen in Kauf nimmt.

7.4 Interpretation and Model Selection

Basierend auf den verschiedenen Metriken und Analysen, die durchgeführt wurden, scheint das Random Forest-Modell mit den besten Parametern und Oversampling insgesamt die beste Leistung zu erbringen ohne zu overfitten.

Die Lernkurve des Modells zeigt, dass die Lücke zwischen den Trainings- und Validierungsscores mit zunehmender Anzahl von Trainingsbeispielen abnimmt. Dies deutet darauf hin, dass das Modell nicht nur gut auf den Trainingsdaten performt, sondern auch effektiv auf neue, unvorhergesehene Daten generalisiert. Darüber hinaus ist die Genauigkeit des Modells auf den Validierungsdaten hoch, was darauf hindeutet, dass es in der Lage ist, sowohl positive als auch negative Klassen effektiv

zu unterscheiden. Die Konfusionsmatrix des Modells zeigt ebenfalls eine hohe Leistung. Es minimiert die Anzahl der falsch negativen (FN) und falsch positiven (FP) Vorhersagen. Der Klassifikationsbericht unterstreicht diese Beobachtungen. Das Modell hat eine hohe Genauigkeit sowie hohe Werte für Precision, Recall und F1-Score. Diese Metriken zeigen, dass das Modell eine hohe Leistung bei der Klassifizierung beider Klassen aufweist und dabei ein gutes Gleichgewicht zwischen Precision und Recall hält.

Insgesamt deutet die Analyse darauf hin, dass das Random Forest-Modell mit den besten Parametern und Oversampling die effektivste Wahl für StayAwhile ist. Dieses Modell zeigt eine hohe Leistung über eine Reihe von Metriken hinweg und scheint effektiv ein Gleichgewicht zwischen der Minimierung von Overfitting und der Maximierung der Vorhersagegenauigkeit zu erreichen. Obwohl das Modell mit dem besten Parameter in einigen Metriken besser abschneidet, ist es relevanter Overfitting zu minimieren und somit wird sich für **Modell 2** entschieden.

8. Deployement

Im folgenden Kapitel wird das Deployement kurz erklärt und ausgeführt, um das Model für StayAwhile bereitzustellen.

1. Modell speichern: Das Modell muss gespeichert werden um dieses außerhalb der Jupyter Notebook Umgebung zu nutzen, dafür wird folgender Code genutzt: „joblib.dump(clf_rf_adjusted_os, 'best_model.pkl')“. Das Modell wird also als „Pickle“ serialisiert, welches dann später wieder abgerufen werden kann.
2. Erstellen einer API: Als nächstes muss eine API erstellt werden, die Vorhersagen mit dem Modell machen kann. Das wird mit dem Flask Framework gemacht. Die API sollte in der Lage sein, Anfragen mit den notwendigen Eingabedaten zu empfangen, diese Daten zu verarbeiten, eine Vorhersage mit dem Modell zu machen und die Vorhersage als Antwort zurückzugeben.
3. API hosten: Die API muss auf einem Server gehostet werden, der dann genutzt werden kann. In dieser Arbeit wird der Server auf replit.com gehostet.
4. Bereitstellung und Nutzung: Sobald die API gehostet wurde, kann Sie von der Hotelkette genutzt werden, um Anfragen mit den notwendigen Eingabedaten zu senden und die API würde eine Vorhersage zurückgeben. Das kann direkt im Buchungstool erfolgen oder es kann eine POST-Funktion an /predict gesendet werden.
5. Im Rahmen dieser Arbeit wird ein Formular bereitgestellt welches, Daten Input liefert und dann an den Flask Server schickt, auf welchem das Modell ausgeführt wird. Das Formular kann unter: https://stayawhile.flitschi7.repl.co/ aufgerufen werden. Der Code für das Formular und den Flask Server ist im GitHub Repository unter Webtool zu finden.

Dadurch ist eine Integration in vorhandene Systeme Möglich.

9. Datenbasierte Services

Die Integration des entwickelten Vorhersagemodells für Hotelbuchungsstornierungen in das bestehende System eines Hotels ermöglicht die Generierung zweier wesentlicher datenbasierter Services.

1. automatisierte Zimmerfreigabe: Durch die Integration des Modells in das Hotelbuchungssystem können Hotelbetreiber ein System implementieren, das automatisch Zimmer freigibt, wenn eine Stornierung vorhergesagt wird. Dieser Service könnte erhebliche Auswirkungen auf die Effizienz der Zimmerverwaltung haben und das Risiko von Über- oder Unterbuchungen minimieren. Angesichts der Komplexität und Dynamik von Hotelbuchungen kann eine solche automatisierte Zimmerfreigabe den Hotelbetreibern dabei helfen, ihre Ressourcen optimal zu nutzen und gleichzeitig die Kundenzufriedenheit zu verbessern, indem sie die Verfügbarkeit von Zimmern maximiert.

2. personalisierte Angebote: Das Modell könnte genutzt werden, um personalisierte Angebote für Kunden zu erstellen, bei denen eine Stornierung vorhergesagt wird. Dieser Service könnte als eine Form der proaktiven Kundenbindung betrachtet werden. Wenn das Modell eine Stornierung vorhersagt, könnten automatische Benachrichtigungen an Kunden gesendet werden, die personalisierte Angebote oder Anreize enthalten, um die Buchung aufrechtzuerhalten. Konkrete Beispiele wären, eventuell Willkommens-Getränke zu schenken, oder Gutscheine für die Bar. Eventuell wären sogar Upgrades in bessere Zimmer möglich, wenn das Hotel ausreichend Kapazität hat. Solche personalisierten Angebote könnten auf den spezifischen Präferenzen und Anforderungen der Kunden basieren und dazu beitragen, ihre Zufriedenheit und Loyalität zu steigern. Darüber hinaus könnte dieser Service dazu beitragen, die Stornierungsrate zu senken und damit die Rentabilität des Hotels zu verbessern.

10. Relevante Effekte für Stay Awhile

Die Implementierung des entwickelten Vorhersagemodells für Hotelbuchungsstornierungen bietet StayAwhile wesentliche Vorteile und hat potenziell transformative Auswirkungen auf verschiedene Aspekte des Hotelbetriebs.

Effizienzsteigerung bei der Zimmerverwaltung: Durch die Vorhersage von Stornierungen und die automatisierte Freigabe von Zimmern kann StayAwhile die Effizienz seiner Zimmerverwaltung erheblich steigern. Dies minimiert das Risiko von Über- oder Unterbuchungen und ermöglicht es dem Hotel, seine Ressourcen optimal zu nutzen. Auf diese Weise kann StayAwhile einen höheren Umsatz erzielen und gleichzeitig unnötige Kosten vermeiden.

Verbesserung der Kundenzufriedenheit und -bindung: Der personalisierte Service, der durch das Modell ermöglicht wird, kann dazu beitragen, die Kundenzufriedenheit und -bindung zu verbessern. Indem StayAwhile personalisierte Angebote oder Anreize für Kunden bereitstellt, bei denen eine hohe Stornierungswahrscheinlichkeit vorliegt, kann das Hotel die Kundenerfahrung verbessern und die Kundenbindung erhöhen. Dies könnte dazu führen, dass mehr Kunden StayAwhile als ihr bevorzugtes Hotel wählen, was die Rentabilität des Hotels weiter steigern würde. Und generell weniger Stornierungen erfolgen.

Verbesserung der strategischen Entscheidungsfindung: Darüber hinaus kann das Modell wertvolle Erkenntnisse für die strategische Entscheidungsfindung liefern. Durch das Verständnis der Faktoren, die zu Stornierungen führen, kann StayAwhile seine Strategien und Praktiken anpassen, um die Kundenzufriedenheit zu verbessern und Stornierungen zu reduzieren. Dies könnte beispielsweise die Anpassung von Preisstrategien, die Verbesserung der Servicequalität oder die Einführung neuer Dienstleistungen beinhalten.

Literaturverzeichnis

Alotaibi, D. E. (2020). Application of Machine Learning in the Hotel Industry: A Critical Review. *Journal of Association of Arab Universities for Tourism and Hospitality*, 78-96.

Bonthu, H. (11. 07 2021). *Analytics Vidhya*. Abgerufen am 11. 07 2023 von https://www.analyticsvidhya.com/blog/2021/07/an-introduction-to-logistic-regression/

Chauhan, N. S. (09. 02 2022). *KD Nuggets*. Abgerufen am 11. 07 2023 von Decision Tree Algorithm, Explained: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

Devisme, B. (15. 03 2019). *Quicktext*. Abgerufen am 11. 07 2023 von https://www.quicktext.im/blog/impact-of-machine-learning-on-hotel-operations/

Donges, N. (14. 03 2023). *Builtin*. Abgerufen am 11. 07 2023 von Random Forest: A Complete Guide for Machine Learning: https://builtin.com/data-science/random-forest-algorithm

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research 18*, 1-5.

Otaris. (10. 07 2022). *Otaris*. Abgerufen am 11. 07 2023 von Data Science Beratung: https://www.otaris.de/beratung-data-science/

Pedregosa, F., & al., e. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2825-2830.

Raschka, S. (01. 10 2018). *STAT 479: Machine Learning Lecture Notes.* Abgerufen am 11. 07 2023 von https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf

Strike, S. -O. (17. 04 2020). *Why is multicollinearity not checked in modern statistics/machine learning*. Abgerufen am 29. 06 2023 von https://stats.stackexchange.com/q/168631

Trotta, F. (27. 06 2022). *How and Why Performing One-Hot Encoding in Your Data Science Project*. Abgerufen am 10. 06 2023 von https://towardsdatascience.com/how-and-why-performing-one-hot-encoding-in-your-data-science-project-a1500ec72d85

Verot, B. (11. 07 2023). *HotelMinder*. Abgerufen am 11. 07 2023 von https://www.hotelminder.com/everything-you-need-to-know-about-hotel-cancellations

Anhang

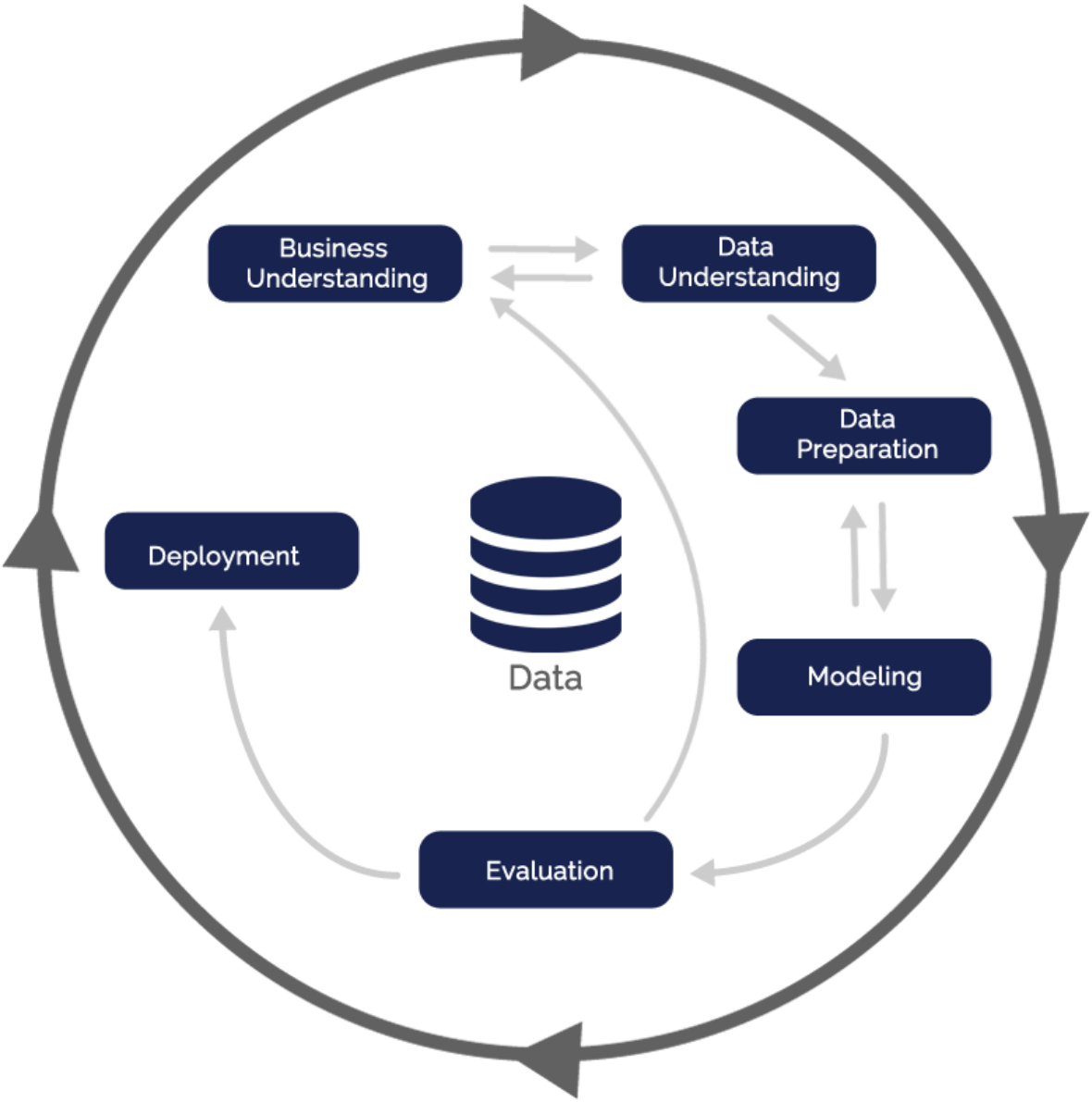

ABBILDUNG 1: CRISP DM CYCLE (OTARIS, 2022)

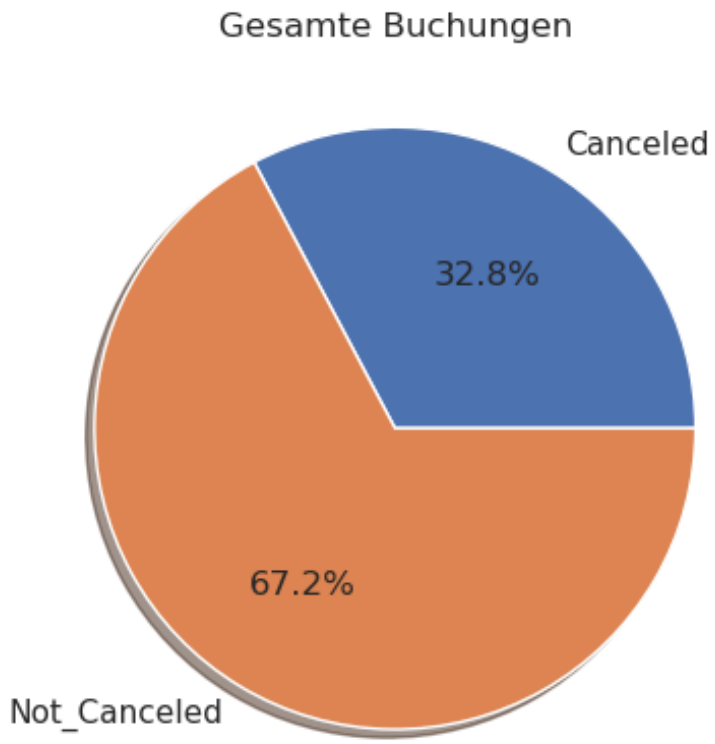

ABBILDUNG 2: VERTEILUNG ZIELVARIABLE

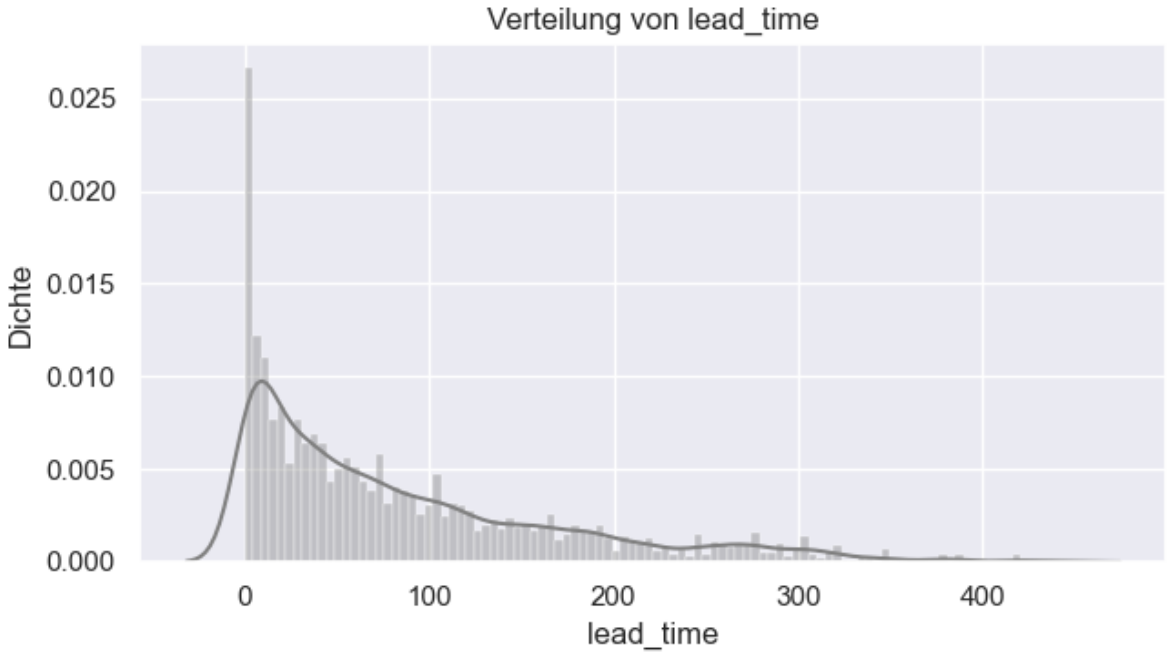

ABBILDUNG 3: HISTOGRAMM FÜR LEAD_TIME

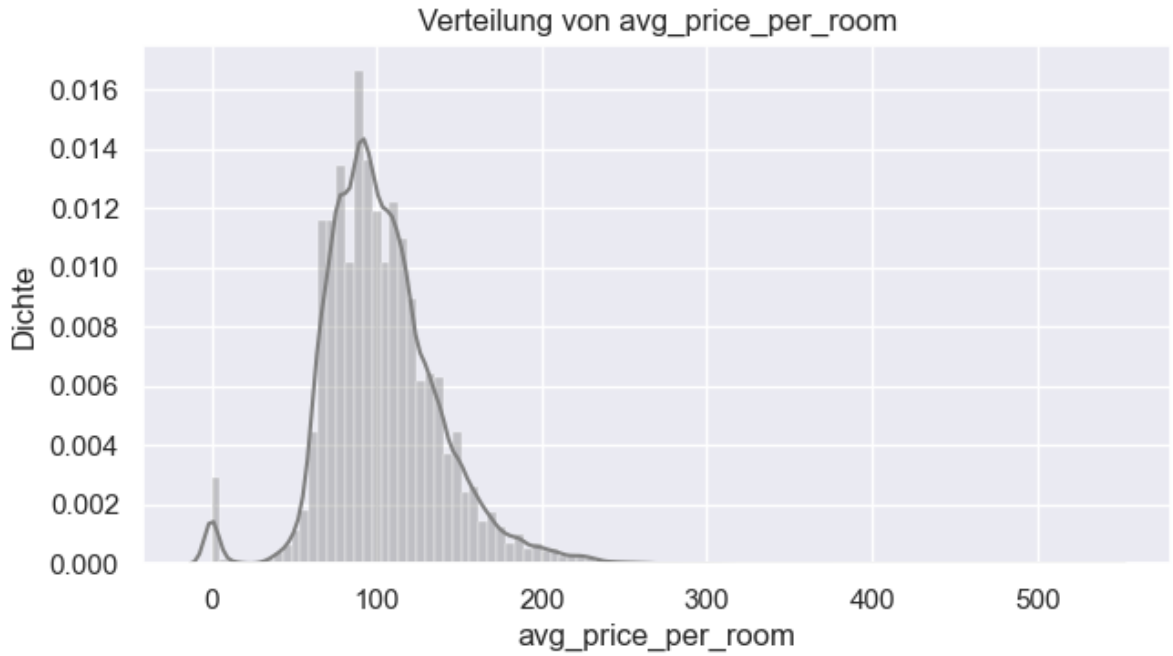

Abbildung 4: Histogramm für avg_price_per_room

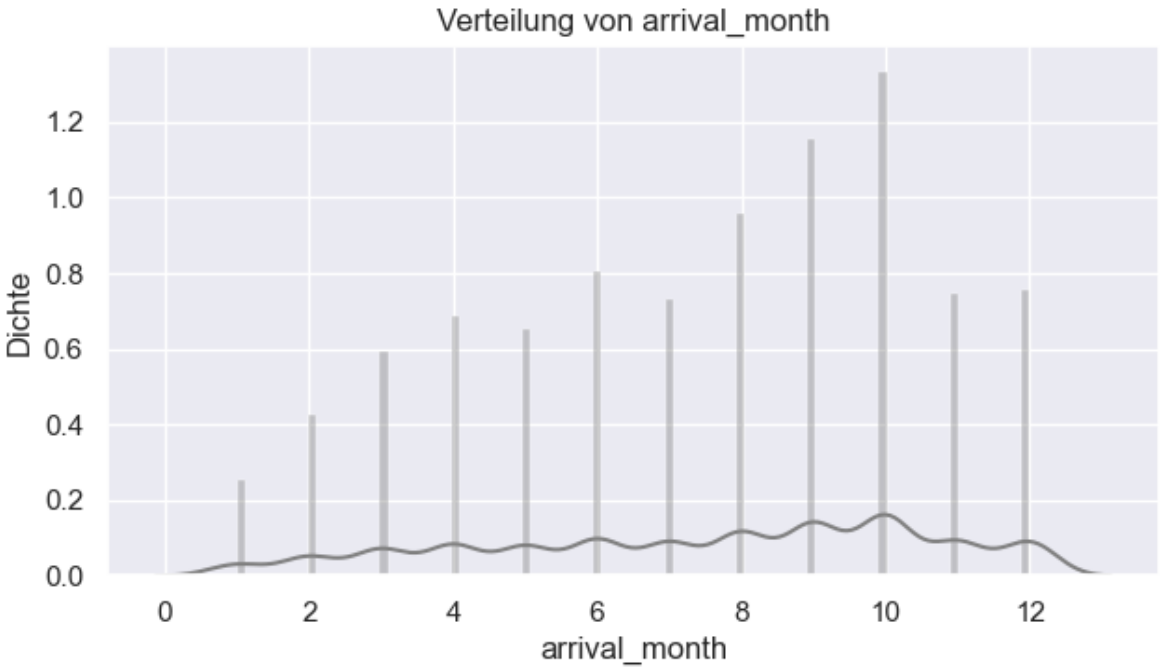

Abbildung 5: Histogramm für arrival_month

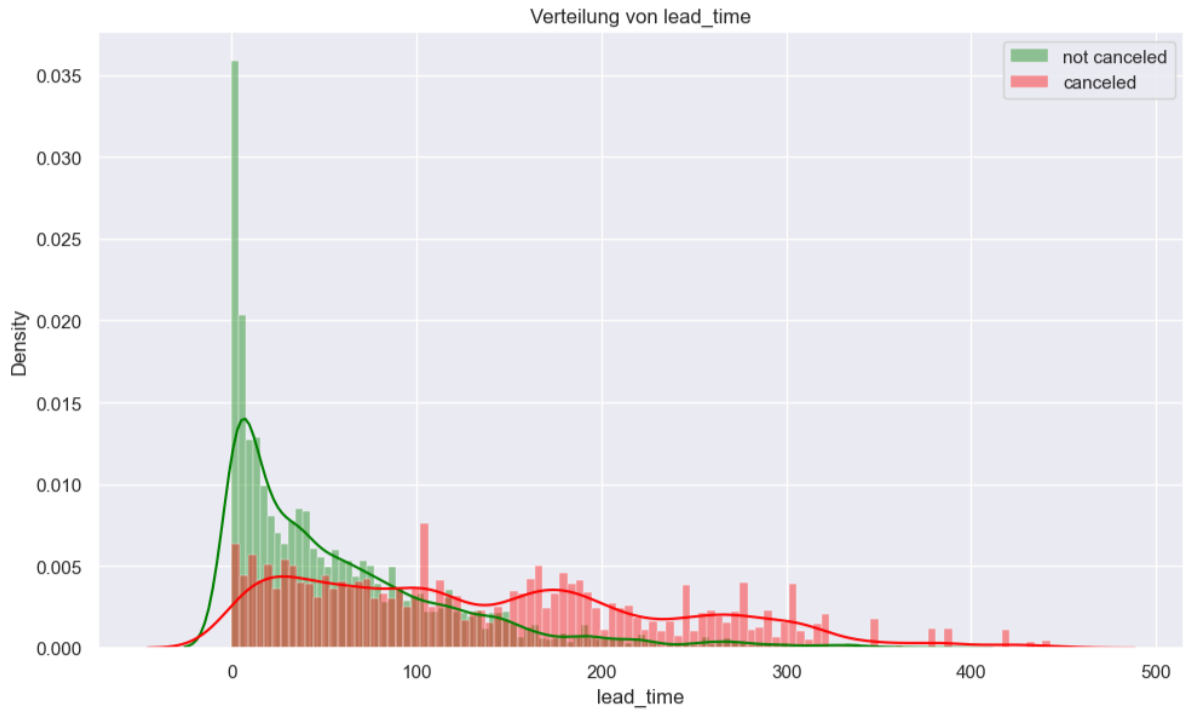

ABBILDUNG 6: HISTOGRAMM FÜR LEAD_TIME NACH ZIELVARIABLE

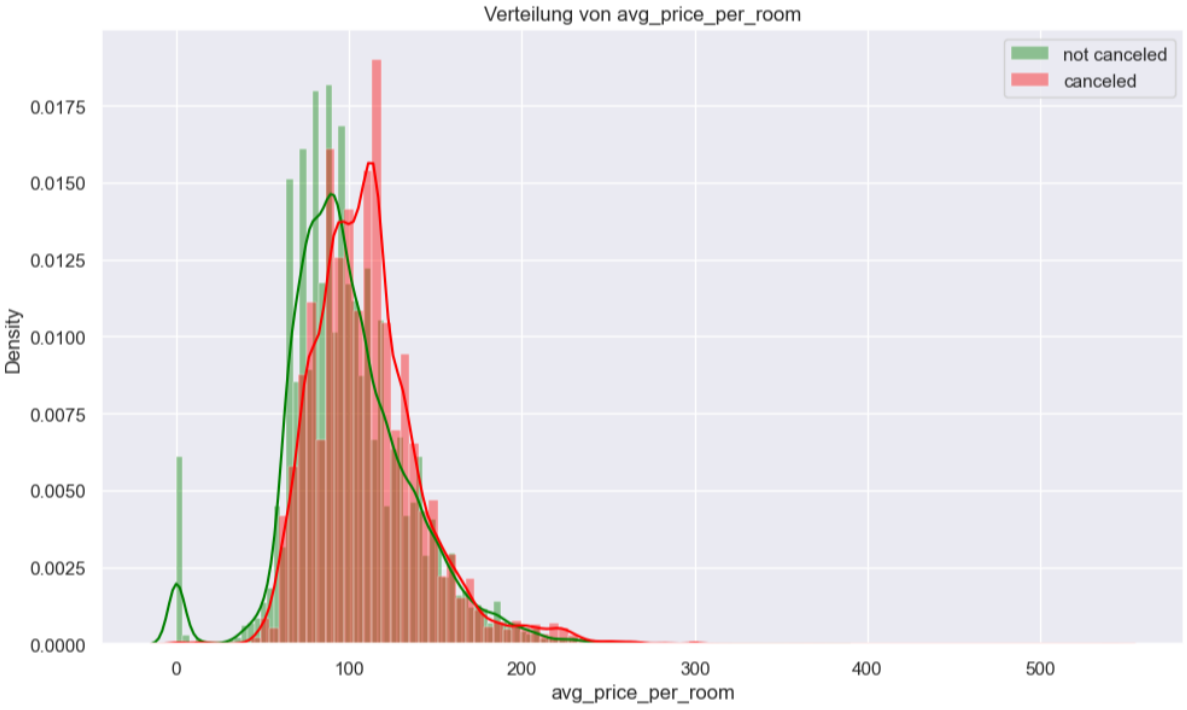

ABBILDUNG 7: HISTOGRAMM FÜR AVG_PRICE_PER_ROOM NACH ZIELVARIABLE

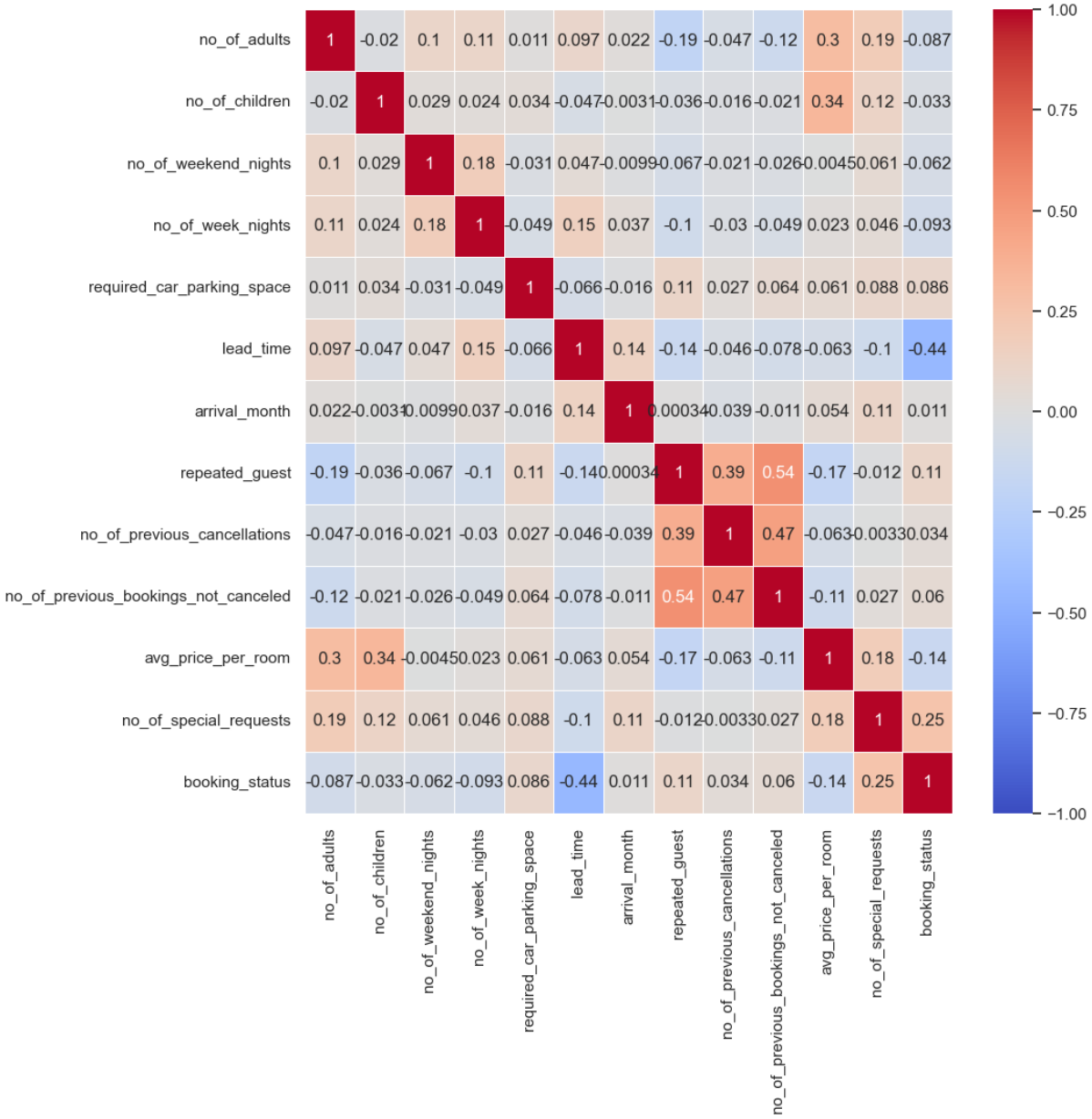

ABBILDUNG 8: KORRELATIONSMATRIX

```python
#Definieren der Parameter für den Random Forest
rf_param_grid = {
    'n_estimators': [100, 200, 300, 400, 500],
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
#intialisieren der RF Klassifikation
rf = RandomForestClassifier()

#Initialisieren des GridSearchCV Objekts
rf_grid_search = GridSearchCV(estimator=rf, param_grid=rf_param_grid, cv=3, n_jobs=-1, verbose=2)

#GridSearchCV Objekt auf die Daten fitten
rf_grid_search.fit(X_train, y_train)

#Ausgabe der besten Parameter
print('Best parameters for RandomForestClassifier: ', rf_grid_search.best_params_)

#Definieren der Parameter für den Decision Tree
dt_param_grid = {
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

#Initialisieren des Decision Tree Klassifikators
dt = DecisionTreeClassifier()

#Initialisieren des GridSearchCV Objekts
dt_grid_search = GridSearchCV(estimator=dt, param_grid=dt_param_grid, cv=3, n_jobs=-1, verbose=2)

#GridSearchCV Objekt auf die Daten fitten
dt_grid_search.fit(X_train, y_train)

#Ausgabe der besten Parameter
print('Best parameters for DecisionTreeClassifier: ', dt_grid_search.best_params_)
```

[34]

ABBILDUNG 9: GRIDSEARCHCV CODE

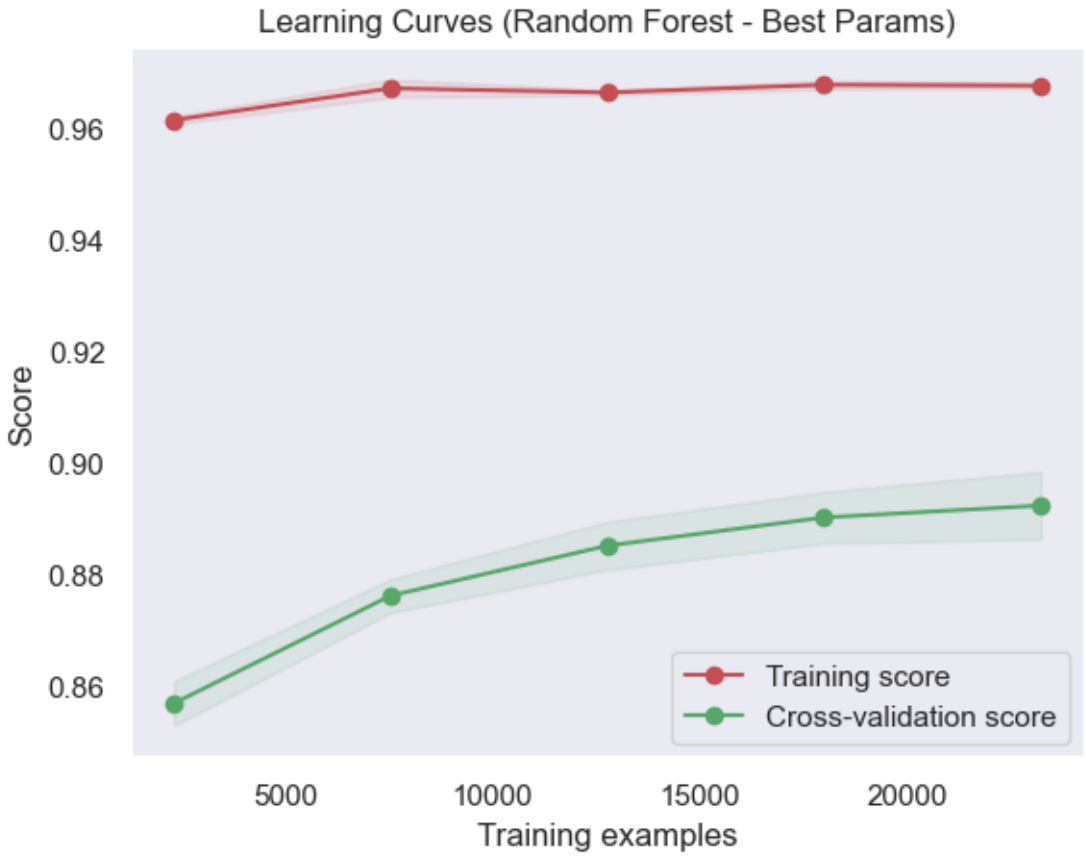

ABBILDUNG 10: LEARNING CURVE RF BEST PARAMS

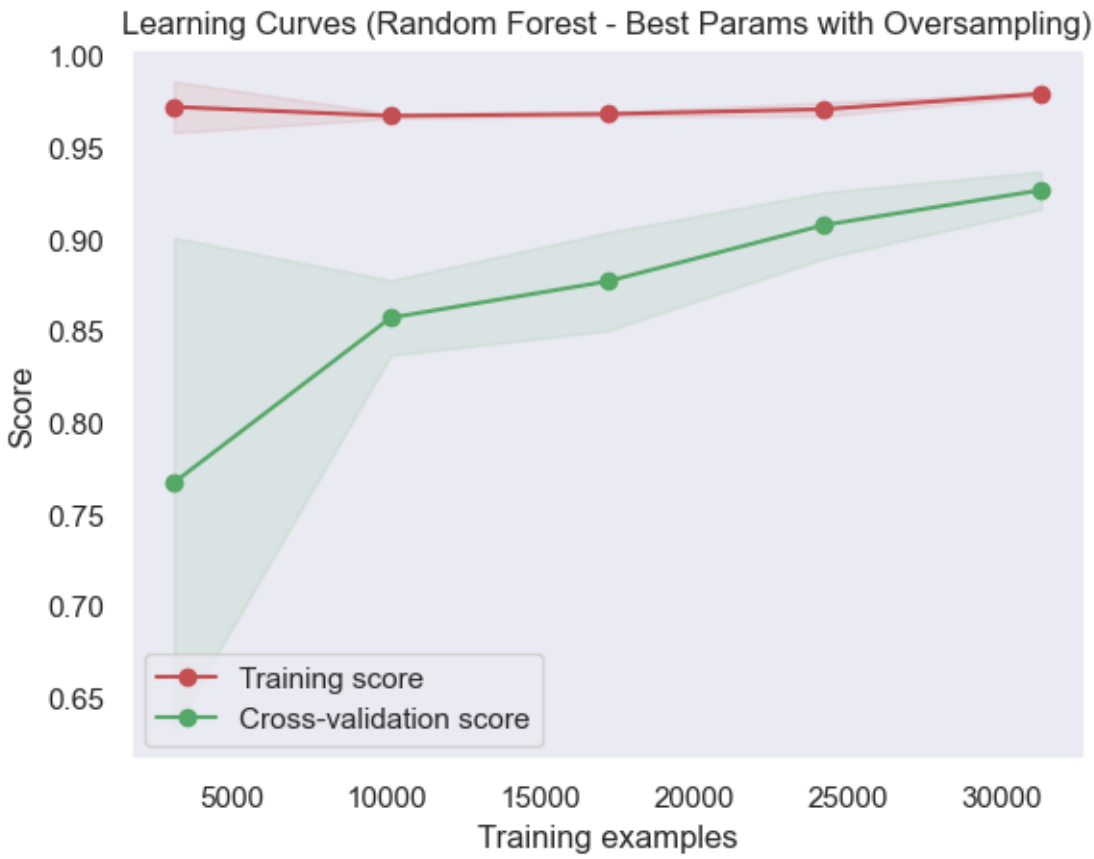

ABBILDUNG 11: LEARNING CURVE BEST PARAMS WITH OVERSAMPLING

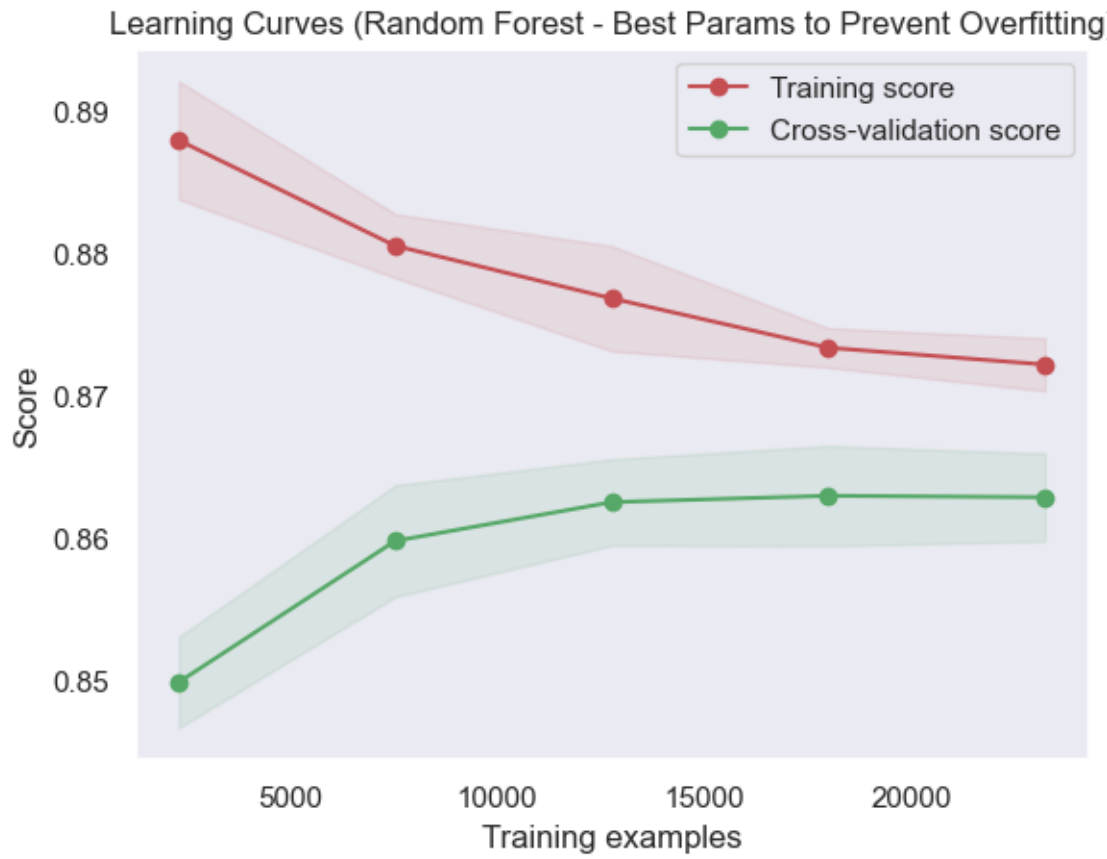

Abbildung 12: Learning Curve Best Params to Prevent Overfitting

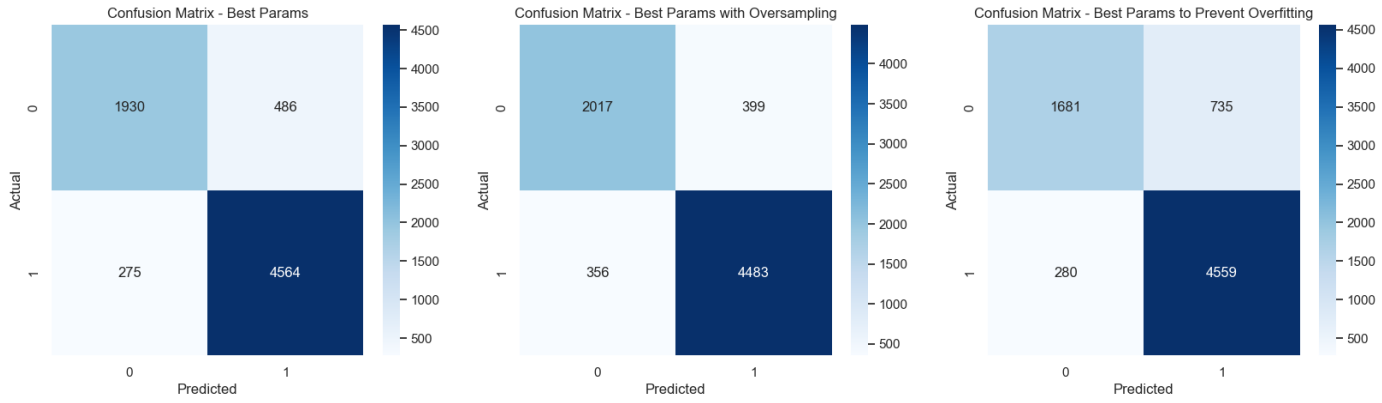

Abbildung 13: Konfusionsmatrix

```
Classification Report - Best Params:
              precision    recall  f1-score   support

           0       0.88      0.80      0.84      2416
           1       0.90      0.94      0.92      4839

    accuracy                           0.90      7255
   macro avg       0.89      0.87      0.88      7255
weighted avg       0.89      0.90      0.89      7255


Classification Report - Best Params with Oversampling:
              precision    recall  f1-score   support

           0       0.85      0.83      0.84      2416
           1       0.92      0.93      0.92      4839

    accuracy                           0.90      7255
   macro avg       0.88      0.88      0.88      7255
weighted avg       0.90      0.90      0.90      7255


Classification Report - Best Params to Prevent Overfitting:
              precision    recall  f1-score   support

           0       0.86      0.70      0.77      2416
           1       0.86      0.94      0.90      4839

    accuracy                           0.86      7255
   macro avg       0.86      0.82      0.83      7255
weighted avg       0.86      0.86      0.86      7255
```

ABBILDUNG 14: CLASSIFICATION REPORTS