

**Entwicklung eines Machine Learning Modells
ohne die Verwendung von Hotel spezifischen
Vergangenheitsdaten**

William Mendat

MASTERARBEIT

zur Erlangung des akademischen Grades Master of Science (M.Sc.)

Studiengang Informatik Master

Fakultät Elektrotechnik, Medizintechnik und Informatik
Hochschule für Technik, Wirtschaft und Medien Offenburg

30.03.2024

Durchgeführt bei der Firma happyhotel

Betreuer

Prof. Dr.-Ing. Janis Keuper, Hochschule Offenburg

Prof. Dr. rer. nat. Klaus Dorer, Hochschule Offenburg

Mendat, William:

Entwicklung eines Machine Learning Modells ohne die Verwendung von Hotel spezifischen Vergangenheitsdaten / William Mendat. –

MASTERARBEIT, Offenburg: Hochschule für Technik, Wirtschaft und Medien Offenburg, 2024. 61 Seiten.

Mendat, William:

Development of a machine learning model without using hotel-specific historical data / William Mendat. –

MASTER THESIS, Offenburg: Offenburg University, 2024. 61 pages.

Vorwort

Die Entwicklung eines Machine Learning Modells ohne die Verwendung von Vergangenheitsdaten ist ein umfangreiches und sehr interessantes Thema. Es ist ein sehr weitreichendes Thema, an dem sehr lange geforscht werden könnte und an sich auch eine Wissenschaft für sich ist. Nicht alle Facetten und Möglichkeiten, dieses doch recht komplexe Thema zu bewältigen, werden in der folgenden Arbeit dargestellt aber es wird Einblick gegeben, wie an dieses Thema herangegangen werden kann.

Ich möchte mich an dieser Stelle bei allen Personen bedanken, die mich während dieser Master-Thesis unterstützt haben. Ein besonderer Dank geht dabei an Prof. Dr.-Ing. Janis Keuper für die Betreuung während der Arbeit und die Möglichkeit, dieses Thema überhaupt bearbeiten zu können. Des Weiteren möchte ich mich bei Kai Schmidt und Marius Müller bedanken, die mich während der Bearbeitung tatkräftig unterstützt haben. Als letztes möchte ich mich bei dem Unternehmen happyhotel bedanken, dass sie mir ermöglicht haben, dieses interessante Thema zu bearbeiten.

Eidesstattliche Erklärung

Hiermit versichere ich eidesstattlich, dass die vorliegende Master-Thesis von mir selbstständig und ohne unerlaubte fremde Hilfe angefertigt worden ist, insbesondere, dass ich alle Stellen, die wörtlich oder annähernd wörtlich oder dem Gedanken nach aus Veröffentlichungen, unveröffentlichten Unterlagen und Gesprächen entnommen worden sind, als solche an den entsprechenden Stellen innerhalb der Arbeit durch Zitate kenntlich gemacht habe, wobei in den Zitaten jeweils der Umfang der entnommenen Originalzitate kenntlich gemacht wurde. Ich bin mir bewusst, dass eine falsche Versicherung rechtliche Folgen haben wird.

Offenburg, 30.03.2024

William Mendat

Sperrvermerk

Die vorliegende Abschlussarbeit beinhaltet vertrauliche Informationen und interne Daten des Unternehmens happyhotel. Sie darf aus diesem Grund nur zu Prüfzwecken verwendet und ohne ausdrückliche Genehmigung durch die happyhotel weder Dritten zugänglich gemacht, noch ganz oder in Auszügen veröffentlicht werden. Die Sperrfrist endet 5 Jahre nach dem Einreichen der Arbeit bei der Hochschule Offenburg. Unbeschadet hiervon bleibt die Weitergabe der Arbeit und Einsicht in die Arbeit an die mit der Prüfung befassten Mitarbeiter der Hochschule und Prüfer möglich, die ihrerseits zur Geheimhaltung verpflichtet sind, sowie die Verwendung der Arbeit in eventuellen prüfungsrechtlichen Rechtsschutzverfahren nach Maßgabe der geltenden verwaltungsprozessualen Regeln.

Zusammenfassung

Entwicklung eines Machine Learning Modells ohne die Verwendung von Hotel spezifischen Vergangenheitsdaten

In dieser Thesis wird ein Machine-Learning-Modell entwickelt, das gezielt darauf ausgelegt ist, Preisvorschläge ohne spezifische historische Daten zu generieren. Es werden verschiedene Ansätze vorgestellt, wobei einer davon detailliert untersucht wird. Anschließend erfolgt eine umfassende Evaluierung dieses Ansatzes, um die Performance zu überprüfen.

Die Arbeit bietet einen Einblick in die Problematik der Datenwissenschaft, wenn keine historischen Daten verfügbar sind, und zeigt auf, wie dieser Herausforderung begegnet werden kann.

Abstract

Development of a machine learning model without using hotel-specific historical data

In this thesis, a machine learning model is developed specifically designed to generate price proposals without specific historical data. Various approaches are introduced, with one of them being elaborated upon in detail. Subsequently, a thorough evaluation of this approach is conducted to assess its performance.

This work provides insight into the data science challenge when historical data is not available and demonstrates how to address this issue effectively.

Inhaltsverzeichnis

1 Einleitung	1
1.1 happyhotel	2
1.1.1 Das Unternehmen	3
1.1.2 Dynamische Preisgenerierung	3
1.1.3 Momentane Problematik	4
1.2 Vorgehensweise	5
1.2.1 Vorgehen in der Datenwissenschaft	5
1.2.2 Benchmark Hotels	7
2 Konzepte	9
2.1 Hotel Daten von vielen Hotels	9
2.2 Mitbewerber Modell	10
2.3 Ähnliche Hotels	12
2.4 Synthetische Daten erstellen	13
2.5 Evaluation	13
3 Ähnliche Hotels	15
3.1 Finden von ähnlichen Hotels	15
3.1.1 Datenbeschaffung	16
3.1.2 Datenvorverarbeitung	19
3.1.3 Allgemeine Datenanalyse	20
3.1.4 Evaluation der ähnlichen Hotels	29
3.1.5 Unbewachtes Lernen	33
3.1.6 Überwachtes Lernen	45
3.1.7 Finden von ähnlichen Hotels Fazit	49
3.2 Modifiziertes RevPAR-Modell	49
3.2.1 Datenvorverarbeitung	50
3.2.2 Datenanalyse	52
3.2.3 Modellbildung	55
3.2.4 Evaluation	56
3.2.5 Zwischenfazit	58
3.2.6 Dynamische Preisgenerierung	58
4 Fazit	60

5 Ausblick	61
Tabellenverzeichnis	
Abbildungsverzeichnis	i
Quellcodeverzeichnis	iii
Literatur	iv

1. Einleitung

In einer Welt, die sich mit rasanter Geschwindigkeit digitalisiert, suchen die Menschen stets nach Wegen, um die Komplexität des modernen Lebens zu bewältigen. Diese Digitalisierung hat eine stetig wachsende Sehnsucht nach der Vorhersage zukünftiger Ereignisse hervorgebracht - sei es in der Wirtschaft, der Gesundheitsbranche oder auch im Bereich des Dienstleistungssektors wie dem Hotelgewerbe. Es ist ein Streben nach Präzision, ein Bestreben, aus Daten und Mustern eine Art Kristallkugel zu formen, um die Zukunft vorhersagen zu können.

Albert Einstein hat es sehr treffend formuliert:

If you want to know the future, look at the past. [1]

Dieser Gedanke illustriert die gängige Annahme, dass die Vergangenheit Hinweise auf die Zukunft liefern kann. Es ist interessant anzumerken, dass dieses Zitat auch als Titel eines Buches von Einstein dient, welches seine philosophischen Ansichten zur Zeit, Raum und Vorhersage behandelt.

Doch was passiert, wenn diese Vergangenheitsdaten nicht verfügbar sind oder nicht genutzt werden können? In Branchen wie der Hotelindustrie, die oft noch auf traditionelle, statische Preisstrategien zurückgreifen, stellt sich die Frage, wie eine effektive Vorhersage ohne spezifische historische Daten möglich ist.

Es wird immer deutlicher, dass ein dynamischer Ansatz im Hotelwesen erforderlich ist, um die Umsatzaufschlüsselung durch Revenue Management zu steigern. Dies erfordert die Anpassung von Preismodellen an sich ändernde Nachfrage und andere Einflussfaktoren. Eine mögliche Lösung liegt in der Verlagerung der traditionellen Rolle des Revenue Managements auf Modelle, die auf breiteren Datenquellen und fortgeschrittenen Methoden des maschinellen Lernens basieren.

Die Suche nach einem solchen Modell, das ohne die spezifischen Vergangenheitsdaten eines bestimmten Hotels auskommt, bildet das Herzstück dieser Forschungsarbeit. Der Fokus liegt darauf, alternative Datenquellen zu erkunden und innovative Ansätze zu entwickeln, um Prognosen und Entscheidungsgrundlagen für das Revenue Management in der Hotellerie zu schaffen. Ziel ist es, dass diese nicht ausschließlich auf vergangenen Daten eines spezifischen Hotels basieren, sondern auf einer Vielzahl von allgemeinen, zugänglichen Informationen und fortschrittlichen Analysemethoden beruhen. Es geht darum, einen Weg zu finden, wie Hotels, selbst ohne ihre spezifischen vergangenen Daten, zukünftige Entscheidungen im Bereich des Revenue Managements treffen können, um ihre Leistung zu optimieren und ihre Wettbewerbsfähigkeit zu stärken.

1.1 happyhotel

Die vorliegende Masterthesis fängt mit einem umfassenden Überblick über die Firma happyhotel an. In diesem ersten Kapitel wird eingehend auf die grundlegende Idee und das herausragende Produkt von happyhotel eingegangen, welches einen signifikanten Beitrag zur Weiterentwicklung der Hotelbranche leistet.

Im Anschluss wird der Fokus auf das gegenwärtige Vorgehen des Unternehmens, der dynamischen Preisgenerierung, gelegt. Diese fortschrittliche Methode, die auf einer Künstlichen Intelligenz und umfassender Datenanalyse basiert, ermöglicht es happyhotel, in Echtzeit auf den momentanen Markt zu reagieren und optimale Preise für Hotels zu generieren.

Abschließend wird in diesem einführenden Kapitel die zugrunde liegende Problematik hervorgehoben, die den Ausgangspunkt dieser Arbeit bildet. Dabei richtet sich der Fokus auf die daraus resultierenden Herausforderungen, die mit der dynamischen Preisgenerierung einhergehen. Diese Problematik dient als Basis für die nachfolgende Analyse und Forschung, die darauf abzielt, innovative Lösungsansätze und Optimierungen im Rahmen der Preisstrategie von happyhotel zu entwickeln.

1.1.1 Das Unternehmen

Die Firma happyhotel wurde im Jahr 2019 von den drei Gründern Sebastian Kuhnhardt, Marius Müller und Rafael Weißmüller gegründet. Sie wollten wie der Name schon vermuten lässt, Hotels glücklicher machen. Angefangen hat es mit der Erkenntnis von Sebastian, dass sich viele Hoteliers nicht mit der dynamischen Preisgestaltung beschäftigen. Meist vertrauen diese Hoteliers einfach auf ihr Bauchgefühl, welcher Preis zur momentanen Situation passen könnte oder passen ihre Preise gar nicht an.

Somit stellten sie sich folgende Fragen:

- Wie können die Preise für die Übernachtung in einem Zimmer besser vorausgesagt werden?
- Wonach sollten sich die Preise richten und wie kann man sie bestimmen?

Eine Lösung musste her um mehr Dynamik in die Preisgestaltung zu bringen. Sie erschufen die Software happyhotel ein sogenanntes Revenue Management System. Mit happyhotel kann der Hotelier sein gesamtes Hotel analysieren und es werden passgenaue Preisvorschläge für seine Zimmerkategorien generiert, um mehr Umsatz zu erzeugen.



Abbildung 1.1: happyhotel

1.1.2 Dynamische Preisgenerierung

Wie im vorherigen Kapitel erwähnt, fokussiert sich happyhotel auf die dynamische Preisgenerierung. Um diese Preise zu generieren, braucht es vor allem zwei Sachen:

- Die Daten des Hotels, wie zum Beispiel Buchungen
- Ein Vorgehen, um aus den gesammelten Daten Preise zu generieren

Die Daten bekommen sie aus den verschiedensten Quellen. Sogenannte Property Management Systeme kurz *PMS*, sind Systeme, um ein Hotel zu verwalten. In diesen Property Management Systemen können Hotels zum Beispiel ihre Zimmer ver-

walten oder aber auch Buchungen anlegen und pflegen. Mit den Herstellern dieser Property Management Systeme arbeitet happyhotel zusammen, um an die Daten des Hotels zu gelangen.

Wenn Daten vorhanden sind, braucht es ein Vorgehen, um aus den Daten einen Preis zu generieren. Dazu ist happyhotel auf die folgenden zwei Ideen gekommen:

- Buchungskurvenmodell
- Kombination aus RevPAR und Buchungskurvenmodell

Das Buchungskurvenmodell war die erste Idee von happyhotel. Bei dem Buchungskurvenmodell wird die Vergangenheit angeschaut, um das zukünftige Buchungsverhalten vorherzusagen. Ziel dabei ist es, die Auslastung für einen Tag vorherzusagen, um anhand dessen einen akkuraten Preis zu bestimmen.

Da sich bei dem Buchungskurvenmodell sich einige Schwächen aufgezeigt haben, wurde ein neues Modell erschaffen, um den Schwachstellen entgegen zu wirken. Es sollte nun der RevPAR-Wert vorhergesagt werden und mit dem Buchungskurvenmodell angepasst werden. RevPAR steht dabei für Revenue per Available Room. Auch bei diesem Modell wird sich die Buchungshistorie des Hotels angeschaut, um dann den zukünftigen Umsatz pro verfügbarem Zimmer vorherzusagen, und darauf basierend den endgültigen Preis zu ermitteln.

1.1.3 Momentane Problematik

Wie es Albert Einstein treffend formulierte: Soll die Zukunft vorhergesagt gesagt werden, so sollte sich die Vergangenheit angeschaut werden. Auf diesem Grundprinzip ist happyhotel auch vorgegangen, sie schauen sich bei beiden Ansätzen die Vergangenheit an, um dann Prognosen über die Zukunft zu generieren.

Doch was passiert, wenn die Daten der Vergangenheit nicht vorhanden sind? Dies kann zum Beispiel passieren, wenn ein Hotel erst in der Zukunft öffnet und noch über keinerlei Daten verfügt. Auch dieses Hotel soll mit adäquaten Preisvorschlägen gefüttert werden. Die soeben beschriebene Situation ist ein generelles Problem im Machine Learning Bereich. Es kommt nicht allzu selten vor, dass keine Vergangenheitsdaten aus den unterschiedlichsten Gründen vorliegen. Dieser Problematik soll in dieser Arbeit auf dem Grund gegangen werden.

Ziel dieser Arbeit ist es, ein Modell zu entwickeln, welches Preisempfehlungen für ein Hotel liefert, für das es bisher noch keine Vergangenheitsdaten gibt. Dieses Modell soll dann für folgende zwei Szenarien genutzt werden können:

- Neue happyhotel Kunden ohne Daten
- Nachfrageeinschätzung für bestimmte Märkte

1.2 Vorgehensweise

Vor dem Eintauchen in die Lösung eines Problems ist es entscheidend, einen klaren Weg dorthin festzulegen. Antoine de Saint-Exupéry hat mit den Worten *Ein Ziel ohne Plan ist nur ein Wunsch* treffend darauf hingewiesen, dass ein bloßes Ziel ohne einen durchdachten Plan lediglich eine vage Vorstellung bleibt. Das Verständnis und die Festlegung einer angemessenen Vorgehensweise sind daher der Schlüssel, um ein Problem effektiv anzugehen. Aufgrund dessen wird in dem folgenden Abschnitt dieser Arbeit auf die Vorgehensweise eingegangen. Zudem werden innerhalb dieser Sektion die Benchmark-Hotels ermittelt, an welchen getestet werden kann, ob die Vorgehensweise ein Erfolg war.

1.2.1 Vorgehen in der Datenwissenschaft

Ein Projekt, welches in der Datenwissenschaft (engl. Data Science) angesiedelt ist, beginnt in der Regel mit einem geschäftlichen Problem, so wie es auch in dieser Thesis der Fall ist. Sobald das Problem klar definiert ist, sollten ein oder mehrere Konzepte ausgearbeitet werden, wie das Problem gelöst werden kann. Diese Konzepte sollen als Leitfaden dienen, um das angestrebte Ziel zu erreichen. In der Regel ist es ratsam, mehr als ein Konzept auszuarbeiten, da so Ausweichmöglichkeiten festgelegt werden können, sollte ein Konzept nicht funktionieren. So werden auch in dieser Arbeit, in dem Kapitel *Konzepte*, für das vorliegende Problem Konzepte ausgearbeitet und evaluiert.

Mit der klaren Definition des Problems und der darauffolgenden Konzeption, kann der Datenwissenschaftler mit Hilfe des *OSEMN*-Vorgangs die Problematik angehen [2].

Der *OSEMN*-Vorgang besteht aus den folgenden Elementen:

- Obtian data (Erhalten von Daten)
- Scrub data (Daten reinigen)
- Explore data (Untersuchen von Daten)
- Model data (Modelldaten)
- Interpret results (Interpretieren von Ergebnissen)

Obtian data

Die wichtigste Ressource des 21. Jahrhundert besteht in den Daten, die zur Verfügung stehen. Dies erläuterte Klaus Schwab, der Gründer des Weltwirtschaftsforums, mit seinen Worten: *Die wertvollste Ressource des 21. Jahrhunderts sind nicht mehr Öl, sondern Daten.* Aufgrund dessen besteht der erste Schritt, nach der Evaluierung der Konzepte, in der Beschaffung der Daten. Es muss zunächst ein Überblick geschaffen werden. Zum Überblick gehören Informationen wie:

- Welche Daten bereits vorhanden sind.
- Welche Daten eventuell noch intern neu erworben werden müssen.
- Welche Daten aus dem Internet gezogen werden können

Sobald die benötigten Daten vorliegen, kann mit dem nächsten Schritt fortgefahren werden.

Scrub data

Der nächste Schritt besteht darin, die Daten zu bereinigen. Zum Bereinigen der Daten gehört der Vorgang, mit dem die Daten in ein standardisiertes Format gebracht werden. Dazu gehört der Umgang mit fehlenden Daten, sowie die Korrektur von Fehlern und das Entfernen von sogenannten *outlier* [2]. Outlier sind Daten, welche im Verhältnis zu der gesamten Datenmenge aus der Reihe tanzen.

Explore data

Die Datenuntersuchung oder auch Datenanalyse dient dazu, um mit den Daten vertraut zu werden und ein besseres Verständnis für die Daten zu gewinnen. Dies ist ein sehr wichtiger Schritt bei einem *Data Science* Projekt, da nur dann ein gutes Ergebnis erzielt werden kann, wenn die Daten, mit den gearbeitet werden kann, verstanden sind. Das Verständnis über die Daten trägt zudem auch maßgeblich dazu bei, den richtigen Ansatz für die Modellierung zu finden.

Model

Nach dem Erforschen der Daten, kann ein *Machine Learning* Modell eingesetzt werden. Es existieren viele verschiedene Modelle, die meist für einen speziellen Fall implementiert worden sind. Die Auswahl des Modells hängt ganz davon ab, welche Art von Problem gelöst werden soll und welche Daten vorhanden sind, um dieses Problem zu lösen.

Interpret results

Der letzte Schritt besteht darin, die gesammelten Ergebnisse zu interpretieren. Dazu gehört die Entscheidung, ob die erzielten Ergebnisse gut oder schlecht sind. Meistens werden zur Hilfe der Entscheidung, ob die Ergebnisse gut oder schlecht sind, Diagramme, Grafiken und Tabellen erstellt. Es soll hier zudem auch entschieden werden, ob die Ergebnisse verwendet angewendet oder noch weiter geforscht werden muss. So entsteht ein Kreislauf, der mit dem Erstellen weiterer Konzepte startet und schlussendlich mit dem Interpretieren neuer Ergebnisse endet.

1.2.2 Benchmark Hotels

Noch vor der Konzeptionierung zur eigentlichen Lösung der Problematik, sollten schon vorhandene Hotels in unserer Datenbank als *Benchmark-Hotels* ausgesucht werden. Die Idee dahinter ist es, Hotels auszusuchen, bei denen viele Daten vorhanden sind und so zu tun, als wären gar keine Daten von diesen Hotels vorhanden. Mithilfe dieser Hotels sollen dann die ausgearbeiteten Konzepte im Kapitel *Konzepte* validiert werden. Ein Konzept wird dann als gut empfunden, wenn es in der

Lage ist, gute Preisvorschläge für die Benchmark-Hotels zu erzeugen.

Da nicht jedes Hotel, welches in der Datenbank vorhanden ist, in Frage kommt, wurden einige Kriterien aufgestellt, die ein Hotel erfüllen muss, um als ein Benchmark-Hotel gelten zu können. Diese Kriterien sehen wie folgt aus:

- Haben Daten von mehr als zwei Jahren
- Haben einen Benutzer mit der Role *Revenue Manager*
- Haben oft Preise geändert
- Haben oft vorgeschlagene Preise von happyhotel nicht angenommen

Die Begründung, warum die letzten drei Kriterien hinzu kamen, liegt darin, dass diese Hotels mit den Preisvorschlägen von happyhotel nicht zufrieden sind und vermutlich auch nur manuell *Revenue Management* betreiben. Die Hoffnung besteht darin, dass das Konzept nicht nur für Hotels ohne Vergangenheitsdaten gute Preisvorschläge liefert, sondern dass das Konzept auch verwendet werden kann, als Alternative zu den vorhandenen zwei Modellen, für Hotels, die noch manuell Dynamische Preise gestalten.

Es wurde zur Analyse der Hotels eine *json*-Datei erstellt. Diese *json*-Datei besteht aus einer Liste von einzelnen Objekten. Jedes Objekt innerhalb der Liste repräsentiert ein Hotel in der Datenbank. Ein Beispiel für diese *json*-Datei könnte wie folgt aussehen:

```
1  [
2  {
3      company_id: "11111111111111",
4      not_accepted_recs: 10,
5      price_change_average: 5
6  }
7  ]
```

Listing 1.1: Beispielhafte json-Datei

Das Feld *not_accepted_recs* beschreibt dabei, wie viele Preisvorschläge das Hotel nicht angenommen hat, beziehungsweise ignoriert oder abgelehnt hat, und das Feld *price_change_average* ist die durchschnittliche Preisänderung pro Tag. Anhand von diesen Informationen konnten zwei Hotels als Benchmark-Hotels ausgesucht werden.

2. Konzepte

Wie bereits in dem vorangegangenen Kapitel *Vorgehensweise* hervorgehoben wurde, bildet die Entwicklung und Ausarbeitung von Konzepten einen essenziellen Eckpfeiler bei der Bewältigung und Lösungsfindung für komplexe Probleme. Im nachfolgenden Abschnitt wird eine ausführliche Vorstellung und Evaluation der erarbeiteten Konzepte präsentiert. Dabei liegt der Fokus darauf zu ergründen, welche Konzepte das Potenzial besitzen, weiterverfolgt zu werden, um die spezifische Herausforderung zu meistern. Besondere Aufmerksamkeit gilt hierbei der grundlegenden Idee jedes Konzeptes sowie der detaillierten Darlegung, wie jedes einzelne Konzept dazu beitragen kann, dynamische Preisgestaltung für ein Hotel auch ohne vorhandene Daten zu generieren

Ziel dieses Kapitels ist es einen umfassenden Überblick über die Konzepte zu geben, ihre Relevanz für die Forschung zu betonen und den Weg für die darauffolgenden Analysen und Schlussfolgerungen zu ebnen.

2.1 Hotel Daten von vielen Hotels

Das Konzept *Hotel Daten von vielen Hotels* verfolgt die elementare Idee, sämtliche bis dato gesammelten Hoteldaten zu konsolidieren und ein umfassendes, übergeordnetes Modell des maschinellen Lernens zu entwickeln.

Die bereits vorhandenen Daten aus zahlreichen Hotels bilden die Basis für den Aufbau eines solchen Modells. Dieses Vorhaben sieht vor, das bereits existierende RevPAR-Modell zu modifizieren und zu erweitern. Zunächst wird angestrebt, alle Hotels in eine vergleichbare Form zu bringen. Eine mögliche Herangehensweise hierbei ist die Definition bestimmter Hotelmerkmale und ihre Auflistung als Vektor,

um eine Vergleichbarkeit zu ermöglichen.

Im nächsten Schritt ist eine Anpassung des RevPAR-Modells erforderlich, da dieses normalerweise auf Buchungsdaten basiert. Für Hotels ohne historische Buchungsdaten ist es offensichtlich nicht möglich, diese als Features zu verwenden, da sie schlichtweg nicht verfügbar sind. Stattdessen sollen die charakteristischen Merkmale jedes Hotels dem jeweiligen RevPAR zugeordnet werden.

Sobald das RevPAR-Modell entsprechend umstrukturiert ist, können sämtliche Hotels in der Datenbank als Datensätze dem Modell zugeführt werden. Falls ein Hotel ohne vergangene Buchungsdaten auftaucht, können basierend auf seinen charakteristischen Eigenschaften Prognosen über den zu erwartenden RevPAR getroffen werden. In diesem Szenario muss das Hotel lediglich, wie alle anderen Hotels auch, eine Zuordnung zwischen dem RevPAR und dem tatsächlichen Preis festlegen.

Die Aufmachung dieses Konzeptes soll im folgenden Schaubild nochmal bildlich verdeutlicht werden:

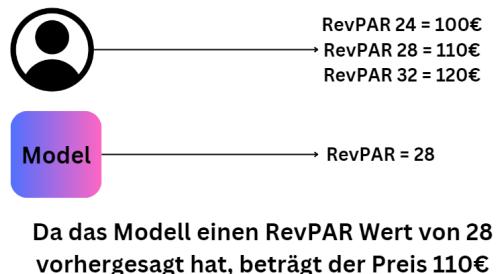


Abbildung 2.1: RevPAR-Modell Vorgehen

Dieser Ansatz zielt darauf ab, eine umfassende Verwendung der vorhandenen Daten zu ermöglichen und somit auch für Hotels ohne historische Buchungsdaten eine Prognose des RevPAR auf der Grundlage ihrer individuellen Eigenschaften zu ermöglichen.

2.2 Mitbewerber Modell

Die entscheidende Idee des Konzeptes: das Mitbewerber Modell ist es, die Daten von der Konkurrenz zu benutzen um daraufhin Preisvorschläge zu generieren.

Durch Drittanbieter wie *HQ-Revenue* können Konkurrenzdaten genutzt werden, um ein Modell aufzubauen. HQ-Revenue ist beispielsweise ein Anbieter, welcher Internetseiten wie Bookings.com oder trivago *scraped*, um an Hotelpreise oder andere Daten zu kommen. Dabei gibt es viele verschiedene Herangehensweisen, um Preise für ein Hotel ohne Vergangenheitsdaten zu entwickeln. Ein primitiver Ansatz dabei wäre es, wenn alle Preise von den Konkurrenten genommen werden und damit der Durchschnitt ermittelt wird. Dies hat natürlich nichts mit Machine Learning oder geschweige denn Data Science zu tun, war aber bislang die gängige Handhabe.

Dieser Ansatz birgt jedoch eine Problematik: nicht jeder Kunde von happyhotel hat auch automatisch Konkurrenzdaten zur Verfügung, diese müssen noch dazu gebucht werden. Deswegen wird folgender Ansatz verfolgt.

So wie im vorherigen Konzept *Hotel Daten von vielen Hotels* werden auch hier die Hotels in eine vergleichbare Form gebracht. Das Ziel ist es dann, ein oder mehrere Hotels zu finden, die vermeintlich in den Rahmenparametern vergleichbar sind. Sobald ein oder mehrere ähnliche Hotels gefunden worden sind, können die Konkurrenzdaten von den ähnlichen Hotels genutzt werden.

Als Zielvariable des Modells werden dann die Preise des ähnlichsten Hotels verwendet. Jedoch sollen hierbei die Preise von den Konkurrenten und von dem korrespondierendem Hotel nicht einfach so benutzt werden, sondern lediglich das Verhältnis. Die Preise sollen anhand von dem durchschnittlichen Preis in Verhältnis gebracht werden, und dieses Verhältnis soll vorhergesagt werden.

Das Hotel ohne Vergangenheitsdaten muss in diesem Fall dann einen durchschnittlichen Preis angeben, anhand dessen mit dem vorhergesagten Verhältnis der tatsächliche Preis abgeleitet werden kann. Dies soll im folgenden Schaubild noch einmal dargestellt werden:

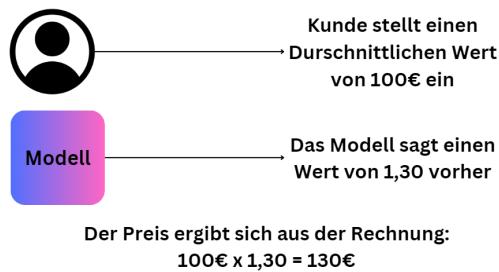


Abbildung 2.2: Mitbewerber Modell

2.3 Ähnliche Hotels

Dieses Konzept der *Ähnlichen Hotels* verschmilzt in gewisser Weise die Ideen der beiden Konzepte *Mitbewerber Modell* und *Hotel Daten von vielen Hotels*. Dieser Ansatz zielt darauf ab, ähnliche Hotels zu identifizieren und basierend auf den Daten dieser Hotels ein Modell zu entwickeln.

Im Gegensatz zum Konzept *Hotel Daten von vielen Hotels* besteht bei diesem Ansatz die Möglichkeit, konkrete Buchungsdaten der jeweiligen Hotels zu verwenden. Die primäre Herausforderung liegt jedoch darin, die ähnlichsten Hotels zu identifizieren. Nachdem diese ähnlichen Hotels ausfindig gemacht wurden, kann das bereits vorhandene Modell *Kombination aus RevPAR und Buchungskurve* ohne jegliche Anpassungen genutzt werden.

Dieses Modell wird dann, ähnlich wie bei anderen Hotels, mit den Buchungsdaten der zugrunde liegenden ähnlichen Hotels gefüttert, um Preise zu generieren. In diesem Szenario muss der Kunde lediglich eine Zuordnung zwischen dem RevPAR-Wert und dem konkreten Preis des Hotels festlegen.

Diese Vorgehensweise kombiniert die Vorteile beider vorherigen Konzepte, indem er sowohl auf die Ähnlichkeitsfindung zwischen Hotels, als auch auf die Nutzung spezifischer Buchungsdaten abzielt. Durch die Verwendung vorhandener Modelle ohne umfangreiche Modifikationen können so gezielt Preisvorhersagen für ähnliche Hotels generiert werden.

2.4 Synthetische Daten erstellen

Das Konzept der *Erstellung synthetischer Daten* markiert einen innovativen Ansatz innerhalb der Konzepte und weicht von den bisherigen Strategien ab. Dieser Ansatz verfolgt die Idee, ein Modell mit sämtlichen verfügbaren Daten zu trainieren und darauf aufbauend synthetische zukünftige Daten zu generieren. Ziel ist es, eine Art Simulation zu erstellen, die den Buchungsverlauf eines Hotels nachbildet.

Mittels dieser Simulation wird angestrebt, Vorhersagen darüber zu treffen, wie viele Buchungen für bestimmte Zimmerkategorien an bestimmten Tagen eingehen werden. Dadurch soll die Möglichkeit geschaffen werden, einen dynamischen Preis entsprechend dem erwarteten Buchungsverlauf zu gestalten.

Die Grundidee hinter dieser Vorgehensweise liegt in der Schaffung eines virtuellen Modells, welches basierend auf vergangenen Daten und Mustern potenzielle zukünftige Buchungen simuliert. Hierbei sollen verschiedene Szenarien durchgespielt werden, um die wahrscheinlichsten Buchungstrends abzuschätzen, und somit einen fundierten Ansatz für die dynamische Preisgestaltung zu generieren.

2.5 Evaluation

Nachdem nun die ausgearbeiteten Konzepte vorgestellt wurden, gilt es diese zu bewerten um festzulegen, mit welchen Konzepten fortgefahrene werden soll. Jedes der vorgestellten Entwürfe ist auf seine Art valide und hat auch die Berechtigung weiterhin verfolgt zu werden. Um entscheiden zu können, welchem Konzept überhaupt nachgegangen werden soll oder in welcher Reihenfolge sie ausprobiert werden sollen, werden die Konzepte nach den folgenden Kriterien bewertet:

- Aufwand
- Erfolgswahrscheinlichkeit
- Impact

Aufwand und Erfolgswahrscheinlichkeit sind selbsterklärend. Der Impact bezieht sich darauf, inwiefern happyhotel im Generellen von dem Konzept profitieren könnte und ob das Konzept nicht auch für schon vorhandene Kunden eingesetzt werden könnte.

Jedes Konzept kann bei jedem Kriterium eine Zahl zwischen 1 bis 5 erzielen, wobei 5 das Beste und 1 das Schlechteste in dem jeweiligen Kriterium bedeutet. Die Ergebnisse der Evaluation sind wie folgt in der Tabelle dargestellt:

Konzepte	Aufwand	Erfolgsw.	Impact	Result
Daten von vielen Hotels	4	4	4	12
Mitbewerber	4	3	3	10
Ähnliche Hotels	4	5	4	13
Synthetischen Daten	1	3	5	9

Tabelle 2.1: Evaluierung der Konzepte

Nach der Evaluierung wurde bestimmt, dass das Konzept *Ähnliche Hotels* das größte Potenzial hat und soll dementsprechend auch verfolgt werden. Je nach Zeit und Ergebnis werden die Konzepte *Mitbewerber Modell* und *Hotel Daten von vielen Hotels* auch verfolgt und evaluiert werden. Die Erstellung einer Simulation durch synthetische Daten ist auch ein sehr interessantes Konzept, würde aber im Rahmen dieser Thesis zu weit gehen.

3. Ähnliche Hotels

Die Evaluierung der verschiedenen Konzepte, wie sie im vorangegangenen Kapitel *Konzepte* ausführlich dargelegt wurden, führte zu einer kritischen Entscheidung bezüglich des anfänglichen Fokus für die strategische Ausrichtung. Nach einer umfassenden Analyse und Bewertung wurde klar, dass das Konzept der *Ähnliche Hotels* sich als überzeugende und vielversprechende Lösung für das vorliegende Problem erweist.

Dieses Kapitel konzentriert sich daher detailliert auf das Konzept der *Ähnliche Hotels*. Es zielt darauf ab, die spezifische Methodik und Herangehensweise dieses Konzepts zu beleuchten. Die kommenden Abschnitte bieten eine eingehende Analyse, die durch umfassende Datenanalysen und Experimente unterstützt wird. Dadurch soll ein tiefgreifendes Verständnis für die erfolgreiche Umsetzung des Konzepts der *Ähnliche Hotels* vermittelt werden.

Wie in der Sektion *Ähnliche Hotels* beschrieben, liegt der Fokus zunächst darauf, ähnliche Hotels zu identifizieren, die als Grundlage für die Anpassung und Anwendung des bereits bestehenden RevPAR-Modells dienen sollen.

3.1 Finden von ähnlichen Hotels

Das Finden von ähnlichen Hotels ist die Grundlage dieses Konzeptes. Um dies zu erreichen, soll das Vorgehen, welches in der Sektion *Ähnliche Hotels* beschrieben wurde, verfolgt werden. Angefangen mit der Datenbeschaffung, muss sich zunächst ein Überblick darüber gemacht werden, welche Daten schon vorhanden sind und welche Daten gegebenenfalls noch besorgt werden müssen, bevor dann mit der Datenanalyse und der Modellierung weiter gemacht werden kann.

3 Ähnliche Hotels

3.1.1 Datenbeschaffung

Durch die verschiedenen Property Management Systeme sind für die einzelnen Hotels bereits Stammdaten wie Zimmeranzahl oder Zimmerkategorien vorhanden. Zudem wird der jeweilige Kunde beim Einrichten von happyhotel auch schon nach verschiedenen Daten befragt. Unter den Daten, die bei einem Kunden abgefragt werden, gehören Informationen wie zum Beispiel die Adresse des jeweiligen Hotels.

Werden nun alle schon vorhanden Informationen zusammengetragen, die bisher in der Datenbank zur Verfügung stehen, entsteht dabei das folgende Dataframe:

company_id	median_min	median_max	areatype_count	area_count	region	city
60dca381ae221ce505d2263a	10950.0	12550.0	6	12	Baden-Württemberg	Ohlsbach
61baffd61e564422dba0903f	17900.0	24900.0	5	16	Schleswig-Holstein	Schleswig
60ce02508af8912a494d2b29	30400.0	37400.0	16	68	Rheinland-Pfalz	Rhdt unter Rietburg
614c8db23707c0c5c3bd91e6	6000.0	8600.0	15	49	Nordrhein-Westfalen	Eschweiler
600e7ff88c1fe4620c48db08	12400.0	18500.0	6	27	Baden-Württemberg	Reutlingen

Abbildung 3.1: Alle schon vorhanden Features

In Abbildung 3.1 sind die Daten zu sehen, die bislang zur Verfügung stehen, wobei sich die *median_min* und *median_max* Werte auf den Median aller Zimmerkategorie-Preise bezieht.

Dadurch, dass *region* und *city* manuell vom Kunden eingetragene Werte sind, ergab sich eine gewisse Skepsis, ob alle Werte norm-konform eingetragen wurden. Aufgrund dieser Skepsis sollte eine kleine Datenanalyse getätigter werden.

Bei der Datenanalyse wurde jeweils nach der Region und nach der Stadt gruppiert, um zu prüfen ob die Daten so benutzt werden können. Im Folgenden sind die Werte jeweils für die Region und für die Stadt zu sehen:

```
Index(['Baden-Württemberg', 'Bavaria', 'Bayern', 'Berlin', 'Brandenburg',
       'Free and Hanseatic City of Hamburg', 'Hansestadt Hamburg', 'Hessen',
       'Kanton Zürich', 'Land Berlin', 'Land Salzburg',
       'Mecklenburg Vorpommern', 'Niedersachsen', 'Nordrhein-Westfalen',
       'Rheinland-Pfalz', 'Saarland', 'Sachsen', 'Sachsen-Anhalt',
       'Schleswig-Holstein', 'Wien'],
      dtype='object', name='region')
```

Abbildung 3.2: Alle vorhanden Regionen

3 Ähnliche Hotels

```
Index(['München', 'Aachen', 'Allenbach', 'Appenweier', 'Aschheim', 'Bad Ems',
       'Bad Kreuznach', 'Bad Schlema', 'Baesweiler', 'Bayreuth', 'Bendorf',
       'Berlin', 'Berlin - Karlshorst', 'Borkum', 'Cuxhaven', 'Dachau',
       'Dortmund', 'Dülmnen', 'Eisenberg (Pfalz)', 'Erlangen', 'Eschweiler',
       'Freiburg im Br.', 'Freiburg im Breisgau', 'Freising',
       'Gerlinden/Maisach', 'Haltern am See', 'Hamburg', 'Harrislee',
       'Isernhagen', 'Karlsruhe', 'Kirchberg an der Jagst', 'Koblenz', 'Köln',
       'Königswinter', 'Lautenbach', 'Lutherstadt Eisleben', 'Mannheim',
       'Monschau', 'München', 'Nürnberg', 'Ohlsbach', 'Olching', 'Parsberg',
       'Parsberg-Hörmannsdorf', 'Plauen', 'Putbus - Lauterbach', 'Remscheid',
       'Reutlingen', 'Rhodt unter Rietburg', 'Rosenheim', 'Rüdesheim am Rhein',
       'Saarlouis', 'Salzburg', 'Sauerlach', 'Schermbeck', 'Scheßlitz',
       'Schleswig', 'Schwabach', 'Schönwald im Schwarzwald',
       'Seehausen am Staffelsee', 'Sonsbeck', 'St. Peter-Ording', 'Stuttgart',
       'Templin', 'Todtnau', 'Todtnau-Muggenbrunn', 'Trassem', 'Trier', 'Ulm',
       'Uster/Zürich', 'Warstein', 'Weilheim im Oberbayern', 'Wemding',
       'Wernau', 'Westerstede', 'Wien', 'Wunstorf'],
      dtype='object', name='city')
```

Abbildung 3.3: Alle vorhanden Städte

Es ist eindeutig zu erkennen, dass es sowohl bei der Region als auch bei der Stadt zu einer Inkonsistenz kommt. So wurde die Region *Berlin* sowohl als *Berlin*, als auch mit *Land Berlin* angegeben. Auch bei der Stadt ist die Diskrepanz deutlich zu erkennen, -so wurde bspw. *München* mit Leerzeichen vorne dran angegeben oder die Stadt Freiburg einmal mit der Abkürzung *Br.* und ein anderes Mal mit komplett ausgeschriebenen *Breisgau* angegeben. Es ergab sich also, dass diese Werte so wie sie sind, nicht verwendet werden können.

Beschaffung von Region, City und Stadtgröße

Durch eine schon im Vorfeld getätigte Arbeit, existieren zu jedem Hotel in unserer Datenbank die Koordinaten, repräsentiert durch die zwei Werte Long- und Latitude. Mithilfe von diesen zwei Werten sollte ein Skript geschrieben werden, um die Region, die Stadt und Größe der Stadt zu beschaffen.

Für das Skript wurde *Nominatim* API verwendet, um die einzelnen Werte zu beschaffen. Im Folgenden wird das Skript präsentiert:

```
1  from geopy.geocoders import Nominatim
2
3 def get_region_city_size(latitude, longitude):
4     geolocator = Nominatim(user_agent="city_size_app")
5     location = geolocator.reverse((latitude, longitude), language='de')
6
7     address = location.raw['address']
8
9     if 'city' in address:
10         region = address["city"]
11         if 'state' in address:
```

```

12     region = address["state"]
13     return {
14         "city": address["city"],
15         "region": region,
16         "size": "Großstadt"
17     }
18 elif 'town' in address:
19     return {
20         "city": address["town"],
21         "region": address["state"],
22         "size": "Kleinstadt"
23     }
24 elif 'village' in address:
25     region = None
26     if 'county' in address:
27         region = address["county"]
28     if 'state' in address:
29         region = address["state"]
30     return {
31         "city": address["village"],
32         "region": region,
33         "size": "Kleinstadt"
34     }
35 else:
36     return dict()

```

Listing 3.1: Einfaches Recommendation System für Film vorschläge

Die Funktion `get_region_city_size` nimmt als Parameter die Long- und Latitude-Werte und erzeugt dadurch ein *Dictionary* mit den Werten `region`, `city` und `size`.

Beschaffung von der Hotelart

Einer der wichtigsten Eigenschaften, die ein Hotel vorweisen kann, ist die Hotelart von dem jeweiligen Hotel. Die Hotelart hängt maßgeblich mit der zur grundlegenden Preisgestaltung ab. Dies ist einfach zu erklären, da Hotels existieren, die eher auf Wellness ausgelegt sind und somit prinzipiell teurer sind als einfache Urlaubshotels. Somit ist die Art eines Hotels essentiell, um ähnliche Hotels zu finden. So soll auch für das Modell die Hotelart vorhanden sein. Dieses Feature muss jedoch erst beschafft werden, da diese Information nicht in der Datenbank hinterlegt ist.

Leider ist der Versuch, die Hotelart auf einem automatisierterem Weg zu bekommen, gescheitert, und es blieb nichts anderes übrig, als die Hotelart eines jeden Hotels manuell herauszufinden.

3 Ähnliche Hotels

Nachdem die Hotelart beschaffen wurde, sieht der Feature-Datensatz wie folgt aus:

company_id	median_min	median_max	areatype_count	area_count	region	city	art
6284f18969588aff1c3ace83	8900.0	19900.0	8	147	Berlin	Berlin	Stadshotel
615f129a52ab7509853d352c	7400.0	23400.0	3	27	Baden-Württemberg	Karlsruhe	Stadshotel
6155bdff03707c0492904de86	8000.0	31000.0	4	19	Bayern	Erlangen	Wellnesshotel
5fc4e11c3c09ba0c169c93b3	9000.0	30000.0	2	188	Berlin	Berlin	Businesshotel
627232f4ac34477a98aba6af	4900.0	12900.0	9	61	Rheinland-Pfalz	Trier	Stadshotel

Abbildung 3.4: Alle schon vorhanden Features 2

3.1.2 Datenvorverarbeitung

Nach der Datenbeschaffung erfolgt die Datenvorverarbeitung, die darauf abzielt, die Qualität und Integrität der Daten sicherzustellen. Ein zentraler Schwerpunkt liegt dabei auf der Identifikation und Eliminierung fehlerhafter oder ungültiger Datensätze. Häufig wird innerhalb der Datensätze nach Nullwerten gesucht, um diese entweder durch valide Daten zu ersetzen oder in einigen Fällen gänzlich zu entfernen. Im vorliegenden Fall ist es wichtig, dass sämtliche als verwendbar gekennzeichneten Hotels in die Analyse einbezogen werden. Die Datensätze sollen nicht einfach verworfen werden; vielmehr erfolgt eine gezielte Substitution von Nullwerten durch valide Daten, um eine konsistente und zuverlässige Grundlage für die weiterführende Analyse zu gewährleisten [3].

Im Folgenden ist die Anzahl aller Nullwerte innerhalb des Datensatzes zu sehen:

```
median_min      0
median_max      0
areatype_count  0
area_count      0
region          0
city            0
art             0
dtype: int64
```

Abbildung 3.5: Summe aller Nullwerte im Datensatz

In Abbildung 3.5 ist zu sehen, dass innerhalb des Datensatzes keine Nullwerte gibt und somit auch keine Datenvorverarbeitung getätigt werden muss.

3.1.3 Allgemeine Datenanalyse

Die Datenanalyse ist ein, wenn nicht sogar der wichtigste Schritt im Data Science Bereich [3]. Dabei sollen die Daten ergründet und verstanden werden. Dieser Schritt soll nicht nur ein Verständnis für die vorliegenden Daten schaffen, sondern auch *outlier* erfassen. Meist wird auch versucht, innerhalb der Datenanalyse Zusammenhänge zu der Zielvariable zu finden. Dies setzt jedoch voraus, dass eine Zielvariable vorhanden ist. In dem vorliegenden Fall existiert keine Zielvariable, da nicht bekannt ist, welche Hotels mit welchen anderen Unterkünften kongruent sind. Aufgrund dessen, dass keine Zielvariable vorhanden ist, wird die folgende Datenanalyse lediglich dazu genutzt, um die Daten besser zu verstehen. Weiterhin soll festgestellt werden, welche Daten in welcher Form als Features verwendet werden können.

Für die Datenanalyse soll die Python-Bibliothek *Seaborn* benutzt werden. Die *Seaborn*-Bibliothek stellt ein leistungsstarkes Werkzeug dar, das speziell für die Erstellung von statistischen Grafiken in Python entwickelt wurde [4]. Das Hauptziel besteht darin, die verschiedenen Verteilungen der Features auf anschauliche Weise darzustellen, um einen umfassenden Überblick über die zugrunde liegenden Daten zu ermöglichen. Durch die Nutzung der Funktionalitäten von *Seaborn* wird eine effiziente und ästhetisch ansprechende Visualisierung erreicht, die es ermöglicht, Muster, Ausreißer oder Trends in den Daten leichter zu identifizieren.

Region Features

Zuallererst soll ein grober Überblick über die Städte der Hotels innerhalb der Datenbank erstellt werden. Dazu wird innerhalb des Datensatzes nach der Stadt um die Anzahl der Hotels einer Stadt zu ermitteln.

3 Ähnliche Hotels

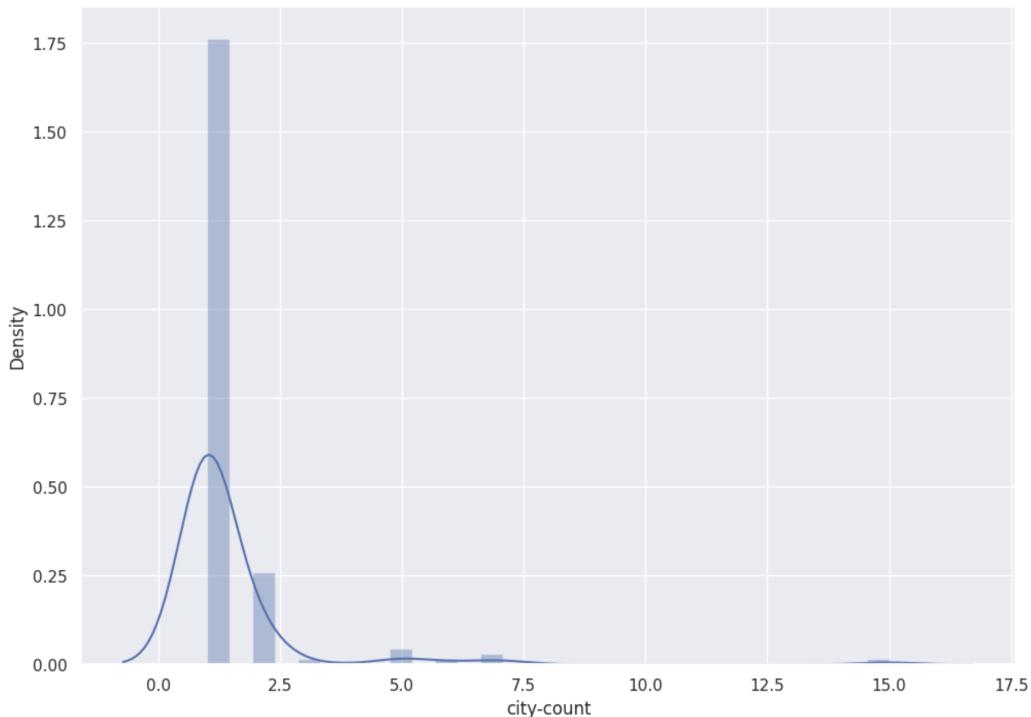


Abbildung 3.6: Verteilung der Städte

Die Analyse der Abbildung 3.6 offenbart, dass die überwiegende Mehrheit der in der Datenbank erfassten Städte lediglich über ein einziges Hotel verfügt, welches happyhotel in Anspruch nimmt. Zugleich verdeutlicht die Abbildung 3.6, dass es lediglich einen geringen Anteil von Städten gibt, in denen fünf oder mehr Hotels registriert sind. Dies ist etwas problematisch, da das Feature *City* zu sehr unausgewogen ist und so nicht mit in das Modell gegeben werden kann. Es muss also in einer anderen Form verwendet werden.

Momentan werden, wie in der Abbildung 3.4 gezeigt, die Stadt und die Region als zwei separate Features aufgelistet. Die Idee ist es nun, die zwei Features zu verschmelzen und die Hotels in Regionen aufzuteilen, um mehr Informationen zu erhalten. Anstatt also Region und Stadt separat zu haben, soll es ein Feature Region geben, welches wie folgt aufgebaut ist:

- Befinden sich innerhalb einer Stadt fünf oder mehr Hotels, so wird die Stadt ohne jegliche Modifikation als Region genommen.
- Befinden sich innerhalb einer Stadt weniger als fünf Hotels, so setzt sich die Region aus der ursprünglichen Region, also dem Bundesland und der Größe der Stadt zusammen nach dem Schema: Region-Größe

3 Ähnliche Hotels

Für die Idee muss zunächst ermittelt werden, in welchen Städten sich fünf oder mehr Hotels befinden. Auch diese Information kann wie folgt visualisiert werden:

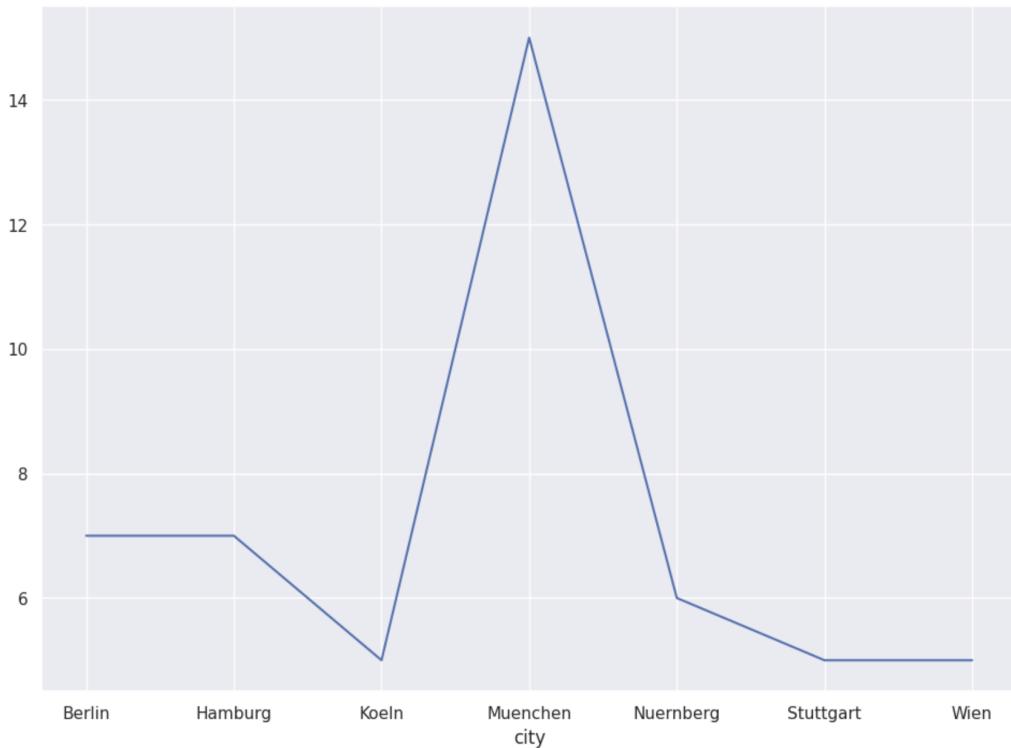


Abbildung 3.7: Städte mit Fünf oder mehr Hotels

Ganz klar ist zu erkennen, dass die sieben Städte Berlin, Hamburg, Köln, München, Nürnberg, Stuttgart, und Wien die Städte sind, in denen fünf oder mehr Hotels vorhanden sind. Die aufgelisteten Städte können dementsprechend so übernommen werden und für alle anderen wird die Regel von oben angewandt.

Nach der Umformulierung von Region und Stadt sieht der Datensatz wie folgt aus:

company_id	median_min	median_max	areatype_count	area_count	art	region2
6247262ad7da93fb2b9ac50d	10500.0	29500.0	7	18	Stadthotel	Wien
5f7db0bb34ee036332daffea	12000.0	60000.0	5	38	Businesshotel	Berlin
5faac8b2448f3913904ffaed	9500.0	35000.0	6	205	Businesshotel	Muenchen
637353f228d9f119bb9e6d2b	12900.0	52900.0	12	75	Stadthotel	Nuernberg
6492f9f68b5ac216293bee08	7500.0	8300.0	5	9	Familienhotel	Bayern-Kleinstadt
65005a8d765491b555190edb	9150.0	17150.0	9	36	Ferienhotel	Baden-Wuerttemberg-Grossstadt
63791654c755fe850934264c	12900.0	30000.0	8	26	Familienhotel	Niedersachsen-Kleinstadt
641039e8b6b33774ec0335d5	11500.0	30000.0	10	189	Apartementhotel	Berlin
61420e574f15836e1a37a173	7900.0	29000.0	3	28	Businesshotel	Nuernberg
6221d0f96b58f774c9890657	8900.0	14900.0	2	42	Apartementhotel	Bayern-Kleinstadt

Abbildung 3.8: Datensatz nach der Umformulierung

3 Ähnliche Hotels

Auch hier soll wieder die Verteilung visualisiert werden

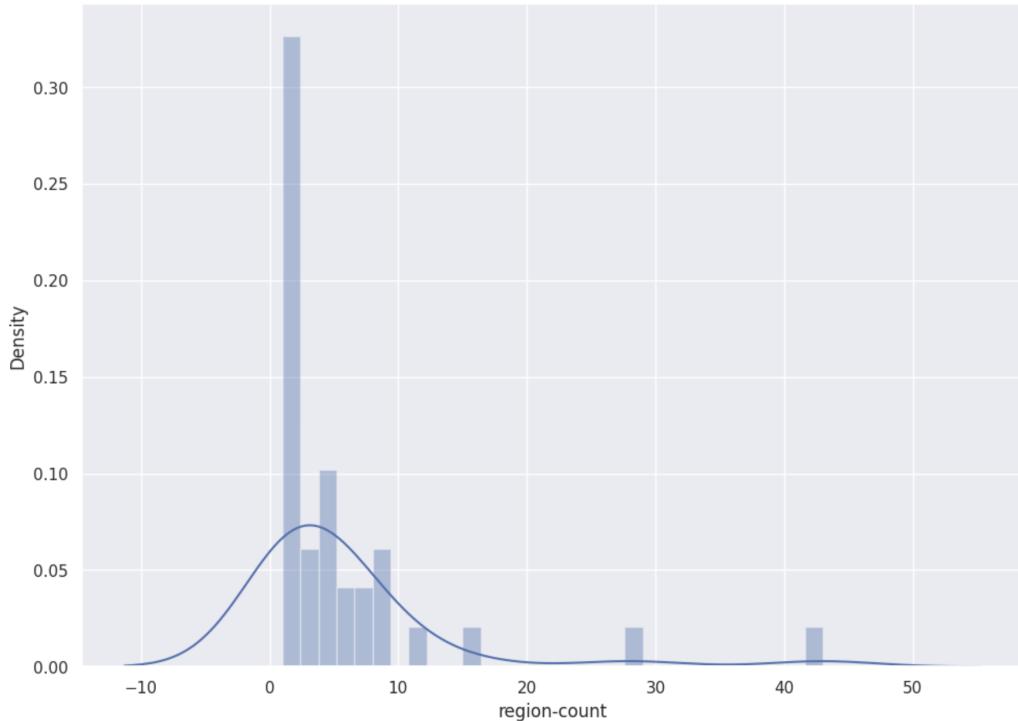


Abbildung 3.9: Verteilung des neuen Features *region2*

Es zeigt sich, dass sich noch immer ein großer Anteil des Datensatzes im niedrigeren Bereich befindet, jedoch konnte mit der Modifikation ein bisschen mehr Varianz in die Daten gebracht werden.

Preis Features

Als nächstes sollen die Preis Features, namentlich betitelt mit *median_min* und *median_max*, erkundet und verdeutlicht werden. Hierzu soll wie auch bei der Region zunächst die Verteilung betrachtet werden:

3 Ähnliche Hotels

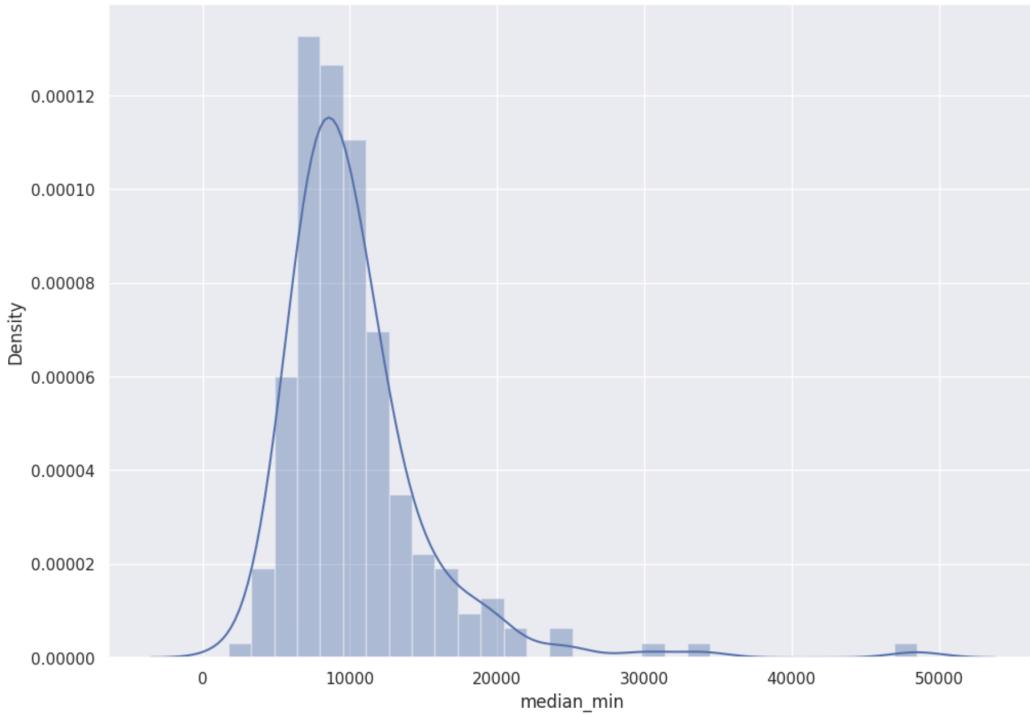


Abbildung 3.10: Verteilung von dem Preis Features *median_min*

Abbildung 3.10 zeigt, dass das Feature *median_min* eine Normalverteilung mit ein paar outlier aufweist. Eine Normalverteilung sagt aus, dass die Verteilung mehr Daten um den Mittelwert herum aufweist. Die Datenverteilung nimmt ab, wenn sich vom Zentrum entfernt wird. Die resultierende Kurve ist symmetrisch zum Mittelwert und bildet eine glockenförmige Verteilung [5].

Eine weitere interessante Information, die noch aus dem Feature *median_min* gelesen werden kann, ist der durchschnittliche Wert Region. Dazu soll der Datensatz nach der Region gruppiert werden, um den durchschnittlichen Wert zu ermitteln.

3 Ähnliche Hotels

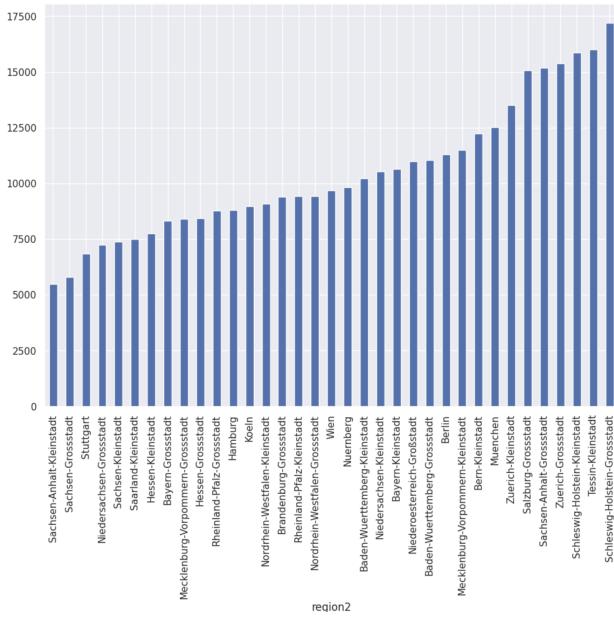


Abbildung 3.11: Durchschnittlicher minimal Preis pro Region

Die Erkenntnis, die aus der Abbildung 3.11 genommen werden kann, ist die, dass es einen deutlichen Unterschied macht, in welcher Region das Hotel liegt, wenn auf den Minimalen Median Preis des Hotel geschaut wird. Das kann auch ebenso mit dem Maximalen Median Preis eines Hotels gemacht werden. Auch hier soll zunächst die Verteilung visualisiert werden:

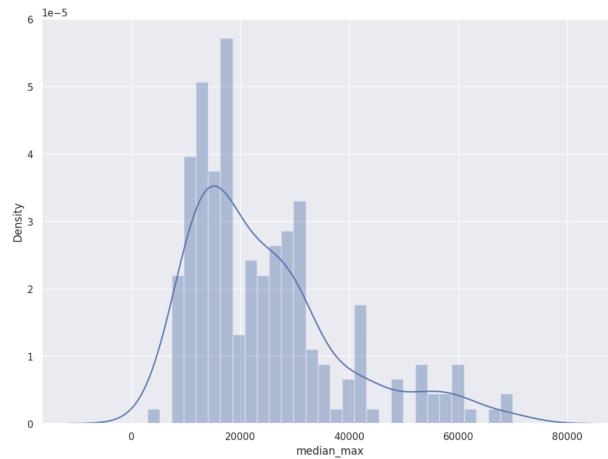


Abbildung 3.12: Verteilung von dem Preis Features *median_max*

Anders als bei dem Minimalen Median Preis, kann bei dem Maximalen Median Preis keine Normalverteilung erkannt werden. Hier wirken die Werte recht verstreut.

3 Ähnliche Hotels

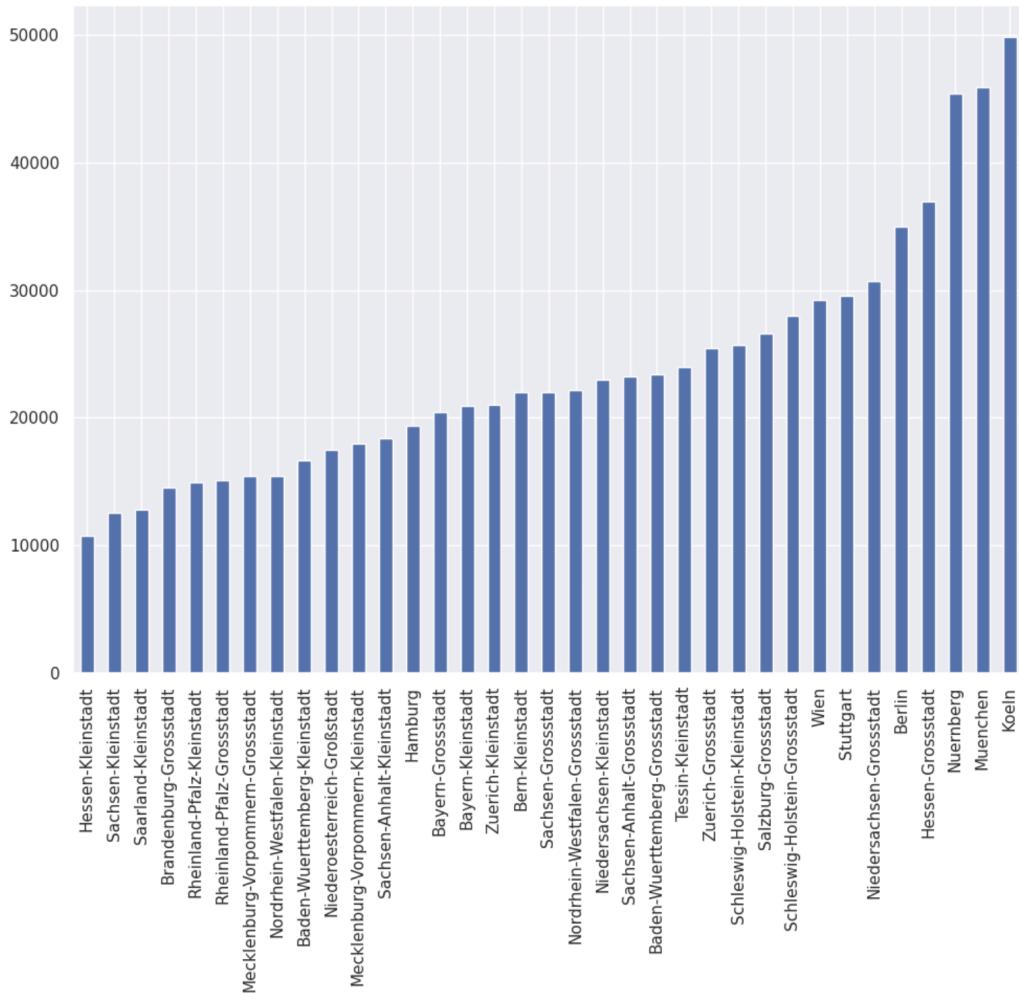


Abbildung 3.13: Durchschnittlicher maximal Preis pro Region

Abbildung 3.13 zeigt, dass es auch deutliche Unterschiede bei den einzelnen Regionen gibt. Eine weitere interessante Erkenntnis ist die, dass es bei dem Maximalen Median Preis eher so ist, dass die Großstädte wie Köln und München eher zu einem höheren Maximalen Preis tendieren.

Hotelart Feature

Die Hotelart bildet einen essenziellen Bestandteil, um umfassende Einblicke in die Charakteristiken eines Hotels zu gewinnen. Sie liefert nicht nur Informationen über den Zweck und die Ausrichtung der Unterkunft, sondern ermöglicht auch eine präzise Identifikation der Zielgruppe, die das Hotel anspricht [6]. Die Zielgruppe eines Hotels ist zudem eine wertvolle Information, wenn es darum geht, Preise für das

3 Ähnliche Hotels

Hotel zu gestalten, zumindest ist so die Annahme. Auch in diesem Fall kann wieder nach der Hotelart gruppiert werden und jeweils der minimale und maximale durchschnittliche Preis angezeigt werden.

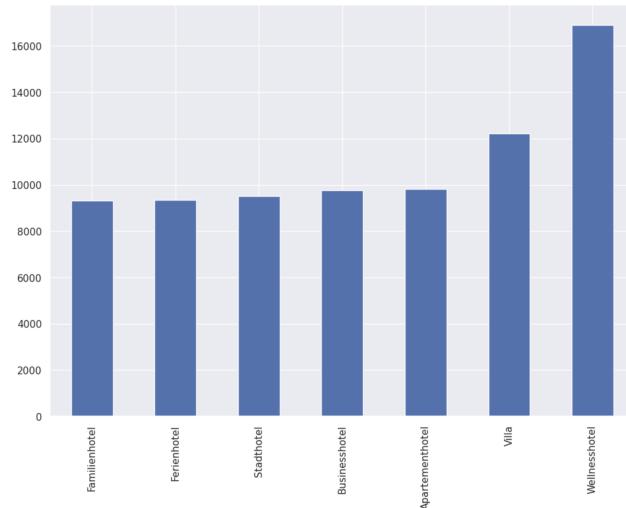


Abbildung 3.14: Durchschnittlicher minimal Preis pro Hotelart

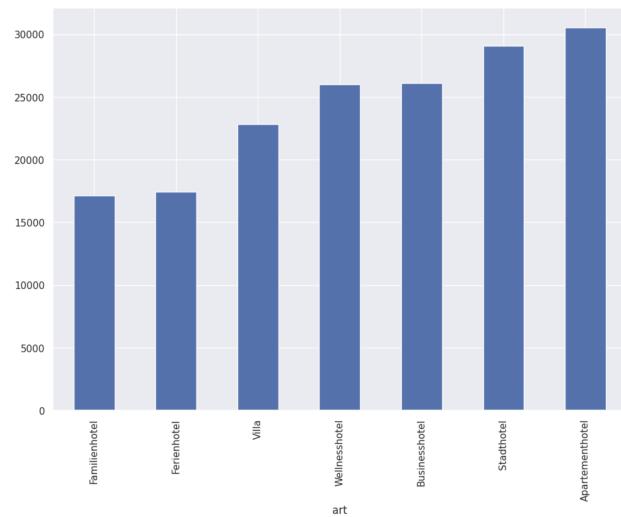


Abbildung 3.15: Durchschnittlicher maximal Preis pro Hotelart

Anhand von den zwei Abbildungen 3.14 und 3.15 zeigt sich, dass es tatsächlich einen Unterschied bei den Preisen auf die Hotelart gibt. Zudem zeigt sich, dass die zwei Hotelarten *Ferienhotel* und *Familienhotel* in beiden Fällen die gleiche Information wiedergibt und somit auch zu *Ferienhotel* zusammengefasst werden kann. Zudem könnten anhand von *median_min* noch weitere Hotelarten zusammengefasst werden, jedoch wenn beide Informationen zusammen betrachtet werden, so bleibt es lediglich bei *Ferienhotel* und *Familienhotel*.

Zimmer Features

Die letzten zwei Features innerhalb des Datensatzes sind die Features *area_count* und *areatype_count*, welche die Größe des jeweiligen Hotels repräsentieren. Die Frage, die sich hier also stellt, ist ob das Preisverhältnis in irgendeiner Art mit der Größe des Hotels zusammenhängen kann. Hierzu soll zunächst auch erstmal die Verteilung der Zimmeranzahl angeschaut werden:

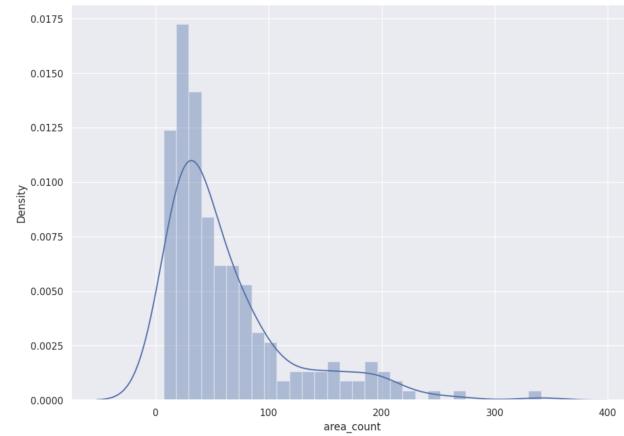


Abbildung 3.16: Verteilung nach Zimmeranzahl

Die Verteilung zeigt, dass es auch hier recht ausgeprägt ist und lediglich im Ansatz einer Normalverteilung gleicht. Des Weiteren soll nach der Anzahl der Zimmer gruppiert werden, um zu überprüfen, wie oft jede Anzahl vorkommt:

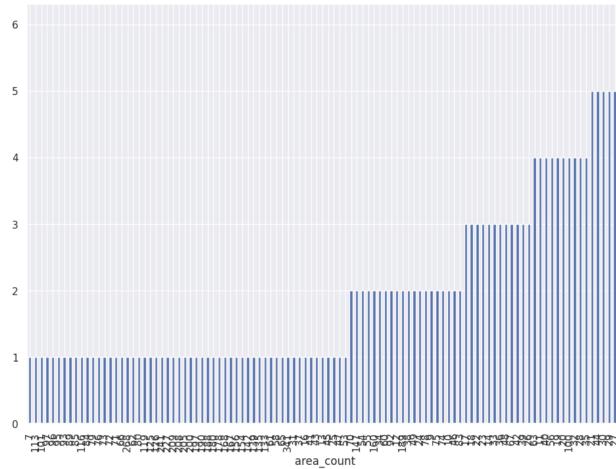


Abbildung 3.17: Häufigkeit der Zimmeranzahl im Datensatz

Abbildung 3.17 hat gezeigt, dass das Feature *area_count* zu ausgeprägt ist. Aufgrund dessen, dass das Feature zu ausgeprägt ist und keine Idee vorhanden ist, wie

3 Ähnliche Hotels

dieses Feature umformuliert werden könnte, wurde beschlossen *area_count* und *areatype_count* aus dem Datensatz zu entfernen.

Der finale Datensatz, welcher für das Modell benutzt werden soll, sieht wie folgt aus:

company_id	median_min	median_max	art	region2
6247262ad7da93fb2b9ac50d	10500.0	29500.0	Stadthotel	Wien
5f7db0bb34ee036332daffea	12000.0	60000.0	Businesshotel	Berlin
5faac8b2448f3913904ffaed	9500.0	35000.0	Businesshotel	Muenchen
637353f228d9f119bb9e6d2b	12900.0	52900.0	Stadthotel	Nuernberg
6492f9f68b5ac216293bee08	7500.0	8300.0	Ferienhotel	Bayern-Kleinstadt
65005a8d765491b555190edb	9150.0	17150.0	Ferienhotel	Baden-Wuerttemberg-Grossstadt
63791654c755fe850934264c	12900.0	30000.0	Ferienhotel	Niedersachsen-Kleinstadt
641039e8b6b33774ec0335d5	11500.0	30000.0	Apartementhotel	Berlin
61420e574f15836e1a37a173	7900.0	29000.0	Businesshotel	Nuernberg
6221d0f96b58f774c9890657	8900.0	14900.0	Apartementhotel	Bayern-Kleinstadt

Abbildung 3.18: Finaler Datensatz für das Modell

3.1.4 Evaluation der ähnlichen Hotels

Der vorliegende Datensatz ist nun verfügbar, und grundsätzlich kann die Modellierung fortgesetzt werden. Allerdings stellt sich die Frage, ob die identifizierten Hotels tatsächlich ähnlich zum ursprünglichen Hotel sind. Es ist von entscheidender Bedeutung, nachzuweisen, dass die ausgewählten Hotels auf irgendeine Weise miteinander vergleichbar sind. Aus diesem Grund wird im nachfolgenden Abschnitt ein Mechanismus entwickelt, um die Ähnlichkeit der Hotels zu überprüfen und zu gewährleisten.

Angesichts des angestrebten Ziels, nämlich der dynamischen Generierung von Preisen, wurde zunächst in Erwägung gezogen, die Preise der einzelnen Hotels zu vergleichen. Zu diesem Zweck wurde initial ein *Dataframe* erstellt, das sämtliche gültigen Hotels sowie ihre Preisinformationen für einen bestimmten Zeitraum umfasst.

3 Ähnliche Hotels

	5eff10592f24eb0df2ef4825	5ea83ceb60b7a90e412a8047	612dd09e6abd9a3651042cb3	614c8b5e3707c081fcbd48e8	6006f4878c1fe4571a439326
0	6900.0	7150.0	99900.0	4800.0	13050.0
1	7000.0	7350.0	99900.0	4600.0	11950.0
2	7000.0	7550.0	99900.0	4700.0	11950.0
3	7000.0	7850.0	99900.0	4600.0	11950.0
4	7000.0	7500.0	99900.0	4700.0	11950.0
5	7000.0	7350.0	99900.0	4700.0	11950.0
6	7000.0	7350.0	99900.0	4700.0	11950.0
7	7000.0	7150.0	99900.0	4600.0	11950.0
8	7000.0	7350.0	7400.0	4600.0	11950.0
9	7000.0	7550.0	7900.0	4600.0	13750.0

Abbildung 3.19: Preise von allen Hotels für das Jahr 2022

Abbildung 3.19 zeigt einen exemplarischen Auszug aus dem DataFrame. Zudem wurde anhand dieses DataFrame noch die dazugehörige Korrelationsmatrix erstellt. Die Korrelationsmatrix ist dafür da, um Zusammenhänge zwischen den Vektoren zu finden. Dabei beschreibt ein Wert nahe 1 einen hohen positiven Zusammenhang der zwei Vektoren und ein Wert nahe -1 einen hohen negativen Zusammenhang der zwei Vektoren. Ein Korrelationswert gegen 0 beschreibt keinerlei Zusammenhang der Vektoren [7]. Die Vermutung legt nah, dass wenn die Preise von 2 Hotels korrelieren und die Preise sich überschneiden, so müssten das ähnliche Hotels sein.

Diese Vermutung soll dementsprechend mit einigen Hotels getestet werden:

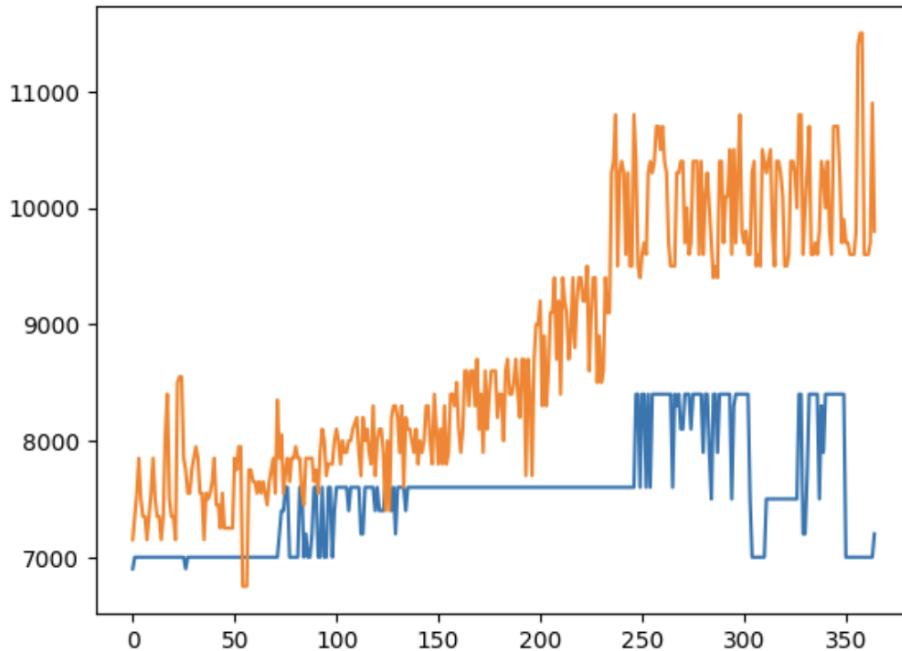


Abbildung 3.20: Visualisierung der Preise zweier Hotels

3 Ähnliche Hotels

	5eff10592f24eb0df2ef4825	5ea83ceb60b7a90e412a8047
5eff10592f24eb0df2ef4825	1.000000	0.600991
5ea83ceb60b7a90e412a8047	0.600991	1.000000

Abbildung 3.21: Korrelationswerte der zwei Hotels

Abbildung 3.20 illustriert den Preisverlauf von zwei Hotels im Jahr 2022, während Abbildung 3.21 die Korrelation zwischen diesen beiden Hotels zeigt. Der festgestellte Korrelationswert von 0,6 erweist sich als bemerkenswert, insbesondere vor dem Hintergrund, dass die Preise in Abbildung 3.20 beträchtlich voneinander abweichen. Infolgedessen wurde die Überlegung angestellt, die Preise zu skalieren und daraufhin miteinander zu vergleichen. Entscheidend für ähnliche Hotels ist lediglich die Tendenz, wie sich die Preise verhalten.

Werden die Preise nun skaliert ändert sich an der Korrelation nichts und die skalierten Preise sehen nun wie folgt aus:

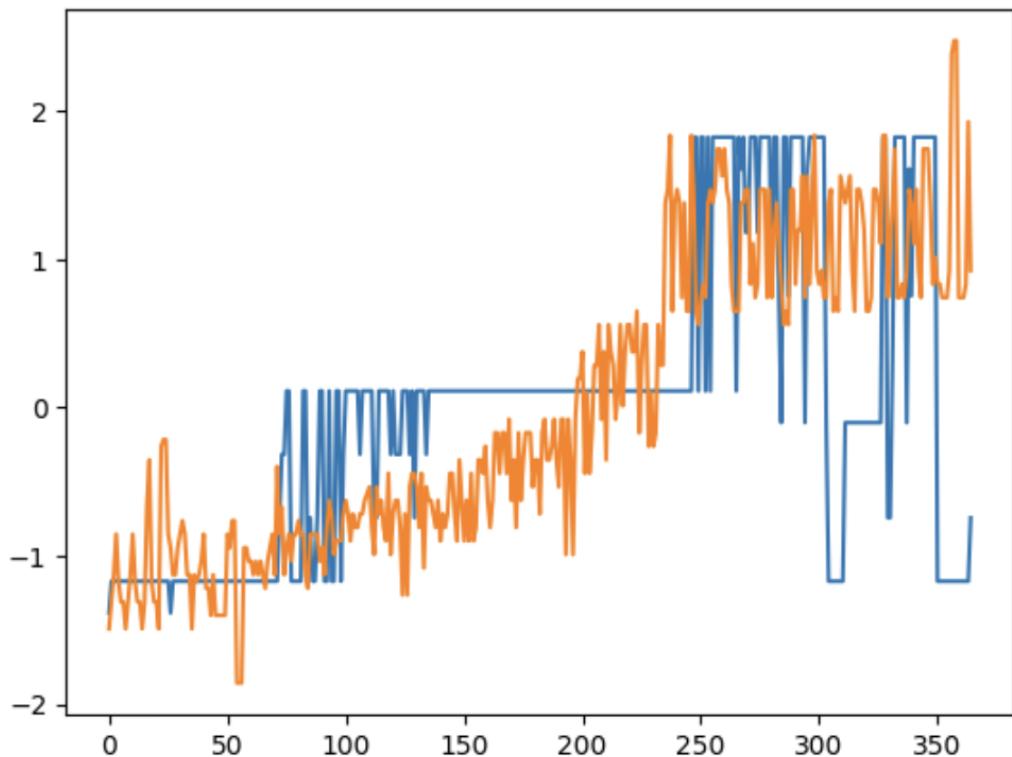


Abbildung 3.22: Visualisierung der Skalierten Preise zweier Hotels

Eine zusätzliche Betrachtung ergab die Frage, ob ein ähnliches Muster nicht auch durch die Verwendung des RevPAR erzielt werden könnte. Die Verwendung des

3 Ähnliche Hotels

RevPAR-Werts erscheint in diesem Kontext sinnvoller als die ausschließliche Be- rücksichtigung der Zimmerpreise, da das nachfolgende Modell letztendlich darauf abzielt, den RevPAR-Wert vorherzusagen.

Im Folgenden werden die gleichen zwei Hotels mit dem RevPAR-Wert verglichen:

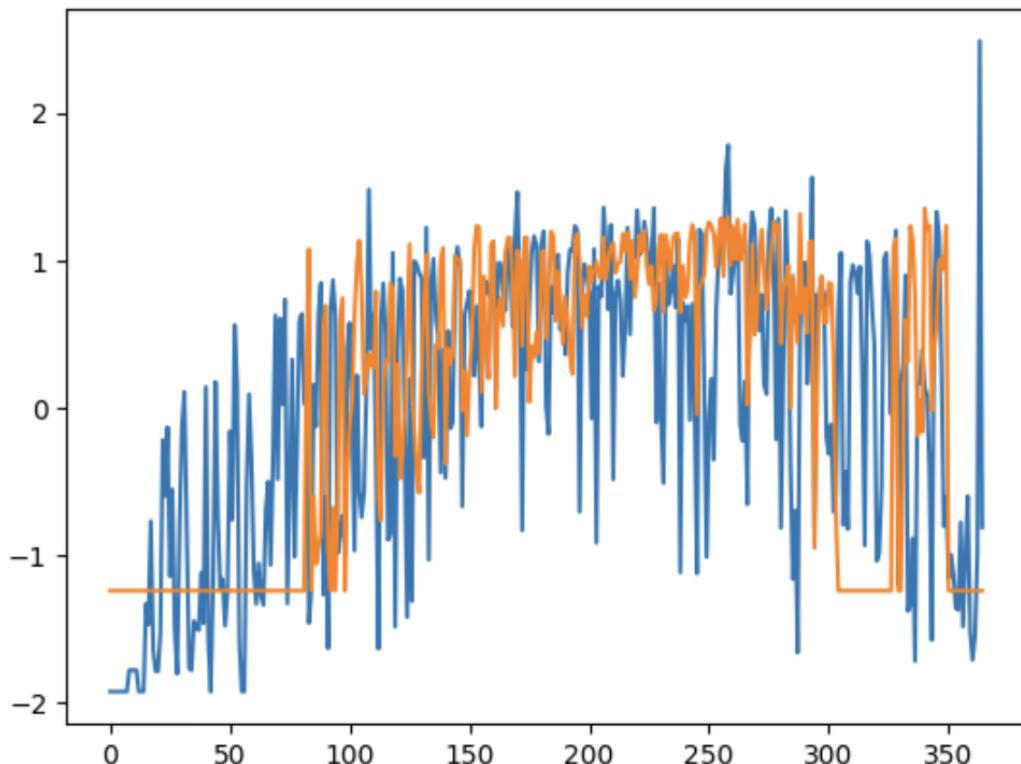


Abbildung 3.23: Visualisierung der Skalierten RevPAR-Werte zweier Hotels

5eff10592f24eb0df2ef4825	5ea83ceb60b7a90e412a8047
5eff10592f24eb0df2ef4825	1.00000
5ea83ceb60b7a90e412a8047	0.26945

Abbildung 3.24: Korrelationswerte der RevPAR-Werte zweier Hotels

Abbildung 3.24 offenbarte einen abweichenden Korrelationswert im Vergleich zu demjenigen, der bei der Betrachtung der Preisentwicklung ermittelt wurde. Darauf- hin wurde die Entscheidung getroffen, dass der Korrelationswert der RevPAR-Werte als aussagekräftiger betrachtet wird als derjenige der reinen Preisentwicklung. In- folgedessen wurde festgelegt, dass dieser Wert als Evaluation für die Ähnlichkeit zwischen den Hotels herangezogen wird.

Dieser Korrelationswert der RevPAR-Werte kann lediglich zur Evaluation der Modelle verwendet werden, da dieser Korrelationswert für ein Hotel nicht vorhanden ist.

3.1.5 Unbewachtes Lernen

Mit dem konstruierten Datensatz und der zusätzlichen Metrik zur Bestimmung ähnlicher Hotels erfolgt nun die Modellierung. Aufgrund der Unbekanntheit im Vorfeld, welche Hotels miteinander vergleichbar sind, wird ein Modell für unbewachtes Lernen implementiert. Unbewachtes Lernen bezeichnet eine Methode des maschinellen Lernens, bei der der Algorithmus eigenständig und ohne Überwachung Muster sowie Zusammenhänge in den Daten explorativ erkennt [8]. Die Grundidee besteht darin, jedes Hotel in einer vergleichbaren Form dem Modell zuzuführen. Dieses Modell erkennt eigenständig Muster und Ähnlichkeiten auf Basis der gegebenen Informationen.

Die Evaluation der unterschiedlichen Ansätze und Modelle erfolgt anhand der Benchmark-Hotels, welche in der Sektion zu den Benchmark-Hotels selektiert wurden. Ein Ansatz oder Modell wird als erfolgreich betrachtet, wenn es für die Benchmark-Hotels ein oder mehrere ähnliche Hotels identifiziert, die Korrelationswert von mindestens 0,8 haben.

Basismodell

Ein Basismodell ist im Grundlegenden ein einfaches Modell, das als Referenz in einem ML-Projekt dient. Seine Hauptfunktion besteht darin, die Ergebnisse trainierter Modelle in einen Kontext zu setzen. Basismodelle sind normalerweise wenig komplex und können nur geringe Vorhersagekraft haben. Trotzdem ist ihre Einbeziehung aus verschiedenen Gründen notwendig [9].

Einerseits dient das Basismodell dazu, einen initialen Einstieg in die Modellierung zu ermöglichen, dessen Ergebnisse anschließend mittels der zuvor ausgewählten Metrik evaluiert werden können. Hierbei wird ein vollständiger Workflow in vergleichsweise kurzer Zeit geschaffen. Andererseits fungiert das Basismodell als Re-

3 Ähnliche Hotels

ferenzpunkt, welcher durch weitere Modelle übertroffen werden soll. Ein beispielhaftes Szenario für ein solches Basismodell könnte ein Empfehlungssystem für Filme sein, bei dem stets die aktuell beliebtesten Filme auf der Plattform angeboten werden.

Im vorliegenden Kontext besteht das Basismodell darin, vier zufällige Hotels aus dem Gesamtdatensatz zu extrahieren und diese als ähnlich zu betrachten. Mit dem Basismodell, welches vier zufällige Hotels liefert, sieht die Evaluation wie folgt aus:

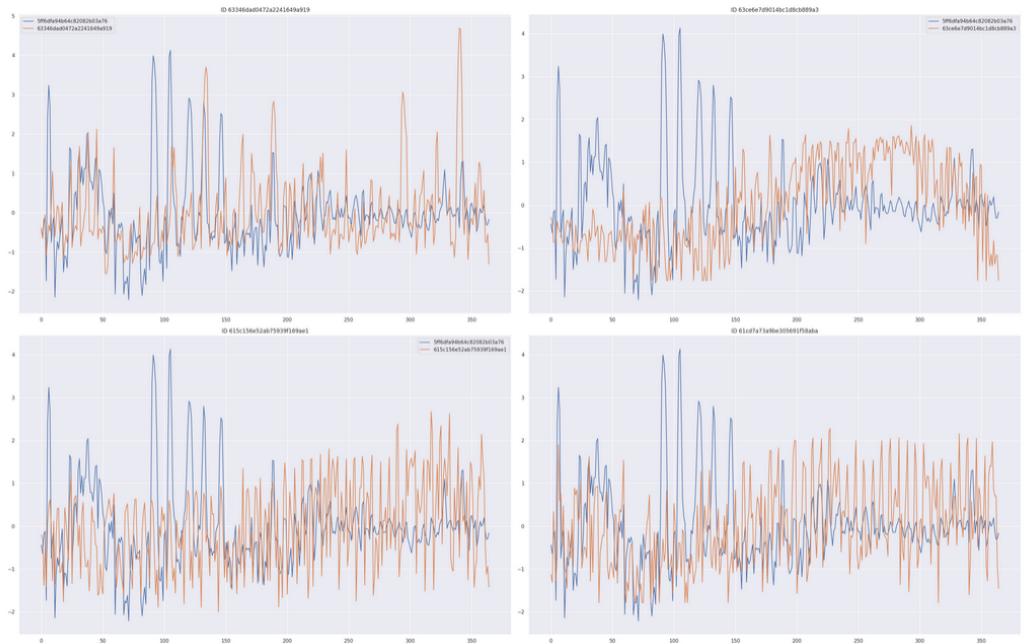


Abbildung 3.25: Basismodell Ergebnisse der RevPAR-Verläufe

5ff6dfa94b64c82082b03a76

63346dad0472a2241649a919	0.280356
63ce6e7d9014bc1d8cb889a3	-0.018887
615c156e52ab75939f169ae1	0.141771
61cd7a73a9be305691f58aba	0.180069
5ff6dfa94b64c82082b03a76	1.000000

Abbildung 3.26: Basismodell Ergebnisse RevPAR-Korrelationen

In den Abbildungen 3.25 und 3.26 sind die Ergebnisse für das Hotel mit der Identifikationsnummer *5ff6dfa94b64c82082b03a76* im Zusammenhang mit dem Basismodell ersichtlich. Sowohl Abbildung 3.25 als auch Abbildung 3.26 verdeutlichen, dass die vier Hotels, welche vom Basismodell ausgewählt wurden, keine deutliche Ähnlichkeit aufweisen. Des Weiteren zeigt Abbildung 3.26 eindeutig, dass das angestrebte Ziel, ein Hotel mit einem Korrelationswert von mindestens 0,8 zu identifizieren, nicht erreicht wurde. Infolgedessen hat das Basismodell die entsprechende Anforderung nicht erfüllt.

DBScan Modell

Die erste Idee nach dem Basismodell, um ähnliche Hotels zu finden, ist es den weitverbreiteten *Density-Based Spatial Clustering of Applications with Noise-Algorithmus* zu nutzen. Der *Density-Based Spatial Clustering of Applications with Noise-Algorithmus* kurz DBScan ist ein leistungsfähiger Ansatz zur Clusteranalyse, der darauf abzielt, Cluster in Datensätzen mit variabler Dichte zu identifizieren. Anders als klassische Clustering-Algorithmen, die auf geometrischen Formen oder Distanzmetriken basieren, nutzt DBScan die Dichte der Datenpunkte, um Cluster zu identifizieren [10].

Die Identifikation von Clustern innerhalb eines Datensatzes erfordert vom DBScan-Algorithmus zwei wesentliche Informationen:

- Mindestanzahl an Datenpunkten für die Bildung eines Clusters.
- Maximale Entfernung, in der Datenpunkte voneinander liegen dürfen.

Ein Vorteil des DBScan-Algorithmus besteht darin, dass keine vorherige Festlegung darüber getroffen werden muss, wie viele Cluster am Ende resultieren sollen. Es genügt die Angabe dieser beiden genannten Informationen, um die Bildung von Clustern zu ermöglichen.

Die zugrunde liegende Hypothese dieser Methodik besagt, dass der DBScan-Algorithmus alle ähnlichen Hotels in einem Cluster zusammenführt.

Vorbereitung des Datensatzes Der in Abbildung 3.18 dargestellte Datensatz enthält sowohl numerische als auch kategoriale Daten. Da der DBScan-Algorithmus ausschließlich mit numerischen Daten arbeiten kann, ist es erforderlich, den vor-

handenen Datensatz vor der Anwendung des Algorithmus zu transformieren. Zu diesem Zweck wird eine Pipeline erstellt, die auf den gesamten Datensatz angewendet wird. In einem ersten Schritt werden die kategorialen Daten mithilfe von *One-Hot-Encoding* umgewandelt. Darüber hinaus erfolgt eine Skalierung der bereits vorhandenen numerischen Daten.

Die Struktur der Pipeline ist wie folgt:

```
1 from sklearn.compose import ColumnTransformer
2 from sklearn.pipeline import make_pipeline
3 from sklearn.preprocessing import StandardScaler, OneHotEncoder
4
5 numerical_features = ['median_min', 'median_max']
6 categorical_features = ['art', 'region2']
7
8 numerical_pipeline = make_pipeline(StandardScaler())
9
10 categorical_pipeline = make_pipeline(OneHotEncoder())
11
12 preprocessor = ColumnTransformer(
13     transformers=[
14         ('num', numerical_pipeline, numerical_features),
15         ('cat', categorical_pipeline, categorical_features)
16     ])
17
18
19 pipeline = make_pipeline(preprocessor)
```

Listing 3.2: Datensatz Pipeline

Die in dem Listing 3.2 gezeigte Pipeline kann nun auf Datensatz angewendet werden. Dazu muss die Pipeline zunächst du den Datensatz abgestimmt werden, um dann den Datensatz zu Transformieren. Dies wird mit dem folgenden Listing veranschaulicht:

```
1 # Anpassen der Daten an die Pipeline
2 pipeline.fit(df)
3
4 # Transformieren der Daten
5 transformed_df = preprocessor.transform(df)
6
7 # Erstelle einen DataFrame aus den transformierten Daten
8 model_df = pd.DataFrame(transformed_df.toarray(),
9                           columns=numerical_features +
10                           list(preprocessor.named_transformers_['cat']
11                               .named_steps['onehotencoder']
12                               .get_feature_names_out(
13                                   categorical_features)))
```

Listing 3.3: Ausführung der Pipeline auf dem Datensatz

Modellbildung Nach erfolgter Transformation des Datensatzes kann dieser in den DBScan-Algorithmus eingegeben werden, um die Cluster zu generieren. Als Mindestanzahl an Datenpunkten für ein Cluster wird der Wert 2 festgelegt, da stets Paare von zwei Hotels gefunden werden sollen. Das Auffinden der Cluster erfolgt daraufhin lediglich durch einen Aufruf.

```

1 from sklearn.cluster import DBSCAN
2
3 # Define variables
4 eps = 0.8
5 min_samples = 2
6
7 # Create the DBScan algorithm
8 dbSCAN = DBSCAN(eps=eps, min_samples=min_samples)
9
10 # Add Cluster to df
11 model_df['Cluster'] = dbSCAN.fit_predict(model_df)

```

Listing 3.4: Ausführung des DBScan-Algorithmus

Das Listing 3.4 veranschaulicht die Initialisierung des DBScan-Algorithmus sowie die Erzeugung der individuellen Cluster. Diese Cluster werden anschließend der Datenreihe in der Spalte *Cluster* zugeordnet.

In einem nachfolgenden Schritt können die erstellten Cluster wie folgt visualisiert werden:

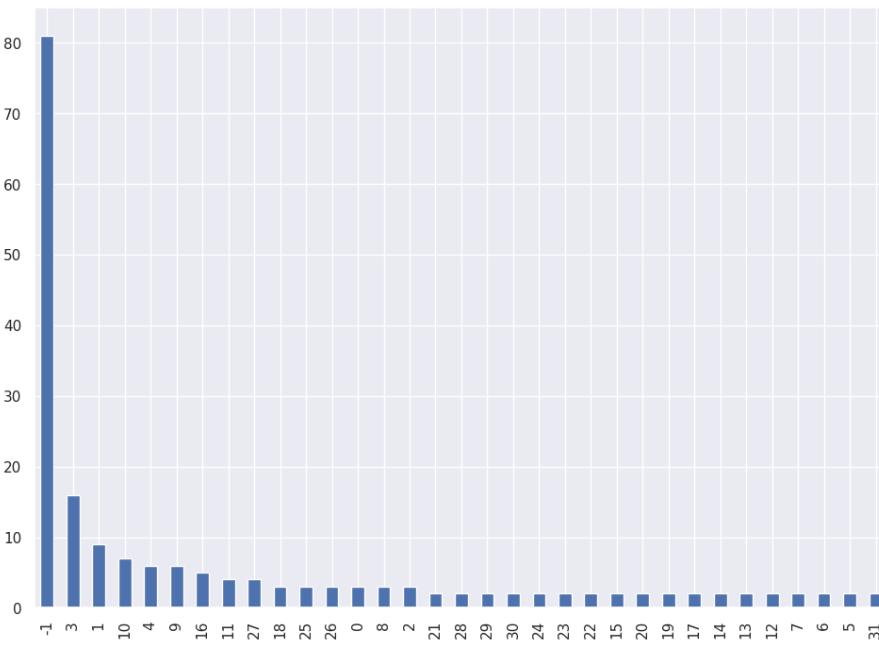


Abbildung 3.27: DBScan Clusters

Abbildung 3.27 zeigt die verschiedenen Cluster, die durch den DBScan-Algorithmus erstellt wurden. Es sind insgesamt 32 Cluster zu erkennen, beginnend mit Cluster 0, sowie eine weitere Kategorie -1, die anzeigt, dass für diese Hotels kein Cluster gefunden wurde. Bemerkenswert ist dabei, dass die Mehrheit dieser Hotels keinem Cluster zugeordnet wurde.

Ungeachtet dieser Beobachtung soll im nächsten Schritt überprüft werden, welche Hotels mit dem Benchmark-Hotel in einem Cluster vorhanden sind. Hierfür müssen die Hotel-IDs in einer zusätzlichen Spalte dem Dataframe hinzugefügt werden. Die nachfolgende Code-Sequenz im angegebenen Listing ermöglicht die Identifikation der Hotels, die mit dem Benchmark-Hotel in einem Cluster sind.

```

1 # Add hotel ids to the dataset with the clusters
2 model_df['hotel_id'] = df.index
3
4 # Get the row for the benchmark hotel
5 benchmark_row = model_df[model_df["hotel_id"] == str(benchmark_hotel_id)]
6
7 # Get the Cluster value of the benchmark hotel
8 benchmark_cluster = list(benchmark_row["Cluster"].values)[0]
9
10 # Get the rows with the same cluster as the benchmark hotel
11 benchmark_cluster_df = model_df[model_df["Cluster"] == benchmark_cluster]
12
13 # Get the hotel ids
14 hotel_ids = list(benchmark_cluster_df["hotel_id"].values)

```

Listing 3.5: Finden alle Hotel ID's innerhalb des gleichen Clusters

Die Variable *hotel_ids* enthält nun sämtliche Hotel-IDs, die zusammen mit dem Benchmark-Hotel in einem Cluster verortet sind. Initial besteht das Interesse darin, die identifizierten Hotels zu visualisieren. Daher soll im Anschluss angezeigt werden, welche Merkmale die identifizierten Hotels aufweisen.

	median_min	median_max	art	region2
company_id				
5ff6dfa94b64c82082b03a76	11500.0	54000.0	Stadthotel	Nuernberg
63734d7e84ee84c5d68a1d86	15900.0	61000.0	Stadthotel	Nuernberg
637353f228d9f119bb9e6d2b	12900.0	52900.0	Stadthotel	Nuernberg

Abbildung 3.28: Gefundene Hotels für das erste Benchmark-Hotel mit dem DBScan-Algorithmus

3 Ähnliche Hotels

Die Hotels, die im identifizierten Cluster zu finden sind, weisen tatsächlich Ähnlichkeiten zu den Merkmalen auf, die das Benchmark-Hotel mit der ID `5ff6dfa94b64c82082b03a76` charakterisieren. Da die Aussagekraft eines einzelnen Hotels begrenzt ist, soll das zweite Benchmark-Hotel ebenfalls mit einbezogen werden. Hierbei sollen auch für dieses Hotel zunächst die identifizierten Hotels anhand ihrer Merkmale bewertet werden.

company_id	median_min	median_max	art	region2
<code>63791654c755fe850934264c</code>	12900.0	30000.0	Ferienhotel	Niedersachsen-Kleinstadt
<code>61b84fc9d0acf9e96f19b205</code>	14500.0	18000.0	Ferienhotel	Niedersachsen-Kleinstadt
<code>64219fde25647f559402e857</code>	12900.0	20000.0	Ferienhotel	Niedersachsen-Kleinstadt
<code>645cb624b9b0cca6912ed4cc</code>	9650.0	13500.0	Ferienhotel	Niedersachsen-Kleinstadt

Abbildung 3.29: Gefundene Hotels für das zweite Benchmark-Hotel mit dem DBScan-Algorithmus

Unter Berücksichtigung des zweiten Hotels wird ersichtlich, dass der DBScan Hotels mit ähnlichen Merkmalen identifiziert. Dennoch lässt sich anhand der vorliegenden Merkmale nicht eindeutig feststellen, ob diese Hotels tatsächlich ähnliche Verläufe in Bezug auf den RevPAR aufweisen.

Evaluation Angesichts der Tatsache, dass allein anhand der Merkmale keine klare Aussage über die tatsächliche Ähnlichkeit der Hotels getroffen werden kann, erfolgt in der nachfolgenden Sektion eine Evaluierung der beiden Benchmark-Hotels.

Der Beginn dieser Evaluierung erfolgt mit dem Hotel, das die ID `5ff6dfa94b64c82082b03a76` trägt. Die Ergebnisse dieser Evaluation werden im Folgenden dargestellt:

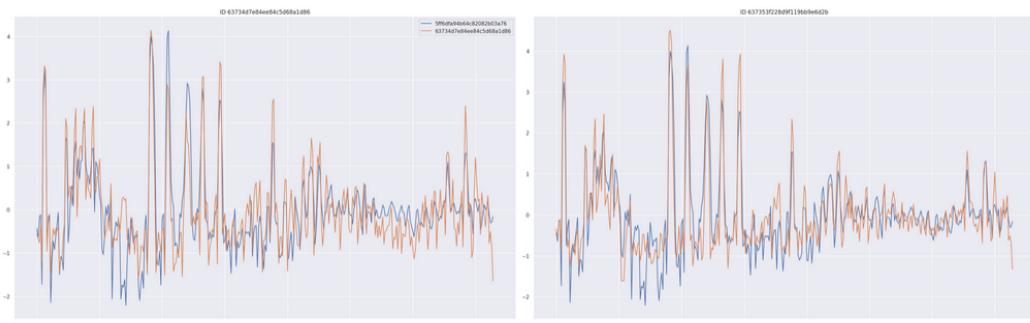


Abbildung 3.30: DBScan Ergebnisse der RevPAR-Verläufe für das erste Benchmark-Hotel

3 Ähnliche Hotels

5ff6dfa94b64c82082b03a76

5ff6dfa94b64c82082b03a76	1.000000
63734d7e84ee84c5d68a1d86	0.815371
637353f228d9f119bb9e6d2b	0.850404

Abbildung 3.31: DBScan Ergebnisse der RevPAR-Korrelationen für das erste Benchmark-Hotel

Die Abbildungen 3.30 und 3.30 verdeutlichen, dass die identifizierten Hotels die Voraussetzungen für einen Korrelationswert von mindestens 0,8 erfüllen. Die Ergebnisse für das Hotel mit der ID **5ff6dfa94b64c82082b03a76** erweisen sich demnach als zufriedenstellend. Allerdings ist eine gewisse Skepsis angebracht, da es auch rein zufällig sein könnte, dass diese Hotels Ähnlichkeiten in Bezug auf den RevPAR Verlauf aufweisen. Aus diesem Grund soll die gleiche Evaluation auch mit dem zweiten Hotel, welches die ID **645cb624b9b0cca6912ed4cc** trägt, durchgeführt werden.

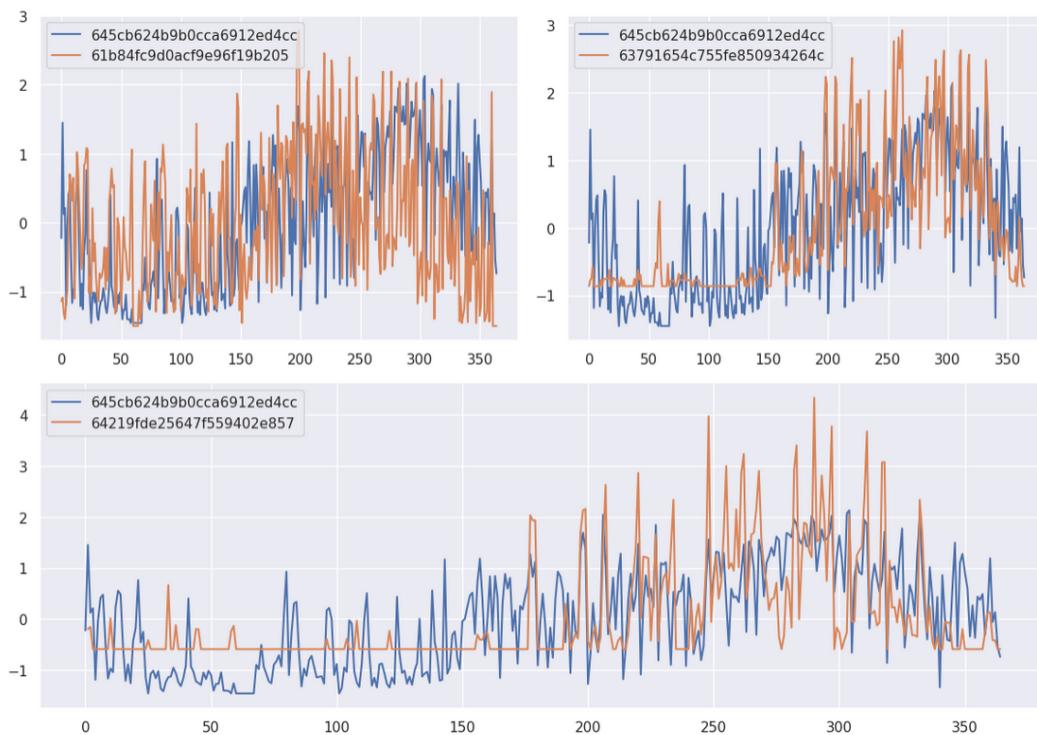


Abbildung 3.32: DBScan Ergebnisse der RevPAR-Verläufe für das zweite Benchmark-Hotel

645cb624b9b0cca6912ed4cc	
63791654c755fe850934264c	0.715351
61b84fc9d0acf9e96f19b205	0.415042
64219fde25647f559402e857	0.610250
645cb624b9b0cca6912ed4cc	1.000000

Abbildung 3.33: DBScan Ergebnisse der RevPAR-Korrelationen für das zweite Benchmark-Hotel

Die Evaluation des Hotels mit der ID *645cb624b9b0cca6912ed4cc* ergab, dass die Anforderungen an einen Korrelationswert von mindestens 0,8 nicht erfüllt wurden. Somit lässt sich ableiten, dass die Ähnlichkeit der Merkmale zwischen den einzelnen Hotels nicht zwangsläufig mit der Ähnlichkeit in Bezug auf die RevPAR-Werte in Verbindung steht.

Obwohl der DBScan-Algorithmus im Vergleich zum Basismodell verbesserte Ergebnisse lieferte, bleibt das Gesamtergebnis dennoch unbefriedigend. Besonders bedenklich ist dabei, dass die überwiegende Mehrheit der Hotels keinem Cluster zugeordnet werden konnten.

Doc2Vec Modell

In Anbetracht des unbefriedigenden Ergebnisses des DBScan-Algorithmus wird nun die Erprobung eines weiteren Modells in Betracht gezogen, um ähnliche Hotels zu identifizieren. Die nächste Alternative besteht in der Anwendung des Doc2Vec-Modells.

Doc2Vec ist ein maschinelles Lernverfahren, das dazu dient, Dokumente in einem kontinuierlichen Vektorraum abzubilden. Es wurde als Weiterentwicklung von Word2Vec konzipiert, einem etablierten Ansatz zur Repräsentation von Wörtern in einem semantischen Vektorraum. Das Hauptziel von Doc2Vec besteht darin, eine kontinuierliche Darstellung von Dokumenten zu generieren, um semantische Ähnlichkeiten zwischen diesen Dokumenten zu erfassen [11].

Die grundlegende Idee besteht darin, jedes Hotel in einem textbasierten Dokument zu repräsentieren und dem Doc2Vec-Modell die Aufgabe zu übertragen, ähnliche Dokumente zu identifizieren. Dieser Ansatz zielt darauf ab, zu verhindern, dass Hotels ohne jegliche Zuordnung zu anderen Hotels vorliegen.

Ähnlich wie beim DBScan, wo der Vorteil darin bestand, keine Cluster angeben zu müssen, weist auch das Doc2Vec-Modell den Vorteil auf, dass keinerlei Cluster explizit angegeben werden müssen. Zudem eröffnet sich hier ein weiterer Vorteil, der beim DBScan-Algorithmus nicht gegeben war. Die beiden Informationen *Art* und *Region* sind beide durch Strings repräsentiert. Da Doc2Vec mit einer textbasierten Repräsentation arbeitet, bedarf es keiner Umwandlung dieser beiden Merkmale mittels *One-Hot-Encoding*.

Vorbereitung des Datensatzes Auch in diesem Kontext erfordert der in Abbildung 3.18 dargestellte Datensatz eine vorherige Verarbeitung, um mit dem Doc2Vec-Modell kompatibel zu sein. Die Vorbereitung besteht darin, dem Datensatz eine zusätzliche Spalte hinzuzufügen, die sämtliche Informationen des Hotels als textbasiertes Dokument repräsentiert.

Im Folgenden wird eine Hilfsfunktion präsentiert, die dazu dient, die textbasierten Dokumente anhand der einzelnen Informationen zu erstellen.

```
1 # Copy the dataset to keep the original clean
2 doc2vec_df = df.copy()
3
4 # Get the columns of the dataset dynamic
5 column_names = doc2vec_df.columns
6
7 # Function to generate the document out of each columns
8 def concatenate_columns(row):
9     result = "Hoteleigenschaften: "
10    for column in column_names:
11        result += column + "=" + str(row[column]) + " "
12    return result
13
14 # Add the document in a new column "doc"
15 doc2vec_df['doc'] = doc2vec_df.apply(concatenate_columns, axis=1)
```

Listing 3.6: Hilfsfunktion zur Erzeugung von textbasierten Dokumenten

Durch die im Listing 3.6 präsentierten Codezeilen, wurde dem Datensatz eine neue Spalte namens *doc* hinzugefügt, die später als Eingabe für das Modell dienen soll.

Zur Veranschaulichung dieser neu erstellten *doc*-Spalte soll das erste Dokument als Beispiel dienen:

```
'Hoteleigenschaften: median_min=11500.0 median_max=54000.0 art=Stadthotel region2=Nuernberg '
```

Abbildung 3.34: Exemplarisches Dokument innerhalb des Datensatzes

Mithilfe dieser neuen Repräsentation kann nun im nächsten Schritt das Modell trainiert werden.

Modellbildung Nach der erfolgten Datenvorbereitung und der Erstellung der textbasierten Dokumente für jedes Hotel kann das Modell nun trainiert werden. Hierzu werden die Hotel-IDs zunächst als Index festgelegt. Anschließend wird der Datensatz in sogenannte *TaggedDocument* Objekte umgewandelt. Diese *TaggedDocument* Objekte erfassen die einzelnen Wörter innerhalb eines Dokuments sowie ein sogenanntes *Tag*, welches zur Identifikation des jeweiligen Dokuments dient. In diesem Kontext repräsentiert das *Tag* die Hotel-ID. Der letzte Schritt besteht darin, die *TaggedDocument* Objekte unter Verwendung einiger Hyperparameter dem Modell zuzuführen und das Training zu starten. Sobald das Modell trainiert ist, können ähnliche Dokumente zu einem ausgewählten Dokument, das durch die Hotel-ID identifiziert wird, gefunden werden.

Dieses Verhalten wird im nachfolgenden Listing dargestellt:

```
1 # Generate the TaggedDocuments with Hotel-ID as Tag
2 documents = [TaggedDocument(words=doc.split(), tags=[str(i)]) for i, doc in
   doc2vec_df["doc"].iteritems()]
3
4 # Train the Model
5 model = Doc2Vec(documents, vector_size=10, window=3, min_count=1, workers=4, epochs
   =1000, alpha=0.01)
6
7 # Get similar documents als tupel (ID, Doc)
8 similar_documents = model.dv.most_similar(str(benchmark_hotel_id), topn=4)
9
10 pred_hotel_ids = [tupel[0] for tupel in similar_documents]
```

Listing 3.7: Ausführung des Doc2Vec Modell

Nun stehen in der Variable *pred_hotel_ids* die Hotel-IDs, die anhand von dem Doc2Vec Modell als am ähnlichsten empfunden wurden. Diese können wie auch schon beim DBScan Algorithmus anhand ihrer Merkmale betrachtet werden.

3 Ähnliche Hotels

company_id	median_min	median_max	art	region2
63734d7e84ee84c5d68a1d86	15900.0	61000.0	Stadthotel	Nuernberg
61976b80479a653b3cd02f84	4750.0	26250.0	Stadthotel	Nuernberg
61e1a086f43cbe2cc4aac8b3	5950.0	10450.0	Businesshotel	Baden-Wuerttemberg-Kleinstadt
5d8ca3d41da4f20e1bef59	18100.0	54000.0	Stadthotel	Baden-Wuerttemberg-Grossstadt

Abbildung 3.35: Gefundene Hotels für das erste Benchmark-Hotel mit dem Doc2Vec Modell

Es zeigt sich, dass das Doc2Vec Modell ganz andere Hotels als ähnlich identifiziert hat. Im Gegensatz zum DBScan-Algorithmus scheinen die Merkmale der gefundenen Hotels lediglich Überlappungen aufzuweisen, anstatt sich zu teilen. Wie bereits in den vorhergehenden Abschnitten ersichtlich wurde, müssen sich die Merkmale nicht zwangsläufig überschneiden.

Evaluation Ungeachtet der jeweiligen Merkmale der einzelnen Hotels besteht die Möglichkeit, dass diese dennoch Ähnlichkeiten in dem RevPAR-Verlauf aufweisen. Diese Annahme soll im Folgenden überprüft werden.

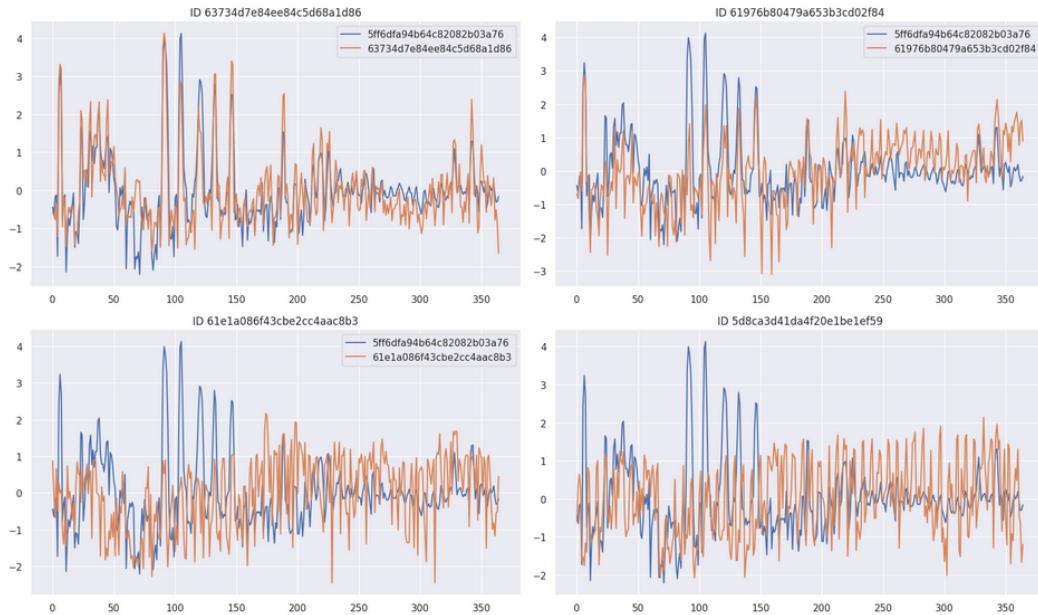


Abbildung 3.36: Doc2Vec Ergebnisse der RevPAR-Verläufe für das erste Benchmark-Hotel

Es offenbart sich, dass das Doc2Vec-Modell erheblich schlechter abschneidet als der DBScan-Algorithmus. Diese Feststellung wird bereits anhand des ersten Benchmark-

5ff6dfa94b64c82082b03a76	
63734d7e84ee84c5d68a1d86	0.815371
61976b80479a653b3cd02f84	0.499787
61e1a086f43cbe2cc4aac8b3	0.154515
5d8ca3d41da4f20e1be1ef59	0.186729
5ff6dfa94b64c82082b03a76	1.000000

Abbildung 3.37: Doc2Vec Ergebnisse der RevPAR-Korrelationen für das erste Benchmark-Hotel

Hotels ersichtlich, und es wird daher auf die detaillierte Evaluation dieses Benchmark-Hotels verzichtet.

Unbewachtes Lernen Fazit

Sowohl der Einsatz des DBScan-Algorithmus als auch der Ansatz des Doc2Vec-Modells haben sich als unzureichend erwiesen. Trotz der Identifikation von Hotels in beiden Ansätzen lässt sich im Vorfeld nicht sicherstellen, ob diese tatsächlich als ähnlich zueinander betrachtet werden können. Aufgrund dieser Erkenntnisse wurde die Entscheidung getroffen, beide Ansätze zu verwerfen und eine neue Methodik zur Identifizierung ähnlicher Hotels zu entwickeln.

3.1.6 Überwachtes Lernen

Beide unbewachte Ansätze erwiesen sich als nicht zufriedenstellend, dennoch haben sie wertvolle Erkenntnisse geliefert. Es wurde festgestellt, dass die erforderlichen Korrelationswerte für eine Evaluierung vorhanden sind, mit Ausnahme von neuen Hotels ohne entsprechende Daten. Nichtsdestotrotz könnten die bestehenden Korrelationswerte trotzdem genutzt werden, um ein Modell zu trainieren. Die Idee besteht darin, die beschreibenden Merkmale eines Hotels in ein trainiertes Modell einzuspeisen, um die Korrelationswerte vorherzusagen, anstatt direkt ähnliche Hotels zu prognostizieren.

Konkret sieht der Ansatz wie folgt aus: Jedes Hotel im Datensatz wird mit jedem anderen Hotel kombiniert, wodurch ein Datensatz der Größe *Anzahl der Hotels* \times *Anzahl der Hotels* entsteht. Zu jeder Kombination wird der Korrelationswert in einer separaten Spalte hinzugefügt, die als Zielvariable für die Vorhersage dient. Ein Modell wird mithilfe dieser Daten und der jeweiligen Zielvariable trainiert. Bei der Vorhersage der Korrelationswerte für ein neues Hotel wird dieses mit jedem anderen Hotel kombiniert und in das Modell eingebracht. Das Modell gibt schließlich die vorhergesagten Korrelationswerte aus.

Um diese theoretische Idee zu konkretisieren, werden im folgenden Abschnitt die einzelnen Schritte veranschaulicht.

Vorbereitung des Datensatzes

Um die beschriebene Idee in die Tat umzusetzen, bedarf es zunächst einer Modifikation des Datensatzes, welcher in Abbildung 3.18 dargestellt ist. Jede Zeile soll mit jeder anderen Zeile kombiniert werden. Dies wird durch die nachfolgenden Codezeilen realisiert:

```
1 # Get Hotel ID as separate column
2 model_df = model_df.reset_index()
3
4 # Repeat the rows 2 times
5 repeat = 2
6
7 # Get all row combinations
8 combinations = list(product(model_df.index, repeat=repeat))
9
10 # Create the empty result dataframe
11 combined_df = pd.DataFrame(columns=[f"{column}_{i+1}" for i in range(repeat) for
12                                     column in model_df.columns])
13
14 # Combine all combinations in the result dataframe
15 for combination in combinations:
16     row1 = model_df.iloc[combination[0]].values
17     row2 = model_df.iloc[combination[1]].values
18     new_row = pd.Series(list(row1) + list(row2), index=combined_df.columns)
     combined_df = combined_df.append(new_row, ignore_index=True)
```

Listing 3.8: Erstellen des kombinierten Datensatzes

Mithilfe der im Listing 3.8 präsentierten Codezeilen wurden sämtliche Zeilen erfolgreich miteinander kombiniert. Anschließend kann der Korrelationswert jeder Kombination als neue Spalte *target* hinzugefügt werden.

3 Ähnliche Hotels

Der resultierende Datensatz, welcher für das Modell verwendet werden soll, sieht wie folgt aus¹:

	median_min_1	median_max_1	art_1	region2_1	median_min_2	median_max_2	art_2	region2_2	target
0	11500.0	54000.0	Stadthotel	Nuernberg	11500.0	54000.0	Stadthotel	Nuernberg	1.000000
1	11500.0	54000.0	Stadthotel	Nuernberg	9000.0	21500.0	Ferienhotel	Baden-Wuerttemberg-Kleinstadt	0.249577
2	11500.0	54000.0	Stadthotel	Nuernberg	8650.0	59650.0	Stadthotel	Koeln	0.280356
3	11500.0	54000.0	Stadthotel	Nuernberg	10350.0	16550.0	Villa	Niedersachsen-Kleinstadt	-0.083118
4	11500.0	54000.0	Stadthotel	Nuernberg	8200.0	41000.0	Stadthotel	Hamburg	0.086656
5	11500.0	54000.0	Stadthotel	Nuernberg	6000.0	27900.0	Ferienhotel	Stuttgart	0.086735
6	11500.0	54000.0	Stadthotel	Nuernberg	10800.0	13900.0	Ferienhotel	Bayern-Kleinstadt	0.320706
7	11500.0	54000.0	Stadthotel	Nuernberg	6800.0	12800.0	Ferienhotel	Nordrhein-Westfalen-Kleinstadt	0.143902
8	11500.0	54000.0	Stadthotel	Nuernberg	6650.0	11650.0	Ferienhotel	Baden-Wuerttemberg-Grossstadt	0.138306
9	11500.0	54000.0	Stadthotel	Nuernberg	11500.0	18000.0	Ferienhotel	Mecklenburg-Vorpommern-Kleinstadt	0.059813

Abbildung 3.38: Datensatz bestehend aus den einzelnen Kombinationen

Modellbildung

Für das überwachte Lernen wird CatBoost verwendet, ein hochentwickelter Machine-Learning-Algorithmus, der gezielt auf die Vorhersage von kategorialen Variablen in Datensätzen abzielt und auf dem Gradient Boosting Framework basiert. Eine signifikante Eigenschaft von CatBoost ist seine einzigartige Methode zur Behandlung von kategorialen Merkmalen, bekannt als *Kategorie-Binning*. Diese Methode befähigt den Algorithmus dazu, die internen Strukturen kategorialer Variablen besser zu erfassen und sie effektiv in den Trainingsprozess einzubeziehen, was zu präziseren Vorhersagen führt [12].

Angesichts der Fülle an kategorialen Variablen im vorliegenden Datensatz erweist sich der CatBoost-Algorithmus als besonders geeignet für diese spezifische Aufgabe. Demzufolge kann der Datensatz in seiner aktuellen Form verwendet werden, ohne dass eine umfangreiche Vorverarbeitung erforderlich ist.

Im folgenden Abschnitt werden die erforderlichen Codezeilen präsentiert, um das Modell zu trainieren und Vorhersagen für das Benchmark-Hotel zu generieren:

```
1 # Get test data
2 test = combined_df[combined_df["company_id_1"] == str(benchmark_hotel_id)]
3 # Get train data
```

¹Die Hotel-IDs wurden zur besseren Veranschaulichung entfernt

3 Ähnliche Hotels

```
4 train = combined_df.drop(combined_df[(combined_df["company_id_1"] == str(benchmark_hotel_id)) | (combined_df["company_id_2"] == str(benchmark_hotel_id))].index)
5 # Get X and y from test and train
6 y_train = train["target"]
7 X_train = train.drop("target", axis=1)
8 y_test = test["target"]
9 X_test = test.drop("target", axis=1)
10 # Save important informations
11 hotel_test_df = X_test[["company_id_1", "company_id_2"]]
12 hotel_test_df["target"] = list(y_test)
13 # Drop Company ID
14 X_train = X_train.drop(['company_id_1', 'company_id_2'], axis=1)
15 X_test = X_test.drop(['company_id_1', 'company_id_2'], axis=1)
16 # Define cat features
17 cat_features = ["art_1", "region2_1", "art_2", "region2_2"]
18 # Define and train model
19 model = cb.CatBoostRegressor()
20 model.fit(X_train, y_train, cat_features=cat_features)
21 # Get predictions
22 y_pred = model.predict(X_test)
23 # Add predictions to the information df
24 hotel_test_df["predictions"] = list(y_pred)
```

Listing 3.9: Erzeugung der Vorhersagen von ähnlichen Hotels mittels CatBoost

Mithilfe der Codezeilen im Listing 3.9 wurde ein zusätzliches DataFrame erstellt, das die Hotel-IDs sowie deren Vorhersagen enthält. Dieses DataFrame wird im Folgenden präsentiert:

	company_id_1	company_id_2	target	predictions
34368	5ff6dfa94b64c82082b03a76	62416aec716e701c3cab690d	0.059813	0.201074
34369	5ff6dfa94b64c82082b03a76	5fc4e11c3c09ba0c169c93b3	0.088331	0.232576
34370	5ff6dfa94b64c82082b03a76	644bc41fe3a0a0bb2ae02f6a	-0.040648	0.146323
34371	5ff6dfa94b64c82082b03a76	637c7fe0389d0e567d998708	0.061575	0.192539
34372	5ff6dfa94b64c82082b03a76	6328577c12d28300fc40e7a9	0.229205	0.257228
34373	5ff6dfa94b64c82082b03a76	61976b80479a653b3cd02f84	0.499787	0.725910
34374	5ff6dfa94b64c82082b03a76	627a57e5f7461fd18585a1c4	0.168608	0.238727
34375	5ff6dfa94b64c82082b03a76	6247262ad7da93fb2b9ac50d	0.087450	0.115829
34376	5ff6dfa94b64c82082b03a76	647af947bd00fc8bc4ccf335	0.015964	0.173107
34377	5ff6dfa94b64c82082b03a76	649eb7253dc142bc919f6c8c	0.265457	0.223732

Abbildung 3.39: Datensatz der Hotel-IDs und deren Korrelationswert

Es besteht nun die Möglichkeit, verschiedene Operationen auf dem DataFrame auszuführen. Zum Beispiel kann auf dem DataFrame gemäß der Bedingung in Ab-

schnitt *Unbewachtes Lernen* gefiltert werden, die besagt, dass Hotelkorrelationen mit mindestens einem Korrelationswert von 0,8 gefunden werden sollen.

Evaluation

Im Gegensatz zu früheren Modellen wurde bei diesem Modell auf eine umfassende Evaluation verzichtet, da der prognostizierte Korrelationswert ausreichend ist, um ähnliche Hotels zu identifizieren. Es wurde lediglich versucht, den R2-Score mithilfe von Hyperparameteranalysen mittels Random Search **AdapRandomSearch4.05.2009** und Grid Search [13] zu verbessern.

3.1.7 Finden von ähnlichen Hotels Fazit

In dieser Sektion wurden verschiedene Ansätze zum Finden von ähnlichen Hotels erforscht, beginnend mit dem Einsatz von unbewachtem Lernen unter Verwendung der Algorithmen DBScan und Doc2Vec. Die Ergebnisse dieser Ansätze waren gemischt, wobei beide Algorithmen gewisse Schwächen aufwiesen, darunter die Tendenz, Hotels als ähnlich zu klassifizieren, die in Wirklichkeit nicht zueinander passen.

Im Anschluss wurde der Fokus auf überwachtes Lernen verlagert, wobei der CatBoost-Algorithmus eingesetzt wurde. Dieser Ansatz erwies sich als vielversprechend, da er in der Lage war, eine Vorhersage darüber zu treffen, wie ähnlich Hotels zueinander sind. Die Entscheidung, den CatBoost-Ansatz zu verfolgen, wurde getroffen, da bei diesem Ansatz auch ein Korrelationswert vorhanden ist, an dem sich orientiert werden kann.

3.2 Modifiziertes RevPAR-Modell

In der vorherigen Sektion wurde ein vielversprechender Ansatz zur Identifizierung ähnlicher Hotels entwickelt und evaluiert, wobei erfolgreich der CatBoost-Algorithmus eingesetzt wurde, um präzise Vorhersagen über die Ähnlichkeit zwischen verschiedenen Hotels zu treffen. Basierend auf diesen Ergebnissen wird im folgenden Abschnitt ein neuer Schwerpunkt gesetzt, der darauf abzielt, die Erkenntnisse aus dem

3 Ähnliche Hotels

Ähnlichkeitsmodell zu nutzen und ein weiteres Modell zu entwickeln.

Der Schwerpunkt dieser Sektion liegt auf der Modifikation des bereits vorhandenen RevPAR-Modells, das darauf abzielt, die RevPAR-Werte für ein Hotel vorherzusagen. Die vorgeschlagene Methode besteht darin, für jedes ähnliche Hotel ein eigenes RevPAR-Modell zu erstellen und den Durchschnitt der Ergebnisse zu ermitteln. Dabei werden die Daten der ähnlichen Hotels so angepasst, dass das Modell unabhängig vom spezifischen Hotel verwendet werden kann. Diese Sektion beginnt ebenfalls mit der Datenvorverarbeitung, bevor mit der Datenanalyse und Modellierung fortgefahren wird.

3.2.1 Datenvorverarbeitung

Die Datenvorverarbeitung dient dazu, zu ermitteln, welche Daten bereits vorhanden sind und wie sie modifiziert werden müssen, um die Entwicklung eines generellen Modells zu ermöglichen.

	stay_date	booking_date	available_rooms	booked_rooms	revenue	occupancy	current_revpar	revPAR	final_occupancy
0	2018-01-31 23:00:00+00:00	2018-01-31 23:00:00+00:00	51.0	38.0	1284211.0	0.745098	25180.607843	25180.607843	0.745098
1	2018-02-01 23:00:00+00:00	2018-01-31 23:00:00+00:00	51.0	31.0	1039557.0	0.607843	20383.470588	20383.470588	0.607843
2	2018-02-01 23:00:00+00:00	2018-02-01 23:00:00+00:00	51.0	31.0	1039557.0	0.607843	20383.470588	20383.470588	0.607843
3	2018-02-02 23:00:00+00:00	2018-01-31 23:00:00+00:00	51.0	20.0	578599.0	0.392157	11345.078431	12329.137255	0.450980
4	2018-02-02 23:00:00+00:00	2018-02-01 23:00:00+00:00	51.0	20.0	578599.0	0.392157	11345.078431	12329.137255	0.450980

Abbildung 3.40: Bereits vorhandene Hotelspezifischen Daten

Abbildung 3.40 veranschaulicht beispielhaft die Vergangenheitsdaten, wie sie aus der Datenbank abgerufen werden können. Dabei repräsentiert die Spalte *revPAR* den zu prognostizierenden Wert. Es ist zu erkennen, dass diese Werte sich spezifisch auf das ausgewählte Hotel beziehen und daher in ihrer aktuellen Form nicht für das Training verwendet werden können. Die vorgeschlagene Vorgehensweise besteht darin, sämtliche hotelspezifischen Spalten durch den Durchschnittswert der jeweiligen Spalte zu skalieren, um einen standardisierten Wert zu erlangen, der für jedes Hotel verwendet werden kann.

Im Folgenden werden mehrere Funktionen definiert, die einerseits weitere Daten zum Datensatz hinzufügen sollen und andererseits die vorhandenen Spalten, wie oben beschrieben, durch den Durchschnittswert normieren sollen:

3 Ähnliche Hotels

```
1 # list holiday
2 list_holidays = ['holiday_Neujahr', 'holiday_Christi Himmelfahrt', 'holiday_Erster
Mai',
3                 'holiday_Karfreitag', 'holiday_Ostermontag', 'holiday_Pfingstmontag',
4                 'holiday_Tag der Deutschen Einheit', 'holiday_Reformationstag', '
5                 'holiday_Erster Weihnachtstag',
6                 'holiday_Zweiter Weihnachtstag',
7                 ]
8
9 # list months
10 list_months = ['start_date_monthName_January', 'start_date_monthName_February', ',
11               'start_date_monthName_March',
12               'start_date_monthName_April', 'start_date_monthName_May', '
13               'start_date_monthName_June',
14               'start_date_monthName_July', 'start_date_monthName_August', '
15               'start_date_monthName_September',
16               'start_date_monthName_October', 'start_date_monthName_November', '
17               'start_date_monthName_December']
18
19 # list days
20 list_days = ['start_date_weekDayName_Monday', 'start_date_weekDayName_Tuesday', ',
21               'start_date_weekDayName_Wednesday',
22               'start_date_weekDayName_Thursday', 'start_date_weekDayName_Friday', '
23               'start_date_weekDayName_Saturday',
24               'start_date_weekDayName_Sunday']
25
26 # Vorbereiten der Hotelfeatures
27 def add_holidays_as_bool(df):
28     ger_holidays = holidays.Germany()
29     df['holiday'] = df.stay_date.apply(lambda x: ger_holidays.get(x.date()))
30     df['holiday'] = df['holiday'].notnull().astype(int)
31     return df
32
33 def add_month_feature(df):
34     df.insert(loc=1, column='stay_date_monthName', value=df['stay_date'].dt.
35               month_name())
36     return df
37
38 def add_weekDay_feature(df):
39     df.insert(loc=1, column='stay_date_weekDayName', value=df['stay_date'].dt.
40               day_name())
41     return df
42
43 def add_year(df):
44     df.insert(loc=1, column='stay_date_year', value=df['stay_date'].dt.year)
45     return df
46
47 def feature_ADR(df):
48     df['ADR'] = df['revenue'] / df['booked_rooms']
49     df.loc[df['booked_rooms'] == 0, 'ADR'] = 0
50
51     durchschnitt = np.mean(df['ADR'])
52     df["scaled_current_adr"] = df['ADR'] / durchschnitt
53
54     return df
```

```

46
47 def add_time_to_arrival(df):
48     df['time_to_arrival'] = df['stay_date'] - df['booking_date']
49     df['time_to_arrival'] = df['time_to_arrival'].dt.days
50     return df
51
52 def add_stay_date_str(df):
53     df["stay_date_str"] = df["stay_date"].dt.strftime("%d.%m.%Y")
54     return df
55
56 def add_scaled_revpar_values(df):
57     durchschnitt = np.mean(df['revPAR'])
58     df["target"] = df['revPAR'] / durchschnitt
59     df["scaled_current_revpar"] = df['current_revpar'] / durchschnitt
60     return df

```

Listing 3.10: Hilfsfunktionen für das RevPAR Modell

	stay_date_year	stay_date_weekDayName	stay_date_monthName	holiday	time_to_arrival	scaled_current_adr	scaled_current_revpar	occupancy	target
0	2018	Wednesday	January	0	0	3.135200	3.507706	0.745098	3.507706
1	2018	Thursday	February	0	1	3.110994	2.839456	0.607843	2.839456
2	2018	Thursday	February	0	0	3.110994	2.839456	0.607843	2.839456
3	2018	Friday	February	0	2	2.683862	1.580391	0.392157	1.717472
4	2018	Friday	February	0	1	2.683862	1.580391	0.392157	1.717472

Abbildung 3.41: Datensatz nach der Ausführung der Hilfsfunktionen

Abbildung 3.41 zeigt den Datensatz nach der Ausführung der Funktionen in Listing 3.10. Es ist zu erkennen, dass der Datensatz nun ausschließlich normierte Werte enthält. Im Folgenden wird eine Datenanalyse auf diesem Datensatz durchgeführt.

3.2.2 Datenanalyse

Wie bereits in Abschnitt *Allgemeine Datenanalyse* erläutert wurde, ist die Datenanalyse eine der essenziellsten Komponenten bei der Modellierung eines Modells. Im Folgenden wird daher untersucht, welche hotelspezifischen Daten vorhanden sind und wie sie effektiv genutzt werden können.

Als erster Schritt soll erforscht werden, ob der gesamte Zeitraum des Datensatzes valide ist und verwendet werden kann. Dazu wird zunächst der durchschnittliche normierte RevPAR, repräsentiert durch die Spalte *target*, mit den einzelnen Jahren verglichen.

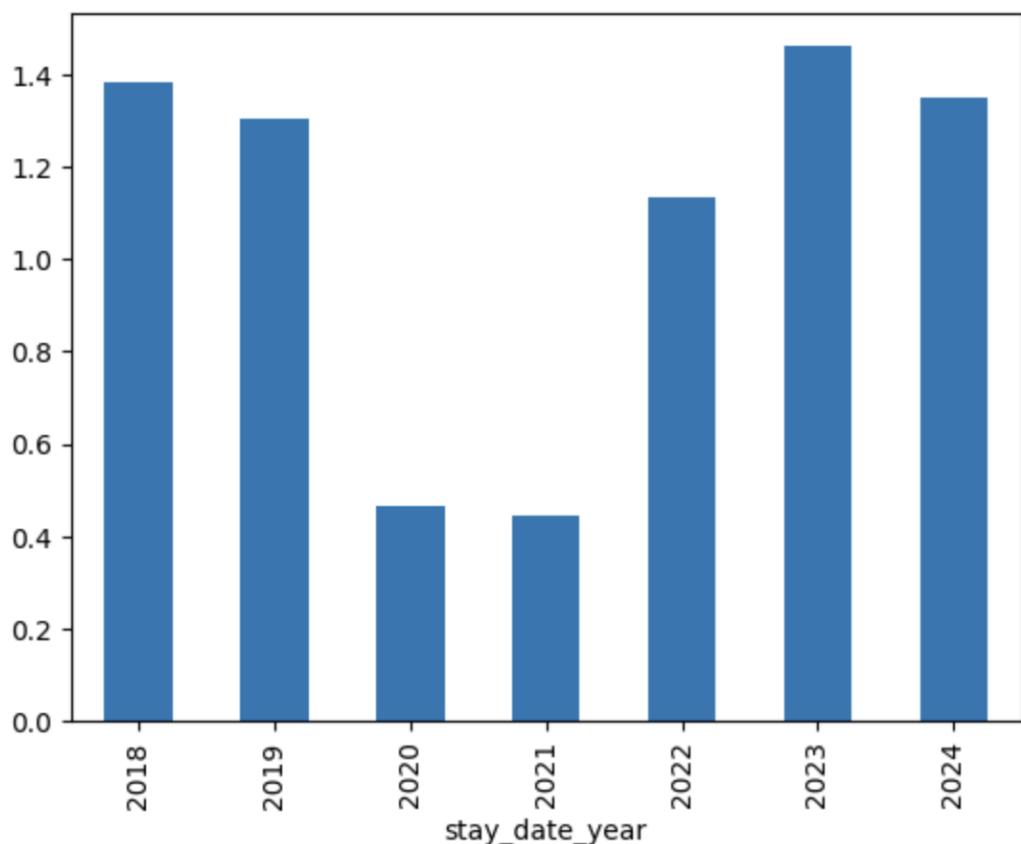


Abbildung 3.42: Durchschnittlich normierte RevPAR-Werte pro Jahr

Eine bedeutsame Erkenntnis aus Abbildung 3.42 ist, dass nicht alle Jahre gleich sind. Insbesondere weisen die Jahre 2020 und 2021 einen deutlich niedrigeren durchschnittlichen RevPAR auf als die anderen Jahre. Diese Unterschiede können auf verschiedene Faktoren zurückzuführen sein, wobei die Zeit der Covid-19-Pandemie einen wesentlichen Beitrag zu dieser Abweichung geleistet haben dürfte, da viele Hotels zu dieser Zeit vorübergehend schließen mussten. Es wäre daher unangemessen, diese Informationen in das Modell einzubeziehen, da sie den Datensatz verfälschen würden. Da die Spalte *stay_date_year* außer Ausreißern keine weiteren relevanten Informationen bietet, wird dieses Feature im weiteren Verlauf aus dem Datensatz entfernt.

Des Weiteren soll mit derselben Methodik überprüft werden, ob die Features *stay_date_weekDayName*, *stay_date_monthName* und *holiday* einen signifikanten Einfluss auf den normierten RevPAR haben.

3 Ähnliche Hotels

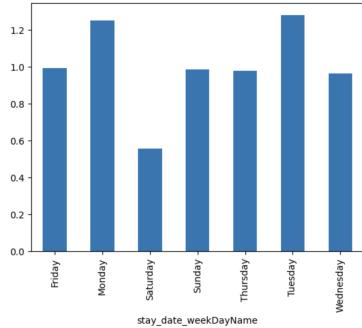


Abbildung 3.43: Durchschnittlich normierte RevPAR-Werte pro Tag

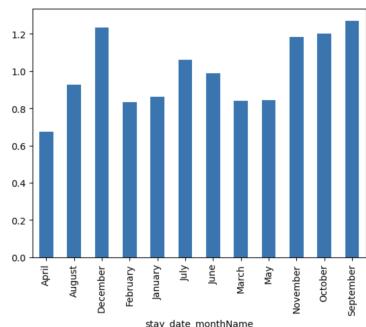


Abbildung 3.44: Durchschnittlich normierte RevPAR-Werte pro Monat

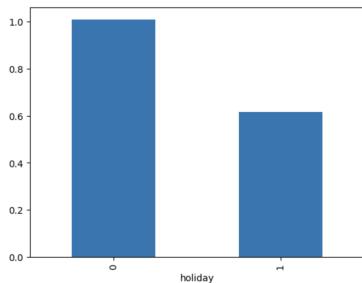


Abbildung 3.45: Durchschnittlich normierte RevPAR-Werte Ferientage vs. nicht-Ferientage

Anhand der Abbildungen 3.43, 3.44 und 3.45 lässt sich erkennen, dass jedes dieser Features einen signifikanten Einfluss auf den normierten RevPAR hat und daher im Datensatz beibehalten werden sollte.

Zum Abschluss sollen noch die Spalten *scaled_current_adr*, *scaled_current_revpar* und *occupancy* auf ihren Einfluss überprüft werden. Hierzu wird im Folgenden eine Korrelationsmatrix erstellt, um die Korrelation dieser Spalten mit dem normierten RevPAR zu bestimmen.

3 Ähnliche Hotels

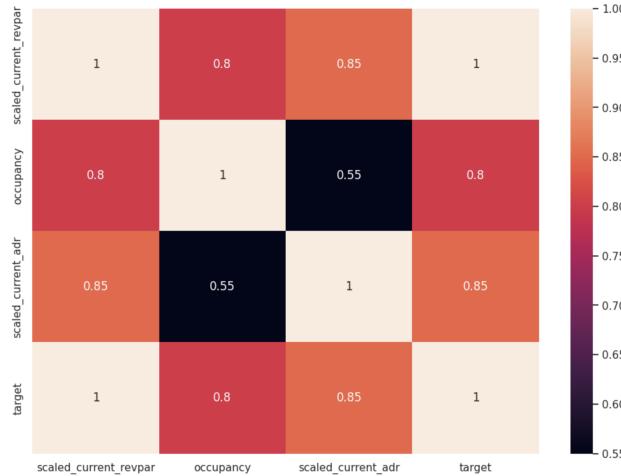


Abbildung 3.46: Korrelationsmatrix der verbleibenden Spalten

Bei Betrachtung der unteren Zeile *target* ist festzustellen, dass alle aufgelisteten Features eine signifikante Korrelation zur Zielvariable aufweisen. Daher sollten sie auch im Datensatz berücksichtigt werden.

Der resultierende Datensatz, welcher nach der Entfernung der Ausreißer und unnötigen Feature entsteht, sieht wie folgt aus:

	stay_date_weekDayName	stay_date_monthName	holiday	time_to_arrival	scaled_current_adr	scaled_current_revpar	occupancy	target
0	Wednesday	January	0	0	3.135200	3.507706	0.745098	3.507706
1	Thursday	February	0	1	3.110994	2.839456	0.607843	2.839456
2	Thursday	February	0	0	3.110994	2.839456	0.607843	2.839456
3	Friday	February	0	2	2.683862	1.580391	0.392157	1.717472
4	Friday	February	0	1	2.683862	1.580391	0.392157	1.717472
5	Friday	February	0	0	2.536224	1.717472	0.450980	1.717472
6	Saturday	February	0	3	2.445435	0.635535	0.173077	0.635535
7	Saturday	February	0	2	2.445435	0.635535	0.173077	0.635535
8	Saturday	February	0	1	2.445435	0.635535	0.173077	0.635535
9	Saturday	February	0	0	2.445435	0.635535	0.173077	0.635535

Abbildung 3.47: Finaler Datensatz für das RevPAR-Modell

3.2.3 Modellbildung

Nachdem der vorbereitete Datensatz vorliegt, wird das Modell implementiert. Aufgrund des Vorhandenseins vieler kategorischer Daten wird erneut das CatBoost-Modell verwendet. Die Implementierung erfolgt ähnlich wie in der vorherigen Sektion zur Identifizierung ähnlicher Hotels.

```
1 import catboost as cb
```

```

2
3 cat_features = [
4     "stay_date_weekDayName",
5     "stay_date_monthName",
6     "holiday",
7     "time_to_arrival"
8 ]
9
10 result_dict = dict()
11
12 for id in silimar_hotel_ids:
13     X_train, y_train, X_test, y_test = get_data(id)
14     model = cb.CatBoostRegressor()
15     model.fit(X_train, y_train, cat_features=cat_features)
16     y_pred = model.predict(X_test)
17     result_dict[id] = y_pred
18
19 df = pd.DataFrame(result_dict)
20 overall_preds = df.astype(float).mean(axis=1)

```

Listing 3.11: Erzeugung der RevPAR-Vorhersagen

Die Code-Zeilen in Listing 3.11 erzeugen die Variable *overall_preds*, die eine Liste von normierten RevPAR-Werten darstellt. Diese normierten RevPAR-Werte werden als Vorhersage für das Hotel ohne Vergangenheitsdaten verwendet.

3.2.4 Evaluation

Nachdem im vorherigen Abschnitt die Vorhersage normierter RevPAR-Werte mittels eines definierten Code-Abschnitts erfolgte, rückt nun die Frage nach der Genauigkeit dieser Vorhersagen in den Vordergrund. Diese Sektion widmet sich daher einer eingehenden Untersuchung der Wirksamkeit der vorhergesagten Werte für das Jahr 2023. Ziel dieser Evaluation ist es, die Übereinstimmung der prognostizierten Werte mit den tatsächlichen Daten zu bewerten und die Zuverlässigkeit des Modells bei der Vorhersage zu prüfen.

Um dieses Ziel zu erreichen, werden verschiedene Metriken wie der R2-Score, der Root-Mean-Square-Error und der Mean-Absolute-Error verwendet, um die Abweichungen zwischen den Vorhersagen und den realen Werten zu quantifizieren. Darüber hinaus erfolgt ein direkter Vergleich zwischen dem entwickelten Modell und dem aktuellen Live-Modell, das derzeit von Kunden verwendet wird. Diese Gegenüberstellung ermöglicht es, fundierte Entscheidungen über das weitere Vorgehen zu

3 Ähnliche Hotels

treffen, basierend auf einem direkten Vergleich der Leistungsfähigkeit beider Modelle.

	similar_model	live_model
r2_score	0.377454	0.526812
RMSE	0.476737	0.415633
MAE	0.374274	0.316122

Abbildung 3.48: Evaluation der beiden Modelle für das erste Benchmark-Hotel

Die Darstellung in Abbildung 3.48 präsentiert die Ergebnisse der einzelnen Modelle und ermöglicht gleichzeitig einen direkten Vergleich zwischen ihnen. Dabei fällt auf, dass das aktuell verwendete Live-Modell im Vergleich zum Similar-Modell eine bessere Leistung für dieses spezifische Hotel aufweist. Es ist jedoch wichtig zu beachten, dass das Similar-Modell gänzlich auf hotel-spezifische Daten verzichtet und trotzdem nur eine geringfügige Differenz aufweist.

Die Berücksichtigung eines einzelnen Hotels allein für diese Evaluation könnte als begrenzt angesehen werden. Daher wird die Analyse um das zweite Benchmark-Hotel erweitert, um festzustellen, wie die einzelnen Modelle dort abschneiden. Dies ermöglicht eine umfassendere Beurteilung der Leistungsfähigkeit der Modelle über verschiedene Kontexte hinweg.

	similar_model	live_model
r2_score	0.503561	0.151640
RMSE	0.548012	0.716386
MAE	0.379931	0.449965

Abbildung 3.49: Evaluation der beiden Modelle für das zweiten Benchmark-Hotel

Die Einbeziehung der Ergebnisse des zweiten Benchmark-Hotels, wie sie in Abbildung 3.49 dargestellt sind, verdeutlicht signifikante Unterschiede in der Leistung des Live-Modells. In diesem Szenario erweist sich das Similar-Modell als deutlich überlegen, obwohl es ohne hotel-spezifische Daten arbeitet. Diese Erkenntnis unterstreicht, dass die Modelle je nach individuellen Hotelkontexten unterschiedlich gut abschneiden können.

3.2.5 Zwischenfazit

Die präsentierten Ergebnisse in Abschnitt *Evaluation* legen nahe, dass das Similar-Modell trotz des Fehlens hotel-spezifischer Daten einen validen Ansatz zur Vorhersage normierter RevPAR-Werte darstellt. Basierend auf diesen Erkenntnissen wurde beschlossen, diesen Ansatz für die dynamische Preisgenerierung von Hotels ohne Verfügbarkeit von Vergangenheitsdaten zu nutzen.

3.2.6 Dynamische Preisgenerierung

Durch die Implementierung des Modells in der vorherigen Sektion werden normierte RevPAR-Werte generiert, die in einem DataFrame wie folgt dargestellt werden können:

	stay_date_str	time_to_arrival	scaled_revpar
0	01.01.2023	0	0.614793
1	02.01.2023	1	0.985979
2	03.01.2023	2	0.919261
3	04.01.2023	3	0.701799
4	05.01.2023	4	0.827744

Abbildung 3.50: Datensatz der Vorhersage

Um die tatsächlichen Preise unter Verwendung des RevPAR-Preis-Mappings für die einzelnen Zimmerkategorien zu generieren, ist es erforderlich, die normierten RevPAR-Werte in ihre ursprüngliche Form zu überführen. Dies erfolgt durch die Multiplikation der normierten RevPAR-Werte mit dem durchschnittlichen RevPAR-Wert des Hotels. Zu diesem Zweck wird dem DataFrame eine zusätzliche Spalte mit dem Namen *revpar* hinzugefügt. Diese Spalte repräsentiert das Produkt aus dem normierten RevPAR und dem durchschnittlichen RevPAR des Hotels.

3 Ähnliche Hotels

	stay_date_str	time_to_arrival	scaled_revpar	revpar
0	01.01.2023	0	0.614793	5866.413912
1	02.01.2023	1	0.985979	9408.305469
2	03.01.2023	2	0.919261	8771.675030
3	04.01.2023	3	0.701799	6696.638411
4	05.01.2023	4	0.827744	7898.412114

Abbildung 3.51: Datensatz mit dem tatsächlichen RevPAR-Werten

Durch diese Vorgehensweise wurde eine vergleichbare Ausgangssituation wie beim ursprünglichen RevPAR-Modell geschaffen. Von diesem Punkt an kann das Mapping vom RevPAR-Wert zum Preis verwendet werden, um die tatsächlichen Zimmerpreise zu generieren.

	stay_date_str	time_to_arrival	scaled_revpar	revpar	price
0	01.01.2023	0	0.614793	5866.413912	15300.0
1	02.01.2023	1	0.985979	9408.305469	15600.0
2	03.01.2023	2	0.919261	8771.675030	15500.0
3	04.01.2023	3	0.701799	6696.638411	15300.0
4	05.01.2023	4	0.827744	7898.412114	15400.0

Abbildung 3.52: Datensatz mit den tatsächlichen Preisen

Die Abbildung 3.52 zeigt beispielhaft die Preise in Cent für eine bestimmte Zimmertypen, die schließlich dem Kunden vorgeschlagen werden.

4. Fazit

Die vorgestellte Arbeit demonstriert die Möglichkeit der Generierung dynamischer Hotelpreise ohne spezifische Vergangenheitsdaten. Dabei wurden verschiedene Methoden erläutert und ein bestimmter Ansatz detailliert umgesetzt. Im gewählten Vorgehen wurden zunächst ähnliche Hotels identifiziert und anhand ihrer Daten ein CatBoost-Modell trainiert, um dynamische Preise für ein Hotel ohne historische Daten zu erzeugen.

Des Weiteren wurde ein Vergleich zwischen dem präsentierten Modell und dem aktuellen Live-Modell durchgeführt. Dieser Vergleich zeigt, dass das vorgestellte Modell trotz des Fehlens spezifischer Hotelinformationen mit dem Live-Modell mithalten konnte.

5. Ausblick

Im Folgenden wird ein kurzer Ausblick auf die weitere Entwicklung der Dynamischen Preisgestaltung von Hotels ohne spezifische Vergangenheitsdaten gegeben. Das ausführlich ausgearbeitete Konzept hat sich als äußerst effektiv erwiesen, um Preise für ein Hotel ohne spezifische Vergangenheitsdaten zu generieren.

Es ist jedoch wichtig anzumerken, dass das vorgestellte Konzept nicht in der Lage ist, den Fall zu bewältigen, wenn keine ähnlichen Hotels gefunden werden können. In solchen Fällen bedarf es eines alternativen Konzepts, um Preise zu generieren.

Des Weiteren haben weitere Analysen ergeben, dass das Konzept nicht nur für Hotels ohne spezifische Vergangenheitsdaten verwendet werden kann, sondern auch für Hotels mit Vergangenheitsdaten, um noch präzisere Preisgestaltungen zu ermöglichen.

Tabellenverzeichnis

2.1 Evaluierung der Konzepte	14
--	----

Abbildungsverzeichnis

1.1	happyhotel	3
2.1	RevPAR-Modell Vorgehen	10
2.2	Mitbewerber Modell	12
3.1	Alle schon vorhanden Features	16
3.2	Alle vorhanden Regionen	16
3.3	Alle vorhanden Städte	17
3.4	Alle schon vorhanden Features 2	19
3.5	Summe aller Nullwerte im Datensatz	19
3.6	Verteilung der Städte	21
3.7	Städte mit Fünf oder mehr Hotels	22
3.8	Datensatz nach der Umformulierung	22
3.9	Verteilung des neuen Features <i>region2</i>	23
3.10	Verteilung von dem Preis Features <i>median_min</i>	24
3.11	Durchschnittlicher minimal Preis pro Region	25
3.12	Verteilung von dem Preis Features <i>median_max</i>	25
3.13	Durchschnittlicher maximal Preis pro Region	26
3.14	Durchschnittlicher minimal Preis pro Hotelart	27
3.15	Durchschnittlicher maximal Preis pro Hotelart	27
3.16	Verteilung nach Zimmeranzahl	28
3.17	Häufigkeit der Zimmeranzahl im Datensatz	28
3.18	Finaler Datensatz für das Modell	29
3.19	Preise von allen Hotels für das Jahr 2022	30
3.20	Visualisierung der Preise zweier Hotels	30
3.21	Korrelationswerte der zwei Hotels	31
3.22	Visualisierung der Skalierten Preise zweier Hotels	31
3.23	Visualisierung der Skalierten RevPAR-Werte zweier Hotels	32
3.24	Korrelationswerte der RevPAR-Werte zweier Hotels	32
3.25	Basismodell Ergebnisse der RevPAR-Verläufe	34
3.26	Basismodell Ergebnisse der RevPAR-Korrelationen	34
3.27	DBScan Clusters	37
3.28	Gefundene Hotels für das erste Benchmark-Hotel mit dem DBScan-Algorithmus	38

3.29 Gefundene Hotels für das zweite Benchmark-Hotel mit dem DBScan-Algorithmus	39
3.30 DBScan Ergebnisse der RevPAR-Verläufe für das erste Benchmark-Hotel	39
3.31 DBScan Ergebnisse der RevPAR-Korrelationen für das erste Benchmark-Hotel	40
3.32 DBScan Ergebnisse der RevPAR-Verläufe für das zweite Benchmark-Hotel	40
3.33 DBScan Ergebnisse der RevPAR-Korrelationen für das zweiten Benchmark-Hotel	41
3.34 Exemplarisches Dokument innerhalb des Datensatzes	43
3.35 Gefundene Hotels für das erste Benchmark-Hotel mit dem Doc2Vec Modell	44
3.36 Doc2Vec Ergebnisse der RevPAR-Verläufe für das erste Benchmark-Hotel	44
3.37 Doc2Vec Ergebnisse der RevPAR-Korrelationen für das erste Benchmark-Hotel	45
3.38 Datensatz bestehend aus den einzelnen Kombinationen	47
3.39 Datensatz der Hotel-IDs und deren Korrelationswert	48
3.40 Bereits vorhandene Hotelspezifischen Daten	50
3.41 Datensatz nach der Ausführung der Hilfsfunktionen	52
3.42 Durchschnittlich normierte RevPAR-Werte pro Jahr	53
3.43 Durchschnittlich normierte RevPAR-Werte pro Tag	54
3.44 Durchschnittlich normierte RevPAR-Werte pro Monat	54
3.45 Durchschnittlich normierte RevPAR-Werte Ferientage vs. nicht-Ferientage	54
3.46 Korrelationsmatrix der verbleibenden Spalten	55
3.47 Finaler Datensatz für das RevPAR-Modell	55
3.48 Evaluation der beiden Modelle für das erste Benchmark-Hotel	57
3.49 Evaluation der beiden Modelle für das zweiten Benchmark-Hotel . .	57
3.50 Datensatz der Vorhersage	58
3.51 Datensatz mit dem tatsächlichen RevPAR-Werten	59
3.52 Datensatz mit dem tatsächlichen Preisen	59

Listings

1.1	Beispielhafte json-Datei	8
3.1	Einfaches Recommendation System für Film vorschläge	17
3.2	Datensatz Pipeline	36
3.3	Ausführung der Pipeline auf dem Datensatz	36
3.4	Ausführung des DBScan-Algorithmus	37
3.5	Finden alle Hotel ID's innerhalb des gleichen Clusters	38
3.6	Hilfsfunktion zur Erzeugung von textbasierten Dokumenten	42
3.7	Ausführung des Doc2Vec Modell	43
3.8	Erstellen des kombinierten Datensatzes	46
3.9	Erzeugung der Vorhersagen von ähnlichen Hotels mittels CatBoost .	47
3.10	Hilfsfunktions für das RevPAR Modell	51
3.11	Erzeugung der RevPAR-Vorhersagen	55

Literatur

- [1] M. KARATAŞ und A. Einstein, *If You Want to Know the Future, Look at the Past.* Amazon Digital Services LLC - KDP Print US, 2017, ISBN: 9781976758331. Adresse: https://books.google.de/books?id=u_14twEACAAJ.
- [2] Amazon Web Services, Inc., *Was ist Datenwissenschaft? – Datenwissenschaft erklärt – AWS*, 15.11.2023. Adresse: <https://aws.amazon.com/de/what-is/data-science/>.
- [3] A. Agarwal, „Linear Regression on Boston Housing Dataset - Towards Data Science“, *Towards Data Science*, 5.10.2018. Adresse: <https://towardsdatascience.com/linear-regression-on-boston-housing-dataset-f409b7e4a155>.
- [4] Melanie, *Seaborn: Alles über das Python-Tool zur Datenvizualisierung - Weiterbildung Data Science | DataScientest.com*, 2023. Adresse: <https://datascientest.com/de/seaborn-alles-ueber-das-python-tool-zur-datenvizualisierung>.
- [5] Shrishty, „What Is Normal Distribution & Standard Deviation in Statistics“, *Simplilearn*, 5.08.2021. Adresse: <https://www.simplilearn.com/tutorials/statistics-tutorial/what-is-normal-distribution>.
- [6] S. User, *Die wichtigsten Hotelarten im Überblick*, 20.01.2024. Adresse: <https://www.toursol.at/de/willkommen/toursol-blog/99-hotelarten>.
- [7] D. S. Team, „Was ist eine Korrelationsmatrix?“, *Data Science*, 3.05.2020. Adresse: <https://datascience.eu/de/mathematik-statistikatik/was-ist-eine-korrelationsmatrix/>.
- [8] datasolut GmbH, *Was ist Unsupervised Learning (Unüberwachtes Lernen)?*, 5.02.2024. Adresse: <https://datasolut.com/wiki/unsupervised-learning/>.
- [9] A. Nair, „Baseline Models: Your Guide For Model Building - Towards Data Science“, *Towards Data Science*, 4.04.2022. Adresse:

<https://towardsdatascience.com/baseline-models-your-guide-for-model-building-1ec3aa244b8d>.

- [10] K. Khan, S. U. Rehman, K. Aziz, S. Fong und S. Sarasvady, „DBSCAN: Past, present and future“, in *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, 2014, S. 232–238. DOI: 10.1109/ICADIWT.2014.6814687.
- [11] Q. Le V und T. Mikolov, *Distributed Representations of Sentences and Documents*. Adresse: <http://arxiv.org/pdf/1405.4053.pdf>.
- [12] J. T. Hancock und T. M. Khoshgoftaar, „CatBoost for big data: an interdisciplinary review“, *Journal of Big Data*, Jg. 7, Nr. 1, S. 94, 2020, ISSN: 2196-1115. DOI: 10.1186/s40537-020-00369-8.
- [13] P. Liashchynskyi und P. Liashchynskyi, *Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS*. Adresse: <http://arxiv.org/pdf/1912.06059.pdf>.