
# BargainBot

**Business Intelligence**
Luca Lenherr & Patrick Nydegger
Frühlingssemester 2025
Prof. Dr. Manuel Renold

# Inhaltsverzeichnis

# 1 Abstract

Mit der Frage, welcher Supermarkt bei einem Wocheneinkauf unter dem Strich wirklich am günstigsten ist, befasst sich die vorliegende Projektarbeit mit der Entwicklung des BargainBot©, einem Proof of Concept zur Einkaufsoptimierung.

Das Herzstück bildet ein in Python entwickelter Prototyp. Dieser ermittelt mit Hilfe eines Optimierungsmodells die kostengünstigste Aufteilung einer Einkaufsliste auf eine vom Nutzer definierte Anzahl an Läden. Als Datengrundlage dienten reale Preisdaten, die um synthetische Daten erweitert und in einer relationalen Datenbank modelliert wurden. Zusätzlich wurden mit R explorative Datenanalysen und Visualisierungen durchgeführt, um Einblicke in die Preisstrukturen der verschiedenen Anbieter sowie die Nutzerverhalten zu gewinnen.

Das Ergebnis ist ein funktionaler Prototyp, der eine optimale Einkaufsstrategie ausgibt und die praktische Anwendung von Business-Intelligence-Konzepten von der Datenmodellierung bis zur Analyse in einem kleinen Rahmen demonstriert.

# 2 Einleitung

Samstagnachmittag, der Wocheneinkauf steht an. Die Einkaufsliste wurde zusammengetragen und es geht los in Richtung Laden – doch in welchen eigentlich? Der Mensch ist bekanntlich ein Gewohnheitstier und wird höchstwahrscheinlich in denselben Laden gehen, wie vor einer Woche. Seit Aldi und Lidl die Schweiz eroberten, wechselten jedoch viele Menschen in einen dieser Discounter, um Geld zu sparen. Bei einzelnen Produkten ist das noch überschaubar, aber welcher Laden ist bei einem ganzen Wocheneinkauf unter dem Strich wirklich günstiger? Verlässliche Preisvergleiche gibt es kaum, nur grobe Faustregeln wie: „Aldi und Denner sind billiger als Coop oder Migros." Doch je nach Einkauf stimmt das eben nicht. Ein konkretes Beispiel: Sojamilch – im *günstigen* Aldi erhältlich für CHF 1.59, doch im *teuren* Coop nur CHF 1.40. … Genau hier wollen wir Klarheit schaffen.

Diese Projektarbeit dokumentiert den Aufbau eines Proof of Concepts in Form eines Python-Prototyps, der für jede Einkaufsliste eine Evaluierung durchführt, den Wunsch nach Anzahl Einkaufsläden berücksichtigt und dann die günstigsten Läden findet und die Produkte für jeden Laden auflistet – der BargainBot©. Um Einsicht in die Daten zu erlangen, wurden zudem mit R verschiedene Analysen durchgeführt. Faktoren wie Qualität, Labels, Packungsgrösse, Herkunft oder Ladestandort wurden dabei bewusst nicht berücksichtigt, wären aber für eine zukünftige Implementation eine spannende Ergänzung. Zudem stützt sich dieser Prototyp zwecks Realisierbarkeit und Vergleichbarkeit auf einem stark reduzierten Produktkatalog.

# 3 Datenquelle (Datenmanagement)

Zu Beginn des Semesters machten wir uns im Internet auf die Suche nach Datenbanken von Supermärkten, welche wir für unser Projekt nutzen wollten. Doch schnell wir stellten fest, dass diese Läden kaum daran interessiert zu sein scheinen, diese Daten öffentlich zugänglich zu machen, weshalb unsere Recherche erfolglos blieb. Also suchten wir nach alternativen Lösungen und stiessen auf eine inoffizielle, aber vielversprechende API auf GitHub. Da sich diese aber nur auf einen einzigen Laden beschränkte, und wir für eine repräsentative Analyse in unserem Projekt mindestens sechs vorsahen, suchten wir weiter.

Die Projektidee schien damit bereits zu scheitern, doch dabei lagen die Daten direkt vor unserer Nase: Supermarkt-Webseiten – eine riesige Sammlung von Daten, darunter Produkte und Preise, welche für jeden frei einsehbar sind. Also beschäftigten wir uns näher mit Web Scraping; einem automatisierten Prozess, um Daten von Webseiten zu extrahieren.

Beim Web Scraping stiessen wir aber schnell auf eine weitere Herausforderung: Die Webseiten der Supermärkte sind alle unterschiedlich aufgebaut und nicht so einfach zugänglich wie erhofft. Die Produkte und deren Metadaten sind stark verschachtelt oder erst gar nicht in Textform vorhanden (möglicherweise absichtlich als Schutz), indem beispielsweise die Preise direkt in den Produktbildern integriert sind. Diese liessen sich in der Theorie zwar durch eine optische Zeichenerkennung (OCR) herauslesen, doch das würde den Rahmen dieser Projektarbeit sprengen, da der Fokus dieser Arbeit primär auf der Analyse von Einkaufsläden liegt. Auch ein Versuch, mit ChatGPT die Daten zu extrahieren, erwies sich als erfolglos.

Ausserdem stellten wir mit der Zeit fest, dass sich die Mehrheit der Produkte zwischen den Läden gar nicht so einfach vergleichen lassen – sei es aufgrund unterschiedlicher Marken, Mengen oder Packungsgrössen. Zwar liessen sich die Werte umrechnen und vereinheitlichen, aber das würde unser gewünschtes Ergebnis entscheidend verfälschen.

Natürlich hätten wir die Produkte und Preise auch synthetisieren können, unser Anspruch war es jedoch, zumindest bei den Preisen reale Daten zu verwenden. Also wagten wir einen letzten Versuch und stiessen wir per Zufall auf einen Preisvergleich von K-Tipp, welcher in einer Tabelle die Preise von 40 Produkten aus fünf verschiedenen Läden gegenüberstellte. Diese Liste bildete natürlich nur einen Bruchteil vom gesamten Sortiment ab, doch damit konnten wir uns anfreunden.

## 3.1 Entity Relationship Diagram

Nachdem wir die Datenherkunft geklärt hatten, machten wir uns erste Überlegungen zur Struktur der Datenbank, welche wir grob in Form eines ERD in Miro skizzierten. In dieser ersten Iteration blieben wir bewusst auf einer hohen Flughöhe, um das Grundkonzept zu erstellen.

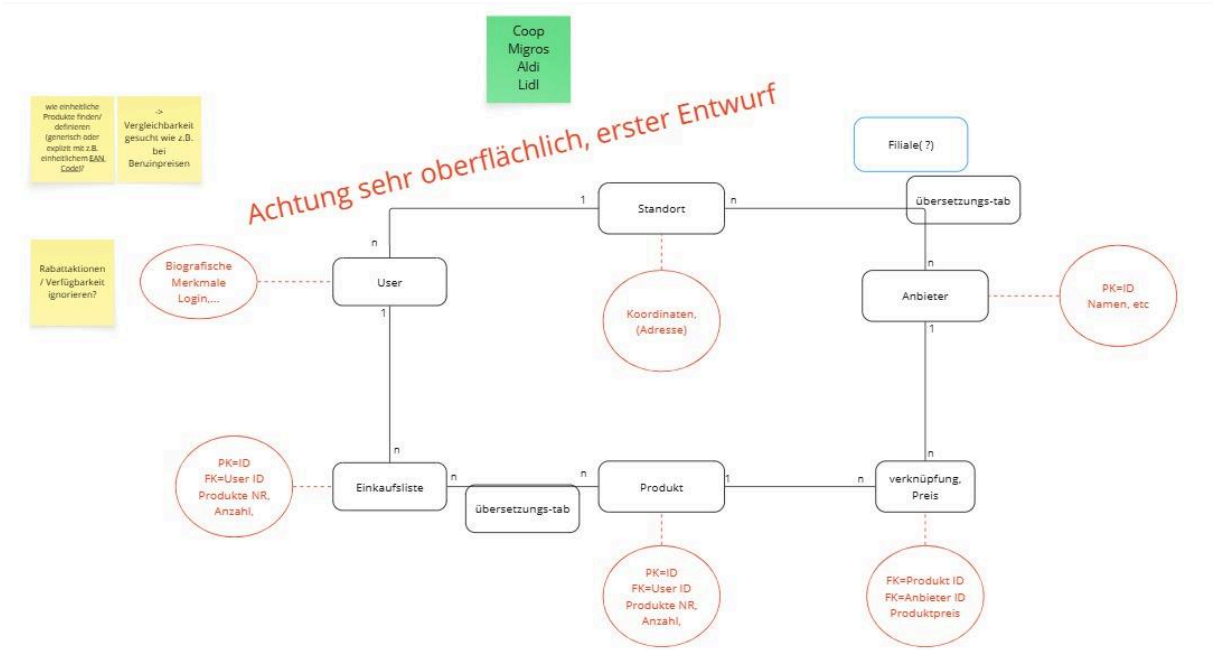


*Abb. 2: Erste Iteration der Datenbank*

Als nächsten Schritt modellierten wir eine zweite Iteration, um die Produkte und die Beziehungen der Tabellen weiter zu definieren. In dieser Iteration wurden bereits die Primär- und Fremdschlüssel festgelegt und die Notation auf die Martin-Notation gewechselt. Dabei wurden noch weitere Ideen und Gedanken in Form von Notizen ergänzt. Zur Vereinfachung entschieden wir uns dafür, die Standorte der Nutzer und Anbieter in der gleiche Tabelle abzubilden. Zusätzlich legten wir fest, dass jede Filiale über das gesamte Sortiment verfügen soll.

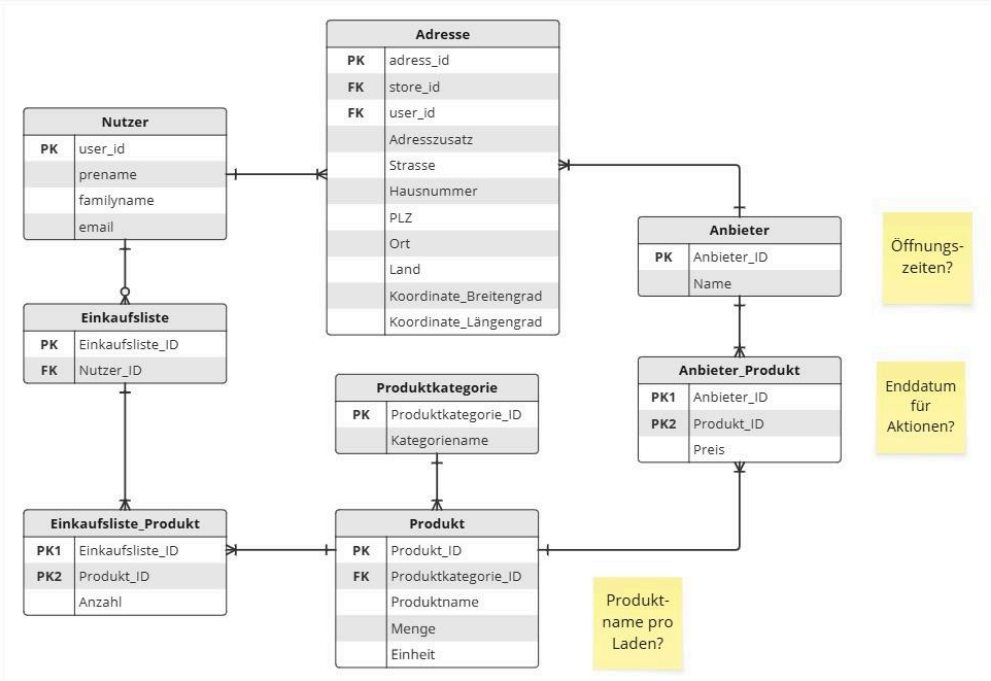


*Abb. 3: Zweite Iteration der Datenbank*

Nun wurden Ideen gesammelt, wie wir die Datenbank von dieser ERD-Sicht in MySQL Workbench überführen könnten. Die manuelle Erstellung von CREATE-Statements erschien uns fehleranfällig und nicht branchenüblich. Daher griffen wir auf Visual Paradigm zurück – ein Tool, das wir bereits im Studium verwendet hatten – und modellierten das ERD mit einigen kleineren Anpassungen. Der entscheidende Vorteil: Die CREATE-Statements konnten so gesammelt und direkt exportiert werden. Auch hier nahmen wir einige Vereinfachungen vor und verzichteten stellenweise bewusst auf eine dritte Normalform, so beispielsweise bei der Abhängigkeit von Adressen und Koordinaten in der gleichen Tabelle.

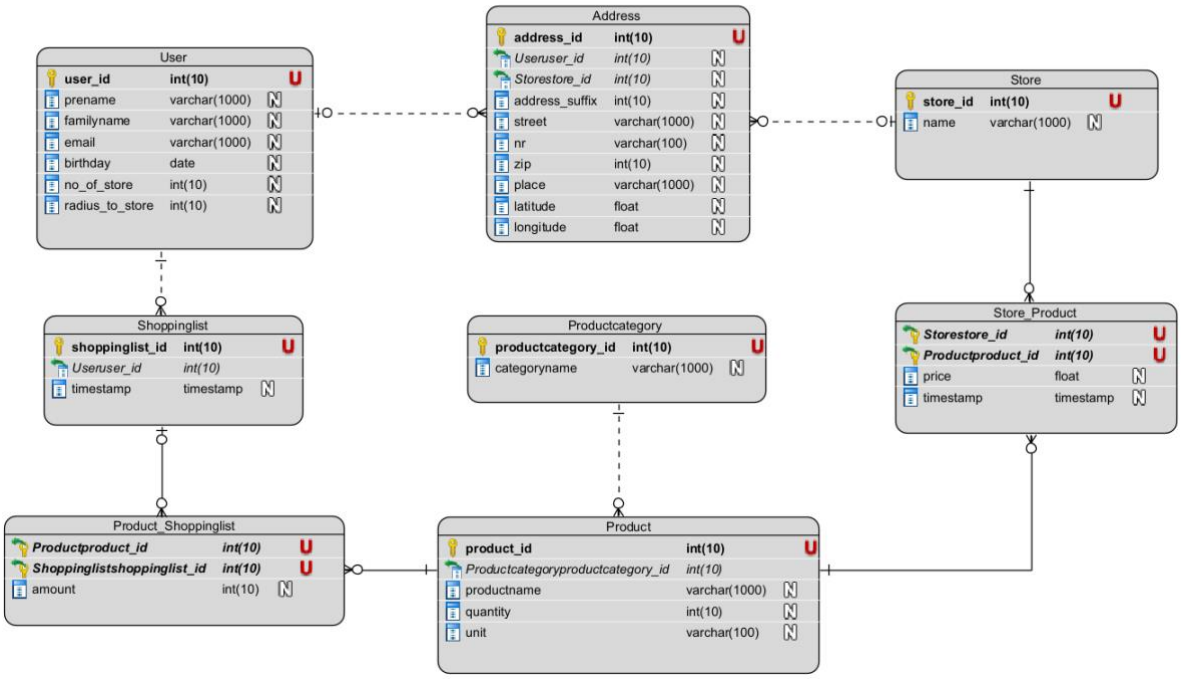


*Abb. 4: Dritte Iteration der Datenbank*

Sobald das ERM fertiggestellt war, exportierten das Schema in einem DDL-File. Als nächstes importierten die CREATE-Statements in MySQL Workbench und erstellten damit einen funktionalen Server – allerdings noch ohne Daten.

Dafür erstellten wir uns eine temporäre Excel-Tabelle, in welcher wir die Tabellenstruktur aus der Datenbank nachbildeten. Als nächstes kopierten wir die Daten (Produktname, Menge und Einheit) aus dem Preisvergleichs-PDF und liessen uns von ChatGPT noch zusätzlich die Preise für den Laden «Spar» generieren, damit wir die anfangs angestrebten sechs Läden erfüllten. Danach erfassten wir einige Spalten manuell, darunter Produktkategorien und reale Adressen von Filialen inkl. Koordinaten. Die restlichen Spalten erzeugten wir aus Zeitgründen synthetisch; Nutzerdaten mit ChatGPT und sämtliche IDs, Zeitstempel sowie Mengenangaben in den Einkaufslisten mit Zufallszahlen in Excel (teilweise mit aufwendigen Verteilungsfunktionen, damit beispielsweise sich die eingekauften Mengen pro Artikel wie in der Realität überwiegend im unteren Bereich aufhalten – mit einigen Ausreissern nach oben).

In einem weiteren Schritt kopierten wir Daten aus den vollständigen Tabellen in einen Online-CSV-to-SQL-Konverter, welcher uns direkt die benötigten INSERT-Statements generierte. Diese importierten wir anschliessend in die Datenbank auf dem MySQL-Server, womit sie für die Datenbankabfragen bereit waren.

# 4 Datentransformation (Data Engineering and Wrangling)

*Dieser Abschnitt ist Teil der Projektarbeit aus dem Modul "Data Engineering and Wrangling", welches zeitgleich zu diesem Projekt stattfand und beschäftigt sich mit dem bereinigen und normalisieren der Daten, welche später vom BargainBot© verwendet werden würden.*

Da die überwiegende Mehrheit unserer Daten synthetischen Ursprungs und in ihrer Anzahl überschaubar sind, wäre für unser Projekt im Grunde kein aufwändiges Engineering und Wrangling erforderlich. Dennoch haben wir uns dazu Gedanken gemacht, welche realistischen Fehlerquellen in unserem Use Case auftreten könnten, wenn es sich um ein

echtes, produktives Tool handeln würde. Da es sich beim BargainBot© um ein unabhängiges Produkt handelt, das nicht gerade im wirtschaftlichen Interesse der Supermärkte operiert, ist eine direkte Anbindung an deren Produktdatenbanken in der Realität kaum denkbar. Stattdessen müsste auf Web Scraping zurückgegriffen werden – eine Methode, die eine grosse Abhängigkeit zur Folge hat und sehr anfällig für unvorhergesehene Änderungen ist. Beispielsweise könnten sich Formatierungen oder Einheiten auf einer der Supermarkt-Webseiten unerwartet ändern, was unweigerlich dazu führen könnte, dass durch das automatisierte Web Scraping fehlerhafte oder inkonsistente Werte unbemerkt in die Datenbank gelangten und damit Konflikte oder falsche Ergebnisse entstünden.

## 4.1 Beispiel-Use-Case zur Datenaufbereitung

Da wir in unserem Projekt mit synthetischen und bereits bereinigten Daten arbeiteten, traten solche Probleme nicht auf. Um das Thema dennoch praktisch zu behandeln, entwickelten wir ein einfaches Beispiel-Szenario, das typische Herausforderungen bei der Datenaufbereitung demonstriert.

Um uns mit Datenaufbereitung zu beschäftigen, haben wir daher den folgenden Use-Case ausgedacht. Die Daten wurden bereits von den Webseiten geholt und müssen nun standardisiert und aufbereitet werden. Die Daten enthalten ein Produktname, eine Menge und eine Einheit. Die Produktkategorie muss also noch ergänzt werden. Danach sollen für die Daten die SQL-INSERT Statements geschrieben werden, um die Daten in die Datenbank zu übertragen.

## 4.2 Technische Umsetzung

In unserer Pipeline wurden die Daten in eine Funktion als Liste abgespeichert, um das Web Scraping zu simulieren. In dieser Liste gibt es absichtlich unterschiedliche Einheiten und Schreibweisen. Um den Code möglichst übersichtlich zu gestalten, haben wir stark objektorientiert programmiert.

Die Bereinigung der Einheiten erfolgt nach der Umwandlung in kleine Buchstaben, durch einen einfachen Textvergleich und simplen Regeln. Z.B wenn ein Ausdruck in der Liste für Gewicht zur Einheit im aktuellen Produkt passt, dann wird die Funktion aufgerufen, die die Menge hoch oder runterrechnet und die Einheit umändert, in diesem Fall auf "g" (Gramm).

Um die Produktkategorie zu bestimmen, wurde es etwas anspruchsvoller. Dank unseren vereinfachten Beispieldaten konnten wir schon mit einfachen NLP-Prozessen die Produktekategorie evaluieren. Dies erreichten wir mit Spacy. Dafür wurde eine etwas umfangreichere Vergleichsliste erstellt und jeweils die entsprechende Produktekategorie ergänzt. Spacy vergleicht nun das aktuelle Produkt mit dieser Liste und wählt die wahrscheinlichste Kategorie. Dies funktioniert nicht perfekt, aber erstaunlich gut. In realen Anwendungen bräuchte es komplexere und grössere Sprachmodelle, um auch längere Produktbeschreibungen verarbeiten zu können. Spannend wäre ein Test mit BERT oder einem kleinen Sprachmodell wie "Google gemma3:4b", das 128k Tokens verarbeiten kann, über 140 Sprachen kennt und trotz seiner geringen Grösse erstaunlich gute Outputs generiert. Möglicherweise wäre sogar die 1-Billion Parameter Version schon ausreichend. Nach der Evaluierung der Produktkategorie werden die Tupels in INSERT-Statements verwandelt und als Simulation in die Konsole geschrieben.

## 4.3 Reflexion und Ausblick

Das Ergebnis dieser Projektarbeit ist eine Miniaturversion der Datenaufbereitung. Es war interessant zu erkennen, wie umfangreich die Datenaufbereitung und Validierung werden kann. Bereits bei unseren Daten wären noch andere Validierungsmöglichkeiten möglich: Erkennen von Ausreissern bei Produktmengen, Kontrolle der Rechtschreibung des Produkts

mit einem Sprachmodell, Senden von Tupels mit Produktname, Menge und Einheit zur Plausibilitätsanalyse an ein Sprachmodell (z.B. «Ist es normal, dass Kartoffeln in 1000ml angegeben werden?») Hier könnte der Output auf unterschiedliche Wichtigkeitsstufen geordnet werden: Daten, die nicht verwendet werden können und der Mensch beachten muss, Daten die das Sprachmodell mit "gutem Gewissen" korrigieren konnte und Daten, die keine Problematik aufweisen. Nun liegt die Frage auf der Hand, wieso wir die Produktnamen nicht einfach durch einen Parsing-Algorithmus geben. Damit könnten dem Produktnamen weitere Informationen entnommen werden. Zum Beispiel: "Pommes-Chips Paprika (in Aktion)" dabei könnte die Geschmacksrichtung "Paprika" und die Bemerkung "in Aktion" erkannt werden. Wir befürchten aber, dass sich dies aufgrund der unterschiedlichen Webseiten stark unterscheiden kann. Es wäre spannend herauszufinden, ab welchem Produktsortiment es sich lohnen würde, eine Parsing-Pipeline aufzubauen oder bis wann man einen komplexeren, aber zugänglicheren Algorithmus wie ein Sprachmodell verwenden sollte.

# 5 Datenaufbereitung (Modellmanagement)

Um eine sinnvolle Aufteilung von Python und R zu erreichen, haben wir entschieden, mit Python das Proof of Concept zu erstellen und mit R diverse Datenauswertungen und Visualisierungen zu tätigen.

## 5.1 Python

### 5.1.1 Erste Datenbankabfragen

Unser Modellmanagement begann mit der Verbindung vom Python-Skript zur MySQL-Datenbank. Um in den Flow zu gelangen, wurden als erstes einfache SELECT- Statements getestet. Als Beispiel wurde der Nutzername anhand vom der user_id abgefragt. Dann machten wir uns an komplexere Abfragen mit mehreren JOINs. Für diese Projektarbeit wollten wir uns mit Python und R befassen, es war uns aber noch nicht klar, wie wir den Use Case in diese beiden Programmiersprachen aufteilen wollen. Erste Experimente bei der wir Python-Code über die Library "reticulate" in R benutzen könnten, wurden nicht mit grossen Erfolgen belohnt. Daher fokussierten wir uns zu Beginn vollständig auf Python und es wurde versucht, mit der Datenbank einfache Datenauswertungen als Baseline zu tätigen. Dabei haben wir eine SELECT-Abfrage getätigt, bei der man den Zeitrahmen auswählen kann, alle Produkte dieses Zeitrahmens pro User extrahiert und die aggregierten Daten der Kaufmenge in absteigender Reihenfolge ausgibt. Dies könnte als einfachste Vorhersage benutzt werden, mit der Philosophie "Wer ein Produkt in den letzten Wochen oft gekauft hat, könnte dieses Produkt wieder kaufen".

### 5.1.2 Implementierung Use Case

Nun ging es darum den Use Case mit Python abzudecken.

Unsere Grundidee übertragen in eine Userstory für einen potenziellen Nutzer lautet:

*"Als User möchte ich die Produkte in meiner Einkaufsliste auf eine gewünschte Anzahl Läden aufgeteilt erhalten, um insgesamt den günstigsten Einkaufspreis zu erzielen"*

Dementsprechend wurden SELECT-Statements definiert, welche in einer hypothetischen App die spezifische Einkaufsliste aus der Datenbank holt.

Um das Produkt zu testen, wurde auch eine manuelle Eingabe von Produkten und deren Mengen implementiert. Dies bedarf schon einer ersten Menüführung, damit der User auswählen kann, welche Art von Daten er verwenden möchte.

Danach ging es an das Herzstück von BargainBot©; der Evaluierung und Zuordnung der optimalen Einkaufsliste. Dies wird im Code mit deterministischen Modellen erreicht. Dabei haben wir die Realität, wie schon bei der Erstellung der Datenbank, stark vereinfach. Variablen wie die Distanz zum Laden, Produktmerkmale oder Vorlieben des Kunden wurden ignoriert.

### 5.1.3 Deterministisches Optimierungsmodell

Als erste Iteration wurde eine Logik implementiert, die im Preistotal der Einkaufsliste den günstigsten Laden berechnet. Ganz simpel, indem für jeden Laden eine Liste mit den Produkten und deren Preisen aufsummiert wird und daraus die günstigste Liste gewählt wird. Um diese Liste darzustellen, wurde extra eine separate Funktion geschrieben und deren Input standardisiert, damit auch komplexere Algorithmen diese Datenausgabe verwenden können.

Nach diesem Erfolg machten wir uns an die Evaluierung mit zwei Läden. Hier wurde uns die Komplexität bewusst, da in unserer Datenbank sechs Läden zum Auswahl stehen. Nun müssten alle Produkte der Einkaufsliste in jeder Kombination bei allen Läden verglichen werden, was zu einem sehr hohen Rechenaufwand führt. Als Beispiel; bereits bei zehn Produkten und sechs Läden ergeben sich $6^{10}$ Kombination, also über 60 Millionen Möglichkeiten – kaum vorzustellen, wenn alle 40 Produkte (ergo $6^{40)}$ getestet werden sollen.

Der Trick liegt darin, der richtige Blickwinkel einzunehmen. Aus Produktsicht müssten alle Kombinationen getestet werden, aus Ladensicht wird es bedeutend einfacher. Unsere Idee: Wir erstellen für jede Ladenkombination eine Liste. Der Einfachheit halber beschränken wir den Einkauf auf zwei Läden. Dies bedeutet 6 nCr 2*,* also 15 Listen von Ladenkombinationen. Nun nehmen wir jedes Produkt der Einkaufsliste einzeln und wählen bei jeder Ladenkombination der jeweils günstigste Laden. Wir ergänzen die 15 Listen mit dem jeweils günstigeren Laden. Als Beispiel: Es gibt die Liste "Coop_Lidl". Für das Produkt "Ruchbrot" der Einkaufsliste wird nun der Preis von Coop mit dem Preis von Lidl verglichen und der günstigere Laden gewählt. Danach wird für alle anderen Paare wie "Migros_Aldi" dasselbe Produkt geprüft und in deren Liste ergänzt. Nun geht es zum nächsten Produkt und so füllen sich allmählich alle 15 Listen – also alle 15 Einkaufsmöglichkeiten. Wenn alle Produkte abgeschlossen sind, werden die Listen miteinander verglichen und die günstigste Liste ausgewählt. Diese wird wie oben erwähnt an die Ausgabefunktion gegeben, welche die Produkte pro Laden auflistet und der Preis darstellt. Es kann vorkommen, dass schliesslich alle Produkte in einem Laden eingekauft werden sollen, da dieser bei allen Produkten günstiger war.

Diese Logik kann auf weitere Anzahl Läden angewandt werden, dies war im Rahmen dieser Projektarbeit aber nicht mehr im Scope. Der Rechenaufwand bleibt auch bei steigender Ladenzahl im kleinen Rahmen, da bei drei Läden nur 6 nCr 3 = 20 Listen, bei vier Läden 6 nCr 4 = 15 Listen, bei fünf Läden 6 nCr 5 = 6 Listen und bei 6 nCr 6 = 1 Liste erstellt werden muss. Die Prüfung der jeweils günstigeren Läden bleibt auch effizient, da nur das Minimum von maximal 6 Werten berechnet werden muss.

## 5.2 R

Der R-Anteil dieser Projektarbeit beschäftig sich wie bereits angedeutet mit den Analysen aus der Sicht von BargainBot©, bzw. dem Anbieter dieses Tools. Diese Erkenntnisse aus den Daten über die Preisvergleiche sowie Nutzerverhalten könnten dann als «wertvolle Insights» an die Supermarktketten verkauft werden, aber alternativ auch als transparente Aufklärung für Kunden (im weitesten Sinne als Konsumentenschutz) verstanden werden.

Um die Analysen durchzuführen, verknüpften wir als erstes RStudio mit der Datenbank. Als nächstes formulierten wir die Datenbankabfragen, um die benötigten Werte für die jeweiligen

Diagramme zu laden. Dann überlegten wir uns, welche Darstellung sich für die Interpretation jeweils am besten eignet und setzten diese in Code um, wobei wir versuchten, den Code möglichst einfach und verständlich zu halten.

# 6 Visualisierung

## 6.1 Python

Unsere Projektarbeit fokussierte sich auf die Implementierung der Logik und des deterministischen Modells. Daher wird die Interaktion mit dem Programm über eine simple Konsoleneingabe ermöglicht. Um trotzdem die Visualisierungsaspekte von Business Intelligence abdecken zu können, wollten wir mit der CustomTkinter-Library ein einfaches GUI bauen. Dies stellte sich als grösseres Projekt heraus als anfangs gedacht, weshalb wir es bei einem GUI für die Eingabe der Einkaufsliste beliessen.

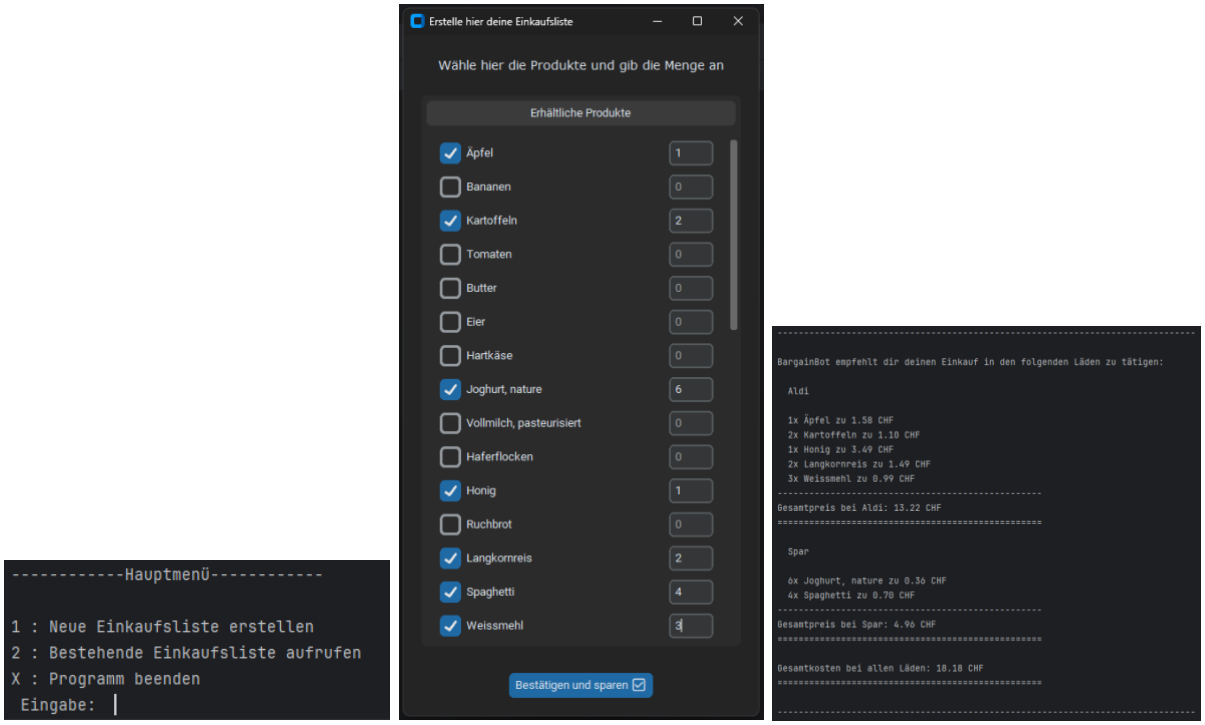


*Abb. 5: Hauptmenü (links), GUI von Einkaufsliste (Mitte) und Output in Konsole (rechts)*

## 6.2 R

Da wir in unserem Projekt abgesehen von den Preisen nahezu alle Daten synthetisch bzw. zufällig generierten, lassen diese folglich keine realen Abhängigkeiten, Muster oder Zeitanalysen zu. Dennoch versuchten wir, die vorhandenen Daten mit R zu explorieren und erstellten dazu mehrere Diagramme:

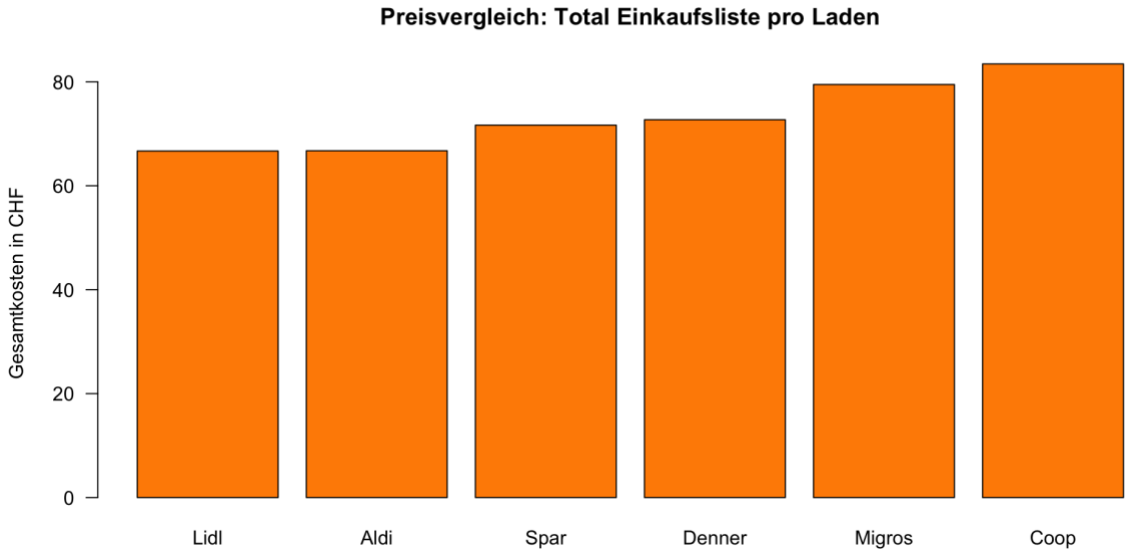


*Abb. 6: Preisvergleich von vollem Warenkorb pro Laden*

Dieses Balkendiagramm zeigt, wie teuer die ausgewählten Supermärkte ausfallen, wenn jedes Produkt aus der Liste genau 1x gekauft wird. Es zeigt sich – wie auch im Bericht von K-Tipp festgestellt wurde – dass Lidl in der Summe der ausgewählten Produkten am günstigsten abschneidet[1].

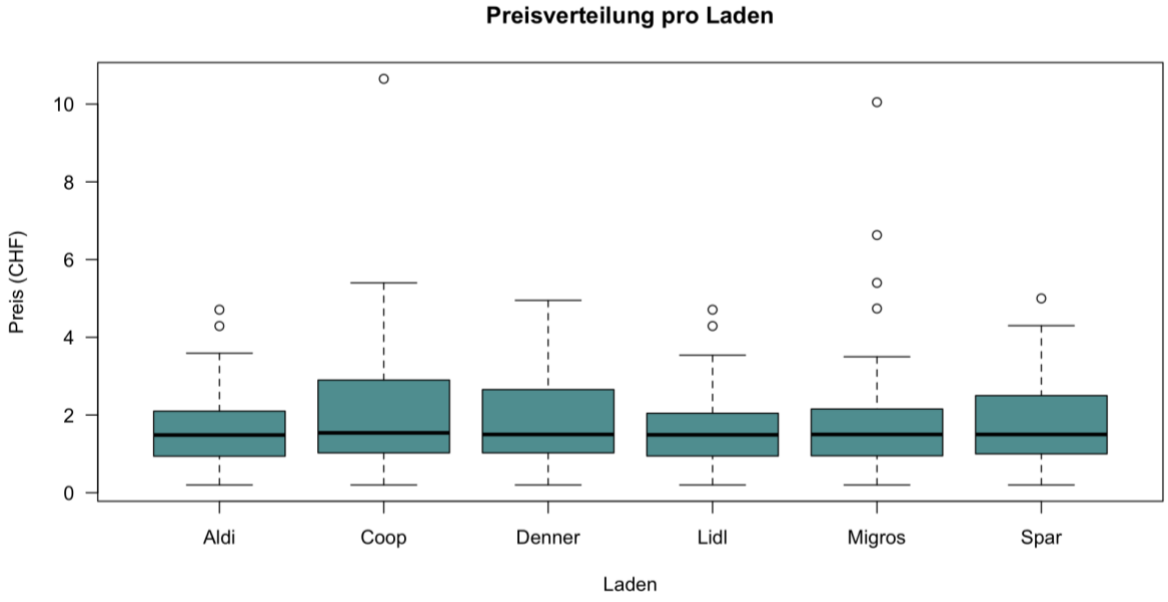


*Abb. 7: Preisverteilung pro Laden*

Diese Darstellung zeigt die Verteilung der Produktpreise zwischen den Läden in einem Boxplot. Grundsätzlich liegen die Medianpreise bei allen Anbietern relativ nahe beieinander. Auffällig ist aber, dass sich beispielsweise die Hälfte des Migros-Sortiments in einer relativ niedrigen sowie schmalen Preisspanne befindet, jedoch so viele Ausreisser vorhanden sind, sodass der Laden in der Summe auf den zweitletzten Platz fällt (vgl. Abb. 6). Lidl und Aldi zeigen im Vergleich erwartungsgemäss eine kompaktere Preisstruktur, was auf ein günstigeres und gleichmässigeres Preisniveau schliessen lässt.

[1] Diese Aussage wurde nicht von Lidl gesponsert! ☺

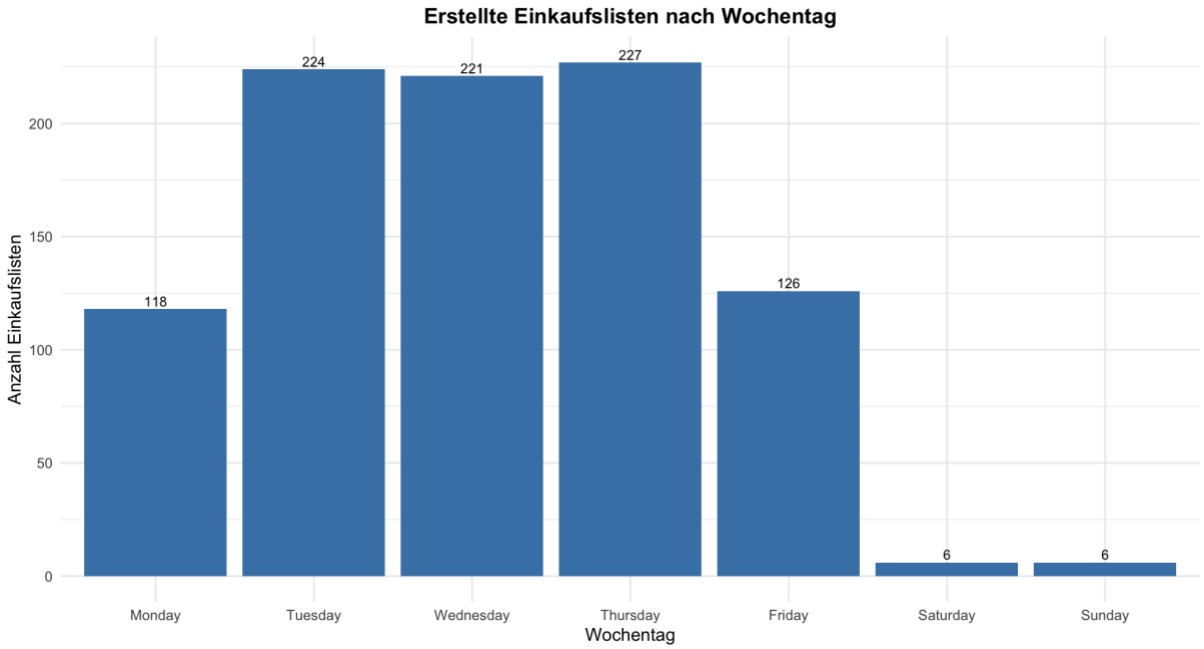


*Abb. 8: Erstellte Einkaufslisten nach Wochentag*

In diesem Balkendiagramm werden die erstellten Einkaufslisten pro Wochentag aufgezählt und es zeigt sich, dass die Nutzer diese fast ausschliesslich unter der Woche zusammenstellen – mit einer scheinbaren Präferenz für die mittleren Wochentage.

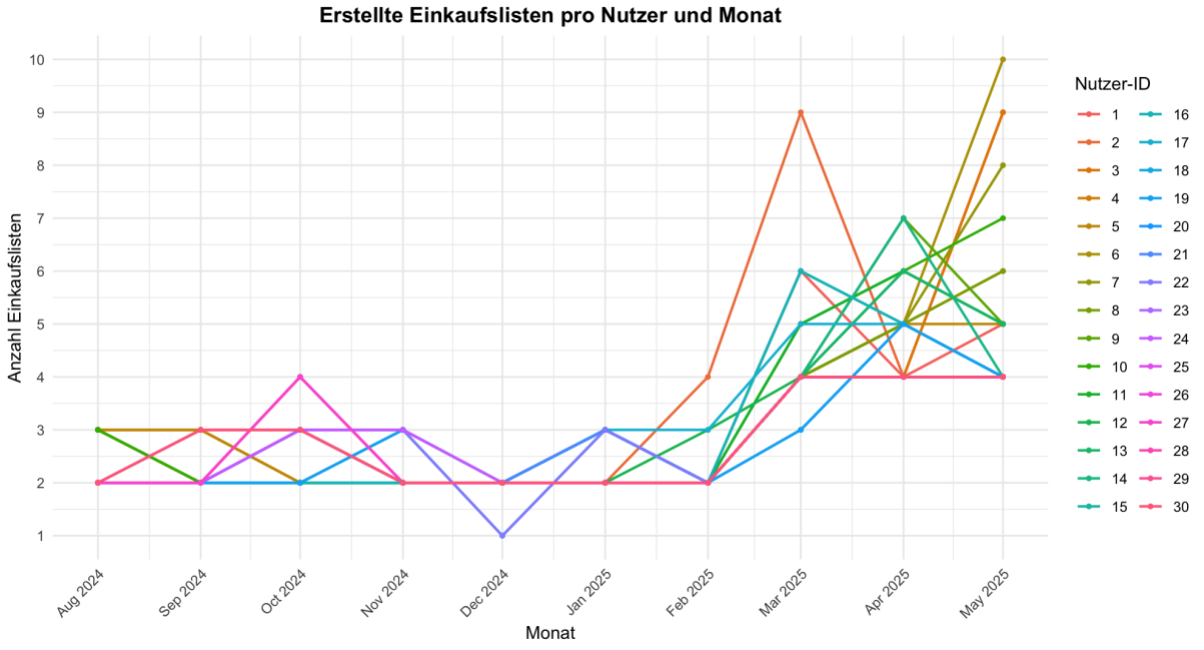


*Abb. 9: Erstellte Einkaufslisten pro Nutzer und Monat*

Dieses Diagramm zeigt, dass im ersten halben Jahr seit der Gründung die Nutzer im Mittel relativ konstant zwischen zwei und drei Einkaufslisten pro Monat erstellten. Diese Anzahl nahm ab März abrupt zu und streute die Werte bis im Mai zwischen vier und zehn erstellten Listen pro Monat und Nutzer! Wir führen diese Veränderung auf ein Update von BargainBot© mit stark optimiertem Tiefpreisalgorithmus zurück, was die Schnäppchenjäger angetrieben haben muss ☺.

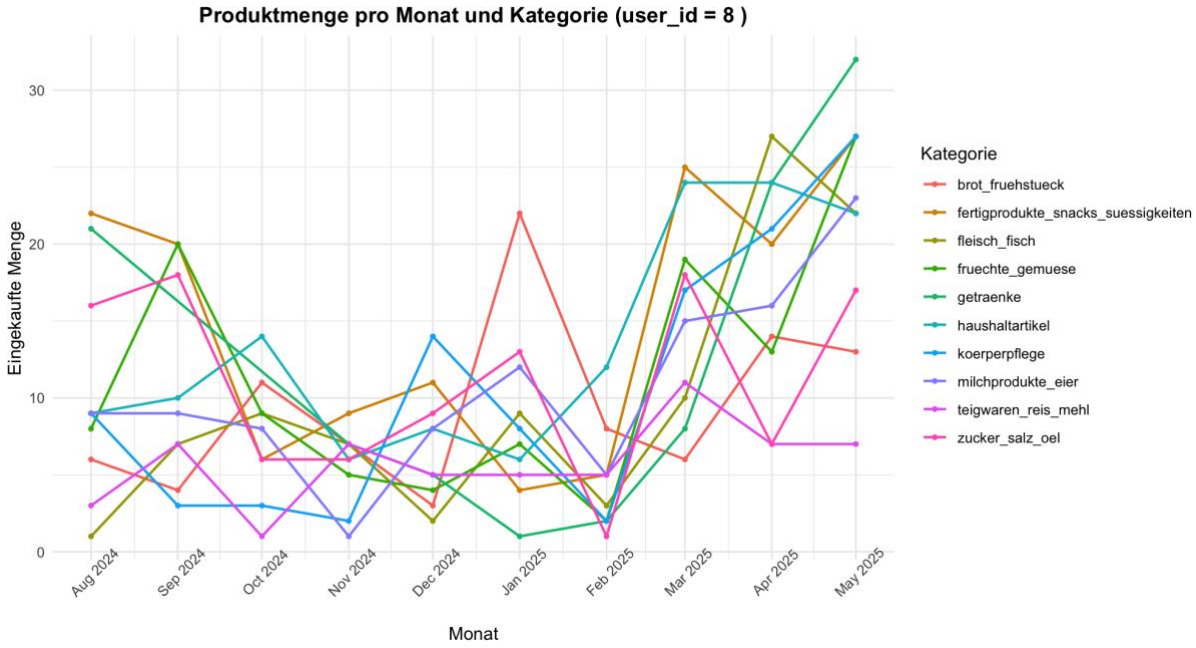


*Abb. 10: Produktmenge pro Monat und Kategorie von einem User*

Diese Darstellung ermöglich einen Einblick in das Kaufverhalten von einem bestimmten Nutzer, gruppiert nach Produktkategorien. Es zeigt, dass im Sommer über alle Kategorien hinweg tendenziell mehr eingekauft wurde als im Winter. Um dabei aber auf eine saisonale Abhängigkeit zu schliessen, wären mehrere Jahre an Daten nötig.

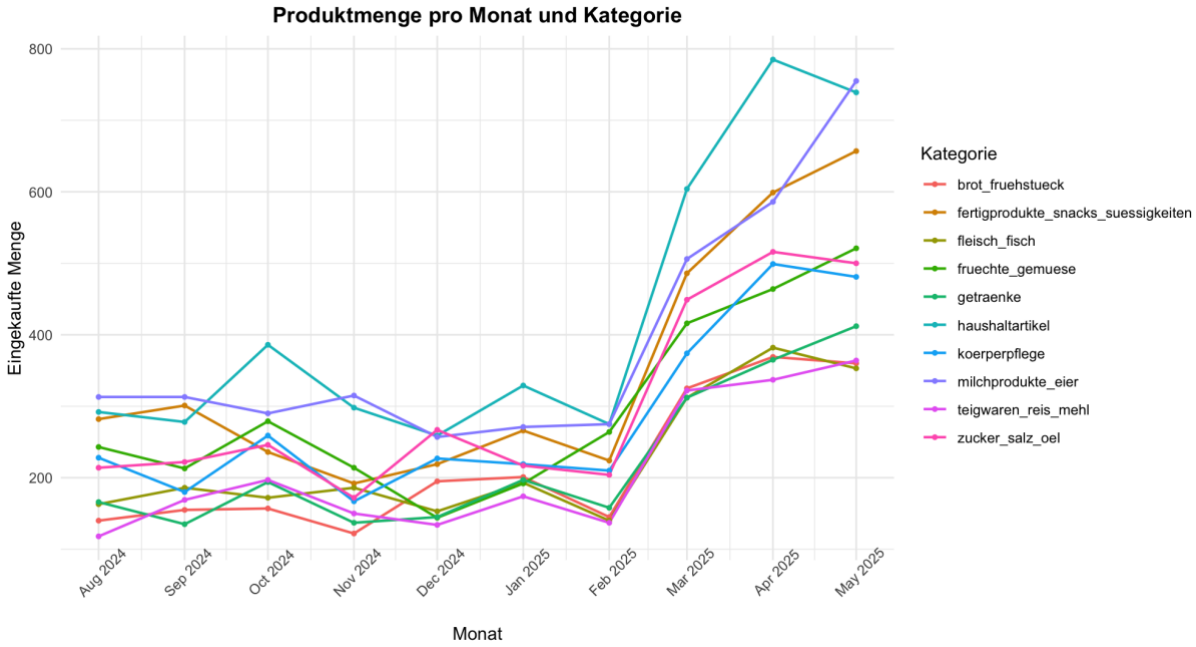


*Abb. 11: Produktmenge pro Monat und Kategorie*

Diese Ansicht ist eine Aggregation von der vorherigen Grafik (vgl. Abb. 10), welche die eingekauften Produkte aller Nutzer summiert und nach Produktkategorie zeigt. Auch hier zeichnet sich (wie bei den erstellten Einkaufslisten, vgl. Abb. 9) eine deutliche Zunahme an Einkäufen über alle Produktkategorien seit März ab. Dabei zeigt sich, dass (gemessen an der Menge) Haushaltsartikel am häufigsten und Teigwaren am wenigsten oft gekauft werden.

## 6.3 Mögliche zukünftige Ergänzungen

Als weiters Vorgehen empfehlen wir das Proof of Concept auszubauen zu einem Prototyp, um danach diesen zu einem Minimum-Loveable-Product zu skalieren.

### 6.3.1 Evaluierungsalgorithmus

Dafür muss der Algorithmus ausgebaut werden, um eine beliebige Anzahl von Einkaufsläden abzudecken und weitere Faktoren berücksichtigen zu können. Beispielsweise wäre die Distanz von User zu Laden sehr interessant, da so eine geografische Selektion von Einkaufsmöglichkeiten umsetzbar wird.

### 6.3.2 Erhalten der Aktuellen Produktdaten

Für die Produkte müssen Partnerschaften mit Läden aufgebaut werden. Möglicherweise sind Aldi, Lidl interessiert, da sie eher günstiger sind als andere. Hier könnte man auch Business hineinbringen und bei einem identischen Preis ein spezifischer Partner bevorzugen. Für Läden die nicht kooperieren möchten, würde eine Web Scraping Pipeline notwendig werden. Diese wird sehr aufwendig sein, da jeder Laden unterschiedliche Webseiten und innerhalb der Webseiten viele verschachtelte Unterseiten besitzt.

### 6.3.3 Aufbereitung der Produktdaten

Diese Produktedaten müssten danach durch die aufgebaute Data-Wrangling- und Cleansing-Pipeline gegeben und in die erweiterte Datenbank importiert werden. Hier müssen komplexere Algorithmen als im aktuellen Proof of Concept angewendet oder eigene Modelle trainiert werden. Damit können die Produktdaten kontrolliert, aufbereitet und in der Datenbank ergänzt werden.

### 6.3.4 Datenbank

Die Datenbank ist im aktuellen Zustand noch nicht umfangreich genug. Es sind fast nur Faktentabellen vorhanden, diese müssten mit Dimensionstabellen ergänzt werden, um umfangreiche Analysen durchführen zu können. Zudem müssen weitere Faktoren der Produktwahl wie in der Einleitung erwähnt berücksichtigt und abgespeichert werden.

## 6.4 Reflexion und Fazit

Durch diese Projektarbeit konnten viele Aspekte der Vorlesungen praktisch angewendet werden und ermöglichte und tiefere Einblicke in die Komplexität von Business Intelligence. Obwohl wir ein stark abstrahiertes Modell der Wirklichkeit verwendet haben, wurden wir mit vielen Herausforderungen konfrontiert, die wir dank kreativen Lösungsansätzen mit gutem Gewissen meistern konnten. Insbesondere die Hürden bei der Datenbeschaffung brachten uns die Erkenntnis: Ein BI-System ist nur so wertvoll wie seine Datengrundlage. Wir haben uns beim Einsatz von Sprachmodellen wie Gemini oder ChatGPT stark eingeschränkt und diese nur für sehr repetitive oder simple Aufgaben verwendet. Als Beispiel wurden die simulierten Produktedaten damit synthetisiert, oder Recherchen getätigt. Obwohl wir dadurch sehr viel gelernt haben, vor allem beim Erstellen von Python- oder R-Code, sind wir der Überzeugung, dass diese Hilfsmittel ausserhalb des Studiums stärker eingesetzt werden sollten.

# Abbildungsverzeichnis

# 7 Anhang

- Lidl PDF mit Preistabelle

- GitHub-Repository
  - SQL Statements
  - Python Code
  - R Scripts

# Alltagsartikel: Lidl am günst

**Coop und Migros werben mit Preissenkungen und dem Ausbau ihrer Billiglinien. Doch diese fehlen in den Filialen oft. Der aktuelle Preisvergleich des K-Tipp zeigt: Lidl und Aldi sind bei Alltagsprodukten weiterhin klar am günstigsten.**

Migros-Chef Mario Irminger versprach im ersten «Migros-Magazin» dieses Jahres: «Es ist unser erklärtes Ziel, die Preise zu senken und damit das Haushaltsbudget der Menschen zu entlasten.» Die Migros vergünstige ab Mitte Januar das Sortiment. Und Coop hielt im Februar in der «Coop-Zeitung» fest, man habe seit Jahresanfang 500 Preise gesenkt und baue die Billiglinie aus.

Der Preisvergleich des K-Tipp zeigt aber: Alltagsartikel sind bei Coop und Migros weiterhin klar teurer als bei Denner, Aldi und Lidl.

Für den Vergleich erhob der K-Tipp Ende März die Preise von 40 Alltagsartikeln in mittelgrossen Filialen der Detailhändler. Berücksichtigt wurden neben Lebensmitteln wie Brot, Milch und Tomaten Fertigwaren wie Tiefkühlpizzas, Haushaltartikel wie Nastücher und Hygieneprodukte wie Duschshampoos.

Der K-Tipp notierte in den Filialen der Detailhändler in Brugg AG und in Lenzburg AG stets den Preis des günstigsten erhältlichen Artikels. Bei Coop und in der Migros fehlten zum Teil Produkte der Billiglinien Prix Garantie und M-Budget. Die Qualität der Produkte bewertet der K-Tipp in Preisvergleichen nicht, sondern in separaten Labortests. Diese zeigen: Teure Artikel sind oft nicht besser als günstige (K-Tipp 5/2023).

Ergebnis des Vergleichs: Bei Lidl und Aldi kostete der Einkauf praktisch gleich viel: bei Lidl Fr. 66.64, bei Aldi Fr. 66.69 (Tabelle). Aldi war bei 27 von 40 Artikeln am günstigsten, Lidl 24 Mal. Am teuersten war Coop: Der K-Tipp-Warenkorb kostete dort Fr. 83.42 – ein Viertel mehr als bei Lidl und Aldi. Coop war bei 23 von 40 Artikeln am teuersten.

Grosse Preisdifferenzen gab es vor allem bei Früchten und Gemüse, bei Fleisch, bei Haushaltartikeln wie Waschmitteln und bei Körperpflegeartikeln wie Deosprays und Shampoos. Bei diesen Waren können Kunden bei Lidl und Aldi viel Geld sparen. Auch Ruchbrot ist dort günstiger. Fast keine Unterschiede gab es bei Eiern, Joghurt, Mehl, Milch, Reis, Teigwaren, Salz und Zucker. Auch Mayonnaise, Guetsli und Pommes-Chips kosteten überall etwa gleich viel. Das Sparpotenzial ist dort also gering.

## Seit 2023 veränderten sich die Preise kaum

Der K-Tipp hatte vor einem Jahr die Preise der gleichen Artikel erhoben. Das Ergebnis war dasselbe wie im aktuellen Vergleich: Lidl war am günstigsten, Coop am teuersten (K-Tipp 7/2023). So haben sich die Preise seither verändert:

- Im Vergleich zu 2023 kostete der Einkauf überall ähnlich viel. Coop wurde geringfügig günstiger (um Fr. 1.70), die Migros um 39 Rappen teurer.
- Bei Denner gibt es die Körperpflegeprodukte der Eigenmarke Isana zum Teil nicht mehr. Der günstigste Deospray ist jetzt ein Markenprodukt von Rexona. Dieses ist mehr als doppelt so teuer wie Isana. Denner sagt dazu, viele Kunden würden Markenartikel wählen.
- Die Läden verändern die Preise ihrer Artikel ständig. Bei über 60 Prozent der Produkte sind die Preise anders als vor einem Jahr.
- In allen Läden günstiger wurde nur Milch. Überall teurer wurden Mehl, Orangensaft, Salz, Reis und Zucker. Reis und Zucker verteuerten sich um einen Viertel, Orangensaft um einen Drittel.

Immerhin: Die Preisvergleiche des K-Tipp zeigen

**Lidl, Frauenfeld TG:** Bei 24 von 40 Produkten am günstigsten

# stigsten, Coop am teuersten

Wirkung. So waren Eier vor einem Jahr bei der Migros am teuersten. Sie senkte den Preis um 14 Prozent – jetzt kosten die Eier gleich viel wie bei Aldi, Coop, Denner und Lidl. Aldi, Coop und Denner vergünstigten ihre Butterguetsli: Diese sind nun nicht teurer als bei Migros und Lidl. Coop senkte zusätzlich den Preis des WC-Papiers.

Fazit: Trotz einigen Preissenkungen bleiben Coop und Migros bis 25 Prozent teurer als die Discounter. Wer sparen will, kauft am besten bei Lidl oder Aldi ein. Denn Coop und Migros bieten ihre Billiglinien in mittelgrossen Filialen zum Teil nicht an. Im Preisvergleich fehlte bei Coop 6 Mal das Prix-Garantie-Produkt, in der Migros 5 Mal der M-Budget-Artikel. Das verteuert den Einkauf. Christian Gurtner

KEYSTONE

| | | Lidl | Aldi | Denner | Migros | Coop |
|---|---|---|---|---|---|---|
| **Früchte, Gemüse** | | | | | | |
| Äpfel | 1 kg | **1.58** | **1.58** | **1.60** | **1.58** | **1.58** |
| Bananen | 1 kg | **1.29** | **1.29** | **1.50** | 1.30 | 1.30 |
| Kartoffeln | 1 kg | **1.10** | **1.10** | 1.42 | **1.10** | **1.58** |
| Tomaten | 1 kg | 1.59 | **1.49** | 2.– | 2.– | **2.95** |
| **Milchprodukte, Eier** | | | | | | |
| Butter | 250 g | **3.39** | **3.39** | **3.40** | **3.40** | **3.40** |
| Eier | 6 Stück | 1.70 | 1.70 | 1.70 | 1.70 | 1.70 |
| Hartkäse | 250 g | **2.74** | **2.74** | 2.81 | 2.75 | **3.88** |
| Joghurt, nature | 200 g | –.36 | –.36 | –.36 | –.36 | –.36 |
| Vollmilch, pasteurisiert | 1 l | 1.35 | 1.35 | 1.35 | 1.35 | 1.35 |
| **Brot, Frühstück** | | | | | | |
| Haferflocken | 1 kg | 1.49 | **1.48** | **1.50** | **1.50** | **1.50** |
| Honig | 500 g | **3.49** | **3.49** | **3.50** | **3.50** | **3.50** |
| Ruchbrot | 500 g | **1.15** | **1.15** | **1.40** | **1.40** | **1.40** |
| **Teigwaren, Reis, Mehl** | | | | | | |
| Langkornreis | 1 kg | **1.49** | **1.49** | **1.50** | **1.50** | **1.50** |
| Spaghetti | 500 g | –.70 | –.70 | –.70 | –.70 | –.70 |
| Weissmehl | 1 kg | **–.99** | **–.99** | **1.–** | **1.–** | **1.–** |
| **Zucker, Salz, Öl** | | | | | | |
| Kristallzucker | 1 kg | **1.59** | **1.59** | **1.60** | **1.60** | **1.60** |
| Mayonnaise, Tube | 265 g | **1.49** | **1.49** | **1.50** | **1.50** | **1.50** |
| Sonnenblumenöl | 1 l | **4.29** | **4.29** | 4.95 | **5.40** | 4.30 |
| Speisesalz | 1 kg | 1.05 | 1.05 | 1.05 | 1.05 | 1.05 |
| **Fleisch, Fisch** | | | | | | |
| Bratwurst | 300 g | 2.31 | 2.31 | **2.56** | 2.31 | **2.30** |
| Forellenfilet, geräuchert | 150 g | 3.54 | 3.59 | **3.–** | 4.74 | **5.40** |
| Pouletgeschnetzeltes | 300 g | **4.71** | **4.71** | **4.71** | 10.05 | **10.65** |
| **Fertigprodukte, Snacks, Süssigkeiten** | | | | | | |
| Butterguetsli, Petit Beurre | 200 g | –.88 | –.88 | –.88 | –.88 | –.88 |
| Lasagne bolognese | 500 g | **2.49** | **2.49** | 2.75 | 2.50 | **2.80** |
| Milchschokolade | 100 g | –.58 | **–.50** | –.58 | **–.50** | **–.59** |
| Pizza Margherita | 400 g | **1.60** | **1.60** | **1.60** | **1.60** | **2.15** |
| Pommes-Chips Paprika, Beutel | 300 g | **3.05** | **3.05** | **3.05** | **3.09** | **3.09** |
| **Getränke** | | | | | | |
| Kaffeekapseln, lungo | 10 Stück | **1.65** | **1.65** | **1.65** | **1.60** | **1.65** |
| Mineralwasser, mit Kohlensäure | 1 l | –.20 | –.20 | –.20 | –.20 | –.20 |
| Orangensaft | 1 l | 1.13 | 1.13 | 1.13 | 1.13 | 1.13 |
| **Haushaltartikel** | | | | | | |
| Allzweckreiniger, flüssig | 1 l | **–.73** | **–.73** | **1.25** | –.75 | **–.73** |
| Geschirrspüler, Pulver | 1 kg | **2.30** | **2.30** | 3.30 | **6.63** | 3.45 |
| Haushaltpapier | 2 Rollen à 50 Blatt | **–.81** | **–.81** | –.97 | –.82 | **1.37** |
| Nastücher | 15 Päckli à 10 Stück | **1.40** | **1.40** | **1.45** | 1.41 | **1.45** |
| Waschmittel (Farbwäsche), flüssig | 1 l | 1.79 | 1.90 | **1.63** | 1.78 | **2.85** |
| WC-Papier | 6 Rollen à 200 Blatt | **1.79** | **1.79** | **1.79** | **1.80** | **1.79** |
| **Körperpflege** | | | | | | |
| Deospray | 150 ml | **1.04** | **1.04** | 2.90 | 1.05 | **2.95** |
| Duschshampoo | 300 ml | **–.36** | –.45 | **–.95** | –.45 | **–.36** |
| Flüssigseife | 500 ml | –.90 | **–.89** | **–.91** | –.90 | –.90 |
| Zahnpasta | 125 ml | **–.59** | **–.59** | **–.60** | **–.60** | **–.60** |
| **Total** | | **66.64** | **66.69** | **72.70** | **79.46** | **83.42** |
| **Differenz zu Lidl** | | | | **Plus 9 %** | **Plus 19 %** | **Plus 25 %** |

Am günstigsten  Am teuersten  Preise in Franken, erhoben am 26. März 2024 in Brugg AG (Coop, Denner, Lidl), Lenzburg AG (Migros) und Niederlenz AG (Aldi). Berücksichtigt wurde stets der günstigste erhältliche Artikel. Bei grösseren Verpackungen wurde der Preis auf die angegebene Menge umgerechnet. Der Preisvergleich berücksichtigt Qualität, Marke und Herkunft nicht. Die Qualität von Artikeln beurteilt der K-Tipp in Labortests.