hochwarth it

DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

CrossSelling im Kontext der Weinbranche: Entwicklung eines Algorithmus zur Berechnung des Geschmacksmusters auf Basis individuellen Kundenverhaltens zum beispielhaften Einsatz im Shopsystem Magento 2.

Bachelorarbeit

für die Prüfung zum

Bachelor of Science

Studiengang Angewandte Informatik

Duale Hochschule Baden-Württemberg Mannheim

von

Sebastian Lember, TINF14AIBI

Abgabedatum:	25.09.2017
Bearbeitungszeitraum:	12 Wochen
Matrikelnummer, Kurs:	8971053, TINF14AIBI
Ausbildungsfirma:	Hochwarth IT GmbH
Betreuer der Ausbildungsfirma:	Sebastian Müller
Gutachter der Dualen Hochschule:	Alexander Radzio

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit mit dem Thema

> *CrossSelling im Kontext der Weinbranche: Entwicklung eines Algorithmus zur Berechnung des Geschmacksmusters auf Basis individuellen Kundenverhaltens zum beispielhaften Einsatz im Shopsystem Magento 2.*

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Mir ist bekannt, dass ich meine Bachelorarbeit zusammen mit dieser Erklärung fristgemäß nach Vergabe des Themas in dreifacher Ausfertigung und gebunden im Sekretariat meines Studiengangs an der DHBW Mannheim abzugeben habe. Als Abgabetermin gilt bei postalischer Übersendung der Eingangsstempel der DHBW, also nicht der Poststempel oder der Zeitpunkt eines Einwurfs in einen Briefkasten der DHBW.

Mannheim, den 25. September 2017

Sebastian Lember

Sperrvermerk

Die Ergebnisse der Arbeit stehen ausschließlich dem auf dem Deckblatt aufgeführten Ausbildungsbetrieb zur Verfügung.

Abstract

Die Aufgabenstellung dieser Arbeit ist es, einen Algorithmus zu entwickeln, der den Weingeschmack eines Kunden von einem Weinhändler anhand seiner gekauften Produkte berechnet. Anhand dieses Weingeschmackes sollen dem Kunden andere Weine vorgeschlagen werden, die ihm aufgrund des ermittelten Geschmacksmusters ebenfalls schmecken könnten. Zunächst werden in dieser Arbeit die Grundlagen der Webtechnologieen HTML, CSS, JavaScript und PHP gesetzt. Anschließend werden künstliche neuronale Netze und die systematische Verkostung von Wein erklärt. Die Umsetzung des Algorithmus erfolg auf Basis eines Magento 2 Onlineshops. Zur Berechnung der Weine wird das künstliche neuronale Netz eingesetzt.

The topic for this thesis is to develop an algorithm that will calculate the wine taste of a customer by his purchased products. On this basis the algorithm suggest wines with the same taste to the customer. The first part of this thesis talks about the basics of web programming. The basics are HTML, CSS, JavaScript and PHP. After that there are some basics about artificial neural networks and the methodic tasting of wine.

Inhaltsverzeichnis

Eidesstattliche Erklärung I

Abkürzungsverzeichnis VI

Abbildungsverzeichnis VII

Tabellenverzeichnis VIII

1 Einleitung 1

1.1 Motivation . . . 1
1.2 Ziel der Arbeit . . . 2
1.3 Aufbau der Arbeit . . . 3

2 Grundlagen 4

2.1 Webtechnologien . . . 4
2.1.1 HTML . . . 4
2.1.1.1 Geschichte . . . 4
2.1.1.2 Aufbau . . . 5
2.1.2 CSS . . . 5
2.1.2.1 Geschichte . . . 7
2.1.2.2 CSS Box Modell . . . 8
2.1.2.3 CSS Syntax . . . 8
2.1.3 JavaScript . . . 9
2.1.3.1 Geschichte . . . 10
2.1.3.2 JavaScript Syntax . . . 10
2.1.4 PHP . . . 11
2.1.4.1 Geschichte . . . 11
2.1.4.2 PHP Syntax . . . 12

2.1.5 Magento 12
2.1.5.1 EAV-Modell 12
2.1.5.2 MVC 14
2.1.5.3 Unterschied von Magento 1 zu Magento 2 15
2.2 Künstliche Neuronale Netze 15
2.3 Grundlagen zur Verkostung von Wein 16

3 Implementierung **18**
3.1 Auswahl von Frameworks und Klassen 18
3.2 Aufsetzen des Online-Shops 18
3.3 Anlegen eines Moduls 19
3.4 Anlegen der Weinattribute 20
3.5 Erstellung einer Matrix 22
3.6 Neuronales Netz 24
3.7 Berechnung der Attributwerte 25
3.8 Anzeige im Frontend 31

4 Zusammenfassung und Ausblick **34**

Literaturverzeichnis **35**
.1 Anhang I

Abkürzungsverzeichnis

Ajax	Asynchronous JavaScript and XML
CSS	Cascading Stylesheet
HTML	Hypertext Markup Language
JS	JavaScript
px	Pixel

Abbildungsverzeichnis

1.1 Umsatz durch E-Commerce (B2C) in Deutschland in den Jahren 1999 bis 2016 sowie eine Prognose für 2017 (in Milliarden Euro) (Quelle: https://de.statista.com/statistik/daten/studie/3979/umfrage/e-commerce-umsatz-in-deutschland-seit-1999/) 2

2.1 Webseite hochwarth-ecom.de mit Stylesheet 6
2.2 Webseite hochwarth-ecom.de ohne Stylesheet 7
2.3 CSS Box Modell 8
2.4 Magento EAV-Modell (Quelle: http://devdocs.magento.com/guides/m1x/magefordev/mage-for-dev-7.html) 13
2.5 caption 14
2.6 Künstliches neuronales Netz (Quelle: https://de.wikipedia.org/wiki/Künstliches_neuronales_Netz) 16

3.1 Anzeige der berechneten Weine im Frontend 33

Tabellenverzeichnis

3.1 Attributematrix 23

1 Einleitung

1.1 Motivation

Der Umsatz durch E-Commerce in Deutschland steigt stetig an. Dies Belegt eine Statistik vom Handelsverband Deutschland aus dem Jahr 2017. Die Statistik in Abb. 1.1 zeigt den Umsatz durch E-Commerce von 1999 bis 2016 in Deutschland in Milliarden Euro und zeigt eine Prognose für das Jahr 2017 auf. So hat sich der Umsatz von 1,1 Milliarden Euro im Jahr 1999 auf 44,2 Milliarden Euro erhöht. Die Prognose für 2017 zeigt eine weitere Umsatzsteigerung von 4,5 Milliarden Euro zum Vorjahr. Diese Umsatzsteigerung liegt zum größten Teil an dem wachsenden Anteil des interaktiven Handelns am Einzelhandel. Aber auch die Händler eines E-Commerce Shops versuchen ihren Umsatz unter anderem durch Cross- und Up-Selling zu erhöhen.

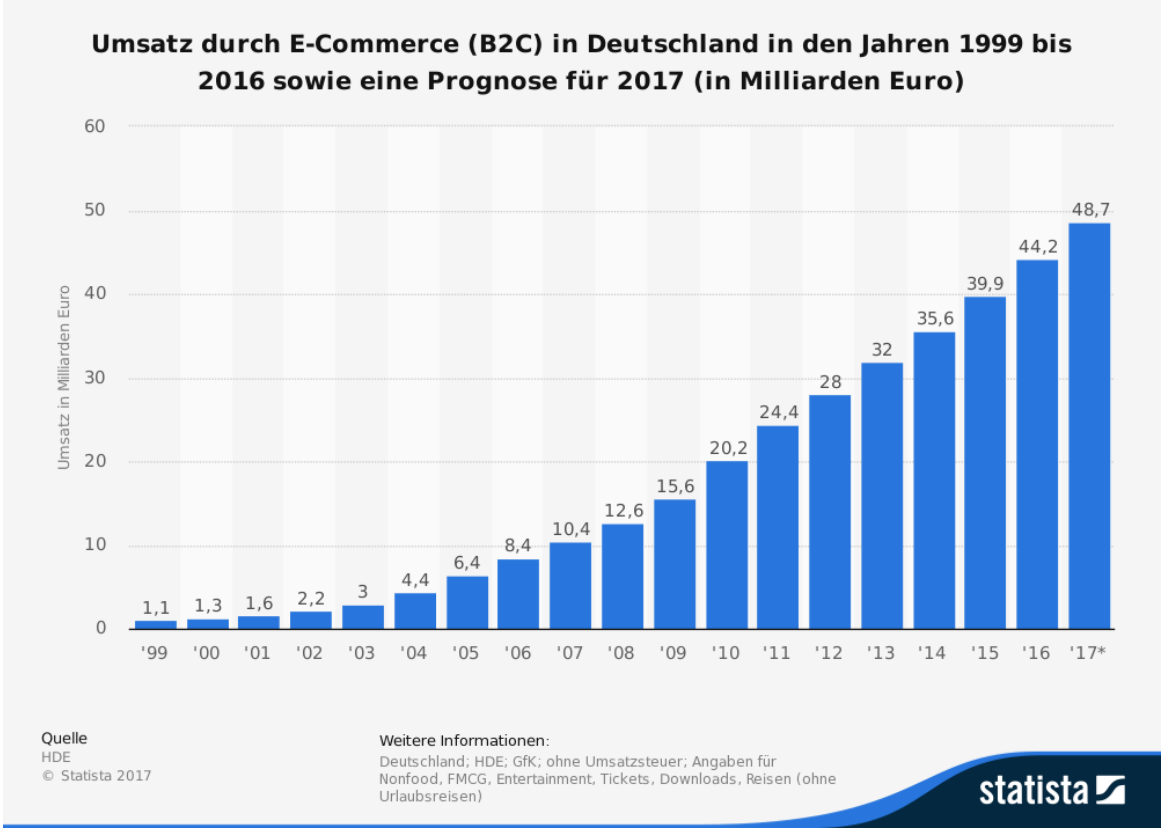

Abbildung 1.1: Umsatz durch E-Commerce (B2C) in Deutschland in den Jahren 1999 bis 2016 sowie eine Prognose für 2017 (in Milliarden Euro) (Quelle: https://de.statista.com/statistik/daten/studie/3979/umfrage/e-commerce-umsatz-in-deutschland-seit-1999/)

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist es, anhand eines Weinshops, einen Algorithmus zu entwickeln, der dem Kunden dieses Shops anhand seiner getätigten Käufe weitere Weine aufzeigt, die auch dem Geschmacksmuster des Kunden entsprechen. Ein weiteres Ziel des Algorithmus ist es, auf lange Sicht dem Umsatz durch Cross-Selling zu steigern.

1.3 Aufbau der Arbeit

Die Arbeit Gliedert sich in drei Teile. Im ersten Teil werden die theoretischen Grundlagen der Arbeit gelegt. Hierbei werden zunächst die benötigten Webtechnologien aufgezeigt. Anschließend werden verschiedene Suchalgorithmen vorgestellt. Im zweiten Teil der Arbeit erfolgt die Durchführung des Projektes und die Implementierung des Algorithmus. Am Ende der Arbeit folgt eine Zusammenfassung, Bewertung und Ausblick des Projektes.

2 Grundlagen

2.1 Webtechnologien

2.1.1 HTML

> HTML ist eine Abkürzung für Hypertext Markup Language. Auf Deutsch bedeutet dies so viel wie „Auszeichnungssprache für verknüpften Text“. HTML ist heute der wichtigste Auszeichnungsstandard im Internet. Er ist Voraussetzung für die Programmierung und das Design von Webinhalten. Andere Standards wie PHP bauen in erheblichem Maße auf HTML auf. [16]

Die erste Version von HTML wurde am 3. November 1992 erstmals veröffentlicht. Die Urversion des HTML orientierte sich zunächst nur an der Darstellung von Text. Somit konnten Überschriften, Absätze, Links und Listen dargestellt werden können. Am 30. April 1993 kam eine weitere Version von HTML in der auch Bildobjekte eingefügt werden konnten. Des weitere wurden Attribute eingeführt, mit denen man das Aussehen des Textes verändern konnte, wie z. B. fette oder kursive Darstellung.

2.1.1.1 Geschichte

Im November 1995 wurde ein neuer HTML Standard (RFC 1866) eingeführt. Somit wurde HTML 2 geboren. Diese Version beinhaltete unter anderem das erzeugen von Formularelementen. Knapp ein Jahr später, am 14. Januar 1997, erschien HTML 3.2. Eine große Neuerung in dieser Version war das Einbinden von Java Applets. Außerdem wurden in dieser Version Tabellen und Textfluss um Bilder hinzugefügt. HTML 4.01, der Nachfolger von HTML 3.2, wurde am 24. Dezember

1999 veröffentlicht. Mit dieser neuen Version konnten Skripte und Stylesheets eingebunden werden. HTML 4.01 war sehr lange Standard, bis am 28. Oktober 2014 HTML 5 eingeführt wurde. Seit dem 1. November 2016 ist HTML 5.1 der neue Standard.

2.1.1.2 Aufbau

Die allgemeine Struktur eines HTML-Dokumentes besteht aus einer Dokumenttypdeklaration, einem HTML-Kopf und einem HTML-Body. Im HTML-Kopf werden meist Dokumenteninformationen angegeben, die im Browser nicht angezeigt werden. Darunter zählen z. B. Metadaten der Seite, Einbindung von Skripten und Stylesheets. Der HTML-Body beinhaltet die Elemente die der Browser darstellt. Dazu gehören z. B. Elemente wie Überschriften, Bilder, Tabellen oder ganz normale Textabsätze. [11]

```
1  <!DOCTYPE html>
2  <html>
3          <head>
4                  <title>Pagetitle</title>
5                  <script src="/js/main.js"></script>
6          </head>
7          <body>
8                  <h1>Heading</h1>
9                  <p>Paragraph</p>
10         </body>
11 </html>
```

Listing 2.1: Aufbau einer HTML-Datei

2.1.2 CSS

CSS steht für Cascading Style Sheets. Es ist eine Sprache die das Aussehen einer Webseite beschreibt. Es wird dafür verwendet, Farbe und Hintergrundbilder der Seite hinzuzufügen, oder um Elemente zu Positionieren. Des weiteren wird es ver-

wendet, um die Nutzbarkeit der Seite zu erhöhen[3]. Abb. 2.1 zeigt eine Webseite in der eine CSS-Datei eingebunden ist. Abb. 2.2 zeigt die gleiche Webseite, jedoch ohne Styles. Man kann gut sehen, dass die Usability der Seite ohne CSS-Datei deutlich schlechter als die Seite mit Stylesheet ist. Es ist zu erkennen, dass das Logo fast die komplette Seite einnimmt. Zudem ist das Menü sehr unübersichtlich, da es nur eine Auflistung von Links ist.

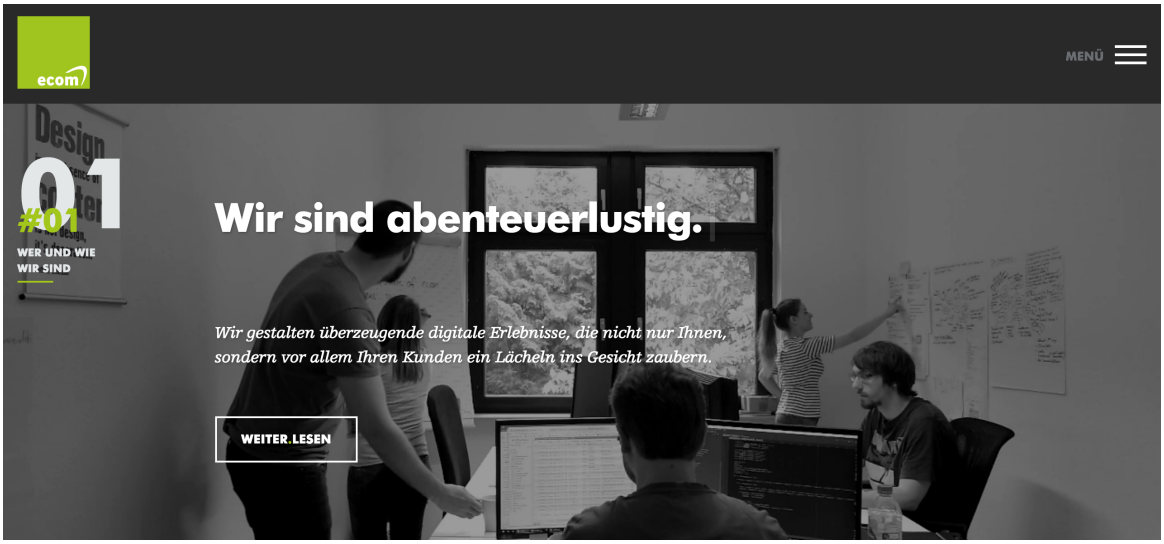

Abbildung 2.1: Webseite hochwarth-ecom.de mit Stylesheet

Abbildung 2.2: Webseite hochwarth-ecom.de ohne Stylesheet

2.1.2.1 Geschichte

CSS wurde erstmals im Dezember 1996 von Håkon Wium Lie und Bert Bos unter dem Namen CSS Level 1 veröffentlicht. Der Nachfolger CSS Level 2 (CSS2) wurde im Mai 1998 veröffentlicht. Jedoch konnten bis Anfang 2010 nur wenige Browser CSS2 umsetzen. Mittlerweile wird an CSS3 gearbeitet. Anders wie seine Vorgänger ist CSS3 Modular aufgebaut. Seit 2014 unterstützen moderne Browser viele CSS3

Module. Nach CSS3 wird es nach jetzigem Stand keinen Nachfolger geben, da die einzelnen Module unter ihren eigenen Versionsnummern laufen sollen. (vgl. [10]

2.1.2.2 CSS Box Modell

Das CSS-Box Modell ist eine Box, die jedes HTML-Element umschließt. Diese Box besteht aus Außenabstand (Margin), Rand (Border), Innenabstand (Padding), Breite und Höhe des Elementes. (vgl. [7])

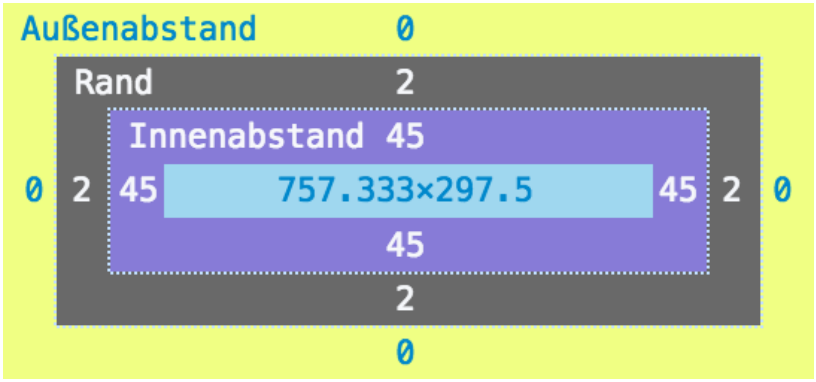

Abbildung 2.3: CSS Box Modell

Abb. 2.3 zeigt ein Box Modell, wie es in den Entwicklertools des Firefox-Browsers zu finden ist. Diese Abbildung zeigt ein Element, das eine Breite von 757,33px und eine Höhe von 297,5px hat. Der Innenabstand dieses Elementes beträgt zu allen Seiten 45px. Außerdem hat das Element einen Rand von 2px.

2.1.2.3 CSS Syntax

Eine CSS-Anweisung besteht aus einem oder mehreren Selektoren, gefolgt von einem Anweisungsblock, der von geschweiften Klammern umgeben ist. Ein Selektor kann ein HTML-Element, eine Klasse oder eine ID sein. Der Selektor bestimmt das Element, welches ein anderes Styling bekommen soll. [9] Um eine genauere Auswahl des zu ändernden Elementes bekommt, gibt es bestimmte Muster, wie die Selektoren angewendet werden können. Nachfolgend gibt es einige Beispiele für solche Muster:

- div, p → Wählt alle div und p Elemente aus.

- div p → Wählt alle p Elemente die innerhalb eines div Elementes vorkommen.
- div > p → Wählt alle p Elemente aus, die direkte Kinder eines div Elementes sind.
- div + p → Wählt alle p Elemente aus, die unmittelbar hinter einem div Element stehen.

(vgl. [8])

Zusätzlich zu den Elementselektoren gibt es noch Selektoren für Pseudoklassen und Pseudoelemente, z. B. :after, :hover. Mit diesen Selektoren können Elemente gestyled werden, über die z. B. mit dem Mauszeiger gefahren wird.

Im Anweisungsblock für einen Selektor steht zunächst die zu verändernde Eigenschaft, gefolgt von einem Doppelpunkt und dem Wert für diese Eigenschaft. Jede Anweisung wird von einem Semikolon abgeschlossen.

```
1 #menu li {
2         background-color: #292929;
3 }
4
5 #menu li.active {
6         background-color: #5c5c5c;
7 }
```

Listing 2.2: CSS-Syntax

Listing 2.2 zeigt ein Beispiel einer CSS-Syntax. In Zeile 1 werden alle li-Elemente ausgwählt, die in einem Element mit der ID "menußind. In Zeile 2 wird die Hintergrundfarbe auf den Hexadezimalwert 292929 gesetzt. In Zeile 6 wird ebenfalls die Hintergrundfarbe gesetzt. Das Element, das dies betrifft, sind alle Listenelemente mit der Klasse äctive", die in einem Element mit der ID "menußind.

2.1.3 JavaScript

JavaScript (kurz JS) ist eine Skriptsprache, die ursprünglich 1995 von Netscape für dynamisches HTML in Webbrowsern entwickelt wurde, um Benutzerinteraktio-

nen auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren und so die Möglichkeiten von HTML und CSS zu erweitern. Heute findet JavaScript auch außerhalb von Browsern Anwendung, so etwa auf Servern und in Microcontrollern. (vgl. [12])

Im Gegensatz zu HTML und CSS ist JavaScript keine Auszeichnungssprache, sondern Programmiersprache und enthält deshalb Variablen, Datentypen, Funktionen und Kontrollstrukturen. Zu den Kontrollstrukturen gehören unter anderem if-else-Bedingungen, switch-case-Statements und Schleifenkonstrukte. JavaScript enthält einige primitive Datentypen, darunter zählen Boolean, Null, Undefined, Number, String, Symbol. Zusätzlich gibt es noch das Objekt als siebten, nicht primitiven Datentyp. (vgl. [6])

2.1.3.1 Geschichte

Javascript wurde zunächst unter dem Namen LiveScript von Netscape am 18. September 1995 veröffentlicht. Noch im selben Jahr, am 4. Dezember 1995, gab es eine Kooperation zwischen Netscape und Sun Microsystems. Dabei sollte die direkte Interaktion mit Java-Applets ermöglicht werden. Somit entstand die Sprache JavaScript. Seitdem wurden viele Versionen veröffentlicht, die immer wieder Neuerungen und geringfügige Updates brachten. Seit 2016 gibt es das ECMAScript 2016, welches jährlich Updates erhalten soll. (vgl. [12])

2.1.3.2 JavaScript Syntax

```
1 function count() {
2         var i = 1;
3         for(i; i <= 5; i++) {
4                 console.log(i);
5         }
6 }
```

Listing 2.3: JavaScript-Syntax

Listing 2.3 zeigt eine Funktion, die eine Variable erzeugt, anschließend eine for-Schleife durchläuft, in der die Variable i so lange hochgezählt wird, bis sie den Wert

5 erreicht hat. In jedem Schleifendurchlauf wird die Variable i in der Konsole ausgegeben. Das Semikolon am Ende jeder Zeile wird nicht von JavaScript verlangt, ist aber ein besserer Programmierstil.

2.1.4 PHP

> PHP (rekursives Akronym für PHP: Hypertext Preprocessor) ist eine weit verbreitete und für den allgemeinen Gebrauch bestimmte Open Source-Skriptsprache, welche speziell für die Webprogrammierung geeignet ist und in HTML eingebettet werden kann.[4]

Durch PHP können dynamisch generierte Webseiten erstellt werden. Der Unterschied hier zwischen clientseitigen Sprachen wie z. B. Javascript ist der, dass PHP auf einem Server ausgeführt wird. Der Code wird also auf dem Server ausgeführt und dieser schickt dann eine Antwort an den Client, in der nur der erzeugte HTML-Code geschickt wird.

2.1.4.1 Geschichte

PHP wurde im Jahr 1995 von Rasmus Lerdorf entwickelt. PHP sollte eine Reihe von Perl-Skripten ersetzen, die Lerdorf geschrieben hatte. Der Nachfolger PHP2 wurde ebenfalls von Rasmus Lerdorf auf Basis der Programmiersprache C entwickelt. 1997 wurde PHP 3 von Andi Gutmans und Zeev Suranski neu geschrieben. Rasmus Lerdorf kooperierte mit den beiden und so wurde die Entwicklung von PHP2 eingestellt und die Verbreitung der Skriptsprache PHP vorangetrieben. Gutmans und Suranski gründeten in Folge die Firma Zend Technologies Ltd., welche die Zend Engine 1 entwickelte. PHP4 baut auf dieser Engine auf und verbesserte die Ausführungsgeschwindigkeit von komplexen Applikationen. Auf Basis der Zend Engine 2 wurde 2004 die 5. Version von PHP veröffentlicht. Diese Version brachte Neuerungen hinsichtlich der Objektorientierung. Mehrere Unterversionen von PHP5 brachten weitere Verbesserungen der Objektorientierung in PHP. Nachdem die Entwicklung des Nachfolgers von PHP5 eingestellt wurde, wurde entschieden, dass die nächste Versionsnummer 7 ist. Mit PHP7 wurde unter anderem die Ausführungsgeschwindigkeit um 30 Prozent reduziert. Seit Dezember 2016 gibt es die Version PHP 7.1. (vgl. [15])

2.1.4.2 PHP Syntax

```
1 function count() {
2         $variable = 0;
3         for($variable; $variable <= 5; $variable++) {
4                 echo $variable;
5                 echo "<br>";
6         }
7 }
```

Listing 2.4: PHP-Syntax

Die Funktion erzeugt das gleiche Ergebnis wie die JavaScript Funktion in Listing 2.3. Die Variable wird allerdings nicht in der Konsole ausgegeben, sondern wird im Browser angezeigt. Bei jeder Iteration wird zusätzlich noch ein Linebreak mittels
 hinter jeder Ausgabe der Variablen hinzugefügt. Variablen erkennt man am Dollar-Zeichen vor dem Variablennamen. Variablen benötigen keine Typdeklaration, da der PHP Interpreter zur Laufzeit aus dem Kontext der Variable den Datentyp festlegt. Im Vergleich zu JavaScript muss bei PHP am Ende jeder Zeile eine Semikolon eingefügt werden. Tut man dies nicht wirft der PHP Interpreter zur Laufzeit einen Fehler.

2.1.5 Magento

Magento ist eine reine Onlineshopsoftware, die von der Firma Varien (heute Magento Inc, im Jahr 2007 entwickelt und am 31. März 2008 veröffentlicht wurde. Magento Inc. war lange Zeit eine Tochterfirma von Ebay und wurde im November 2011 von der Firma Permira übernommen. Am 17. November 2015 wurde eine neue Major Release von Magento veröffentlicht und ist seitdem in der Version 2 verfügbar.

Magento wurde unter Zuhilfenahme des Zend Frameworks entwickelt, wodurch es völlig objektorientiert realisiert werden konnte. Für die Datenspeicherung benutzt Magento das EAV-Modell in einer MySQL Datenbank. Zudem besitzt Magento ein eigenes MVC-Pattern.(vlg. [14], [17])

2.1.5.1 EAV-Modell

EAV ist ein Datenmodell für den Fall, dass die Anzahl der Attribute eines Objektes nicht vorhersehbar sind.

(vgl. [2]) Dadurch ist es Möglich die Datenstruktur eines Objektes möglichst flexibel zu gestalten. Nachteile dieses Modells sind jedoch längere Ladezeiten und kompliziertere SQL-Queries, da mehrere Tabellen miteinander Verknüpft werden müssen.

Abb. 2.4 zeigt das Magento EAV-Modell anhand der Produkte.

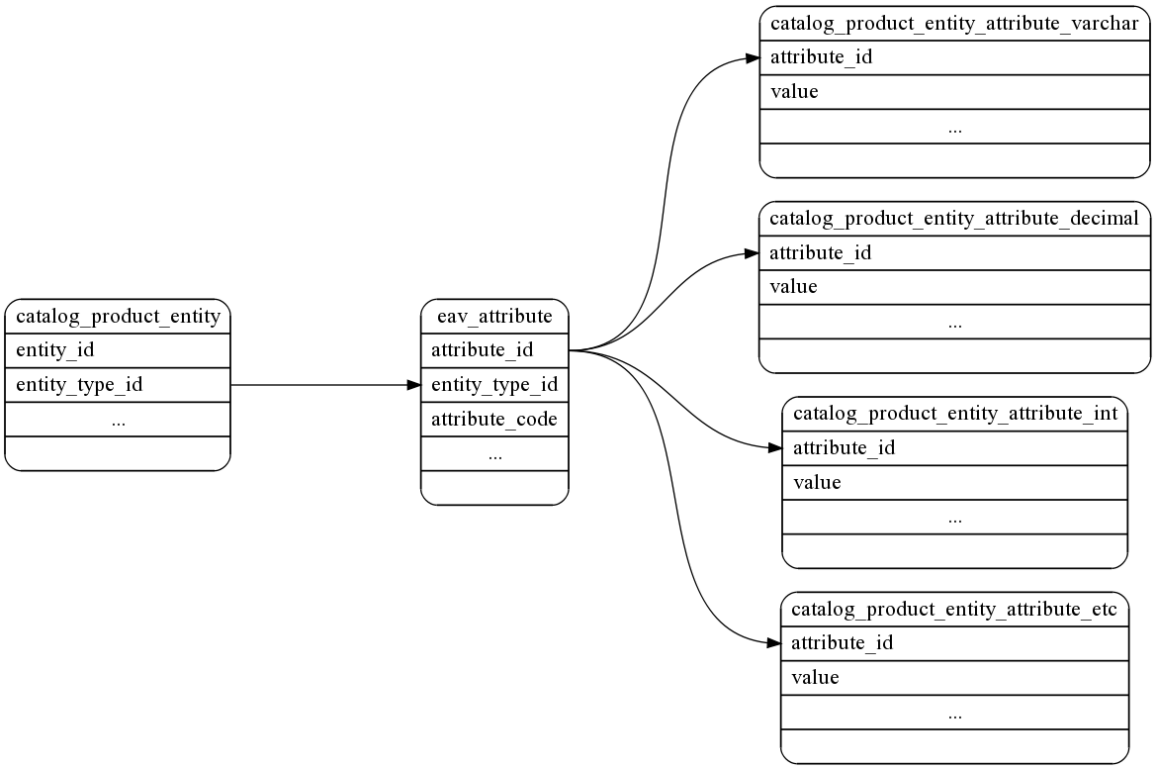

Abbildung 2.4: Magento EAV-Modell
(Quelle: http://devdocs.magento.com/guides/m1x/magefordev/mage-for-dev-7.html)

Jedes einzelne Produkt is in der `catalog_product_entity` Tabelle vorhanden. Über die `entity_type_id` werden in der `eav_attribute` Tabelle die entsprechenden Attribute geladen. Der Wert dieser Attribute wird in den Tabellen catalog_product_entity_attribute_ und dem Datentyp des Attributwertes, z. B. catalog_product_entity_attribute_varchar, gespeichert. (vgl. [5])

2.1.5.2 MVC

MVC ist ein Entwurfsmuster, das die Software in Datenschicht (Model), Präsentationsschicht (View) und Programmsteuerungsschicht (Controller) unterteilt. Ziel dieses Musters ist es, die spätere Änderung oder Erweiterung einzelner Komponenten zu vereinfachen. Abb. 2.5 zeigt die Funktionsweise des MVC. Der Benutzer

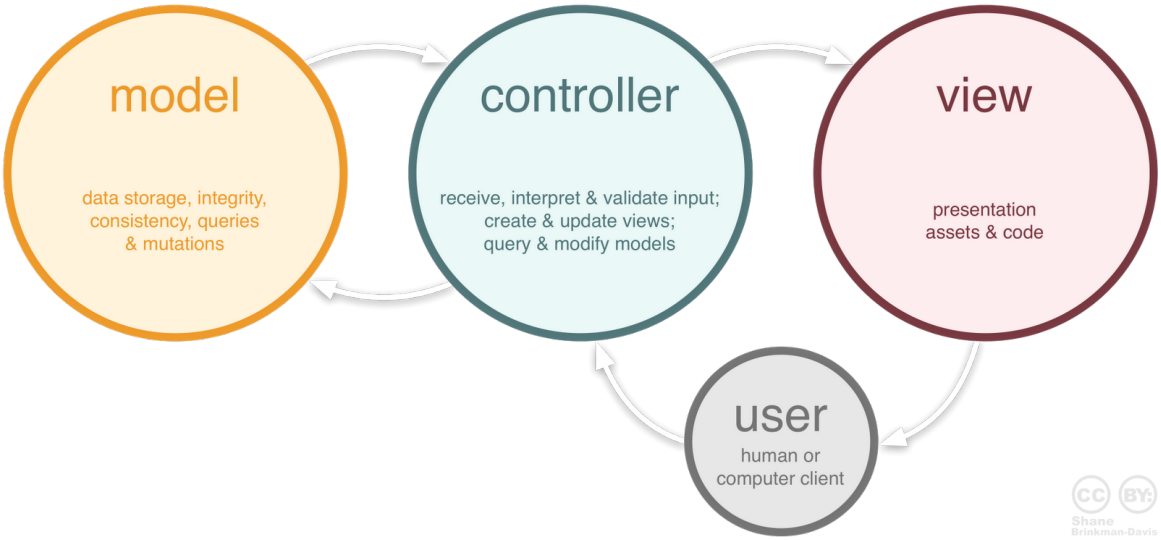

Abbildung 2.5: MVC-Pattern
(Quelle: https://www.codeproject.com/Articles/879896/Programming-in-Java-using-the-MVC-architecture)

tätigt eine Eingabe. Diese Eingabe wird an den Controller geschickt. Anhand der Benutzereingabe manipuliert oder liest der Controller das Model aus und schickt dann die Daten an den View. Der View gibt diese Daten am Bildschirm des Benutzers aus.

In vielen MVC Frameworks wird jeweils das Model, der Controller und der View in einer eigenen Datein angelegt. Magento hat für Model und Controller auch jeweils eine Datei. Der View setzt sich in Magento allerdings aus mehreren Dateien zusammen. Für den View benutzt Magento Layout-, Block- und Template-Dateien. In den Layout Dateien wird festgelegt, an welcher Position im Elementfluss ein bestimmter Block angezeigt wird und welche Template-Datei dieser Block rendern soll. Ein View in Magento besteht also aus einer zusammengesetzten Layout-Datei, vielen Block und Template-Dateien.

2.1.5.3 Unterschied von Magento 1 zu Magento 2

Die Einführung von Magento 2 bringt natürlich einige Neuerungen mit sich. Zunächst einmal baut Magento 2 auf eine neuen Codebasis auf. Des weiteren soll mit Magento 2 die Usability der Benutzer verbessert werden. Außerdem wurde viel an der Performance geschraubt. Somit sollen durch den Einsatz von PHP7 und einem Full Page Cache die Ladezeiten der Seite verkürzen. Allerdings konnte die Performance Verbesserung zwischen einem optimierten Magento 1 und einem Magento 2 Shop noch nicht eindeutig belegt werden. (vgl. [1])

2.2 Künstliche Neuronale Netze

Ein künstliches neuronales Netz ist eine Verkettung mehrerer künstlicher Neuronen. Künstliche neuronale Netze sind einen zweig der künstlichen Intelligenz dar. Hierbei gibt es mehrere Eingabeneuronen, Ausgabeneuronen und eine Reihe von versteckten Neuronen, die zur Lösung verschiedenster Aufgaben trainiert werden können.

Abb 2.6 zeigt ein künstliches neuronales Netz mit zwei Eingabeknoten, einem Ausgabeknoten und 5 versteckten Knoten. Jedes Neuron ist mit jedem Neuron der nächsten Schicht verbunden. Anhand von Gewichtungsvektoren und Aktivierungsfunktionen der einzelnen Neuronen wird somit eine Ausgabe für das Ausgabeneuron erzeugt.

Es gibt mehrere Typen von künstlichen neuronalen Netzen. Darunter zählen feedforward-Netze und feedbackward-Netze, die sich durch ihre Netztopologie unterscheiden. Zudem gibt es auch mehrere Arten diese Netze zu trainieren. Unter anderem kann hierbei überwachtes oder unüberwachtes Lernen eingesetzt werden. Beim überwachten Lernen wird hierbei ein Eingabemuster gegeben. Das neuronale Netz erzeugt in seinem aktuellen Zustand eine Ausgabe. Diese wird anschließend mit dem Ergebnis verglichen, was das neuronale Netz hätte ausgeben sollen. Anhand dieses Vergleiches können Änderungen an der Konfiguration des Netzes vorgenommen werden. Beim unüberwachten Lernen wird dem neuronalen Netz nur Eingabemuster gegeben. Anhand dieser Muster konfiguriert sich das neuronale Netz von alleine. (vgl. [13])

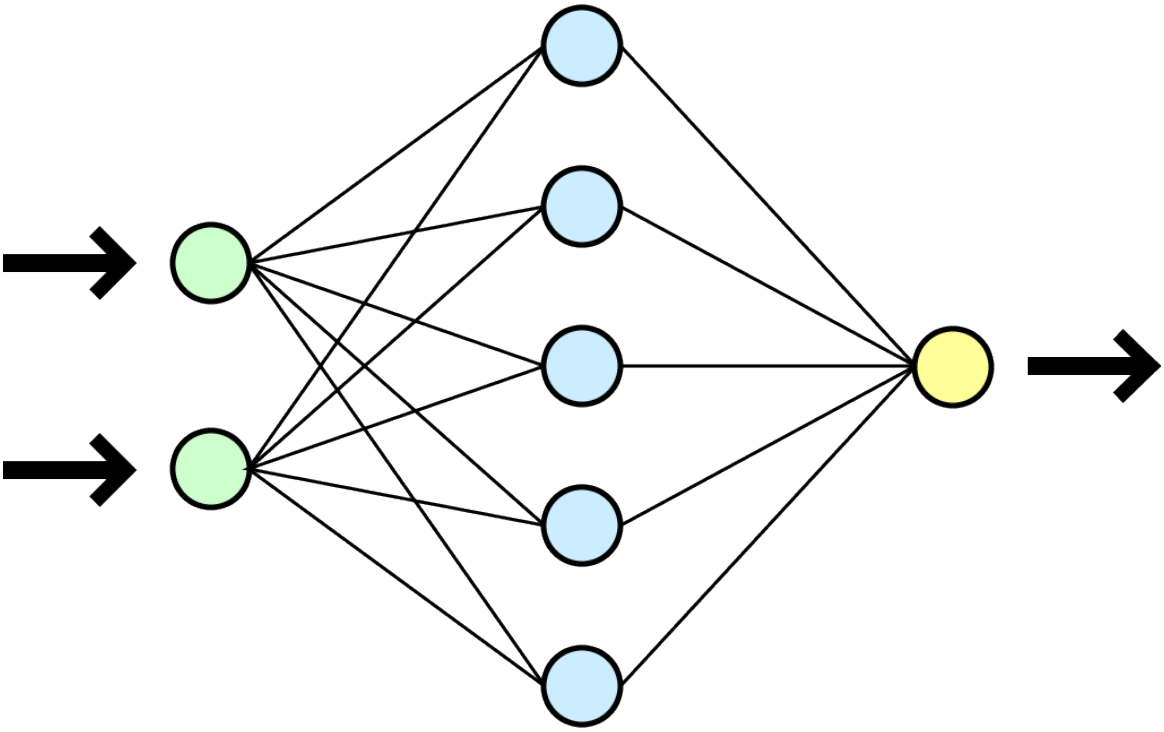

Abbildung 2.6: Künstliches neuronales Netz
(Quelle: https://de.wikipedia.org/wiki/Künstliches_neuronales_Netz)

Einige Anwendungsgebiete von künstlichen neuronalen Netzen sind:

- Schrift- und Spracherkennung
- Bilderkennung und -verarbeitung
- Frühwarnsysteme
- Übersetzung

2.3 Grundlagen zur Verkostung von Wein

Bei der Verkostung von Wein zählt nicht nur der geschmackliche Eindruck. Zunächst einmal wird der Wein nach seinem optischen Erscheinungsbild eingeordnet. Bestimmt wird hierbei die Klarheit des Weines, anschließend der Farbton und

die Farbtiefe. Für die Zuordnung der Farbe der verschiedenen Weinarten gibt es eine Vielzahl von untergeordneten Farbtönen.

Im weiteren Verlauf der Verkostung werden die geruchlichen Merkmale des Weines bestimmt. Dazu gehören die Reintönigkeit und Inntensität des Weines. Außerdem können Aromaausprägungen im Wein gerochen und bestimmt werden. (Vgl. WSET Level 2 im Anhang).

Nach den Geruchsmerkmalen wird noch der Geschmack identifiziert. Neben Süße und Säure wird auch der Abgang des Weines bestimmt. Ähnlich wie beim Geruch werden die Geschmacksausprägungen des Weines charakterisiert. (Vgl. WSET Level 2 im Anhang).

WSET Level2 im Anhang zeigt eine Vorlage zum Verkosten von Wein. In der Vorlage sieht man eine Reihe von Attributen, die bei der Verkostung von Wein ins Spiel kommen. Diese Vorlage enthält jedoch lange nicht alle Attribute, die bei der Verkostung bestimmt werden können.

3 Implementierung

3.1 Auswahl von Frameworks und Klassen

Da die Basis für den Algorithmus ein Online-Shop sein soll und die Firma Hochwarth IT schon seit der Einführung von Magento auf diese Shoplösung setzt, wurde Magento als E-Commerce Framework ausgewählt. Im speziellen Magento 2, da Ursprünglich der Support von Magento 1 eingestellt werden sollte.

Für die Berechnung der zum Kunden passenden Weine wurde auf ein neuronales Netz entschieden, weshalb die Klasse `"class_neuralnetwork.class"` verwendet wurde. (https://github.com/infostreams/neural-network)

3.2 Aufsetzen des Online-Shops

Zuerst wird das Magento 2 Community Paket über Composer heruntergeladen.

```
composer create-project
        --repository-url=https://repo.magento.com/
        magento/project-community-edition
        <installation directory name>
```

Listing 3.1: Magento 2 Composer Projekt erzeugen

Magento 2 hat ein Kommandozeilentool, über dieses anschließend die Installation von Magento gestartet wird.

```
bin/magento setup:install --admin-firstname=Sebastian
        --admin-lastname=Lember --admin-email=lember@hochwarth-it.de
        --admin-user=admin --admin-password=admin123
        --base-url=http://bachelorarbeit.dev/
        --backend-frontname=admin --db-host=localhost
```

```
6           --db-name=bachelorarbeit --db-user=root --db-password=root
7           --language=de_DE --currency=EUR --use-rewrites=1
```

Listing 3.2: Magento 2 installation

Bei der Installation wird ein Backendbenutzer angelegt, außerdem wird dem Skript die Datenbankverbindung und der Datenbankname übergeben. In diese Datenbank installiert das Tool alle Datenbanken, die Magento benötigt.

3.3 Anlegen eines Moduls

Der nächste Schritt wäre, die ganzen Attribute für die Weine anzulegen. Dazu benötigt man ein Modul, welches zuerst angelegt werden muss. Um ein Modul anzulegen werden zwei Dateien benötigt, die nachfolgend gezeigt werden.

```
1 <?xml version="1.0"?>
2
3 <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:noNamespaceSchemaLocation=
5                 "urn:magento:framework:Module/etc/module.xsd">
6     <module name="HochwarthIT_CustomerWine" setup_version="0.0.1">
7     </module>
8 </config>
```

Listing 3.3: Magento 2 Modul - module.xml

In der Module XML wird in Zeile 6 der Name und die Setup Version des Moduls festgelegt. Anhand der Setup Version werden bestimmte Dateien zur Änderung an der Datenbankstruktur ausgeführt, falls diese Vorhanden sind.

```
1 <?php
2
3     use Magento\Framework\Component\ComponentRegistrar;
4
5     ComponentRegistrar::register(
6     ComponentRegistrar::MODULE,
7     'HochwarthIT_CustomerWine',
8     __DIR__
```

```
9 );
```

Listing 3.4: Magento 2 Modul - registration.php

Die registration.php registriert das Modul im Magento System. Dies geschieht in Zeile 5 durch die ComponentRegistrar Class. In Zeile 6 wird festgelegt, dass ein Modul registriert werden soll.

3.4 Anlegen der Weinattribute

Nachdem das Modul installiert ist, wird ein Setup-Skript für dieses Modul angelegt. Dies ermöglicht es, die Attribute für den Wein hinzuzufügen. Das Setup-Skript besteht aus einer Datei, die eine Klasse InstallData und eine Funktion install enthält. Aufgrund der hohen Komplexität der Attribute wurden neun Attribute aus dem WSET Level 2 Bogen ausgewählt. Das sind die Attribute Klarheit, Farbtiefe, Farbton, Reintönigkeit, Intensität, Aromaausprägungen, Süße, Säure und Geschmacksausprägung.

```
1  public function install(
2          ModuleDataSetupInterface $setup,
3          ModuleContextInterface $context
4  )
5          {
6              $eavSetup = $this->eavSetupFactory->create(["setup" => $setup]);
7
8              $eavSetup->addAttribute(
9                  Product::ENTITY,
10                 "clearness",
11                 [
12                     "type" => "int",
13                     "label" => "Klarheit",
14                     "input" => "select",
15                     "global" => true,
16                     "visible" => true,
17                     "required" => false,
18                     "visible_on_front" => true,
19                     "default" => null,
20                     "option" => [
21                         "values" => [
```

```
22                              "klar",
23                              "trüb"
24                          ]
25                      ]
26                  ]
27              );
28
29              $eavSetup->addAttribute(
30                  Product::ENTITY,
31                  "shade",
32                  [
33                      "type" => "int",
34                      "label" => "Farbtiefe",
35                      "input" => "select",
36                      "global" => true,
37                      "visible" => true,
38                      "required" => false,
39                      "visible_on_front" => true,
40                      "default" => null,
41                      "option" => [
42                          "values" => [
43                              "blass",
44                              "mittel",
45                              "tief"
46                          ]
47                      ]
48                  ]
49              );
50
51              //...
52          }
```

Listing 3.5: Magento 2 Installlationsskript für Weinattribute - InstallData.php

In Zeile 6 wird ein neues EAV-Setup erzeugt. Diese Klasse enthält die Methoden um ein Attribut zu den Produkten hinzuzufügen. In Zeile 8 wird eine neues Attribut hinzugefügt. Der Funktion addAttribute wird in Zeile 9 übergeben, ob es sich um eine Produkt Entität oder eine Kunden Entität handelt. Zeile 12 enthält den Namen des Attributes. Der letzte Parameter der addAttribute-Funktion ist ein Array mit Optionen die das Attribut enthält. In den Optionen wird unter anderem

der Datentyp in der Datenbank festgelegt (type). Zudem kommen der Input-Typ (input), ob das Feld benötigt wird (required) oder ob es im Frontend sichtbar ist (visible_on_front). Zudem können über öption"die Eingabewerte des select-Feldes eingetragen werden. Hat man alle neun Attribute in der Datei eingetragen, wird über die Konsole

```
1 bin/magento setup:upgrade
```

Listing 3.6: Magento 2 Datenbankänderungen installieren

ausgeführt. Diese Funktion schreibt die erstellten Attribute in der Datenbank.

3.5 Erstellung einer Matrix

Als nächstes wird eine Matrix erstellt, die zu jeder Attributkombination einen Wert liefert. Da die komplexität sehr hoch ist, wurde die Anzahl der Attribute auf drei gekürzt. Die maximale Anzahl an an Werten die die Attribute annehmen können wurde auf fünf festgelegt. Die drei Attribute sind der Farbton, die Aromaausprägung und die Geschmacksausprägung. Für den Farbton wurden folgende Werte festgelegt.

- 1 → zitronengelb
- 2 → bernsteingelb
- 3 → hellrosa
- 4 → purpurrot
- 5 → granatrot

Für die Aromaausprägung und Geschmacksausprägung wurden folgende Werte festgelegt.

- 1 → Früchte
- 2 → Blumen
- 3 → Gewürze

- 4 → pflanzlich

- 5 → Eichennoten

Die nachfolgende Tabelle enthält einen kleinen Ausschnitt aus der Matrix, da die Attribute 125 Kombinationen annehmen können.

Farbton	Aromaausprägung	Geschmackausprägung	Gesamtwert
1	1	1	100
1	1	2	600
1	1	3	1100
1	1	4	1600
1	1	5	2100
1	2	1	150
1	2	2	650
1	2	3	1150
1	2	4	1650
1	2	5	2150
1	3	1	200
1	3	2	700
1	3	3	1200
1	3	4	1700
1	3	5	2200
1	4	1	250
1	4	2	750
1	4	3	1250
1	4	4	1750
1	4	5	2250
1	5	1	300
1	5	2	800
1	5	3	1300
1	5	4	1800
1	5	5	2300

Tabelle 3.1: Attributematrix

Der Ausschnitt der Matrix zeigt alle Aroma- und Geschmackausprägungsvariationen mit dem gleichen Farbton.

3.6 Neuronales Netz

Nachdem die Matrix erstellt wurde, kann nun das neuronale Netz erzeugt werden. Ziel des neuronalen Netzes ist, anhand der Attributkombinationen zu errechnen, ob die Attributkombinationen gut zu einander passen, weniger gut zueinander passen oder gar nicht zueinander passen. Dazu wird das neuronale Netz mit zwei Inputneuronen, einem Outputneuronen und einer versteckten Schicht mit 4 Neuronen konfiguriert. Anschließend wird das neuronale Netz mit Testdaten trainiert.

```
1  $n = new NeuralNetwork(array(2, 4, 1));
2  $n->setVerbose(false);
3
4  $n->addTestData( array (1, 1), array (1));
5  $n->addTestData( array (0.4, 1), array (1));
6  $n->addTestData( array (0.25, 1), array (1));
7  $n->addTestData( array (0.06, 1), array (-1));
8  $n->addTestData( array (0.09, 1), array (-1));
9  $n->addTestData( array (0.03, 1), array (-1));
10
11 $max = 3;
12 $i = 0;
13
14 while (!($success = $n->train(1000, 0.01)) && ++$i<$max) {
15         echo "Round $i: No success...<br />";
16 }
17
18 if ($success) {
19     $epochs = $n->getEpoch();
20     echo "Success in $epochs training rounds!<br />";
21     $n->save("my_network.ini");
22 }
```

Listing 3.7: Künstliches neuronales Netz Training

Das neuronale Netz wird mit sechs Testdatensätzen trainiert. Die Testdatensätze bestehen immer aus den Input-Werten und dem Output-Wert. Es werden drei Trainings mit jeweils 1000 Runden durchlaufen (Zeile 14). Entsprechen die berechneten Werte den Output-Werten mit einer maximalen Abweichung von 0,01, so ist das Training erfolgreich und die Gewichtsvektoren werden in der my_network.ini gespeichert (Zeile 21).

3.7 Berechnung der Attributwerte

Um die Werte der Attributkombinationen zu berechnen, wird zunächst eine Tabelle angelegt, die bei einem Einkauf die Artikel-ID, die Kunden-ID und die Menge abspeichert. Die Tabelle dient zum schnelleren Zugriff auf die eingekauften Produkte eines Kunden, um bei einem weiteren Einkauf entsprechende Weine vorzuschlagen. Um die Tabelle zu erzeugen wird in dem bereits angelegten Modul eine neue Klasse generiert. Diese Klasse erhält eine Funktion install, die unser Datenbankschema erzeugt.

```
1  public function install(
2          SchemaSetupInterface $setup,
3          ModuleContextInterface $context
4  )
5          {
6              $setup->startSetup();
7
8              $tableName = $setup->getTable("hochwarthit_customer_purchases");
9
10             if(!$setup->getConnection()->isTableExists($tableName)) {
11                 $table = $setup->getConnection()
12                     ->newTable($tableName)
13                     ->addColumn(
14                         "id",
15                         Table::TYPE_INTEGER,
16                         null,
17                         [
18                             "identity" => true,
19                             "unsigned" => true,
20                             "nullable" => false,
21                             "primary" => true
22                         ],
23                         "ID"
24                     )
25                     ->addColumn(
26                         "customer_id",
27                         Table::TYPE_INTEGER,
```

```
28                              null,
29                              [
30                                  "unsigned" => true,
31                                  "nullable" => false,
32                              ],
33                              "Customer ID"
34                          )
35                          //...
36                          ->setOption("type", "InnoDB")
37                          ->setOption("charset", "utf8");
38
39                      $setup->getConnection()->createTable($table);
40                  }
41
42                  $setup->endSetup();
43              }
```

Listing 3.8: Magento 2 Installlationsskript für Datenbankschemas - InstallSchema.php

Diese Funktion erzeugt in Zeile 11 eine neue Tabelle. Dieser werden in Zeile 13 und den darauffolgenden Zeilen die Spalten id, `customer_id`, `product_id` und qty hinzugefügt. Die Funktion addColumn enthält als Parameter den Spaltennamen, den Datentyp der Spalte, die Größe der Spalte und weitere Eigenschaften, die die Spalte bekommen soll, z. B. kann sie den Wert null annehmen (nullable) oder wird sie automatisch hochgezählt (identity). In Zeile 39 wird die Tabelle letztendlich der Datenbank hinzugefügt. Zusätzlich zur Tabelle wird noch ein Model angelegt, damit über Magento-Wege auf die Tabelle zugegriffen werden kann.

Anschließend wird eine Funktion geschrieben, die nach einem erfolgreichen Einkauf die Tabelle befüllt. Hierbei wird auf ein Magento Event mit dem Namen save_order_after zugegriffen. Dieses Event wird ausgeführt, nachdem die Bestellung in der Datenbank gespeichert wurde. In einer events.xml, die ebenfalls innerhalb des Modules liegt, wird festgelegt, welche Observer-Klasse bei diesem Event aufgerufen wird.

```
1 <?xml version="1.0"?>
2 <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3         xsi:noNamespaceSchemaLocation="urn:magento:framework:Event/etc/events.xsd">
```

```
4       <event name="save_order_after">
5           <observer name="hochwarthitSaveOrderAfter"
6                   instance="HochwarthIT\CustomerWine\Observer\OrderAfterObserver"/>
7       </event>
8   </config>
```

Listing 3.9: Magento 2 Event Observer - events.xml

In Zeile 4 ist zu erkennen, welches Event aufgerufen wird. In Zeile 6 wird festgelegt, dass die Klasse HochwarthIT\CustomerWine\Observer\OrderAfterObserver bei dem Event ausgeführt wird.
Anschließend wird die Observer-Klasse erzeugt. Die Klasse enthält die Funktion execute, die beim Aufruf des Events ausgeführt wird. Das Event übergibt die Bestellung.

```
1   public function execute(\Magento\Framework\Event\Observer $observer)
2   {
3       /** @var Order $order */
4       $order = $observer->getEvent()->getOrder();
5       $customerId = $order->getCustomerId();
6       $items = $order->getAllItems();
7
8       foreach ($items as $item) {
9
10          $collection = $this->collectionFactory->create();
11          $customerWinePurchase =
12          $this->collectionFactory
13                  ->addFieldToFilter("customer_id", $customerId)
14                  ->addFieldToFilter("product_id", $item->getProductId())
15                  ->getFirstItem();
16
17          if($customerWinePurchase->getId()) {
18              $customerWinePurchase
19                      ->setData("qty",
20                      $customerWinePurchase->getData("qty") + $item->getQtyOrdered());
21          }
22          else {
23
24              $this->customerPurchasesFactory->create();
25              $this->customerPurchasesFactory
26                  ->setData("customer_id", $customerId);
```

```
27                 $this->customerPurchasesFactory
28                     ->setData("product_id", $item->getProductId());
29                 $this->customerPurchasesFactory
30                     ->setData("qty", $item->getQtyOrdered());
31                 $this->customerPurchasesFactory->save();
32
33             }
34         }
35
36 }
```

Listing 3.10: Magento 2 Observer - OrderAfterObserver.php

In Zeile 4 wird die Bestellung aus dem Event geladen. Anschließend wird aus der Bestellung die Kunden-ID extrahiert. In Zeile 6 werden alle Produkte, die die Bestellung enthält geladen und in einer foreach-Schleife in Zeile 8 durchlaufen. In Zeile 10 bis 15 wird aus der zuvor erstellten Tabelle der Eintrag geladen, der die entsprechende Kunden-ID und Produkt-ID hat. Gibt es einen Eintrag, so wird die Menge des Artikels um die Bestellte Menge erhöht (Zeile 20). Wenn es noch keinen Eintrag mit der Kunden-Artikel-Kombination gibt, so wird in den Zeilen 24 bis 31 ein neuer Eintrag angelegt.
Anschließend wird ein Cronjob erstellt, der bei Kunden, der mehr als 5 Weine bestellt hat, für jeden Wein den Wert der Attributkombination aus der Matrixtabelle ausliest. Danach werden die Werte miteinander verglichen und der Mittelwert von dem am meisten zusammenliegenden Werten berechnet. Dieser Wert wird anschließend in einer Tabelle gespeichert. Des weiteren liest der Cronjob für jeden Wein die Attributkombination aus und speichert diese im Produkt.

```
1  protected function execute()
2  {
3      $customerPurchasesCollection =
4          $this->customerPurchasesCollectionFactory->create();
5      $productFactory = $this->productFactory->create();
6      $customerPurchases = array();
7
8      foreach ($customerPurchasesCollection as $item) {
9          $customerPurchases[$item->getData("customer_id")]
10                 [$item->getData("product_id")] = $item->getData("qty");
11     }
```

```
12
13      foreach ($customerPurchases as $key => $products) {
14          foreach ($products as $qty) {
15              isset($qty[$key]) ? $qty[$key] += $qty : $qty[$key] = $qty;
16          }
17      }
18
19      foreach ($qty as $key => $value) {
20          $attribute_values = array();
21          if($value > 5) {
22              foreach ($customerPurchases[$key] as $productId) {
23                  $product = $productFactory->load($productId);
24                  $attribute_values[] = $this->matrixFactory
25                      ->create()
26                      ->addFieldToFilter("color", $product->getData("color"))
27                      ->addFieldToFilter("aroma", $product->getData("aroma"))
28                      ->addFieldToFilter("taste", $product->getData("taste"))
29                      ->getFirstItem()
30                      ->getData("value");
31              }
32              for($i = 0; $i < count($attribute_values); $i++) {
33                  $selectedAttributes[$attribute_values[$i]][] =
34                          $attribute_values[$i];
35                  for($j = $i + 1; $j < count($attribute_values); $j++) {
36                      if(abs($attribute_values[$i] - $attribute_values[$j]) <= 500) {
37                          $selectedAttributes[$attribute_values[$i]][] =
38                                  $attribute_values[$j];
39                      }
40                  }
41              }
42              $max = null;
43              foreach($selectedAttributes as $selectedAttribute) {
44                  if($max && count($max) < count($selectedAttribute)) {
45                      $max = $selectedAttribute;
46                  }
47              }
48              $attributeValue = array_sum($max) / count($max);
49
50              $customerAttributes = $this->customerAttributeValuesFactory->create();
51              $customerAttributes->setData("customer_id", $key);
```

```
52                 $customerAttributes->setData("value", $attributeValue);
53                 $customerAttributes->save();
54             }
55         }
56 
57         $productCollection = $this->productCollectionFactory->create();
58         foreach ($productCollection as $product) {
59             $attributeValue = $this->matrixCollectionFactory->create()
60                 ->addFieldToFilter("color", $product->getData("color"))
61                 ->addFieldToFilter("aroma", $product->getData("aroma"))
62                 ->addFieldToFilter("taste", $product->getData("taste"))
63                 ->getFirstItem()
64                 ->getData("value");
65 
66             $product->setData("attribute_value", $attributeValue);
67             $product->save();
68         }
69 }
```

Listing 3.11: Magento 2 Cronjob - Cron.php

In Zeile 3 und 4 werden alle Einträge der customer_purchases Tabelle geladen. In den Zeilen 8 bis 11 werden die Einträge nach Kunde sortiert. Anschließend wird in Zeile 13 bis 17 die Menge der eingekauften Artikel nach Kunde sortiert. In Zeile 19 folgende werden alle Mengen durchlaufen und in Zeile 21 überprüft, ob die Zahl größer oder gleich fünf ist. Wenn dies der Fall ist, wird für jeden Kunden alle gekauften Produkte geladen (Zeile 23). In Zeile 24 bis 31 wird für jedes Produkt der Attributwert aus der Matrixtabelle geladen. Danach werden in Zeile 32 bis 40 die Attribute verglichen und alle Werte die im Bereich von 500 um einen Wert liegen zusammen in ein Array geschrieben. In den Zeilen 42 bis 46 wird der Array mit den meisten Einträgen gesucht. Anschließend wird in Zeile 47 der Mittelwert dieser Werte berechnet und in den Zeilen 49 bis 52 in eine neue Tabelle geschrieben, die den Mittelwert zu einem Kunden speichert. In den Zeilen 56 bis 67 werden alle Produkte geladen, der zugehörige Attributwert ausgerechnet und im Produkt gespeichert.

3.8 Anzeige im Frontend

Nachdem die Attributwerte von allen Einkäufen und allen Produkten berechnet sind, können nun passende Weine im Frontend angezeigt werden können. Dazu wird zunächst ein Ajax-Request ausgeführt. Dieser ruft eine Funktion auf, die den Attributwert des Kunden und aller Produkte ausliest. Anschließend wird der kleinere Attributwert durch den größeren geteilt. Das geschieht für jedes Produkt. Die Zahl die dabei entsteht wird dem neuronalen Netz übergeben, welches berechnet, ob die Weine einen ähnlichen Geschmack für den Kunden haben könnten. Ist die Berechnung erfolgt, werden alle Weine die dem Geschmack des Kunden ähnlich sind angezeigt.

```
1 require(["jquery"], function ($) {
2     $.ajax({
3         url: "http://shop.dev/customerwine/calculate",
4         type: "post",
5         success: function(data){
6             $("#customerProducts").html(data);
7         }
8     })
9 });
```

Listing 3.12: Magento 2 Ajax Aufruf - customerWine.js

In Zeile 1 wird die jQuery-Bibliothek geladen. In den Zeilen 2 bis 8 wird ein Ajax-Requets an die URL http://shop.dev/customerwine/calculate, welche die Weine berechnet, die zum eingeloggten Kunden passen. In Zeile 6 wird das Resultat in das HTML-Element mit der ID "customerProducts"geschrieben.

```
1  protected function execute()
2  {
3      $customerId = $this->customerSession->getCustomerId();
4      $products = $this->productCollectionFactory->create()->order(rand());
5
6      $customerAttributeValue = $this->customerAttributeCollectionFactory->create()
7          ->addFieldToFilter("customer_id", $customerId)
8          ->getFirstItem()
9          ->getData("value");
10
11     $productsToShow = array();
```

```
12
13      foreach ($products as $product) {
14          $productAttributeValue = $product->getData("attribute_value");
15
16          $neuralNetworkValue = min($customerAttributeValue, $productAttributeValue)
17              / max($customerAttributeValue, $productAttributeValue);
18
19          $n = new NeuralNetwork(array(3, 4, 1));
20          $n->load("my_network.ini");
21          $result = $n->calculate($neuralNetworkValue);
22
23          if($result) {
24              $productsToShow[] = $product;
25          }
26      }
27
28      $output = "";
29
30      foreach ($productsToShow as $product) {
31          $output .= "<li>
32                  <img src='" . $product->getImage() . "' />
33          <a href='" . $product->getUrl() . "'>" . $product->getName() . "</a>
34          </li>";
35      }
36
37      echo $output;
38  }
```

Listing 3.13: Magento 2 Controller - Calculate.php

In Zeile 6 bis 10 wird der Attributwert des Kunden geladen. Danach wird über alle Produkte iteriert, die zuvor in Zeile 4 in zufälliger Reihenfolge geladen wurden. In den Zeilen 20 bis 22 berechnet das neuronale Netz, ob der Wein geschmacklich zum Kunden passt. In den Zeilen 31 bis 36 wird ein Produktlisting erstellt, was durch anschließend im Frontend angezeigt wird.

Abbildung 3.1: Anzeige der berechneten Weine im Frontend

4 Zusammenfassung und Ausblick

In dieser Bachelorarbeit wurde erfolgreich ein Algorithmus geschrieben, der anhand von Weinattributen und den Einkäufen eines Kunden berechnet, welche anderen Weine dem Geschmacksmuster des Kunden entsprechen. Allerdings ist es notwendig, die ganzen Attribute, die ein Wein haben kann sorgfältig zu pflege, da der Algorithmus genau auf diesen Attributen aufbaut. Um dem Weinhändler das pflegen dieser Attribute zu erleichtern, könnte man ein Tool schreiben, welches er bei der Verkostung eines Weines auf seinem Tablet hat, und direkt die Attribute beim Verkosten belegen kann.

Der Algorithmus könnte sogar noch weiterentwickelt werden, indem man dem Kunden ein Eingabeformular erstellt, indem er einige Weinattribute eingeben kann und anhand dieser Informationen dann Vorschläge zum kauf erhält. Außerdem kann der Algorithmus modifiziert werden, um ihn in anderen Gebieten, z. B. Whisky, Bier, einzusetzen.

Literaturverzeichnis

[1] ? Unterschiede von magento 1 und magento 2. http://www.ecommerce-werkstatt.de/magazin/unterschiede-magento-1-magento-2/, 10.08.2016.

[2] Daniel Nitz. Was ist eigentlich eav? https://de.slideshare.net/danielnitz/was-ist-eigentlich-eav, 12.01.2010.

[3] Jon Penland. Css - cascading style sheets. http://html.com/css/, Abgerufen: 21.07.2017.

[4] php. Was ist php? php.net/manual/intro-whatis.php, Abgerufen: 30.07.2017.

[5] Alan Storm. Magento for developers: Part 7—advanced orm: Entity attribute value. http://devdocs.magento.com/guides/m1x/magefordev/mage-for-dev-7.html, Abgerufen: 20.08.2017.

[6] ChristianLuxem nodexo fscholz siggi-heltau FabianBeiner spiegelp twarncke, yampus. Javascript datentypen und datenstrukturen. https://developer.mozilla.org/de/docs/Web/JavaScript/Datenstrukturen, 24.04.2017.

[7] W3Schools. The css box model. https://www.w3schools.com/css/css_boxmodel.asp, Abgerufen: 21.07.2017.

[8] W3Schools. Css selector reference. https://www.w3schools.com/cssref/css_selectors.asp, Abgerufen: 21.07.2017.

[9] W3Schools. Css syntax and selectors. https://www.w3schools.com/css/css_syntax.asp, Abgerufen: 21.07.2017.

[10] Wikipedia. Cascading style sheets. https://de.wikipedia.org/wiki/Cascading_Style_Sheets, 06.07.2017.

[11] Wikipedia. Hypertext markup language. https://de.wikipedia.org/wiki/Hypertext_Markup_Language, 24.08.2017.

[12] Wikipedia. Javascript. https://de.wikipedia.org/wiki/JavaScript, 17.09.2017.

[13] Wikipedia. Künstliches neuronales netz. https://de.wikipedia.org/wiki/Künstliches_neuronales_Netz, 01.09.2017.

[14] Wikipedia. Magento. https://de.wikipedia.org/wiki/Magento, 15.08.2017.

[15] Wikipedia. Php. https://de.wikipedia.org/wiki/PHP, 19.09.2017.

[16] Online Marketing Wissen. Was ist html? http://www.omkt.de/html/, Abgerufen: 18.07.2017.

[17] Xovi. Magento. https://www.xovi.de/wiki/Magento, Abgerufen: 20.08.2017.

.1 Anhang

1. WSET Level 2 Systematisches Verkosten von Wein (Quelle: https://www.wsetglobal.com/media/1658/l2-wines-sat-2014-ger.pdf)

WSET Level 2 Systematisches Verkosten von Wein®

OPTISCHER EINDRUCK		
Klarheit		klar – trüb
Farbtiefe		blass – mittel – tief
Farbton	*weiß*	zitronengelb – goldgelb – bernsteingelb
	rosé	hellrosa – lachsfarben – orange
	rot	purpurrot – rubinrot – granatrot – braunrot

GERUCH	
Reintönigkeit	sauber – unsauber
Intensität	verhalten – mittel – ausgeprägt
Aromaausprägung	*z. B.* Früchte, Blumen, Gewürze, pflanzlich, Eichennoten, andere

GESCHMACK	
Süße	trocken – halbtrocken – mittel – süß
Säure	niedrig – mittel – hoch
Tannin	niedrig – mittel – hoch
Körper	schlank – mittel – voll
Geschmack-sausprägung	*z. B.* Früchte, Blumen, Gewürze, pflanzlich, Eichennoten, andere
Abgang	kurz – mittel – lang

GESAMTEINDRUCK	
Qualität	fehlerhaft – schwach – durchschnittlich – gut – sehr gut – hervorragend

WSET Level 2 Wein-Lexikon:

Anhang zum WSET Level 2 Systematischen Verkosten von Wein®

AROMA- UND GESCHMACKSAUSPRÄGUNGEN	
FLORAL / FRUCHTIG: Sind die Aromen/Geschmacksnoten einfach/allgemein oder spezifisch? Frisch oder gekocht? Reif oder unreif?	
Floral	Blüten, Rose, Veilchen
Grüne Früchte	grüner Apfel, roter Apfel, Stachelbeere, Birne, Traube
Zitrusfrüchte	Grapefruit, Zitrone, Limette (Saft oder Schale?)
Steinobst	Pfirsich, Aprikose, Nektarine
Tropische Früchte	Banane, Litschi, Mango, Melone, Passionsfrucht, Ananas
Rote Früchte	Rote Johannisbeere, Preiselbeere, Himbeere, Erdbeere, rote Kirsche, Pflaume
Schwarze Früchte	Schwarze Johannisbeere, Brombeere, Heidelbeere, schwarze Kirsche
Trockenfrüchte	Feige, Backpflaume, Rosine, Sultanine, Kirschwasser, Konfitüre, gekocht, gebacken, Fruchtkompott, Fruchtkonserve

GEWÜRZE / PFLANZLICH	
Unreif	grüne Paprikaschote, Gras, weißer Pfeffer, grüne Blätter, Tomate, Kartoffel
Krautig	Gras, Spargel, Schwarze Johannisbeerblätter
Kräuterwürzig	Eukalyptus, Minze, medizinisch, Lavendel, Fenchel, Dill
Gemüse	Kohl, Erbsen, Bohnen, schwarze Oliven, grüne Oliven
Süße Gewürze	Zimt, Gewürznelke, Ingwer, Muskatnuss, Vanille
Scharfe Gewürze	schwarzer/weißer Pfeffer, Lakritze, Wacholder

EICHE / ANDERE	
Einfach / neutral	einfach, neutral, unbestimmt
Hefe-Autolyse	Hefe, Biskuit, Brot, Toast, Gebäck, Hefesatz
Milchprodukte	Butter, Käse, Sahne, Joghurt
Eiche	Vanille, Toast, Zedernholz, Holzkohle, Rauch, Harz
Kerne	Mandel, Kokosnuss, Haselnuss, Walnuss, Schokolade, Kaffee
Animalisch	Leder, fleischig, Bauernhof
Reife	vegetabil, Pilze, Heu, feuchtes Laub, Waldboden, Wildbret, deftig, Tabak, Zedernholz, Honig, Getreide
Mineralisch	Erde, Benzin/Petrolnote, Gummi, Teer, Stein/Stahl, nasse Wolle