LAB University of Applied Sciences

Personoitujen tuote-ehdotusten tekeminen verkkokaupassa

LAB-ammattikorkeakoulu
Insinööri (AMK)
2022
Sami Aartolahti

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Aartolahti, Sami	Opinnäytetyö, AMK	2022
	Sivumäärä	
	52	

Työn nimi

Personoitujen tuote-ehdotusten tekeminen verkkokaupassa

Tutkinto ja koulutusala

Insinööri (AMK), tieto- ja viestintätekniikan koulutus

Toimeksiantajan nimi, titteli ja organisaatio (jos opinnäytetyöllä on toimeksiantaja)

Antti Immonen, Teknologiajohtaja, Tammi Digital Oy

Tiivistelmä

Opinnäytetyön toimeksianto saatiin Tammi Digital Oy:ltä. Työssä tehtiin tuote-ehdotusjärjestelmä WooCommerce-verkkokauppoihin WordPress-lisäosana, jolla haluttiin helpottaa kiinnostavien tuotteiden löytämistä verkkokaupoista. Tavoitteena oli tehdä järjestelmästä helposti ja nopeasti käyttöönotettava huolehtien samalla riittävästä suorituskyvystä. Opinnäytetyö keskittyy tuote-ehdotusten tekemiseen ja siinä käytettyihin menetelmiin, joten WordPressin toimintaa ei käydä tarkemmin läpi.

Lopullinen tuote-ehdotusjärjestelmä käyttää tuote-ehdotusten tekemiseen koneoppimismallia, joka voidaan valita eri vaihtoehdoista. Työssä kuitenkin ehdotettiin mallia, joka luo ehdotukset yhteistoiminnallisin menetelmin Cosine Similarity -menetelmän (CS) ja K-d Tree -algoritmin (KDT) avulla. Ensin algoritmille syötettävä data kuitenkin tiivistettiin Reduced Singular Value Decomposition -menetelmällä (RSVD) järjestelmän suorituskyvyn sekä koneoppimismallin yleistettävyyden parantamiseksi. Järjestelmän ja ehdotetun koneoppimismallin suorituskyky ja tuote-ehdotusten mielekkyys testattiin.

Testitulosten perusteella tuote-ehdotusjärjestelmä on hyvä pohja jatkokehitykselle, sillä kaikki suunnitellut ominaisuudet, kuten koneoppimismallin valinta ja algoritmien hienosäädöt, saatiin toimimaan. Uusien koneoppimismallien lisääminen järjestelmään tehtiin myös helpoksi. Järjestelmän toiminnassa on kuitenkin joitakin kehityskohteita, jotka on syytä ratkaista ennen sen käyttöönottoa. Esimerkiksi tuote-ehdotusten tekemisestä olisi järkevää tehdä asynkroninen prosessi käyttökokemuksen parantamiseksi.

Asiasanat

tuote-ehdotusjärjestelmä, yhteistoiminnallinen menetelmä, tuotekeskeinen tekniikka, malliperusteinen tekniikka, ulottuvuuksien vähentäminen, koneoppimismalli

Abstract

Author(s)	Type of Publication	Published
Aartolahti, Sami	Thesis, UAS	2022
	Number of Pages	
	52	

Title of Publication

Creating personalized product recommendations in e-commerce site

Degree and field of study

Bachelor of Engineering (UAS), Information and communication technology

Name, title and organisation of the client (if the thesis work is commissioned by another party)

Antti Immonen, Chief Technology Officer, Tammi Digital Oy

Abstract

Thesis was commissioned by Tammi Digital Oy. Product recommendation system for WooCommerce sites was created as a WordPress plugin, which was supposed to make finding interesting products easier from e-commerce sites. The objective of the project was to create a system that is easy and fast to add into any WooCommerce site. The performance of the system also needed to be sufficient. The thesis focuses on the process of creating recommendations and the methods that were used in the process. WordPress is not discussed in detail.

The final recommendation system uses a machine learning model to make predictions and the model can be chosen from a few options. A model was proposed which uses a collaborative filtering method, Cosine Similarity (CS) method, and K-d Tree (KDT) algorithm. First, the input data was reduced using the Reduced Singular Value Decomposition (RSVD) method to improve the system's performance and generalize the machine learning model. The performance of the product recommendation system and the proposed model was tested. The recommendations were also assessed.

The test results show that the product recommendation system is a good base for further development because it has all the basic functionality that was originally planned. For example, the possibilities to choose the machine learning model and fine tune the algorithms were successfully created. Also, adding new machine learning models to the system was made relatively easy. The recommendation system still has a few areas that could be improved. Those improvements should be done before starting to use the system in real e-commerce sites. For example, it would be a good idea to make product recommendations asynchronously. This could improve the user experience.

Keywords

product recommendation system, collaborative filtering, item-based filtering, model-based technique, dimensionality reduction, machine learning model

Sisällys

1 Johdanto

Verkkokauppojen tuotevalikoimat ovat kasvaneet vuosien saatossa huomattaviin mittasuhteisiin. Tämän vuoksi asiakkaat saattavat joutua käyttämään tuotteiden manuaaliseen selaamiseen merkittävästikin aikaa, ennen kuin he löytävät kiinnostavia tuotteita. Mikäli asiakkaat joutuvat näkemään liikaa vaivaa, he siirtyvät todennäköisesti toiseen kauppapaikkaan, jossa käyttökokemus on sujuvampi. Tällöin verkkokaupan tuottama voitto jää potentiaalia pienemmäksi, joten tuotteiden etsimistä on järkevää helpottaa.

Tuote-ehdotusjärjestelmillä on edellä kuvatun ongelman ratkaisemisessa suuri rooli. Tuote-ehdotuksia voidaan tehdä esimerkiksi analysoimalla käyttäjän vuorovaikutushistoriaa eri tuotteiden kanssa, jonka perusteella luodaan lista käyttäjää todennäköisesti eniten kiinnostavista tuotteista. Tämän listan sisältämiä tuotteita voidaan sitten ehdottaa käyttäjälle. (Aljunid & Huchaiah 2021, 480.) Tuote-ehdotusjärjestelmät voivat näin auttaa käyttäjiä löytämään heitä kiinnostavia tuotteita ja lisäämään siten verkkokaupan myyntiä tehokkaasti (Zhang ym. 2015, 257).

Työn toimeksianto saatiin Tammi Digital Oy:ltä, jonka toimialaa ovat erityisesti WordPress-pohjaiset verkkosivut ja -kaupat. Työn tavoitteena on luoda edellä kuvatun kaltainen tuote-ehdotusjärjestelmä, jota voidaan käyttää missä tahansa WooCommerce-verkkokaupassa. Tämä edellyttää järjestelmältä mukautumiskykyä eri verkkokauppojen tarpeisiin. Lisäksi järjestelmän käyttöönoton tulee olla mahdollisimman helppoa ja nopeaa, ja sen suorituskykyyn on kiinnitettävä huomiota. Tuote-ehdotusjärjestelmää myös testataan WordPress-testiympäristössä. Näin selvitetään, ovatko tuote-ehdotukset tarkoituksenmukaisia, ja kuinka paljon järjestelmä vaikuttaa WordPress-sivuston suorituskykyyn.

Työ keskittyy tuote-ehdotusten tekemiseen ja siinä käytettyihin menetelmiin. Se voidaan jakaa teoriaosuuteen ja käytäntöön. Teoriaosuudessa selvitetään alan artikkeleiden ja tutkimusaineistojen avulla, miten tuote-ehdotusjärjestelmään valitut menetelmät toimivat. Lisäksi esitellään järjestelmän tekemiseen käytetyt teknologiat. Käytännön osuudessa kuvataan toteutetun tuote-ehdotusjärjestelmän rakenne ja olennaisimmat luokat sekä osoitetaan järjestelmän toimivuus. Lopuksi työssä pohditaan, miten järjestelmää voitaisiin kehittää eteenpäin.

2 Tuote-ehdotusten tekeminen

2.1 Menetelmän valitseminen

Tuote-ehdotusten tekemiseen käytetään yleensä sisältöperusteista tai yhteistoiminnallista suosittelumenetelmää (Li ym. 2013, 177). Yhteistoiminnallisen menetelmän etuna sisältöperusteiseen menetelmään verrattuna on sen kyky toimia luotettavasti tilanteissa, joissa tietoa eri tuotteiden tarkoista ominaisuuksista ei ole saatavilla (Ren ym. 2010, 176).

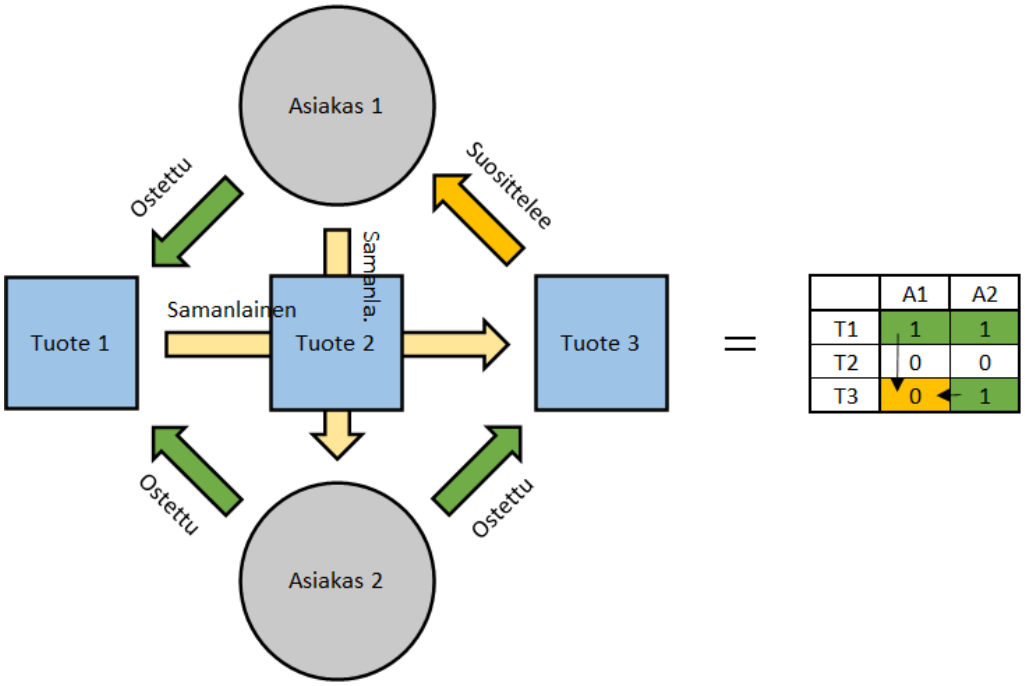
Tämä voi olla hyödyllistä esimerkiksi elokuvia myyvässä verkkokaupassa, koska ennen elokuvan katsomista siitä voidaan tietää lähinnä tyylilaji, ohjaaja ja näyttelijät. Tarkemmat ominaisuudet, kuten hidas tempo ja haikea tunnelma, voivat kuitenkin olla merkityksellisempiä tietoja elokuvan kiinnostavuuden arvioinnissa. Yhteistoiminnallinen menetelmä toimii siis luotettavammin yleiskäyttöisen tuote-ehdotusjärjestelmän pohjana.

2.2 Yhteistoiminnallinen menetelmä

Yhteistoiminnallisessa suosittelumenetelmässä luodaan yleensä matriisi käyttäjien ja tuotteiden välisistä vuorovaikutustapahtumista (Ren ym. 2010, 176). Tapahtumat ilmaistaan numeerisina arvoina, ja kyseiset tapahtumat voivat olla suoria, kuten käyttäjän antaessa numeerisen arvion tuotteesta, tai epäsuoria, esimerkiksi käyttäjän ostaessa tätä kiinnostavan tuotteen. Tämän vuorovaikutusmatriisin avulla selvitetään, mitkä käyttäjät tai tuotteet muistuttavat eniten toisiaan. (Google 2021.)

Mikäli vertaillaan käyttäjiä, on kyse käyttäjäkeskeisestä tekniikasta. Jos taas vertaillaan tuotteita, kyse on tuotekeskeisestä tekniikasta. Tuotekeskeinen tekniikka tuottaa käyttäjäkeskeiseen tekniikkaan nähden tarkempia tuote-ehdotuksia ja sitä voidaan pitää paremmin skaalautuvana. (Ren ym. 2010, 176.) Tyypillisesti tuotteista pyritään lopulta löytämään koneoppimisen avulla ne, joilla on vahvin yhteys viitekäyttäjään tai -tuotteeseen (Li ym. 2013, 177).

Yhteistoiminnallisen menetelmän toimintalogiikka on havainnollistettu kuviossa 1. Siitä nähdään, että eri asiakkaat voidaan tulkita keskenään samanlaisiksi, jos ne ovat ostaneet samoja tuotteita. Asiakkaille voidaan sitten ehdottaa tuotteita, joita muut samankaltaiset asiakkaat ovat ostaneet. Tällöin kyse on käyttäjäkeskeisestä tekniikasta. Tuotekeskeisessä tekniikassa taas selvitetään ensin, mitkä asiakkaat ovat ostaneet kunkin tuotteen. Asiakkaan sitten osoittaessa kiinnostusta jotakin tuotetta kohtaan, tälle voidaan ehdottaa tuotteita, joiden ostohistoria muistuttaa kiinnostuksen kohteena olevaa tuotetta.



Kuvio 1. Yhteistoiminnallinen menetelmä

Yhteistoiminnallinen menetelmä jaetaan yleensä vielä malli- ja muistiperusteisiin kategorioihin (Desrossiers & Karypis 2011, Zhangin ym. 2015, 257 mukaan). Ensin mainitussa kategoriassa koneoppimismalli opetetaan saatavilla olevalla datalla, jonka jälkeen opetettua mallia voidaan käyttää tuote-ehdotusten tekemiseen kerta toisensa jälkeen (Zhang ym. 2015, 257). Muistiperusteisessa ratkaisussa sen sijaan haetaan tietokannasta ajantasainen data aina ennen uusien tuote-ehdotusten tekemistä (Bobadilla ym. 2013, 113). Tästä voidaan päätellä, että malliperusteisella ratkaisulla saavutetaan parempi suorituskyky, kun tuote-ehdotuksia haetaan suuresta tietokannasta.

Jotta tuote-ehdotusjärjestelmä taas voisi ehdottaa käyttäjälle mahdollisimman kiinnostavia tuotteita, järjestelmän olisi hyvä löytää keino mukauttaa tuote-ehdotuksia käyttäjän tarpeisiin. Osa käyttäjistä kuitenkin asioi verkkokaupoissa sisään kirjautumatta, joten käyttäjästä ei aina tiedetä etukäteen mitään. Tuote-ehdotuksia voidaan kuitenkin mukauttaa tuotekeskeisellä tekniikalla meneillään olevan istunnon tietojen pohjalta, esimerkiksi ehdottamalla käyttäjälle samankaltaisia tuotteita viimeksi katseltuun tuotteeseen nähden (Hidasi ym. 2016, 2). Näin tuote-ehdotuksia voitaisiin muuttaa meneillään olevan istunnon sisällä, riippuen tuotteista, joita käyttäjä on katsellut. Tämä tekee tuote-ehdotuksista ajankohtaisia myös kirjautumattomille käyttäjille.

2.2.1 Tuotekeskeinen tekniikka

Tuotekeskeisessä tekniikassa vuorovaikutusmatriisi rakennetaan siten, että jokainen vaakarivi kuvaa tiettyä tuotetta, ja jokainen pystyrivi tietyn käyttäjän mieltymyksiä. Vaakarivejä eli tuotteita vertaillaan keskenään, ja käyttäjälle ehdotetaan tuotteita, jotka ovat mahdollisimman samankaltaisia jonkin viitetuotteen kanssa. (Ren ym. 2010, 176.) Viitetuotteena voidaan käyttää edellisessä luvussa mainittua viimeksi katseltua tuotetta.

Esimerkki tuote-käyttäjä-matriisista nähdään kuviossa 2. Siinä numero 0 merkitsee, ettei tuotteen ja käyttäjän välillä ole vuorovaikutushistoriaa. Numero 1 taas ilmaisee käyttäjän olevan kiinnostunut tuotteesta eli hän on voinut esimerkiksi ostaa sen. Vihreä ruutu tarkoittaa, että myös viitetuotteella on numero 1 kyseisellä pystyrivillä. Keltainen taas tarkoittaa, että viitetuotteella ei ole samassa kohdassa numeroa 1. Matriisin oikealle puolelle on summattu jokaisen rivin vihreiden ja keltaisten ruutujen lukumäärä. Vihreiden ruutujen suuri määrä merkitsee, että tuote on hyvin samanlainen kuin viitetuote. Ensimmäisen ja viimeisen rivin tuotteet ovat siis lähimpänä viitetuotetta, joten niitä ehdotettaisiin asiakkaalle ensin.

		Käyttäjä (ID)											
		1	2	3	4	5	6	7	8	9	10		
Tuote (ID)	1	0	0	1	0	0	1	1	0	0	1	4	0
	2	0	0	0	0	0	1	0	0	0	0	1	0
	3	1	0	0	0	0	0	0	0	1	0	0	2
	4	0	0	0	1	0	0	0	0	1	0	0	2
	5	0	0	1	0	0	1	1	0	0	1	Viitetuote	
	6	0	0	1	0	0	0	0	1	0	0	1	1
	7	1	1	0	0	0	0	0	0	0	1	1	2
	8	0	1	0	1	1	0	0	0	0	0	0	3
	9	0	0	0	0	1	0	0	0	1	0	0	2
	10	1	0	1	0	0	1	1	1	0	1	4	2

Kuvio 2. Tuote-käyttäjä-matriisi

2.2.2 Menetelmän tunnetut ongelmat

Vuorovaikutusmatriisit koostuvat usein suurimmaksi osaksi nollista, koska verkkokaupoissa voi olla paljon tuotteita, mutta yksittäisillä käyttäjillä on yleensä vuorovaikutushistoriaa vain muutaman tuotteen kanssa, jos yhdenkään. Tällaista matriisia kutsutaan harvaksi matriisiksi (sparse matrix), jonka nollasta poikkeavien arvojen vähäinen määrä heikentää tuote-

ehdotusten laatua. (Li ym. 2021, 1.) Matriisin harvuuteen liittyviä ongelmia voidaan ratkaista vähentämällä sen ulottuvuuksien määrää (Li ym. 2013, 176). Tämä voidaan tehdä esimerkiksi Singular Value Decomposition -menetelmällä (SVD) (Bobadilla ym. 2013, 113).

Myös uusien verkkokauppaan lisättyjen tuotteiden ehdottaminen voi olla haastavaa. Uudet tuotteet nimittäin jäävät huomaamatta käyttäjiltä, koska niitä ei voida ehdottaa kenellekään, ennen kuin niillä on vuorovaikutushistoriaa käyttäjien kanssa. Toisaalta vuorovaikutushistoriaa ei tule, jos kukaan ei löydä tuotetta. Kyseinen ongelma ratkaistaan usein muutamien aktiivisten käyttäjien avulla, joiden velvollisuutena on käydä arvostelemassa uudet tuotteet. (Bobadilla ym. 2013, 113.) Uusille tuotteille täytyy siis saada vuorovaikutusdataa jotenkin.

Kolmas yhteistoiminnalliseen menetelmään liittyvä haaste on siinä perinteisesti käytettyjen Nearest Neighbors -algoritmien (NN) suorituskyvyn tasainen lasku tuotteiden ja käyttäjien määrän kasvaessa verkkokaupassa. Tämäkin ongelma voidaan ratkaista datan ulottuvuuksien vähentämisellä. (Sarwar ym. 2000, 7; Li ym. 2013, 176.) Wangin ym. (2017, 156) mukaan myös likimääräisillä Approximate Nearest Neighbors -algoritmeilla (ANN) voidaan merkittävästi parantaa suorituskykyä hitaampiin NN-algoritmeihin nähden.

2.3 Malliperusteinen tekniikka

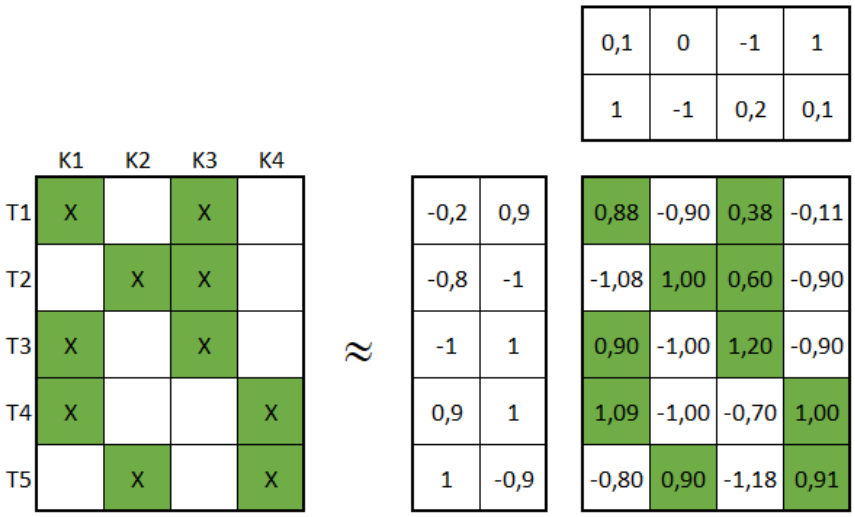
Ratkaisu lasketaan malliperusteiseksi, jos se vanhenee, kun uutta dataa lisätään vuorovaikutusmatriisiin (Bobadilla ym. 2013, 113). Yhteistoiminnallisessa malliperusteisessa ratkaisussa voitaisiin esimerkiksi luoda koneoppimismalli, joka opetettaisiin laskemaan arvosana kullekin tuote-käyttäjä-parille silloisen vuorovaikutushistorian pohjalta. Tällaisen mallin avulla voitaisiin luoda uusi vuorovaikutusmatriisi ohjelmallisesti, joka vastaisi likimäärin alkuperäistä matriisia, jonka avulla koneoppimismalli opetettiin. (Zhang ym. 2015, 257.) Matriisin yksittäiset arvot kertoisivat, kuinka vahva yhteys tietyllä tuote-käyttäjä-parilla arvioitaisiin olevan. Toisin sanoen, kuinka todennäköisesti jokin käyttäjä pitäisi tietystä tuotteesta.

Malliperusteista tekniikkaa ei kuitenkaan ole rajoitettu edellä kuvattuun koneoppimismalliin. Koneoppimismallissa on siis mahdollista käyttää monenlaisia algoritmeja ja menetelmiä, joilla voidaan yrittää korjata yhteistoiminnallisen menetelmän ongelmia. Seuraavissa luvuissa käsitellään tuote-käyttäjä-matriisin ulottuvuuksien vähentämistä, ryhmittelyä ja tuotteiden samankaltaisuuden selvittämistä. Näiden menetelmien avulla pyritään minimoimaan aiemmin mainitut ongelmat tuote-ehdotusprosessissa. Algoritmeiksi valittiin Reduced Singular Value Decomposition (RSVD), K-d Tree (KDT) ja Cosine Similarity (CS).

2.3.1 Datan ulottuvuuksien vähentäminen

Datan ulottuvuuksien vähentämisellä on tarkoitus tuote-ehdotusjärjestelmissä nopeuttaa tuote-ehdotusten tekemistä, kun käytettävä tietokanta on suuri (Bobadilla ym. 2013, 113). Sillä voidaan myös löytää alkuperäisestä vuorovaikutusmatriisista latentteja eli piileviä suhteita, joiden avulla voidaan arvioida käyttäjien kiinnostus eri tuotteita kohtaan (Sarwar ym. 2000, 8). Lisäksi ulottuvuuksien vähentämisellä voidaan ratkaista harvaan vuorovaikutusmatriisin liittyviä ongelmia. Ulottuvuuksien vähentämismenetelmät perustuvat Matrix Factorization -menetelmään (MF). (Bobadilla ym. 2013, 113.)

MF:ssä alkuperäisestä vuorovaikutusmatriisista luodaan useampia pienempiä matriiseja, joiden pistetulo on hyvin lähellä alkuperäistä matriisia. Kuviossa 3 on havainnollistettu MF-menetelmän perusajatus. Kyseisen kuvion oikealla puolella näkyvä keskimmäinen matriisi saadaan laskemalla pistetulo ulompien matriisien välillä, ja lopputuloksena saatu matriisi vastaa likimäärin alkuperäistä vuorovaikutusmatriisia. (Google 2020.)



Kuvio 3. Matrix Factorization (mukailtu Google 2020)

Kahden yhtä pitkän vektorin välisen pistetulon laskemiseksi niiden vastaavat komponentit kerrotaan keskenään ja saadut tulot summataan (Lipschutz & Lipson 2009, 4). Tämä voidaan kuvata matemaattisella kaavalla, joka nähdään kaavassa 1. Pistetulo voidaan laskea myös matriisien välillä. Tällöin kummastakin matriisista valitaan kaksi yhtä pitkää vektoria, joiden vastaavat komponentit kerrotaan keskenään, ja saadut tulot summataan. Summa sijoitetaan uuteen matriisiin vaakariville, jolta ensimmäisestä matriisista valittiin vektori, ja

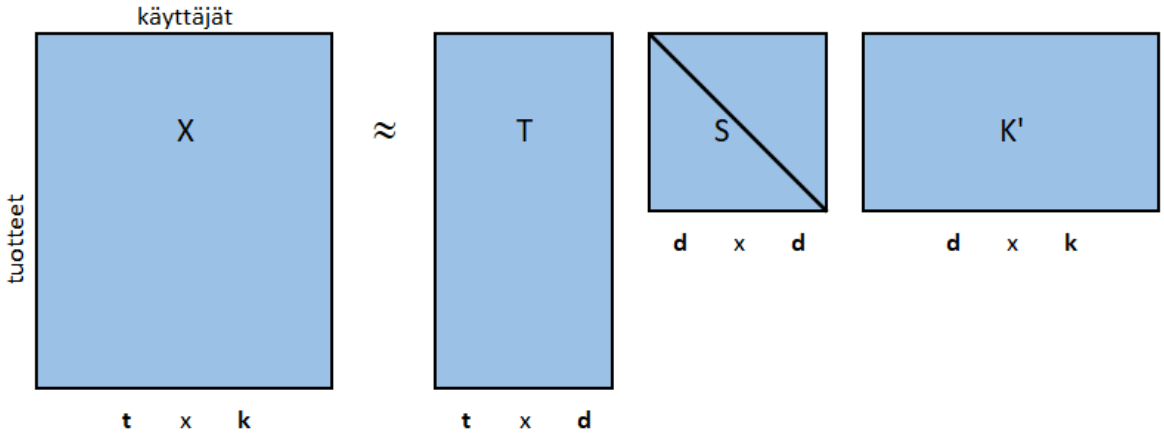
sille pystyriville, jonka järjestysnumero on sama kuin toisesta matriisista valitun vektorin järjestysnumero. Edellä kuvattu laskutoimitus tehdään kaikille matriisien vektoripareille.

$$A \cdot B = \sum_{i=1}^{n} A_i B_i = A_1 B_1 + A_2 B_2 + \cdots + A_n B_n \quad (1)$$

Deerwesterin ym. (1990, 397) mukaan RSVD-menetelmää voidaan käyttää datan ulottuvuuksien vähentämiseen. Siinä alkuperäinen vuorovaikutusmatriisi jaetaan kolmeen pienempään siten, että kahden matriisin pystyrivit ovat ortonormaaleja ja yksi on diagonaalinen singulaariarvomatriisi. Diagonaalisen matriisin kaikki arvot ovat positiivisia ja järjestetty suurimmasta pienimpään.

Jotta matriisin sisältämät pystyrivit olisivat ortonormaaleja, sen jokaisen pystyriviparin tulisi olla ortogonaalinen ja pystyrivin yksikkövektori (Lipschutz & Lipson 2009, 37). Kahden vektorin välinen ortogonaalisuus tarkoittaa, että ne ovat kohtisuorassa, joten niiden pistetulo on 0. Yksikkövektori taas on vektori, jonka pituus on 1. (Lipschutz & Lipson 2009, 4–5.) Diagonaalinen matriisi taas tarkoittaa matriisia, jossa kaikki nollasta poikkeavat arvot sijaitsevat sen lävistäjällä (Lipschutz & Lipson 2009, 35).

RSVD:n toimintalogiikka on havainnollistettu kuviossa 4. K’ on transponoitu matriisi K. Tämä tarkoittaa, että matriisin vaakarivit on muutettu pystyriveiksi ja päinvastoin (Lipschutz & Lipson 2009, 32). Matriisi S on diagonaalinen, ja matriisien T ja K pystyrivit ovat ortonormaaleja. Matriisin T ja singulaariarvomatriisin S pistetulo luo uuden matriisin TS, jonka avulla voidaan verrata tuotteita keskenään. Tämä matriisi pitää siis sisällään tuotteiden datan tiivistetyssä muodossa. Oikeanpuoleisen matriisin K ja singulaariarvomatriisin S välinen pistetulo taas saa aikaan uuden matriisin SK, jonka avulla voidaan vertailla käyttäjiä. Tämä matriisi pitää vastaavasti sisällään käyttäjien datan tiivistetyssä muodossa. Matriisien TS ja SK keskinäinen pistetulo taas luo likimäärin alkuperäistä vastaavan vuorovaikutusmatriisin. (Deerwester ym. 1990, 397–399.)

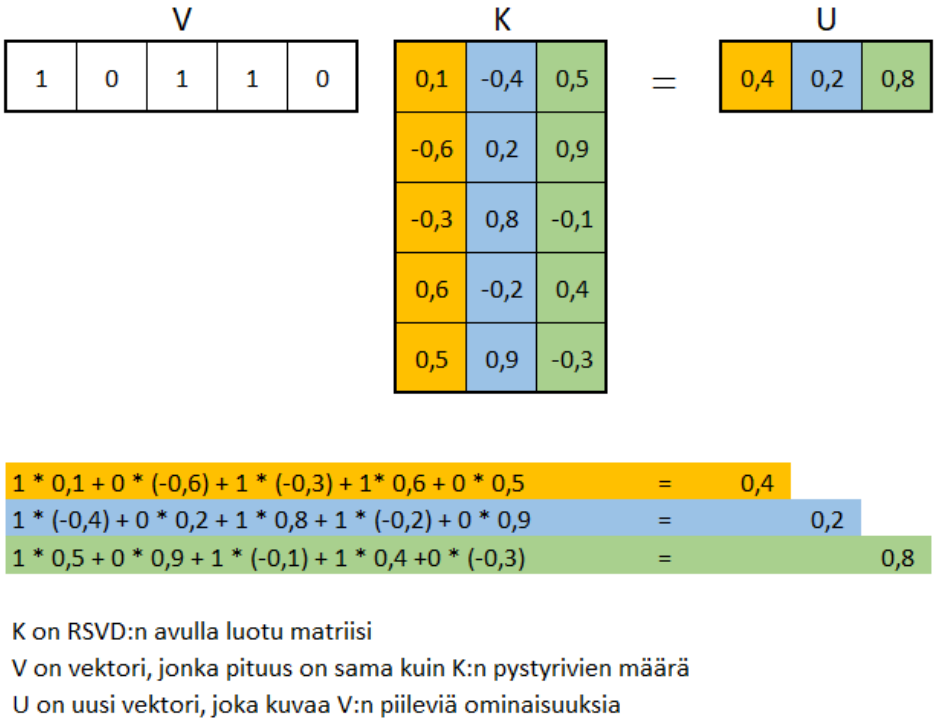


X on alkuperäinen tuote-käyttäjä-matriisi
T ja K ovat matriiseja, joiden pystyrivit ovat ortonormaaleja
K' on transponoitu matriisi K
S on diagonaalinen singulaariarvomatriisi
t on matriisin X vaakarivien määrä
k on matriisin X pystyrivien määrä
d on valittu ulottuvuuksien määrä

Kuvio 4. Reduced Singular Value Decomposition (mukailtu Deerwester ym. 1990, 401)

SVD-menetelmät ovat erittäin raskaita prosesseja (Bobadilla ym. 2013, 113). Tämän vuoksi alkuperäisen vuorovaikutusmatriisin jakaminen pienempiin matriiseihin RSVD-menetelmällä on paras toteuttaa koneoppimismallin opetuksen aikana, jotta tuote-ehdotusten tekemiseen kuluva aika saadaan minimoitua. RSVD:llä luodun matriisin K avulla voidaan vähentää minkä tahansa vektorin ulottuvuuksia, joka on verrannollinen alkuperäisen tuote-käyttäjä-matriisin vaakariveihin. Vektorin pituus on kuitenkin oltava yhtä suuri kuin matriisin K vaakarivien määrä. Tämä tehdään vektorin ja matriisin K välisellä pistetulolla.

Edellä kuvattu vektorin ja matriisin K välinen pistetulo on havainnollistettu kuviossa 5. Siinä vektorin jokainen arvo kerrotaan matriisin K pystyrivien vastaavilla arvoilla, ja kultakin pystyriviltä saadut tulot summataan. Summat asetetaan uuteen vektoriin, ja lopputuloksena saadaan alkuperäistä tiiviimpi vektori. Kokeilemalla voidaan huomata, että alkuperäisen matriisin ja matriisin K pistetulon lopputulos on sama kuin RSVD:llä luotujen matriisien T ja S välinen pistetulo. Kun alkuperäisen tuote-käyttäjä-matriisin ulottuvuudet on vähennetty, tiiviimpi matriisi voidaan tallentaa esimerkiksi tietokantaan myöhempää käyttöä varten.



Kuvio 5. Vektorin ulottuvuuksien vähentäminen pistetulon avulla

RSVD-menetelmä toimii parhaiten, kun opettamiseen käytetty matriisi ei ole äärimmäisen harva. Esimerkiksi 95,4 prosenttisesti tyhjästä matriisista saadaan huomattavasti parempia tuote-ehdotuksia kuin 99,996 prosenttisesti tyhjästä matriisista. Tiiviimmistä matriiseista voidaan löytää tuote-ehdotukset erittäin hyvin, vaikka ulottuvuuksien määrä vähennettäisiin vain 2 prosenttiin alkuperäisestä. Äärimmäisen harvoista matriiseista taas voi olla vaikea löytää luotettavia tuote-ehdotuksia sellaisilla määrillä ulottuvuuksia, joilla suorituskyky pysyisi vielä hyvänä. (Sarwar ym. 2000, 13.) Tästä voidaan päätellä, että datan ulottuvuuksien paras mahdollinen määrä riippuu käytetystä datasta ja löytyy vain kokeilemalla.

2.3.2 Tuotteiden jakaminen ryhmiin

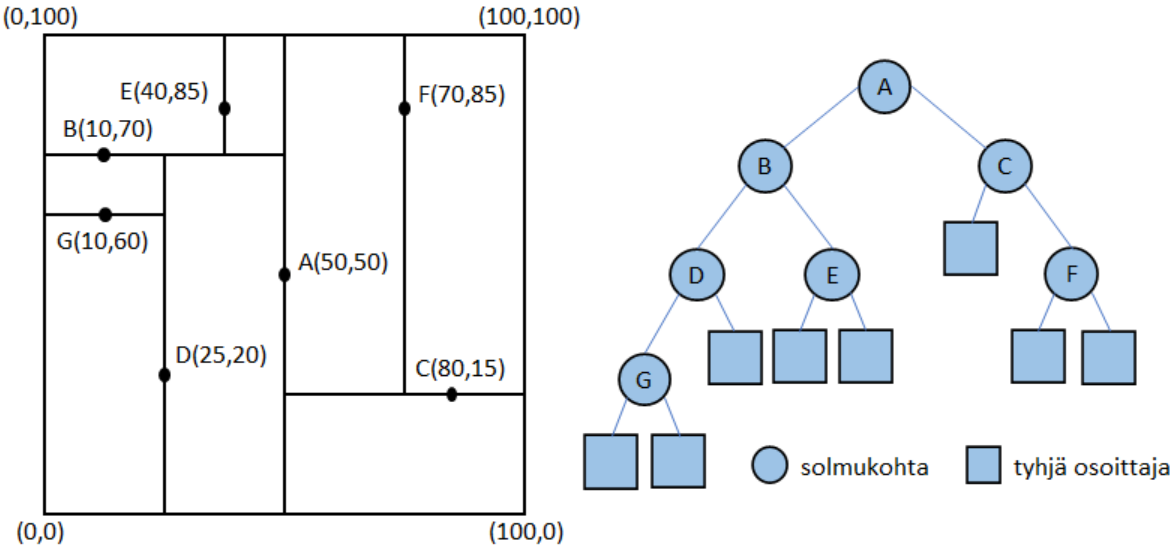
Friedmanin ym. (1977, 3–4) mukaan tuotteita voidaan myös jakaa erilaisiin ryhmiin koneoppimismallin opettamisen aikana, mikä tehdään ennen tuote-ehdotusten tekemistä. Näin varsinaisen tuote-ehdotusprosessin aikana ei tarvitse vertailla jokaista tuotetta viitetuotteeseen. Kokonaisia ryhmiä voidaan nimittäin sulkea pois prosessista, mikäli niiden todetaan olevan kauempana viitetuotteesta kuin jokin toinen ryhmä. Tämä nopeuttaa tuote-ehdotusten tekemistä huomattavasti.

Tuotteiden jakamisessa erilaisiin ryhmiin voidaan käyttää esimerkiksi KDT-algoritmia. Siinä jokainen tuote tallennetaan binääripuuhun solmukohtana, ja jokaisella solmukohdalla on

tuotteen datan ulottuvuuksien määrän verran avainarvoja. (Bentley 1975, 510.) Avainarvot kertovat solmukohdan sijainnin avaruudessa, ja ne ovat tuotetta kuvaavan vektorin sisältämät lukuarvot. Lisäksi jokaisella solmukohdalla on kaksi osoittajaa, jotka osoittavat muihin solmukohtiin tai tyhjään osoitteeseen (Bentley 1975, 510).

Solmukohdan paikka binääripuussa riippuu, miten vertailussa oleva avainarvo vertautuu muiden solmukohtien vastaaviin avainarvoihin. Ne solmukohdat, joiden vertailussa oleva avainarvo on sama tai vähemmän kuin viitesolmukohdalla, kuuluvat viitesolmukohdan vasemmalle puolelle. Jos taas avainarvo on suurempi, solmukohta kuuluu viitesolmukohdan oikealle puolelle. (Bentley 1975, 510.)

Edellä kuvattu datan jakaminen binääripuuhun KDT-algoritmin mukaisesti kaksi ulotteisessa tilassa on esitetty kuviossa 6. Siitä nähdään esimerkiksi, että solmukohta F kuuluu solmukohdan A oikealle puolelle. Tämä johtuu siitä, että A jakaa tilan Y-akselin suuntaisesti X-akselin sijainnista 50 ja F:n arvo X-akselilla on A:ta suurempi. Solmukohta F on sijoitettu myös solmukohdan C oikealle puolelle. C nimittäin jakaa A:n oikealle puolelle jäävän tilan X-akselin suuntaisesti Y-akselin sijainnista 15 ja F:n arvo Y-akselilla on C:tä suurempi.



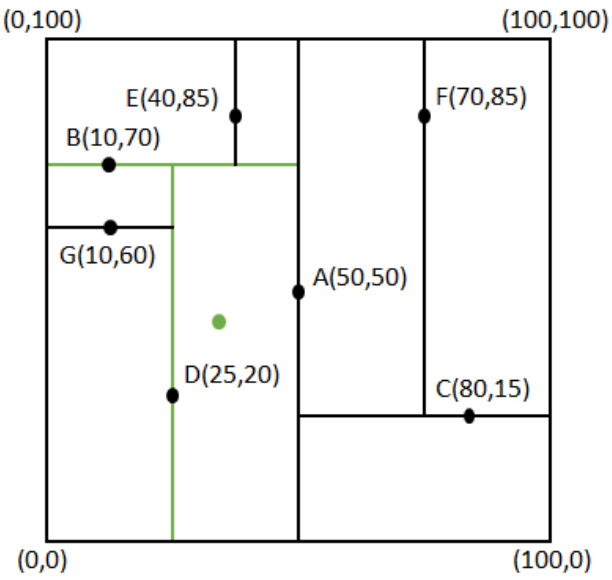
Kuvio 6. K-d Tree kaksi ulotteisessa tilassa (mukailtu Bentley 1975, 510)

2.3.3 Tuotteiden samankaltaisuuden selvittäminen

Friedmanin ym. (1977, 5) mukaan KDT-algoritmilla voidaan myös etsiä sopivia eli viitetuotteen kanssa samankaltaisia tuotteita käymällä binääripuun solmukohtia rekursiivisesti läpi ylhäältä alaspäin. Puurakenteessa edetään aina seuraavaan solmukohtaan, jonka

avainarvo on lähempänä viitetuotetta, kunnes saavutaan binääripuun oksan päättävään solmukohtaan. Tässä vaiheessa oksan kaikkien tuotteiden etäisyys eli erilaisuus viitedataan selvitetään. Näin tuotteet, jotka käytiin läpi, voidaan järjestää sopivuusjärjestykseen.

Edellä kuvatusta tuotteiden hakuprosessista nähdään esimerkki kuviossa 7, jossa vihreä piste esittää viitetuotetta. Tuotteiden etsiminen aloitettaisiin solmukohdasta A, josta mentäisiin seuraavaksi solmukohtaan B, koska viitetuotteen sijainti on A:han nähden vasemmalla. Sitten edettäisiin solmukohtaan D, joka kuuluu B:n vasemmalle osoittajalle. Tässä kohtaa todettaisiin, että viitetuote on D:stä katsottuna oikealla, mutta koska D:n oikeanpuoleinen osoittaja osoittaa tyhjää, tuotteiden etsintä päättyisi. Lopuksi solmukohtien A, B ja D vastaavien tuotteiden erilaisuus viitetuotteeseen nähden laskettaisiin.



Kuvio 7. Tuotteiden hakeminen binääripuusta (mukailtu Bentley 1975, 510)

Tuotteiden samankaltaisuus tai erilaisuus viitekäyttäjään tai -tuotteeseen lasketaan usein CS-menetelmän avulla (Sarwar ym. 2000, 6). Siinä vertailtavia käyttäjiä tai tuotteita kohdellaan vektoreina, ja niiden kulma lasketaan suhteessa viitekäyttäjä- tai -tuotevektoriin. Suuri ero kulmassa viittaa suureen vektoreiden väliseen eroon. Kulma lasketaan esimerkiksi kahden tuotevektorin pistetulon avulla, josta saatu lukema jaetaan kyseisten vektoreiden pituuksien tulolla. (Aljunid & Huchaiah 2021, 483.) CS-menetelmän matemaattinen kaava nähdään kaavassa 2.

$$\frac{A \cdot B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sum_{i=1}^{n} A_i^2 \sum_{i=1}^{n} B_i^2} \quad (2)$$

Esimerkki CS-menetelmän laskutoimituksesta on esitelty kuviossa 8. Siinä nähdään kahden vektorin tiedot taulukkona ja näiden vektoreiden välisen CS-menetelmän laskutoimitus. Saatu tulos kuitenkin vähennetään lopuksi luvusta 1. Rubix ML:n (2021a) mukaan vähennys voidaan tehdä, jotta tuloksesta ei tule negatiivista, ja näin CS-menetelmää voidaan käyttää vektoreiden välisen etäisyyden eli erilaisuuden mittana. Normaalisti CS-menetelmässä lopputulos olisi siis jotain lukujen -1 ja 1 väliltä. Tällöin -1 tarkoittaisi, että vektorit ovat päinvastaisia, ja luku 1 kuvaisi vektoreiden identtisyyttä. Kun saatu lukema sitten vähennetään luvusta 1, tuloksena on jotain lukujen 0 ja 2 väliltä. Tällöin 0 tarkoittaa, että vektoreilla on sama kulma, ja luku 2 kuvaa päinvastaisia vektoreita.

	K1	K2	K3	K4	K5
T1	0	1	1	0	1
T2	1	1	0	0	1

$$Cosine = 1 - \frac{0 \times 1 + 1 \times 1 + 1 \times 0 + 0 \times 0 + 1 \times 1}{\sqrt{0^2 + 1^2 + 1^2 + 0^2 + 1^2} \times \sqrt{1^2 + 1^2 + 0^2 + 0^2 + 1^2}}$$

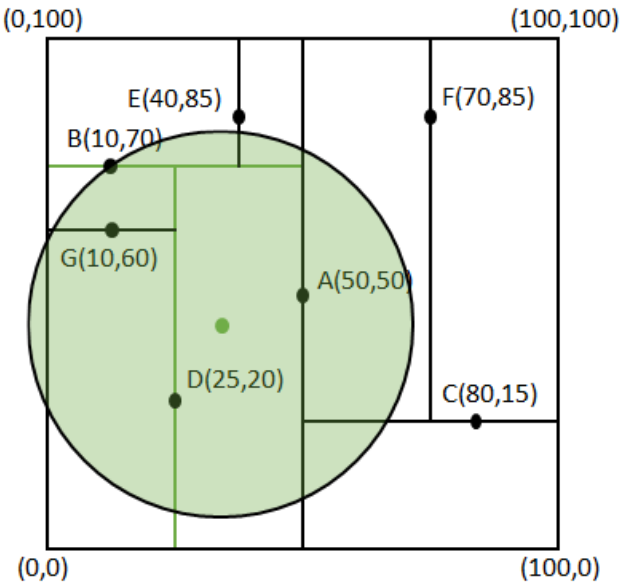
$$Cosine = 1 - \frac{2}{3} = \frac{1}{3} \approx 0{,}33$$

Kuvio 8. Cosine Similarity etäisyyden mittana (mukailtu Rubix ML 2021a)

Kun binääripuun tarkastettujen solmukohtien kaikkien tuotteiden etäisyys viitetuotteeseen on lopulta laskettu ja tuotteista on valittu k määrä, niin selvitetään, onko joissakin tarkastamatta jääneistä binääripuun osuuksista solmukohtia, joita vastaavat tuotteet olisivat lähempänä viitedataa kuin jo tarkastettujen tuotteiden k-sopivin tuote. Jos esimerkiksi viitetuotteen sijainti binääripuussa on ylimmän solmukohdan vasemmalla puolella, niin selvitetään, olisiko syytä harkita hakuprosessissa myös binääripuun oikeanpuoleisia solmukohtia. Sama pitäisi selvittää kaikkien tähän asti tarkastettujen solmukohtien oksanhaarojen osalta. Tämä tehdään ikään kuin keskittämällä viitedatan sijainnin kohdalle geometrinen pallo, jonka halkaisija on viitedatan etäisyys k-sopivimpaan tuotteeseen, ja katsotaan jääkö pallon alle sellaisia solmukohtia, joita ei ole vielä tarkastettu. Jos ei jää, hakuprosessi päättyy ja k-sopivimmat tuotteet palautetaan. (Friedman ym. 1977, 5–6.)

Mikäli tarkastamatta jätettyjä solmukohtia taas jää pallon alle, aiemmin kuvattu hakuprosessi tehdään myös niille binääripuun osuuksille, joihin pallon alle jääneet solmukohdat kuuluvat. Tällä kertaa hakuprosessi aloitettaisiin niistä solmukohdista, jotka sisältävät kyseiset tarkastamattomat binääripuun osuudet. Myös näiden jatkohakuprosessien päätteeksi lasketaan tarkastettujen solmukohtien etäisyys viitedataan, ja kaikista tarkastetuista tuotteista valitaan k-määrä. (Friedman ym. 1977, 6.)

Edellä kuvattu pallotesti on havainnollistettu kuviossa 9. Siinä esitetyssä esimerkkitilanteessa haetaan 3 läheisintä tuotetta vihreällä merkitylle viitetuotteelle. Koska ensimmäisen hakuprosessin aikana kuljetaan solmukohtien A, B ja D kautta, joista B on kauimmaisin, pallotestissä käytetyn pallon säde on sama kuin B:n etäisyys viitetuotteesta. Pallotestissä huomattaisiin, että solmukohta G jää pallon alle, joten myös solmukohdan D vasemmanpuoleinen osuus jouduttaisiin tarkistamaan. Tällöin jatkohaku aloitettaisiin solmukohdasta D. Lopulta palautettaisiin solmukohtia A, D ja G vastaavat tuotteet.



Kuvio 9. K-d Tree pallotesti (mukailtu Bentley 1975, 510; Friedman ym. 1977, 6)

3 Käytetyt teknologiat

3.1 WordPress

WordPress on verkkosivujen hallinnoimiseen ja julkaisemiseen kehitetty avoimen lähdekoodin järjestelmä (WordPress a). Sen käytön aloittaminen on pyritty tekemään helpoksi, ja sen ominaisuuksia voidaan muokata kunkin kehitystiimin tarpeisiin. WordPressiin voidaan esimerkiksi ladata ja luoda erilaisia lisäosia. (WordPress b.) Lisäosien luominen vaatii kuitenkin ohjelmointitaitoa. Lisäksi on huomioitava, että WordPress asennetaan web-palvelimelle.

WordPressin vähimmäisvaatimuksiin kuuluu PHP:n versio 7.4 ja MySQL:n versio 5.7. MySQL:n tilalle voidaan asentaa myös MariaDB:n versio 10.2. WordPress-sivusto jaetaan käytettäväksi esimerkiksi Apache- tai Nginx-palvelinohjelmistolla. (WordPress c.) Monet isännöintipalveluntarjoajat käyttävät WordPress-asennustyökaluja, joiden avulla WordPress asennetaan automaattisesti (WordPress d). Omalle palvelimelle se voidaan asentaa itse. WordPressin asentaminen ei ole olennaista tuote-ehdotuslisäosan tekemisen kannalta, joten sen asennusprosessia ei käydä läpi. WordPress kuitenkin tarvitaan järjestelmän testaamiseen, joten se oletetaan valmiiksi asennetuksi.

3.2 WooCommerce-lisäosa

WooCommerce on WordPress-lisäosa, jonka avulla voidaan luoda kokonainen verkkokauppasivusto maksujärjestelmineen varsin helposti (Automattic 2022). WooCommerce-lisäosan asentamisprosessissa luodaan tuote-ehdotusjärjestelmän tarvitsemia tietokantatauluja, joihin tallennetaan muun muassa käyttäjien ostohistoria. WooCommerce toimii siis tuote-ehdotuslisäosan pohjana, jonka ominaisuuksia sitten laajennetaan.

Kuten kaikki muutkin WordPress-lisäosat, WooCommerce voidaan asentaa WP-sivustolle lataamalla ja sitten aktivoimalla se järjestelmänvalvojan näkymässä. Tämä tehdään Plugins-sivulla. Onnistuneen asennuksen jälkeen järjestelmänvalvojan näkymän vasemman laidan palkkiin pitäisi ilmestyä uusia painikkeita, kuten WooCommerce-painike. Sitä klikkaamalla päästään määrittämään WooCommerce-lisäosan asetuksia. Tämä prosessi ei kuitenkaan ole olennainen tuote-ehdotusjärjestelmän tekemisen kannalta, joten WooCommerce-verkkokauppa oletetaan valmiiksi määritetyksi.

3.3 PHP-ohjelmointikieli

PHP on avoimen lähdekoodin skriptauskieli, jota voidaan esimerkiksi sisällyttää HTML-sivuille (PHP Group a; PHP Group b). Kielellä tehdäänkin pääasiassa dynaamisia verkkosivuja. (PHP Group a.) PHP-skriptit ajetaan siis palvelimella, minkä tuloksena generoidaan

HTML-teksti, joka lopulta lähetetään asiakkaan päätelaitteelle tulkittavaksi. (PHP Group b.) PHP tukee olio-ohjelmointia, ja sitä voidaan käyttää ohjelmointiin laajemminkin. PHP:ta voidaan käyttää ainakin Linuxilla, MacOS:llä ja Windowsilla. Palvelinohjelmistot tukevat PHP:tä laajalti, mukaan lukien Apache, IIS ja nginx. (PHP Group c.)

WordPress-palvelimille ei välttämättä tarvitse asentaa PHP:tä itse, mikäli käyttää isännöintipalveluntarjoajaa, joka asentaa WordPressin. Jos WordPress kuitenkin asennetaan ja konfiguroidaan itse, niin silloin PHP täytyy asentaa ennen WordPressiä. (WordPress c.) PHP tulee lisäksi asentaa paikalliseen ohjelmointiympäristöön, jotta Composer-riippuvuusmanageri voidaan asentaa (Composer a). Tästä saadaan lisätietoja seuraavassa luvussa.

3.4 Composer-riippuvuusmanageri

Composer on työkalu PHP-projektien riippuvuuksien hallintaan. Sen avulla voidaan esimerkiksi määrittää kirjastot, joita projekti tarvitsee toimiakseen. Tarvittavat kirjastot voidaan myös ladata ja päivittää Composerin avulla. (Composer a.) Kyseinen riippuvuusmanageri tulee asentaa ohjelmointiympäristöön edellä kuvattua tarkoitusta varten.

Riippuvuuksia ladataan Composerin avulla require-komennolla, jonka käyttäminen käydään läpi myöhemmissä luvuissa, joissa käsitellään projektissa tarvittavia kirjastoja. Kyseinen komento myös lisää uudet riippuvuudet composer.json-tiedostoon. Mikäli kyseistä tiedostoa ei löydetä kansiosta, jossa komento ajetaan, composer.json luodaan sinne automaattisesti komennon ajamisen aikana. Tiedosto myös päivitetään automaattisesti, kun riippuvuuksia päivitetään update-komennolla. (Composer b.)

Riippuvuudet merkitään versionumeroineen composer.json-tiedostoon ja se luodaan yleensä projektikansion juureen. Composer lataa riippuvuuksien lähdekoodit kansioon nimeltä vendor, johon luodaan autoload.php-tiedosto. Riippuvuudet saadaan projektin käyttöön lataamalla autoload.php sellaisessa PHP-tiedostossa, jossa Composerilla ladattuja kirjastoja tarvitaan. Tämä voidaan tehdä yhdellä koodirivillä (Kuva 1). (Composer c.) Kuvan esimerkki voitaisiin lisätä tiedostoon, joka sijaitsee samassa kansiossa kuin vendor-kansio.

```
require_once __DIR__ . '/vendor/autoload.php';
```

Kuva 1. Composer-kirjastojen tuominen PHP-projektiin (mukailtu Composer c)

3.5 PECL-laajennusvarasto

PECL on varasto PHP-laajennuksille, josta voidaan ladata laajennuksia esimerkiksi komentokehotteen kautta (PHP Group 2020; PHP Group f). PECL:ssä on listattu kaikki tunnetut PHP-laajennukset ja niiden latausosoitteet. PECL käyttää pakettimanagerina PEAR:ia, joka täytyy asentaa PHP-palvelimelle, jotta PECL toimii. (PHP Group 2020.) PECL asennetaan siis WordPress-palvelimelle.

Mikäli palvelin on jokin Linux-järjestelmä, PEAR asennetaan yleensä automaattisesti PHP:n asennuksen yhteydessä (PHP Group g). Tästä johtuen PECL todennäköisesti toimii ilman erillisiä toimenpiteitä WordPress-palvelimilla. Jos PEAR:ia ei kuitenkaan jostain syystä asenneta PHP:n asennuksen yhteydessä, se täytyy tehdä manuaalisesti.

PECL:stä voidaan ladata PHP-laajennuksia install-komennon avulla (PHP Group f). Lataamisen lisäksi laajennukset tulee merkitä php.ini-tiedostoon, jotta ne saadaan käyttöön (PHP Group h). Laajennusten asennuskomennon käyttäminen ja php.ini tiedoston muokkaaminen selitetään tarkemmin myöhemmässä Tensor-laajennusta käsittelevässä luvussa.

3.6 Rubix-kirjasto

Rubix on avoimen lähdekoodin ja korkean tason koneoppimiskirjasto PHP-ohjelmointikielelle. Kirjasto tarjoaa työkaluja koneoppimisprosessin kaikkiin vaiheisiin datan purkamisesta mallin harjoittamiseen ja tuotantoon siirtämiseen asti. Tarjolla on yhteensä yli 40 koneoppimisalgoritmia, jotka käyttävät ohjattua ja ohjaamatonta oppimistapaa. (Rubix ML 2021b.) Rubix asennetaan tuote-ehdotusjärjestelmän projektikansioon.

Rubix-kirjaston vähimmäisvaatimuksiin kuuluu PHP 7.4 tai uudempi versio kyseisestä ohjelmointikielestä. Rubix asennetaan Composer-pakettimanagerin avulla, joten se tulee olla asennettuna ohjelmointiympäristöön. Rubix-kirjasto asennetaan ajamalla kirjaston asennuskomento (Kuva 2) komentokehotteessa tuote-ehdotusjärjestelmän projektikansion juuressa. (Rubix ML 2021c.)

Kuva 2. Rubix-kirjaston asentaminen (mukailtu Rubix ML 2021c)

3.7 Tensor-kirjasto ja -laajennus

Tensor on PHP-kirjasto ja -lisäosa, jota käytetään tieteelliseen laskemiseen. Rubix-kirjasto käyttää sitä muun muassa datan ulottuvuuksien vähentämiseen. Tensor-laajennuksen prosessit ajetaan monisäikeisesti, ja se on jopa 230 kertaa nopeampi kuin Tensor-kirjasto, riippuen käyttötilanteesta. (GitHub 2021.) Tensor-kirjasto asennetaan tuote-ehdotusjärjestelmän projektikansioon. Tensor-laajennus taas tulee asentaa WordPress-palvelimelle.

Rubix-kirjasto suosittelee Tensor-laajennusta (Rubix ML 2021c). Tensor-kirjaston lähdekoodia tutkimalla voidaankin huomata, että datan ulottuvuuksien vähentämiseen käytettyjä algoritmeja ei ole määritelty. Tensor-laajennuksen lähdekoodissa taas on, joten laajennus täytyy asentaa, jotta kaikki Rubix-kirjaston ominaisuudet saadaan käyttöön.

PHP-laajennusten asentaminen tosin vaatii järjestelmänvalvojan oikeudet palvelimelle, mikä voi olla ongelma, jos käytössä oleva palvelin on ostettu isännöintipalveluntarjoajalta. Tämän vuoksi Tensor-kirjasto asennetaan siltä varalta, että laajennusta ei voida asentaa. Jos sekä kirjasto, että laajennus on asennettu, laajennusta suositaan (GitHub 2021).

Tensor-kirjaston ja -laajennuksen vähimmäisvaatimuksena on PHP 7.4 tai uudempi versio. Kirjasto asennetaan Composerin avulla, joten sekin tulee olla asennettuna ohjelmointiympäristöön. Tensor-kirjasto asennetaan ajamalla sen asennuskomento (Kuva 3) komentokehotteessa tuote-ehdotusjärjestelmän projektikansion juuressa. (GitHub 2021.)

```
composer require scienide/tensor
```

Kuva 3. Tensor-kirjaston asentaminen (mukailtu GitHub 2021)

Tensor-laajennus asennetaan komentokehotekomennolla (GitHub 2021). Kyseinen komento käyttää PECL:iä ja voidaan nähdä kuvan 4 vasemmassa laidassa. Lisäksi palvelimelta löytyvään php.ini-tiedostoon lisätään rivi, jolla laajennus aktivoidaan (GitHub 2021). Se lisätään otsikon Dynamic Extensions alle. Lisättävä rivi nähdään kuvan 4 oikeassa laidassa. Tiedosto php.ini sijaitsee WordPress-palvelimella PHP:n asennuskansiossa.

```
pecl install tensor   extension=tensor.so
```

Kuva 4. Tensor-laajennuksen asentaminen (mukailtu GitHub 2021)

4 Tuote-ehdotusjärjestelmän tekeminen

4.1 Järjestelmän kuvaus

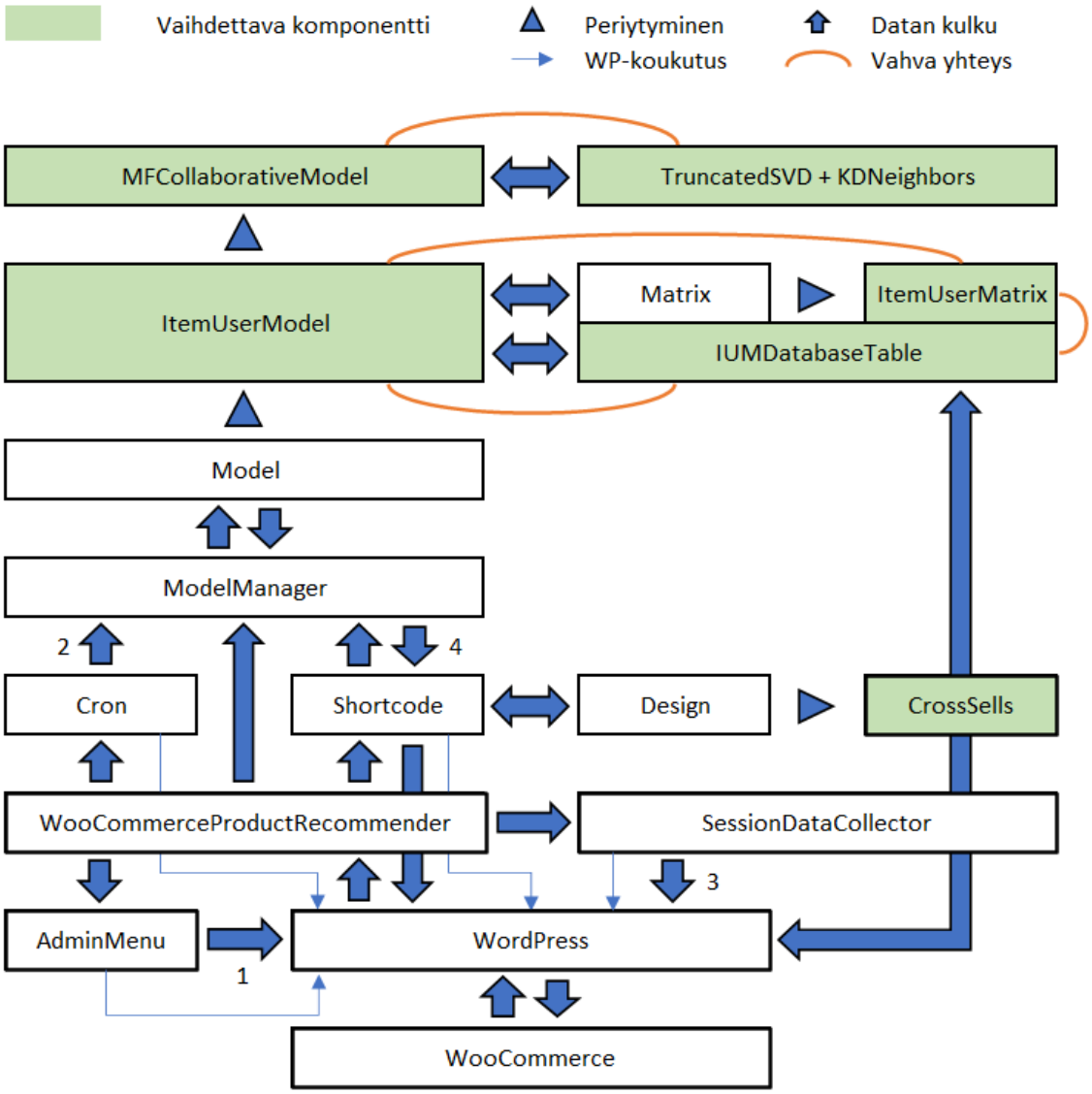
Tuote-ehdotusjärjestelmästä tehtiin WordPress-lisäosa, jota voidaan käyttää monissa eri WooCommerce-verkkokaupoissa. Koska eri verkkosivuprojekteissa on erilaisia tarpeita, järjestelmästä tehtiin mukautettava. Tätä varten WordPressiin luotiin omat asetussivut, joiden kautta tehtävät WP-asetusten muutokset vaikuttavat järjestelmän toimintaan.

Asetussivuilla voidaan esimerkiksi hienosäätää tuote-ehdotusjärjestelmän koneoppimismalleissa käytettyjä algoritmeja tai vaihtaa käytössä olevaa koneoppimismallia kokonaan. Muutetut algoritmiasetukset asetettiin tulemaan voimaan koneoppimismallin luomisen ja opettamisen yhteydessä, koska olemassa olevan mallin säätöarvoja voidaan muuttaa ainoastaan luomalla malli kokonaan uudestaan. Vanhan koneoppimismallin hävittäminen ja uuden luominen koettiin loogisimmaksi toteuttaa mallin opettamisprosessin alussa.

Myös tuote-ehdotusjärjestelmän ulkoasun luomiseen käytettävää designia ja siihen liittyviä asetuksia voidaan muuttaa. Näin järjestelmästä voidaan tehdä useita erilaisia ilmentymiä eri sivuille, ja kyseisille ilmentymille voidaan antaa tarpeen vaatiessa myös erilaisia toimintoja. Esimerkiksi tuote-ehdotukset voidaan näyttää karusellissa, jos niin halutaan. Toisaalta tuote-ehdotukset voidaan näyttää myös useammassa rivissä ilman karuselliakin.

Tuote-ehdotusjärjestelmän koneoppimismallien opettaminen ajastettiin, jotta järjestelmän käyttäminen olisi mahdollisimman helppoa ja koneoppimismallien opettamisesta ilmenisi mahdollisimman vähän haittaa käyttäjien käyttökokemusta ajatellen. Opetusprosessi voi nimittäin olla varsin raskas ja hidastaa hetkellisesti palvelinta. Helppokäyttöisyyden vuoksi koneoppimismalli kuitenkin haluttiin opettaa WordPressissä, joten opetusprosessin ajastus mahdollisimman rauhalliseen ajankohtaan koettiin ainoaksi järkeväksi vaihtoehdoksi.

Kuviossa 10 esitellään järjestelmän tärkeimmät luokat, rakenne ja toimintalogiikka. Edellisissä kappaleissa esitellyt tuote-ehdotusjärjestelmän päätoiminnot voidaan erottaa kyseisestä kuviosta. Esimerkiksi asetussivut luodaan AdminMenu-luokan avulla, Shortcode-luokka vastaa tuote-ehdotusten näkyviin asettamisesta, ja Cron-luokan avulla koneoppimismallin opettaminen ajastetaan. Kuviosta nähdään myös, että ModelManager-luokan kautta hallinnoidaan koneoppimismalleja, ja SessionDataCollector-luokka tallentaa istunnossa viimeksi katsellun tuotteen, jota käytetään viitetuotteena tuote-ehdotusprosessissa. Lisäksi kuviosta nähdään, että joidenkin luokkien välille on määritetty vahva yhteys. Tämä tarkoittaa, että kyseisiä luokkia käytetään vain toistensa kanssa.

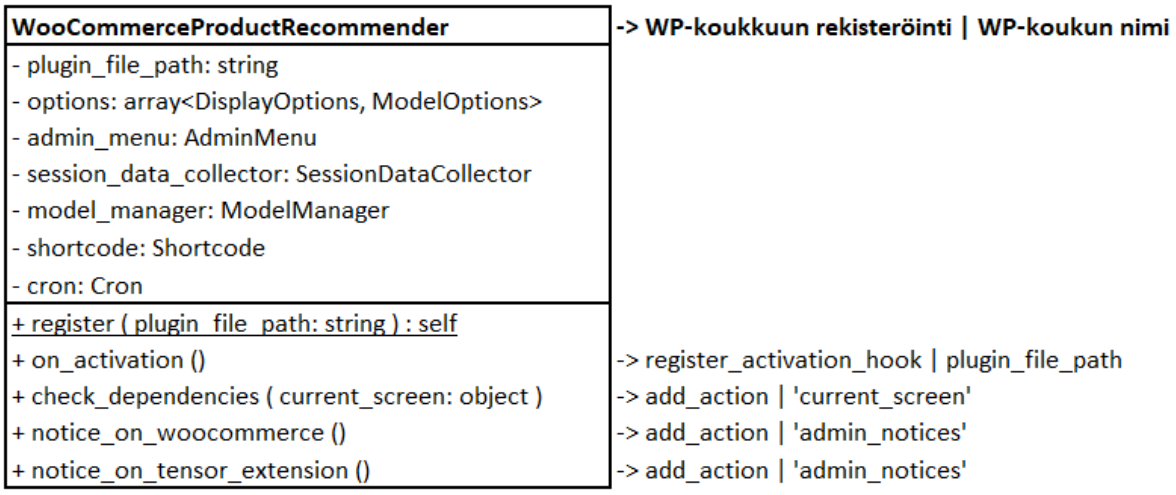


Kuvio 10. Järjestelmän rakenteen ja toiminnan kuvaus

4.1.1 Pääluokka

WooCommerceProductRecommender on tuote-ehdotusjärjestelmän pääluokka, jonka tärkeimmät muuttujat ja funktiot nähdään kuviossa 11. Siitä nähdään, että luokalla hallinnoidaan järjestelmän pääkomponentteja, joita ovat luokkien AdminMenu, SessionDataCollector, ModelManager, Shortcode ja Cron esiintymät. Lisäksi se tarkistaa, että tarvittavat laajennukset ja lisäosat on asennettu. WooCommerceProductRecommender-luokasta rekisteröidään esiintymä heti ensimmäisenä järjestelmän ajon alussa. Tämä tehdään tuote-

ehdotuslisäosan päätiedostossa nimeltä woocommerce_product_recommender.php. Tämä tiedosto sijaitsee lisäosan pääkansion juuressa.



Kuvio 11. WooCommerceProductRecommender-luokan tärkeimmät muuttujat ja funktiot

WooCommerceProductRecommender-luokalle luodaan esiintymä rekisteröintifunktiolla nimeltä register. Rekisteröintifunktion kutsu nähdään kuvassa 5. Kyseiselle funktiolle lähetetään parametrina ainoastaan tuote-ehdotuslisäosan päätiedoston polku. Kyseisen esiintymän luonnin aikana rekisteröidään esiintymät myös järjestelmän pääkomponenttiluokille sekä DisplayOptions- ja ModelOptions-luokille. Lisäksi WooCommerceProductRecommender-luokan loput funktiot rekisteröidään WP-koukkuihin. WordPress ajaa WP-koukkuihin liitetyt funktiot automaattisesti ennalta määrätyissä kohdissa (WordPress e).

```
$wcpr = WoocommerceProductRecommender::register(__FILE__);
```

Kuva 5. WooCommerceProductRecommender-luokan esiintymän luominen

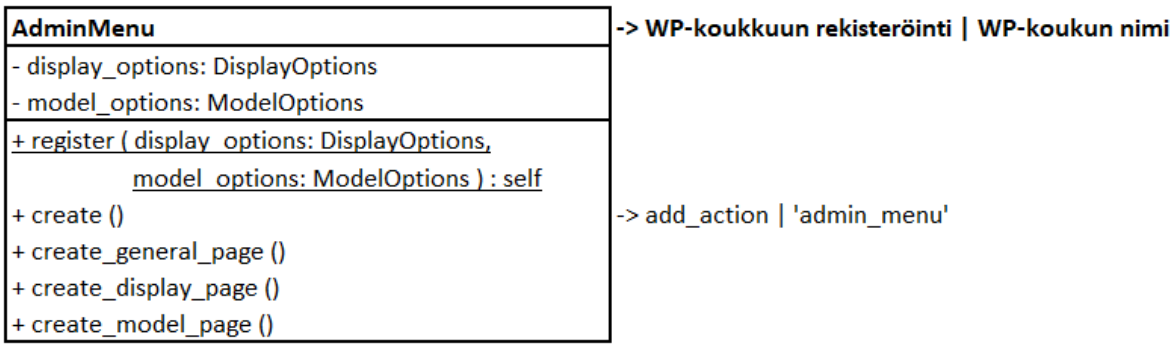
Tuote-ehdotusjärjestelmän aktivoinnin yhteydessä ajettavaksi funktioksi asetetaan on_activation-funktio. Sen aikana luodaan uusi WP-Cron-ajastus, johon liitetään ModelManager-luokan koneoppimismallien opetusfunktio. Näin koneoppimismallit saadaan opetettua automaattisesti haluttuna ajankohtana.

Kun WordPress-sivulla vieraillaan, ajetaan WooCommerceProductRecommender-luokan check_dependencies-funktio. Sillä tarkistetaan, että WooCommerce-lisäosa ja Tensor-

laajennus on kumpikin aktivoitu. Mikäli näissä havaitaan puutteita, tuote-ehdotusten mahdollinen lisääminen sivulle keskeytetään. Mikäli käyttäjä on parhaillaan järjestelmänvalvojan näkymässä, havaituista puutteista luodaan huomautukset näkymän ylälaitaan funktioilla notice_on_woocommerce ja notice_on_tensor_extension.

4.1.2 Asetussivut

AdminMenu-luokalla hallinnoidaan tuote-ehdotusjärjestelmän asetussivuja, ja sen tärkeimmät muuttujat ja funktiot nähdään kuviossa 12. Siitä nähdään, että kyseinen luokka luo yhteensä kolme asetussivua, jotka luodaan WordPress-sivuston järjestelmänvalvojan näkymään. Niiden avulla voidaan muokata järjestelmän asetuksia, joita esimerkiksi design-luokat ja koneoppimismallit käyttävät. AdminMenu-luokan esiintymä rekisteröidään tuote-ehdotusjärjestelmän pääluokan esiintymän rekisteröinnin yhteydessä.



Kuvio 12. AdminMenu-luokan tärkeimmät muuttujat ja funktiot

Rekisteröintifunktion register ajamalla luodaan esiintymä AdminMenu-luokasta. Rekisteröintifunktion kutsu nähdään kuvassa 6. Kuvasta nähdään, että funktiolle annetaan parametreina esiintymät luokista DisplayOptions ja ModelOptions, joiden avulla hallinnoidaan design-valintoihin ja koneoppimismalleihin liittyviä WP-asetuksia. Kyseiset esiintymät on luotu tuote-ehdotusjärjestelmän pääluokassa, ja ne asetetaan rekisteröidyn AdminMenu-luokan esiintymän muuttujiin. Parametreina annetut esiintymät sisältävät myös WP-asetusten päivitysfunktiot, joita kutsutaan, kun asetussivuilla painetaan tallennuspainiketta. Tämän jälkeen päivitetyt WP-asetukset voidaan hakea samojen esiintymien kautta.

```
$this->admin_menu = AdminMenu::register(
    $this->options['display'],
    $this->options['model']
);
```

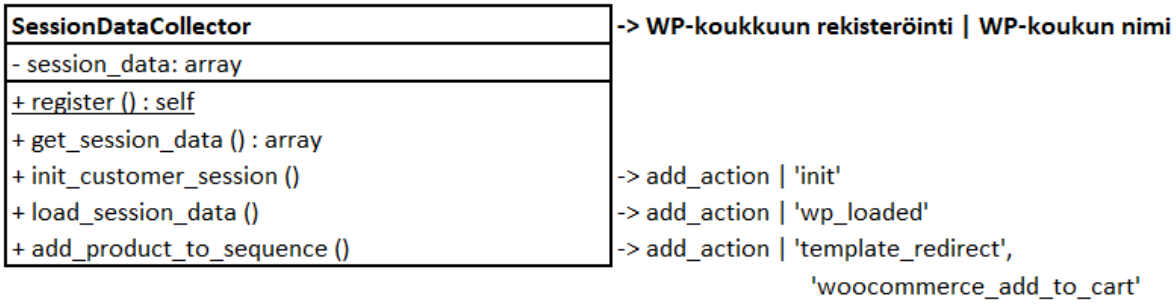
Kuva 6. AdminMenu-luokan esiintymän luominen

Rekisteröintifunktion aikana admin_menu nimiseen WP-koukkuun liitetään create-funktio, jolla luodaan tarvittavat asetussivut. Kyseisen funktion sisällä kutsutaan apufunktioita create_general_page, create_display_page ja create_model_page. Nämä apufunktiot vastaavat yksittäisten asetussivujen luomisesta, joita luodaan siis yhteensä kolme erilaista.

Yksi sivu on nimeltään General, ja se sisältää ohjeet tuote-ehdotusjärjestelmän lisäämiseksi WordPress-sivuille. Toisen sivun nimi on Display, ja sillä voidaan tarkastella ja päivittää tuote-ehdotusten ulkomuotoon liittyviä WP-asetuksia. Näitä ovat muun muassa yhdelle riville asetettavien tuote-ehdotusten määrä ja tuote-ehdotusten kokonaismäärä. Kolmas asetussivu on nimeltään Model, ja sen kautta päästään tarkastelemaan ja päivittämään koneoppimismalleihin liittyviä WP-asetuksia. Näitä ovat esimerkiksi valittu koneoppimismalli, päivitettynä pidettävät mallit ja eri mallien algoritmeihin liittyvät hienosäädöt.

4.1.3 Istunnon tapahtumien tallentaminen

SessionDataCollector-luokalla hallinnoidaan käyttäjän istunnosta kerättäviä tietoja, ja sen tärkeimmät muuttujat ja funktiot nähdään kuviossa 13. Kerättäviä tietoja tarvitaan tuote-ehdotusten tekemiseen, ja kuviosta nähdään, että kerätyt tiedot tallennetaan taulukkoon. Näitä tietoja ovat käyttäjän viimeisimpien tuotteisiin liittyvien tapahtumien tiedot. Käytännössä tämä tarkoittaa ostoskoriin lisättyjen tai käyttäjän katselemien tuotteiden tunnistenumeroita. SessionDataCollector-luokan esiintymä rekisteröidään tuote-ehdotusjärjestelmän pääluokan esiintymän rekisteröinnin yhteydessä.



Kuvio 13. SessionDataCollector-luokan tärkeimmät muuttujat ja funktiot

Rekisteröintifunktion register ajamalla luodaan esiintymä SessionDataCollector-luokasta. Funktion kutsu nähdään kuvassa 7. Kuvasta nähdään, että kyseiselle funktiolle ei tarvitse antaa parametreja. Rekisteröinnin aikana session_data-muuttuja alustetaan ja funktiot init_customer_session, load_session_data ja add_product_to_sequence liitetään sopiviin WP-koukkuihin, jotka WordPress ajaa myöhemmin automaattisesti.

```
$this->session_data_collector = SessionDataCollector::register();
```

Kuva 7. SessionDataCollector-luokan esiintymän luominen

Funktiolla init_customer_session luodaan käyttäjälle WooCommerce-istunto, jos sitä ei ole vielä tehty. Istuntoa tarvitaan tuotteisiin liittyvien tapahtumien tallentamiseen ja tallennetun tapahtumataulukon hakemiseen. Kyseistä funktiota kutsutaan heti WordPressin lataamisen jälkeen ja ennen SessionDataCollector-luokan muita WP-koukkuihin liitettyjä funktioita.

Funktio load_session_data taas hakee WooCommerce-istunnosta sinne tallennetut sen hetkisen käyttäjän viimeisimmät tuotteisiin liittyvät tapahtumat. Tapahtumataulukko asetetaan SessionDataCollector-luokan esiintymän session_data-muuttujaan, kunhan WordPress on saanut WooCommercen alustettua.

Tämän jälkeen viimeisinä klikatut tuotteet sisältävä taulukko voidaan hakea SessionDataCollector-luokan esiintymän kautta sen julkisella get_session_data-funktiolla. Tätä käytetään pääasiassa asetussivuilla valitussa koneoppimismalliluokassa tuote-ehdotusten tekemiseen tarkoitetun funktion aikana.

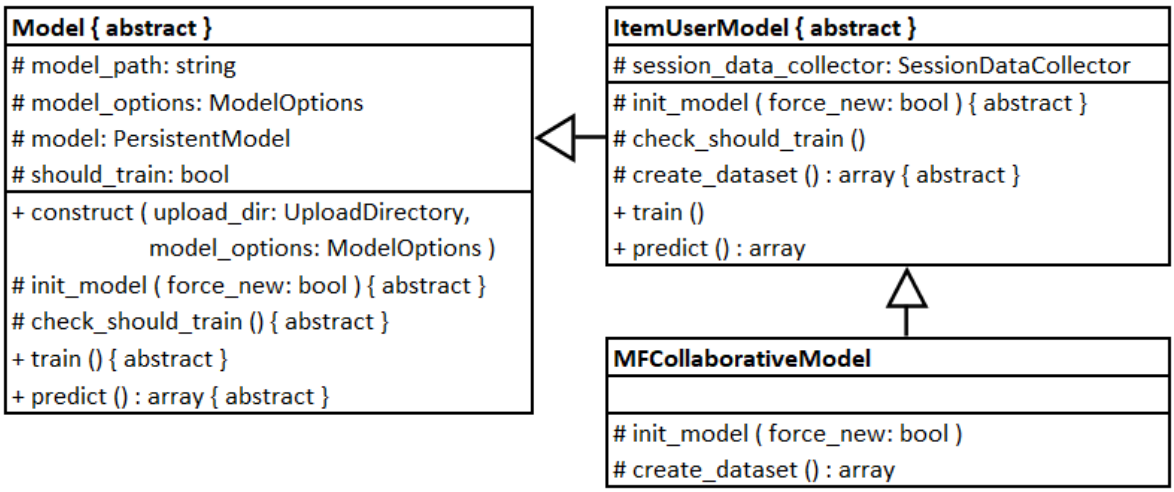
Funktio add_product_to_sequence ajetaan kahden eri WP-koukun yhteydessä: kun käyttäjä vierailee jollakin tuotesivulla tai lisää tuotteen ostoskoriin. Kyseinen funktio tallentaa

käyttäjän viimeisimpien tuotteisiin liittyvien tapahtumien tiedot taulukkoon WooCommerce-istuntoon. Aina kun uusi tapahtuma tallennetaan, vanhin poistetaan tapahtumataulukosta.

4.1.4 Koneoppimismalli

Tuote-ehdotusjärjestelmässä on useita erilaisia koneoppimismalleja, joista kerrallaan voidaan valita yksi. Valitulla mallilla tehdään tuote-ehdotukset. Jotta päällekkäistä koodia ei tulisi paljon, koneoppimismalliluokat on jaettu kolmeen tasoon: kahteen abstraktiin ja yhteen konkreettiseen. Ainoastaan konkreettisista eli kolmannen tason malliluokista voi luoda esiintymiä. Seuraavissa kappaleissa esitellään malliluokkien tasot, ja niiden tarkoitus, perehtymällä MFCollaborativeModel-koneoppimismallin rakenteeseen ja toimintaan.

Koneoppimismalliluokkien tärkeimmät muuttujat ja funktiot on esitelty kuviossa 14. Siitä nähdään, että Model on koneoppimismallien abstrakti pääluokka, joka muodostaa yksin malliluokkien ensimmäisen abstraktin tason. Model-luokasta periytyvät luokat siis tunnistetaan koneoppimismalleiksi. Kaikilla malleilla on model-muuttuja, joka sisältää Rubix-kirjastolla luodun koneoppimisalgoritmin, jolla tuote-ehdotukset tehdään. Mallin käyttämä algoritmi määritetään kolmannen tason koneoppimismalliluokassa. Model-luokasta periytyviä esiintymiä luodaan vain ModelManager-luokan sisällä, joka hallitsee koneoppimismalleja.



Kuvio 14. Koneoppimismalliluokkien tärkeimmät muuttujat ja funktiot

ItemUserModel on myös koneoppimismallien abstrakti luokka, joka kuuluu malliluokkien toiseen abstraktiin tasoon. Toisen tason luokka määrittää mallin opetuksen tarpeen arviointiin käytetyn check_should_train-funktion, train-opetusfunktion ja tuote-ehdotusten tekemiseen käytetyn predict-funktion sisällön. ItemUserModel periytyy Model-luokasta. Kaikki

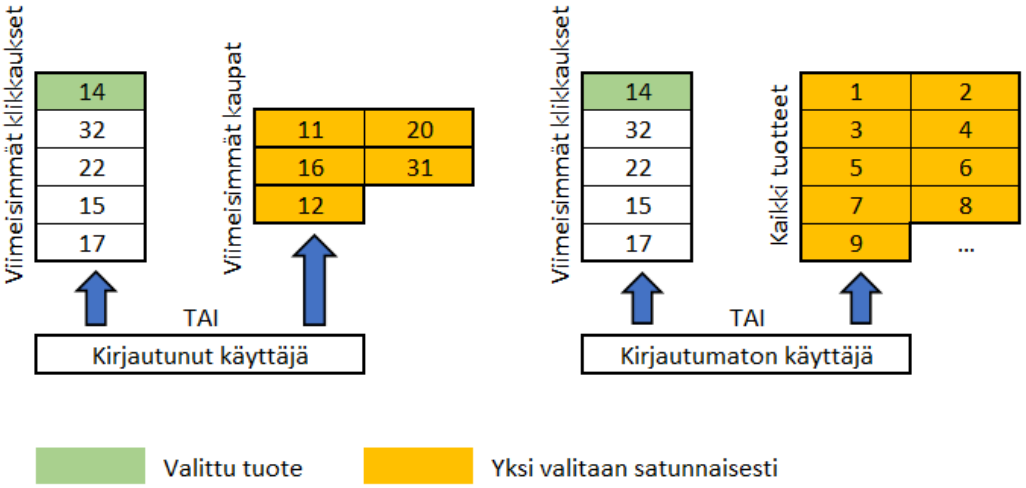
koneoppimismalliluokat, jotka haluavat käyttää ItemUserMatrix-luokan tuote-käyttäjä-matriisia, periytyvät ItemUserModel-luokasta.

Funktio check_should_train ajetaan aina koneoppimismalliluokan esiintymän luomisen yhteydessä, ja sillä siis arvioidaan mallin opettamisen tarve. Jos esimerkiksi mallia ei ole vielä aiemmin opetettu, se tulee opettaa ajankohtaisella datalla. Kyseisestä funktiota voidaan käyttää myös rajoittamaan koneoppimismallin opettamista, ja sen lopputulos tallennetaan should_train-muuttujaan, jonka arvo voidaan tarkistaa julkisella get_should_train-funktiolla. Sitä käytetään esimerkiksi ModelManager-luokassa ennen koneoppimismallin opettamista, jottei mallia opetettaisi turhaan.

ItemUserModel-luokan opetusfunktion aikana luodaan ItemUserMatrix-luokan avulla tuote-käyttäjä-matriisi, jonka tekemiseen käytetään kaikkia tuotteita ja käyttäjiä. Kyseinen matriisi luodaan kutsumalla create_dataset-funktiota, joka määritetään vasta kolmannen tason koneoppimismalliluokassa. Kyseisessä funktiossa luotu matriisi voidaan myös esikäsitellä, ja funktion palauttama matriisi tallennetaan IUMDatabaseTable-luokan kautta tietokantaan ja syötetään sitten koneoppimisalgoritmille sen opettamiseksi.

Edellä mainittua IUMDatabaseTable-luokkaa käytetään myös tuote-ehdotusten tekemisprosessissa viitetuotetta kuvaavan vektorin hakemiseen tietokannasta. Kyseinen vektori on siis koneoppimismallin opetusprosessin yhteydessä tietokantaan tallennetun tuote-käyttäjä-matriisin viitetuotetta vastaava rivi. Näin viitetuotteelle ei tarvitse luoda uutta datavektoria ostohistorian pohjalta tuote-ehdotusprosessin aikana, mikä nopeuttaa kyseistä prosessia.

Viitetuotteen valitsemisprosessi on havainnollistettu kuviossa 15. Viitetuote on yleensä viimeisin katseltu tai ostoskoriin lisätty tuote, ja sen tunnistenumero on tallennettu session_data_collector-muuttujan SessionDataCollector-luokan esiintymään. Sieltä haetaan tapahtumataulukko, josta valitaan viimeisen tuotteen tunnistenumero. Jos käyttäjä ei ole vielä katsellut tai lisännyt ostoskoriin yhtäkään tuotetta meneillään olevan istunnon aikana, viitetuotteeksi valitaan jokin tuote, jonka käyttäjä on ostanut jossakin viimeisistä ostotapahtumistaan. Mikäli käyttäjä ei ole kirjautunut sisään, viitetuote valitaan satunnaisotannalla kaikkien tuotteiden joukosta, jos siis tapahtumataulukkoa ei vielä ole olemassa.



Kuvio 15. Viitetuotteen valitseminen tuote-ehdotusprosessia varten

Lopulta viitetuotetta vastaava datavektori syötetään koneoppimisalgoritmille tuote-ehdotusten luomiseksi. Tuote-ehdotuksista poistetaan tuotteet, jotka käyttäjä on jo lisännyt ostoskoriin tai ostanut aiemmin. Myös viimeisin klikattu tuote poistetaan tuote-ehdotuksista ja jäljelle jäävät tuote-ehdotukset palautetaan.

MFCollaborativeModel kuuluu koneoppimismalliluokkien kolmannelle tasolle, joten se määrittää koneoppimismallin käyttämät algoritmit sekä datan esikäsittelyyn, että tuote-ehdotusten tekemiseen. Se käyttää datan ulottuvuuksien vähentämiseen RSVD:tä ja datan ryhmittelyyn sekä läpikäyntiin KDT:tä. Lisäksi se käyttää CS:ää tuotteiden vertailuun. MFCollaborativeModel periytyy ItemUserModel-luokasta, ja sen esiintymän luomisesta nähdään esimerkki kuvassa 8. Luokalle annetaan parametreina tarvittavat luokkien esiintymät, joiden avulla määritetään mallin tallennuskansio, asetukset ja istunnon aikana kerätyt tiedot. Annettavista parametreista kerrotaan enemmän myöhemmässä kappaleessa, joka käsittelee construct-funktiota.

```
new MFCollaborativeModel(
    $this->upload_dir,
    $this->model_options,
    $this->session_data_collector
);
```

Kuva 8. MFCollaborativeModel-luokan esiintymän luominen

MFCollaborativeModel-luokassa RSVD:llä tehtävä datan ulottuvuuksien vähentäminen tehdään Rubix-kirjaston TruncatedSVD-luokan avulla. Tämä tehdään opetusprosessin aikana create_dataset-funktion yhteydessä. Datan ulottuvuuksia ei siis enää vähennetä tuote-ehdotusten tekemisprosessissa. TruncatedSVD-luokan esiintymän luomisen yhteydessä sille annetaan parametrina datan tavoiteltu ulottuvuuksien määrä (Rubix ML 2021d), jonka arvo määritetään tuote-ehdotusjärjestelmän asetussivuilla. TruncatedSVD-luokan esiintymän luominen nähdään kuvassa 9. Kyseiselle luokalle annettava parametri haetaan WP-asetuksista ModelOptions-luokan esiintymästä, joka on annettu MFCollaborativeModel-luokan esiintymälle parametrina sen luonnin yhteydessä.

```
$current_model_settings = $this->model_options->get_options(
    $this->model_name . '_settings'
);
$transformer = new TruncatedSVD(
    $current_model_settings['dimensions']
);
```

Kuva 9. Datan ulottuvuuksien vähentäminen TruncatedSVD-luokalla

RSVD:n käsittelemä data ryhmitellään KDT:llä, jolla ryhmitellystä datasta myös haetaan tuote-ehdotukset. Ryhmittely tehdään koneoppimismallin opetuksen yhteydessä. KDT:tä käytetään Rubix-kirjaston KDNeighbors-luokan kautta. Kyseiselle luokalle annetaan parametreina palautettavien tuote-ehdotusten määrä, lupa järjestää tuote-ehdotukset niiden etäisyyksien mukaan viitetuotteesta, ja Rubix-hakualgoritmiluokka (Rubix ML 2021e) eli KDTree. KDTreelle annetaan parametrina yksittäisen ryhmän maksimaalinen tuotemäärä ja tuotteiden etäisyyden arviointiin käytettävä Rubix-menetelmäluokka (Rubix ML 2021f) eli Cosine. Cosine-luokalle ei anneta parametreja (Rubix ML 2021a). Tässä kappaleessa kuvattujen Rubix-algoritmiluokkien esiintymien luominen nähdään kuvassa 10. Siinä Rubix-luokkien esiintymät määritetään tallennettavaksi koneoppimisalgoritmiksi PersistentModel-luokan avulla, jolle annetaan parametrina muun muassa opetettava koneoppimisalgoritmi (Rubix ML 2021g) eli KDNeighbors.

Koneoppimismallien construct-funktio ajetaan mallin esiintymän luomisen yhteydessä. Sille annetaan parametreina samat esiintymät, jotka annetaan myös ModelManager-luokan esiintymälle. Ensimmäinen näistä parametreista on UploadDirectory-luokan esiintymä, jonka avulla hallinnoidaan koneoppimismallien tallennuskansioita. Tämän avulla luodaan model_path-muuttujan sisältö. Toisena parametrina annetaan ModelOptions-luokan

esiintymä, jonka avulla hallinnoidaan koneoppimismalleihin liittyviä WP-asetuksia. Tämä parametri asetetaan model_options-muuttujaan. Kolmantena parametrina toimii SessionDataCollector-luokan esiintymä, jonka kautta WooCommerce-istunnosta haetaan viimeisin käyttäjän katselema tai ostoskoriin lisäämä tuote. Tämä parametri asetetaan session_data_collector-muuttujaan.

Kyseisen construct-funktion yhteydessä ajetaan check_should_train-funktio, joka on määritetty toisen tason koneoppimismalliluokassa, kuten ItemUserModelissa. Lisäksi construct-funktiossa ajetaan init_model-funktio, joka luo tuote-ehdotusten tekemiseen tarvittavien Rubix-koneoppimisluokkien esiintymät ja liittää ne tallennettavaksi algoritmiksi.

Funktio init_model määritetään kolmannen tason malliluokissa, kuten MFCollaborativeModel-luokassa. Esimerkki kyseisestä funktiosta nähdään kuvassa 10. Kyseiselle init_model-funktiolle annetaan parametrina muuttuja, joka määrittää, pakotetaanko uuden koneoppimisalgoritmiluokan esiintymän luominen, vaikka sellainen olisi jo olemassa. Koneoppimisalgoritmiluokan esiintymän luomisen yhteydessä sille annettavat parametrit haetaan koneoppimismalliluokan model_options-muuttujasta, joka sisältää Model-asetussivulla annettujen WP-asetusten tuoreimmat arvot. Luotu algoritmi asetetaan malliluokan esiintymän model-muuttujaan.

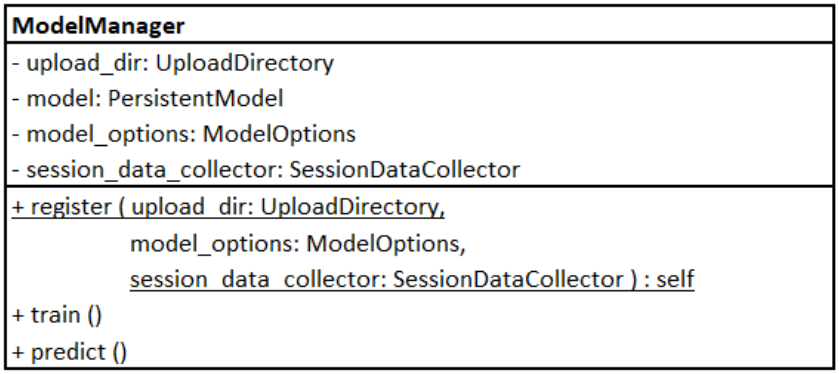
Koneoppimismallin opetuksen jälkeen tuote-ehdotusten tekemiseen opetettu Rubix-koneoppimisalgoritmi tallennetaan tiedostoon. Kyseinen tiedosto luodaan woocommerce_product_recommender-kansioon, joka sijaitsee WordPressin uploads-kansiossa. Tallentamisen jälkeen opetettua algoritmia voidaan käyttää myöhemminkin lataamalla se tallennetusta tiedostosta. Lataaminen voidaan tehdä init_model-funktiossa uuden algoritmiluokan esiintymän luonnin sijaan, mistä nähdään esimerkki kuvassa 10. Kuvasta ilmenee, että jos algoritmin tallennustiedosto löydetään, eikä funktiolle ole annettu parametrina käskyä pakottaa algoritmiluokan uuden esiintymän luomista, tallennettu algoritmiluokka ladataan tuote-ehdotusjärjestelmän käyttöön. Algoritmiluokan uuden esiintymän luominen pakotetaan, kun koneoppimismalli opetetaan, jotta järjestelmänvalvojan näkymässä päivitetyt algoritmiasetukset saataisiin voimaan uuden mallin luomisen kautta.

```
protected function init_model(bool $force_new = false) {
    if (file_exists($this->model_path) && !$force_new) {
        $this->model = PersistentModel::load(
            new Filesystem($this->model_path),
            new RBX()
        );
    } else {
        $current_model_settings = $this->model_options->get_options(
            $this->model_name . '_settings'
        );
        $this->model = new PersistentModel(
            new KDNeighbors(
                $current_model_settings['k'],
                true,
                new KDTree(
                    $current_model_settings['max_leaf_size'],
                    new Cosine()
                )
            ),
            new Filesystem($this->model_path),
            new RBX()
        );
    }
}
```

Kuva 10. MFCollaborativeModel-luokan algoritmiputken luominen ja lataaminen

4.1.5 Koneoppimismallin hallinnointi

ModelManager-luokan tehtävä on hallinnoida koneoppimismalleja, ja sen tärkeimmät muuttujat ja funktiot nähdään kuviossa 16. Kuviosta nähdään, että kyseisen luokan avulla voidaan esimerkiksi opettaa malleja ja tehdä tuote-ehdotuksia. Lisäksi ModelManager luo tarvittavien koneoppimismalliluokkien esiintymät. ModelManager-luokan esiintymä rekisteröidään tuote-ehdotusjärjestelmän pääluokan esiintymän luomisen yhteydessä.



Kuvio 16. ModelManager-luokan tärkeimmät muuttujat ja funktiot

ModelManager-luokan esiintymä luodaan rekisteröintifunktiolla nimeltä register. Esimerkki rekisteröintifunktion kutsusta nähdään kuvassa 11. Sille annetaan parametreina esiintymät luokista UploadDirectory, ModelOptions ja SessionDataCollector. Niiden avulla hallinnoidaan latauskansioita, koneoppimismallien WP-asetuksia ja WooCommerce-istunnon keräämää dataa. UploadDirectory-luokan esiintymän avulla muun muassa määritetään kansio, jonne koneoppimismallit tallennetaan. ModelOptions-luokan esiintymällä taas haetaan eri mallien WP-asetukset, joita käytetään muun muassa koneoppimismalliluokkien esiintymien luomiseen. Lisäksi nämä WP-asetukset määrittävät tuote-ehdotusten tekemiseen käytettävän koneoppimismallin. SessionDataCollector-luokan esiintymä taas toimii viitetuotteen selvittämisessä.

```
$this->model_manager = ModelManager::register(
    new UploadDirectory(),
    $this->options['model'],
    $this->session_data_collector
);
```

Kuva 11. ModelManager-luokan esiintymän luominen

Nämä parametreilla annetut esiintymät luodaan tuote-ehdotusjärjestelmän pääluokassa, ja ne asetetaan ModelManager-luokan muuttujiin upload_dir, model_options ja session_data_collector. ModelManager-luokan esiintymän luomisen yhteydessä luodaan myös esiintymä tuote-ehdotusten tekemiseen valitusta koneoppimismalliluokasta, jolle myös annetaan edellä mainitut parametrit. Koneoppimismalliluokan esiintymän muut funktiot ovat

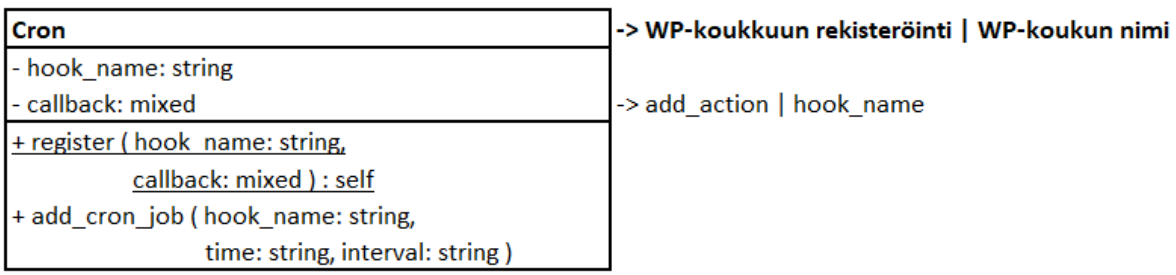
tämän jälkeen kutsuttavissa sen itsensä kautta. Lisäksi se asetetaan ModelManager-luokan esiintymän model-muuttujaan.

Opetusfunktio train luo esiintymät koneoppimismalleista, jotka on valittu pidettäviksi päivitettyinä model_options-muuttujaan asetetuissa WP-asetuksissa. Lisäksi opetusfunktio tarkistaa päivitettävien mallien päivitystarpeen kutsumalla malleille tähän tarkoitukseen määritettyä julkista funktiota. Päivitystä odottavat mallit opetetaan kutsumalla koneoppimismalliluokkien julkista opetusfunktiota. Myös tuote-ehdotusten tekemiseen valittu koneoppimismalli opetetaan. Opettamiseen käytetty data riippuu käytetystä mallista.

Mallin opettamisen jälkeen sillä voidaan tehdä tuote-ehdotuksia predict-funktion avulla. Tämä funktio voi käyttää tuote-ehdotusten tekemiseen ainoastaan tuote-ehdotusjärjestelmän asetussivuilla valittua koneoppimismallia, joka on asetettu ModelManager-luokan model_options-muuttujaan.

4.1.6 Koneoppimismallin opetuksen ajastaminen

Koneoppimismallien opettaminen ajastetaan Cron-luokan avulla, jolla hallitaan WP-Cron-ajastuksia. Kuvaus luokan tärkeimmistä muuttujista ja funktioista nähdään kuviossa 17. Siitä nähdään, että luokalla voidaan luoda uusia ajastettuja WP-Cron-koukkuja, joihin voidaan liittää funktioita, jotka sitten ajetaan tietyin väliajoin. Ajastettujen funktioiden ajamisen tarve arvioidaan vain WordPress-sivun vierailun yhteydessä, joten ajankohta, jolloin ajastettu funktio tulisi ajaa, ei ole kovinkaan tarkka (WordPress f). Cron-luokan esiintymä rekisteröidään tuote-ehdotusjärjestelmän pääluokan esiintymän rekisteröinnin yhteydessä.



Kuvio 17. Cron-luokan tärkeimmät muuttujat ja funktiot

Uusi ajastettu WP-Cron-koukku voidaan luoda ajamalla add_cron_job-funktio. Tästä nähdään esimerkki kuvan 12 vasemmassa laidassa. Funktiolle annetaan parametreina WP-Cron-koukun nimi, ajastuksen aloitusajankohta ja intervalli, jonka välein koukkuun liitetyt funktiot ajetaan. Rekisteröintifunktiolla nimeltä register luodaan sitten Cron-luokasta uusi

esiintymä. Tästä nähdään esimerkki kuvan 12 oikeassa laidassa. Rekisteröintifunktiolle annetaan parametreinä sekä edellä luotu WP-Cron-koukku, että ajastettava funktio, joka liitetään kyseiseen koukkuun. Parametrit asetetaan rekisteröidyn Cron-luokan esiintymän muuttujiin.

```
Cron::add_cron_job(
    'wcpr_cron_hook',
    'today',
    'daily'
);
```

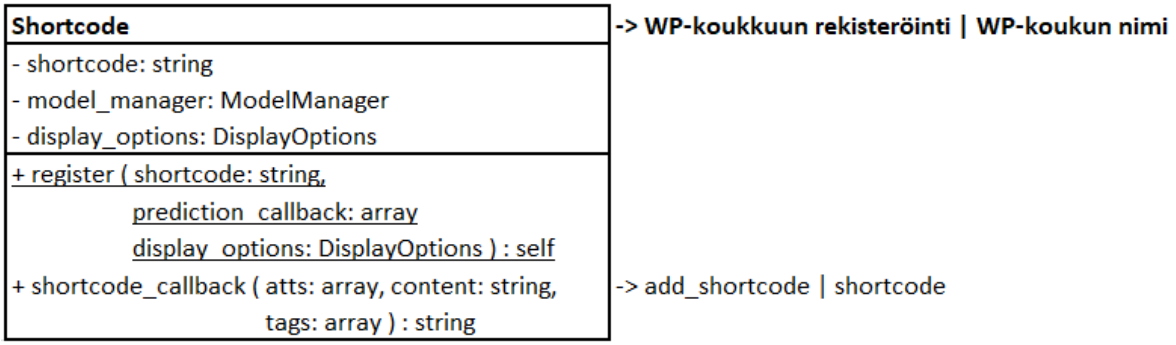
```
$this->cron = Cron::register(
    'wcpr_cron_hook',
    [$this->model_manager, 'train']
);
```

Kuva 12. WP-Cron-ajastuksen luominen ja funktion liittäminen siihen

WP-Cron-ajastus luodaan tuote-ehdotusjärjestelmän aktivoinnin yhteydessä. Ajastettuun koukkuun liitetään ModelManager-luokan koneoppimismallien opetusfunktio. Ajastettu opetusfunktio pyritään ajamaan kerran päivässä keskiyöllä. Ajastuksen avulla koneoppimismallit voidaan pitää päivitettyinä ilman manuaalisia toimia. Ajastus myös mahdollistaa koneoppimismallien päivittämisen mahdollisimman rauhallisena ajankohtana, kuten yöllä. Tällöin mallien opettaminen vaikuttaa mahdollisimman vähän WordPress-sivuston suorituskykyyn.

4.1.7 Tuote-ehdotusten tuominen verkkosivulle

Shortcode-luokka vastaa tuote-ehdotusten tuomisesta verkkosivuille ja tuote-ehdotusten design-valinnan eli ulkomuodon kontrolloimisesta. Kyseisen luokan tärkeimmät muuttujat ja funktiot nähdään kuviossa 18. Siitä nähdään, että edellä kuvattu tuote-ehdotusten tuominen toteutetaan verkkosivuille lisättävän lyhytkoodin avulla, jolla myös määritetään toivottu design. Tuote-ehdotusten tuominen verkkosivuille on kuitenkin mahdollista vasta, kun käytössä oleva koneoppimismalli on opetettu. Shortcode-luokasta rekisteröidään esiintymä tuote-ehdotusjärjestelmän pääluokassa.



Kuvio 18. Shortcode-luokan tärkeimmät muuttujat ja funktiot

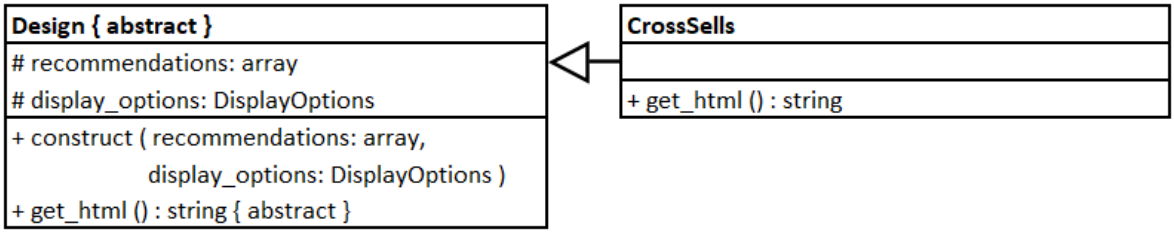
Shortcode-luokasta luodaan uusi esiintymä register-rekisteröintifunktion avulla. Funktion toimintalogiikka myös määrittää käytettävissä olevat design-vaihtoehdot. Shortcode-luokan esiintymän rekisteröiminen nähdään kuvassa 13. Rekisteröintifunktiolle annetaan parametrina muun muassa lyhytkoodi, jolla tuote-ehdotukset lisätään verkkosivuille. Toisena parametrina annetaan esiintymä ModelManager-luokasta, jota käytetään tuote-ehdotusten luomiseen. Kolmantena parametrina annetaan DisplayOptions-luokan esiintymä, jolla hallitaan designeihin liittyviä asetuksia. Parametrit asetetaan rekisteröidyn Shortcode-luokan esiintymän muuttujiin. Lisäksi parametrina annettu lyhytkoodi rekisteröidään, ja siihen liitetään shortcode_callback-funktio. Rekisteröity lyhytkoodi aktivoidaan, kun käyttäjä vierailee verkkosivulla, johon se on asetettu. Tällöin koukkuun liitetty funktio ajetaan.

```
$this->shortcode = Shortcode::register(
    'woocommerce_product_recommender',
    $this->model_manager,
    $this->options['display']
);
```

Kuva 13. Shortcode-luokan esiintymän luominen

Edellä mainittu shortcode_callback-funktio erottaa ensin verkkosivulle asetetusta lyhytkoodista design-valinnan. Design-valintoja on monia, joista yksi kerrallaan valitaan. Samalle sivulle voidaan kuitenkin tuoda monta tuote-ehdotusjärjestelmän ilmentymää, joihin kaikkiin on mahdollista valita eri design. Slider, CrossSells ja UpSells ovat design-luokkia, ja design-valinta määrittää, mitä näistä luokista käytetään tuote-ehdotusten HTML-koodin luomiseen.

Luokat Slider, CrossSells ja UpSells periytyvät abstraktista Design-luokasta, joka on design-valintojen pääluokka. Siitä periytyvät luokat tunnistetaan siis design-valinnoiksi. Tärkeimmät design-luokkien muuttujat ja funktiot nähdään kuviossa 19. Kuviosta nähdään, että edellä mainittujen design-luokkien avulla luodaan tuote-ehdotusten HTML-koodi, joka perustuu design-luokassa määritettyyn PHP-sapluunaan. Esimerkiksi CrossSells-luokka käyttää WooCommercesta valmiiksi löytyvää cross-sells.php-sapluunaa. Kyseisen sapluunan pitäisi noudattaa aktiivisen WordPress-teeman tyylejä automaattisesti.



Kuvio 19. Design-luokkien tärkeimmät muuttujat ja funktiot

Funktio construct määritetään abstraktissa Design-luokassa ja ajetaan Design-luokasta periytyvän luokan esiintymän luomisen yhteydessä. Esimerkki CrossSells-luokan esiintymän luomisesta on esitetty kuvassa 14. Kyseiselle construct-funktiolle annetaan parametreina tuote-ehdotukset sisältävä taulukko ja DisplayOptions-luokan esiintymä, joka hallinnoi design-valintojen asetuksia. Tuote-ehdotukset luodaan Shortcode-luokan model_manager-muuttujan predict-funktiolla, ja DisplayOptions-luokan esiintymänä voidaan käyttää Shortcode-luokan display_options-muuttujaa. Parametrina annetut tuote-ehdotukset asetetaan sitten design-luokan muuttujiin.

```
new CrossSells(
    $this->model_manager->predict(),
    $this->display_options
);
```

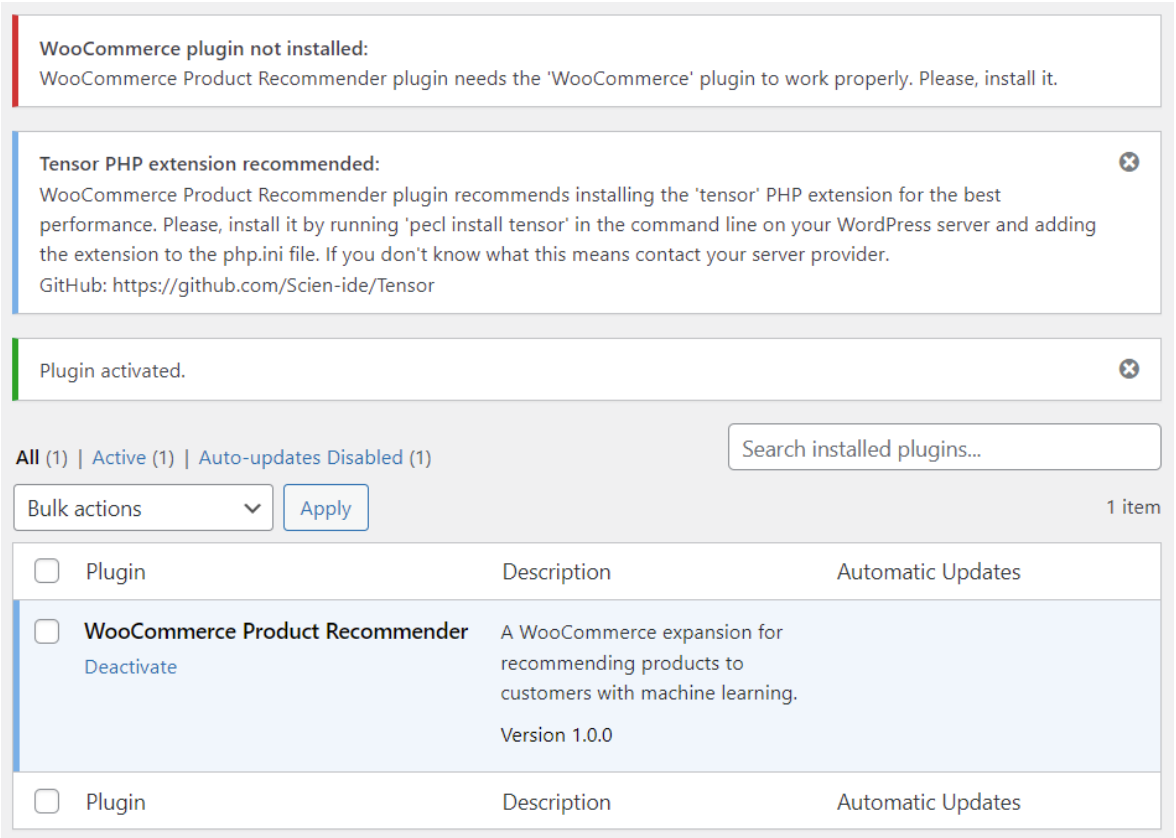
Kuva 14. CrossSells-luokan esiintymän luominen

Luokissa Slider, CrossSells ja UpSells määritetään ainoastaan Design-luokalta peritty get_html-funktio. Sillä haetaan ensin design-luokan display_options-muuttujasta design-valinnan asetukset, kuten näytettävien tuote-ehdotusten kokonaismäärä ja yksittäisellä rivillä

näytettävien tuote-ehdotusten lukumäärä. Näitä arvoja ja tuote-ehdotukset sisältävää recommendations-muuttujaa käytetään tuote-ehdotukset sisältävän HTML-koodin luomiseen. Lopulta luotu HTML-koodi palautetaan ja asetetaan verkkosivulle lyhytkoodin tilalle.

4.2 Järjestelmän havainnollistaminen käytännössä

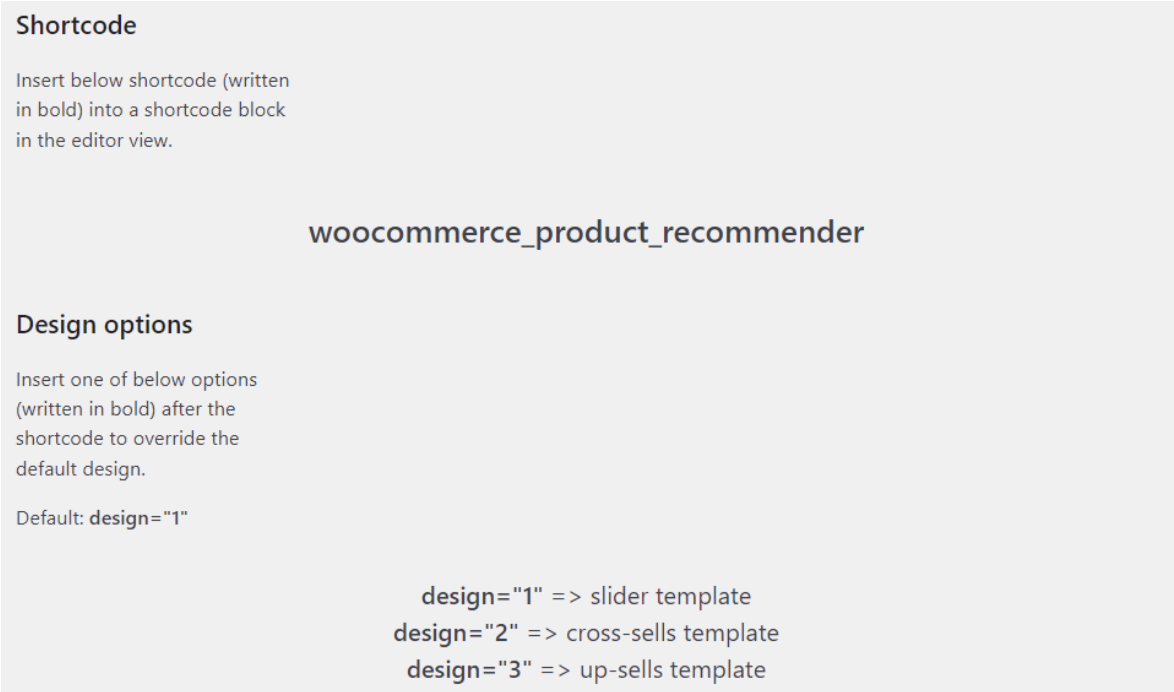
Tuote-ehdotusjärjestelmä asennetaan, kuten muutkin WordPress-lisäosat, siirtämällä järjestelmän projektikansio WordPress-kansiorakenteessa polkuun /wp-content/plugins, ja klikkaamalla lisäosa sitten aktiiviseksi WordPressin järjestelmänvalvojan näkymässä Plugins-sivulla. Jos tuote-ehdotusjärjestelmä on aktivoitu, mutta WooCommerce-lisäosaa ei ole asennettu, asiasta esitetään huomautus WordPress-sivuston järjestelmänvalvojan näkymässä. Myös Tensor-laajennuksen puuttuminen WordPress-palvelimelta aiheuttaa samankaltaisen huomautuksen. Edellä mainitut huomautukset voidaan nähdä kuvassa 15. Siitä nähdään, että tuote-ehdotusjärjestelmä vaatii WooCommerce-lisäosan toimiakseen, mutta Tensor-laajennuksen asentamista vain suositellaan. Tämä voidaan tulkita huomautusten vasemman laidan väristä.



Kuva 15. Huomautukset WooCommerce-lisäosan ja Tensor-laajennuksen puuttumisesta

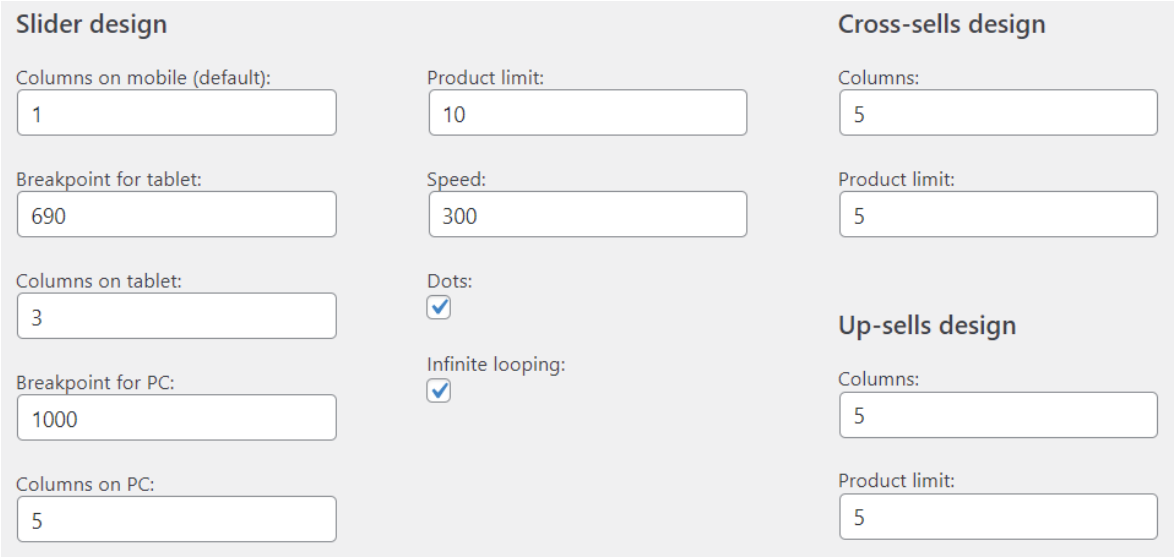
Kun tuote-ehdotusjärjestelmä, WooCommerce-laajennus ja Tensor-lisäosa on asennettu, WordPressin järjestelmänvalvojan näkymään vasemman laidan palkkiin ilmestyy Product Recommender -painike. Kun hiiren vie sen päälle, näkyviin tulee kaikkien tuote-ehdotusjärjestelmän asetussivujen otsikot, joita painamalla voidaan vierailla kyseisillä sivuilla. Asetussivuja on 3 kappaletta: General (Kuva 16), Display (Kuva 17) ja Model (Kuva 18).

Kuvassa 16 näkyvältä General-sivulta löytyvät ohjeet tuote-ehdotusten lisäämiseksi verkkosivuille. Kuvasta nähdään, että tuote-ehdotusjärjestelmä tarjoaa kolme erilaista design-vaihtoehtoa: slider, cross-sells ja up-sells. Tässä työssä keskitytään edellä mainituista designeista vain cross-sellsiin.



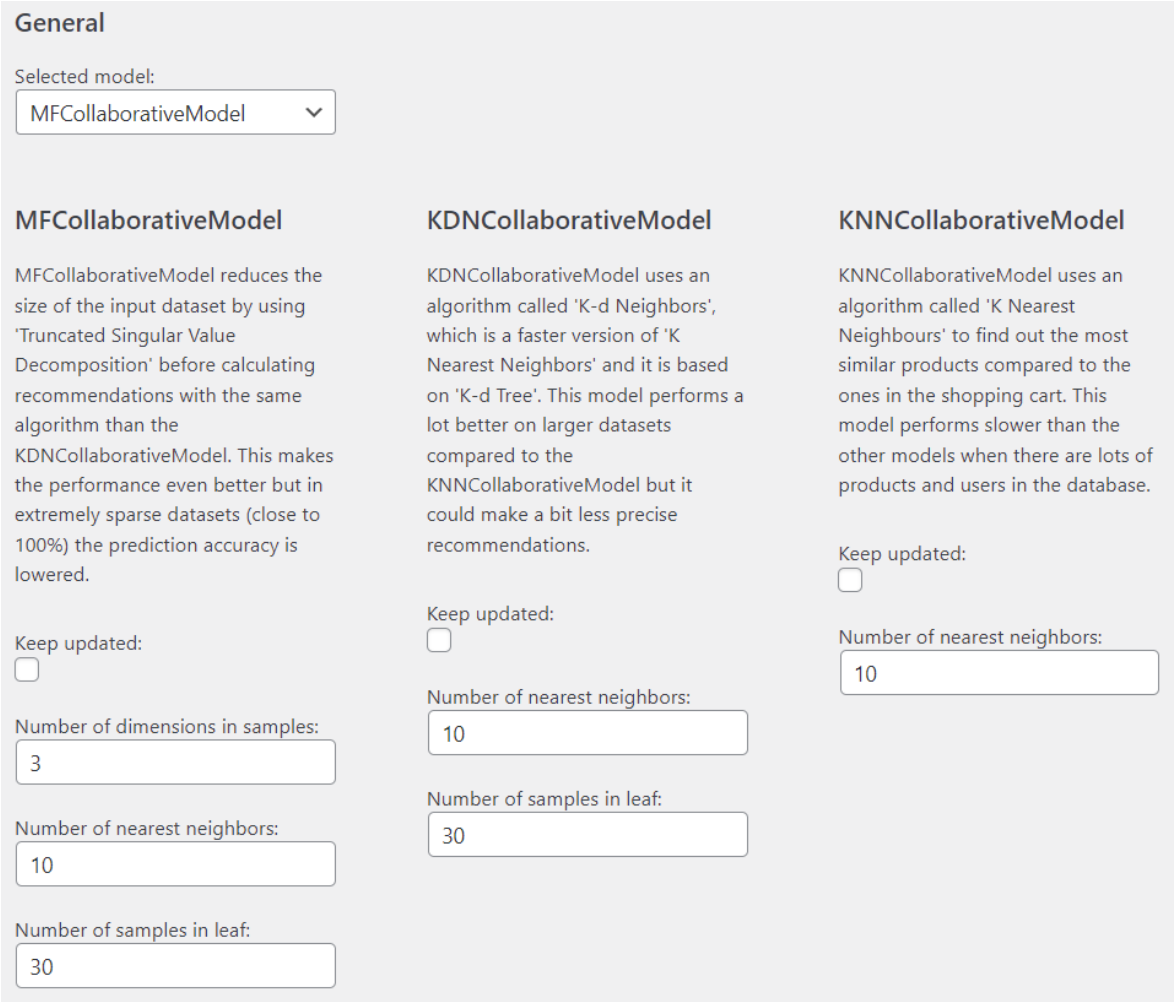
Kuva 16. General-asetussivu

Kuvassa 17 näkyvällä Display-sivulla nähdään eri design-vaihtoehtoihin liittyvät asetukset. Esimerkiksi cross-sells-designia käytettäessä verkkosivulla näytetään yhteensä korkeintaan 5 tuote-ehdotusta yhdellä rivillä. Tuote-ehdotusten sallittu kokonaismäärä on nimittäin asetettu yhtä suureksi kuin yhdelle riville mahtuvien tuote-ehdotusten lukumäärä.



Kuva 17. Display-asetussivu

Kuvassa 18 näkyvällä Model-sivulla nähdään eri koneoppimisluokkiin liittyvät asetukset. Esimerkiksi MFCollaborativeModel-koneoppimismalli tiivistää kuvan perusteella sille annetun datan korkeintaan 3 ulottuvuuteen. Se myös ryhmittelee datan korkeintaan 30 tuotteen ryhmiin. Lisäksi se valitsee datan joukosta 10 mahdollista tuote-ehdotusta, joista poistetaan ne, jotka käyttäjä on jo lisännyt ostoskoriin tai ostanut. Tässä raportissa keskitytään nimenomaan MFCollaborativeModel-koneoppimismallilla tehtyihin tuote-ehdotuksiin.

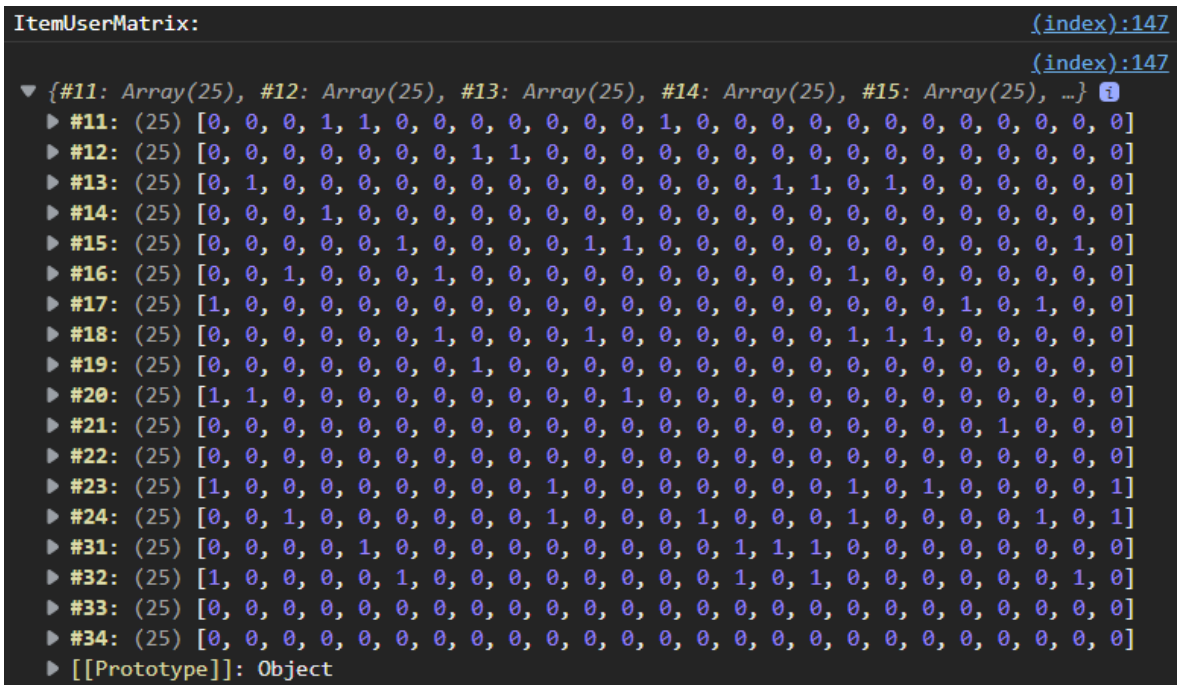


Kuva 18. Model-asetussivu

Tuote-ehdotusjärjestelmän lyhytkoodi asetetaan halutulle WordPress-sivulle kyseisen sivun muokkausnäkymässä. Esimerkki lyhytkoodista nähdään kuvassa 19. Siinä designiksi on valittu cross-sells. Lyhytkoodin lisäämisen jälkeen tuote-ehdotukset näkyvät sivulla vierailtaessa, kunhan koneoppimismalli on opetettu. Mallien opettaminen tehdään ajastetusti kerran vuorokaudessa.

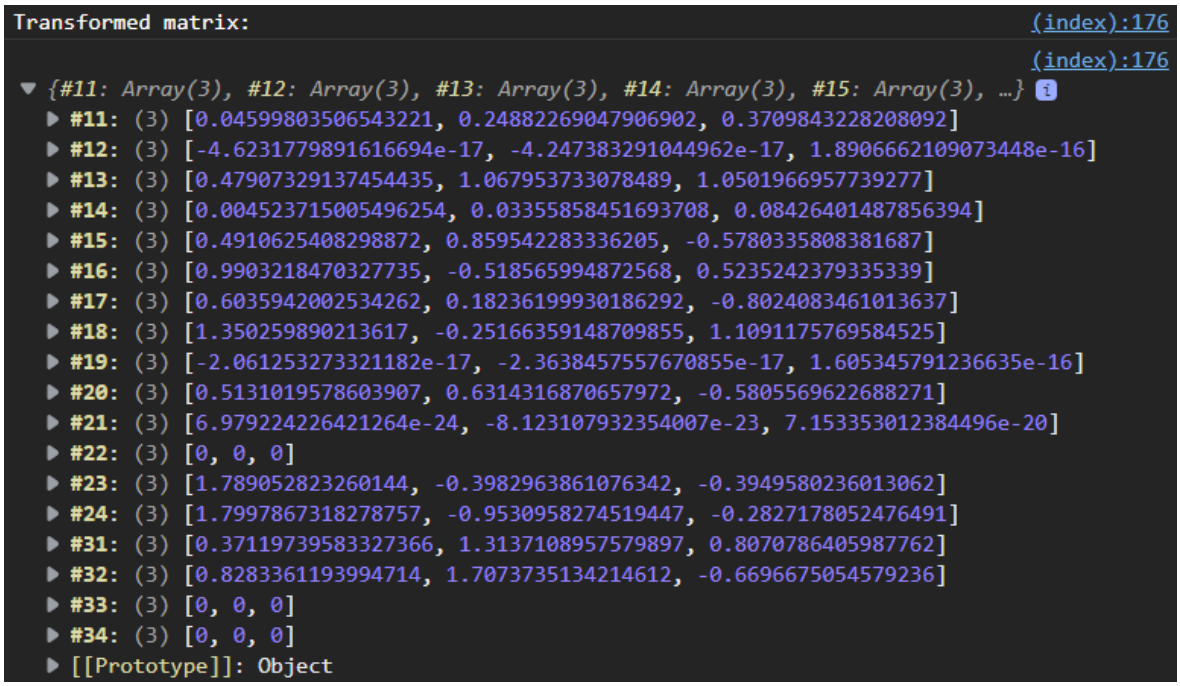
Kuva 19. Tuote-ehdotusten lisääminen WordPress-sivulle

MFCollaborativeModel-koneoppimismallin opettamista varten luodaan tuote-käyttäjä-matriisi. Tuote-ehdotusprosessin toiminnan havainnollistamista varten tietokantaan lisättiin vain 18 tuotetta ja 25 käyttäjää, jotta tuote-ehdotusten arvioiminen olisi helpompi tehdä visuaalisesti. Mallin opettamiseen käytetyssä tuote-käyttäjä-matriisissa on siis vain 18 vaakariviä ja 25 pystyriviä eli ulottuvuutta, mikä nähdään kuvasta 20.



Kuva 20. Tuote-käyttäjä-matriisi ennen ulottuvuuksien vähentämistä

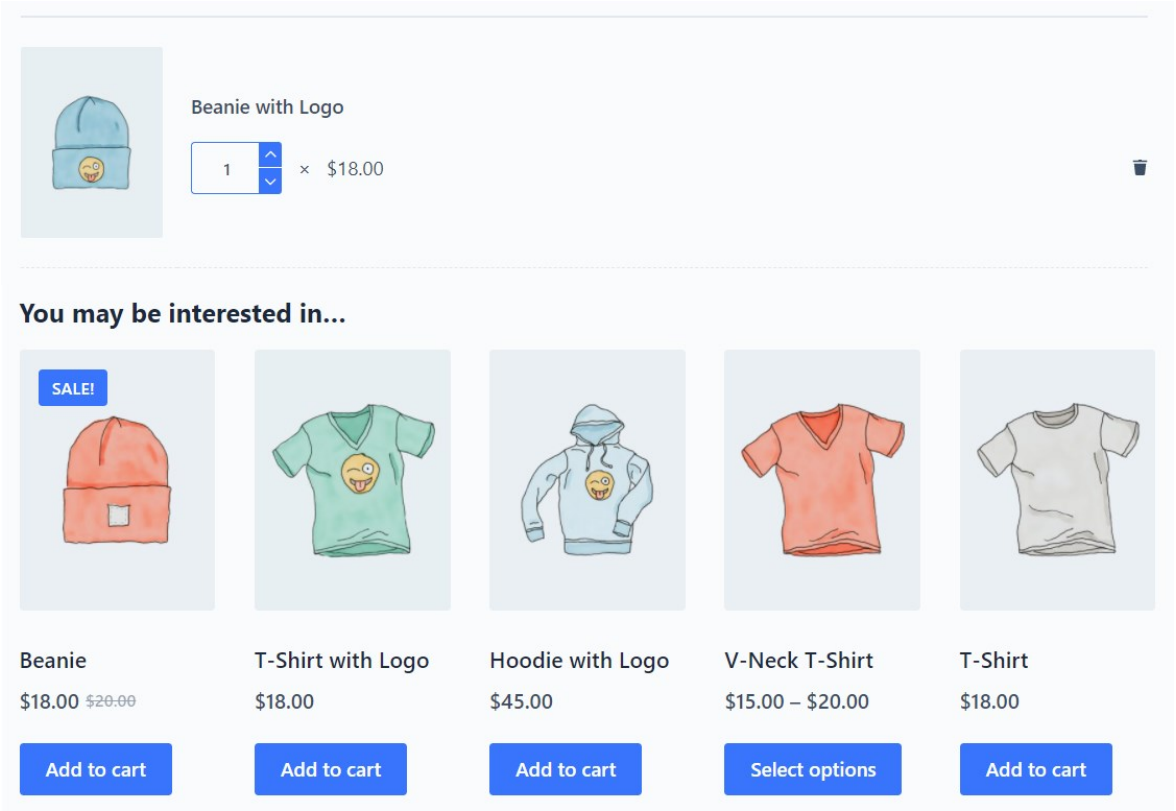
Tuote-käyttäjä-matriisin ulottuvuudet vähennetään RSVD:llä ennen kuin sitä käytetään koneoppimismallin opettamiseen. Ulottuvuuksien määrä on valittu Model-asetussivulla. Kuvassa 21 nähdään esimerkki tuote-käyttäjä-matriisista ulottuvuuksien vähentämisen jälkeen. Ulottuvuuksia on jäljellä 3 kappaletta. Koneoppimismalli opetetaan tiivistetyllä tuote-käyttäjä-matriisilla, jonka jälkeen kyseinen matriisi tallennetaan tietokantaan tuote-ehdotusten tekemisprosessia varten.



Kuva 21. Tuote-käyttäjä-matriisi ulottuvuuksien vähentämisen jälkeen

Tuote-ehdotusten tekemiseen käytettävä koneoppimisalgoritmi luodaan myös Model-asetussivulla annettujen arvojen mukaan. Näin tuote-ehdotuksia voidaan hienosäätää. Käyttäjälle kuitenkin pyritään ehdottamaan tuotteita, jotka ovat samankaltaisia kuin häntä viime aikoina kiinnostaneet tai juuri sillä hetkellä kiinnostavat tuotteet. Tähän pyritään mukauttamalla tuote-ehdotukset valitun viitetuotteen tai mahdollisuuksien mukaan. Viitetuotetta kuvaava vektori saadaan tuote-ehdotusprosessia varten haettua tietokantaan tallennetusta tiivistetystä tuote-käyttäjä-matriisista.

Tuote-ehdotusten ulkoasun luomiseen käytetään lyhytkoodissa valittua designia ja Display-asetussivulla määritettyjä asetuksia. Näin tuote-ehdotusten ulkoasuun voidaan vaikuttaa. Lopullinen ulkomuoto kuitenkin riippuu myös WordPress-teemasta. Kuvassa 22 nähdään esimerkki, miltä tuote-ehdotukset näyttävät, kun ne luodaan aiemmin tässä luvussa mainittujen valintojen ja asetusten pohjalta. Viitetuote eli tässä tapauksessa ostoskorin sisältö näkyy kuvan ylälaidassa ja tuote-ehdotukset alalaidassa. Tuote-ehdotuksia näytetään yhteensä 5 kappaletta, joista on poistettu käyttäjän ostamat tuotteet, ostoskorin sisältö sekä viitetuote.



Kuva 22. Ostoskorin sisältö ja tuote-ehdotukset

5 Järjestelmän toimivuuden arvioiminen

5.1 Tuote-ehdotusten mielekkyys

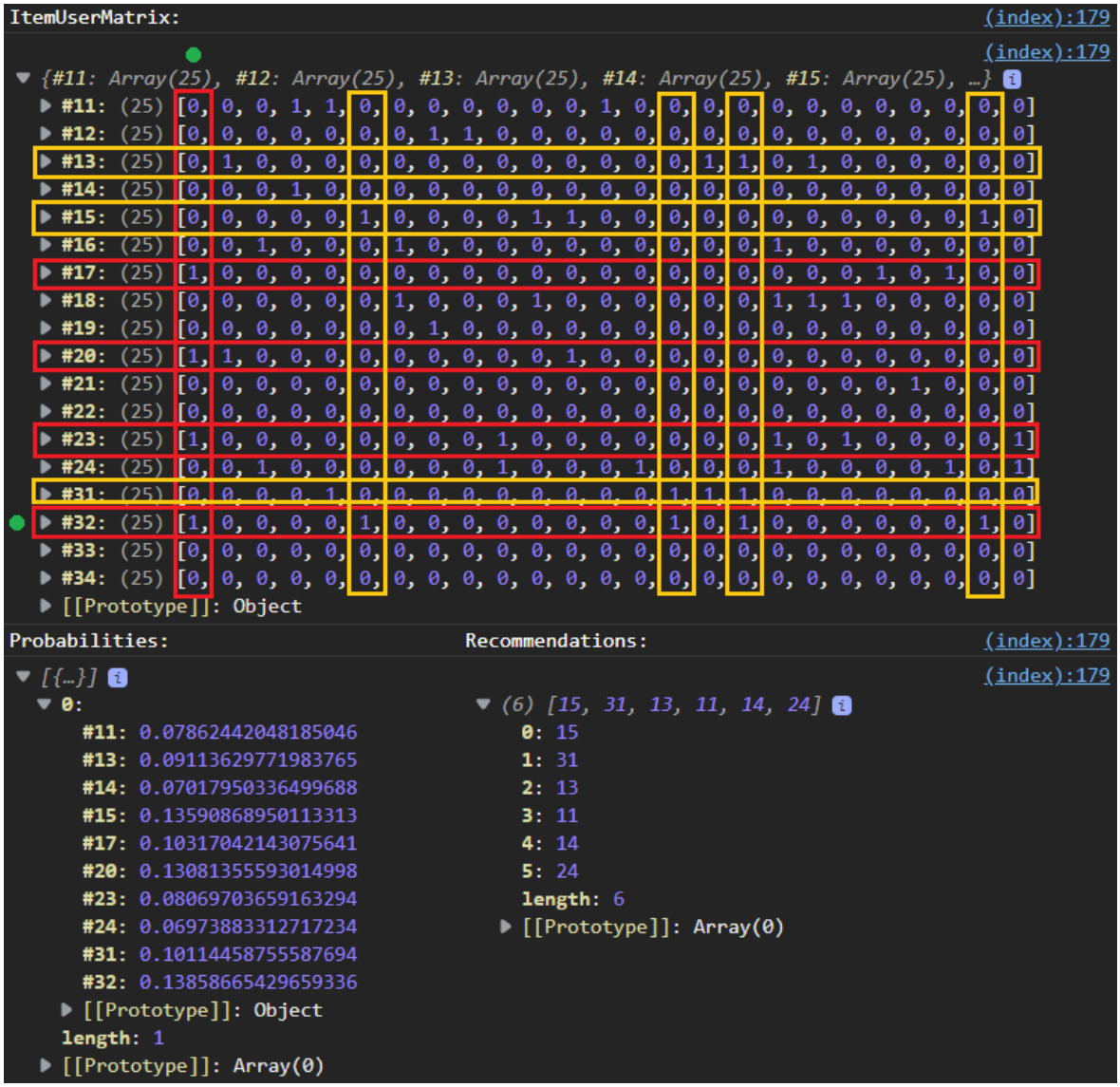
Tuote-ehdotusten mielekkyyteen voi vaikuttaa moni asia. Jos ne ovat esimerkiksi liian samanlaisia kuin käyttäjän aiemmin ostamat tuotteet, tuote-ehdotukset eivät tietenkään auta käyttäjää löytämään uudenlaisia tuotteita, joita hän ei ehkä aiemmin tullut ajatelleeksi. Jos taas tuote-ehdotukset poikkeavat liian paljon käyttäjän mielenkiinnon kohteista, ne saatetaan kokea turhiksi. Toisin sanoen tuote-käyttäjä-matriisi tulisi onnistua yleistämään niin, että löytyisi hyvä tasapaino uusien ja tuttujen tuotteiden ehdottamisen välille. Edellä kuvattu ongelma ratkaistaan tässä työssä RSVD-menetelmällä.

Tuote-ehdotusten tarkkuuteen kuitenkin vaikuttaa datalle valittu ulottuvuuksien määrä. Mitä enemmän ulottuvuuksia käytetään, sitä tarkempia tuote-ehdotukset ovat. Sama ajatus toimii myös toisin päin, eli mitä vähemmän ulottuvuuksia käytetään, sitä enemmän käyttäjälle ehdotetaan hänen kiinnostuksenkohteistaan poikkeavia tuotteita. (Sarwar ym. 2000, 13.) Täytyy siis vain yrittää löytää sopiva ulottuvuuksien määrä, jotta saadaan tehtyä parhaita mahdollisia tuote-ehdotuksia. Tämän löytämiseksi ei kuitenkaan ole mitään sääntöä, koska ulottuvuuksien oikea määrä näyttäisi riippuvan ainakin tuote-käyttäjä-matriisin harvuudesta. Sopiva arvo voidaan siis löytää vain kokeilemalla.

Tietokantaan oli tässä vaiheessa asetettu 18 tuotetta ja 25 käyttäjää. Koneoppimismallina toimi MFCollaborativeModel, jolle syötetyn tuote-käyttäjä-matriisin ulottuvuudet vähennettiin 3:een, ja tuote-ehdotuksia haettiin yhteensä 10. Koska käyttäjät muodostavat tuote-käyttäjä-matriisin pystyrivit eli ulottuvuudet, ulottuvuuksien määrä vähennettiin noin 12 prosenttiin alkuperäisestä. Lisäksi tietokantaan oli lisätty 26 ostotapahtumaa, ja yhteen ostotapahtumaan kuuluu keskimäärin 2 tuotetta. Lopullisessa tuote-käyttäjä-matriisissa oli siis nollasta poikkeavia arvoja noin 12 prosenttia kaikkien arvojen lukumäärästä.

Kuvasta 23 voidaan päätellä, millaisia tuote-ehdotuksia edellä kuvatuilla arvoilla luodun koneoppimismallin tulisi tehdä. Alkuperäinen tuote-käyttäjä-matriisi sijaitsee kuvan yläosassa. Siihen on merkitty vihreät pallot kirjautuneen käyttäjän ja valitun viitetuotteen kohdalle. Viitetuotteen kanssa lähimpiä ovat kuvan perusteella tuotteet 13, 15, 17, 20, 23 ja 31. Kyseisiä tuotteita ovat nimittäin ostaneet samat käyttäjät, jotka viitetuotettakin ovat ostaneet. Näistä kuitenkin tulisi poistaa tuotteet 17, 20 ja 23, koska kirjautunut käyttäjä on jo ostanut ne. Kyseiset tuotteet on merkitty punaisella. Keltaisella taas on merkitty viitetuotteen ostaneet käyttäjät ja heidän aiemmin ostamansa tuotteet, joita ei poisteta lopullisista tuote-ehdotuksista. Jotta koneoppimismallia voitaisiin sanoa tarkaksi, sen tekemien tuote-ehdotusten pitäisi siis sisältää ainakin tuotteet 13, 15 ja 31.

Kuvasta 23 nähdään myös, kuinka hyvin lopulliset tuote-ehdotukset mukailivat sopivimmiksi pääteltyjä tuotteita. Kuvan vasemman alalaidan taulukosta nähdään, että koneoppimismalli arvioi viitetuotteeseen nähden sopivimmiksi tuotteet 11, 13, 14, 15, 17, 20, 23, 24, 31 ja 32. Tuotteet eivät ole sopivuusjärjestyksessä, mutta järjestys voidaan päätellä tuotteiden tunnisteiden oikealla puolella näkyvistä arvoista. Suurempi arvo tarkoittaa sopivampaa tuotetta. Tuote-ehdotuksista poistettiin viitetuote ja kirjautuneen käyttäjän aiemmin ostamat tuotteet. Ehdotuksiksi hyväksyttiin sopivuusjärjestyksessä tuotteet 15, 31, 13, 11, 14 ja 24. Kolmen kärki sisälsi siis juuri ne tuotteet, jotka tuote-käyttäjä-matriisin perusteella pääteltiin sopivimmiksi. Tämän perusteella voidaan sanoa, että käytetty koneoppimismalli kykenee tuottamaan tarkkoja tuote-ehdotuksia ainakin silloin, kun tuote-käyttäjä-matriisissa on matriisin kokoon nähden suhteellisen paljon nollasta poikkeavia arvoja.



Kuva 23. Tuote-ehdotusten arvioiminen

5.2 Suorituskyky

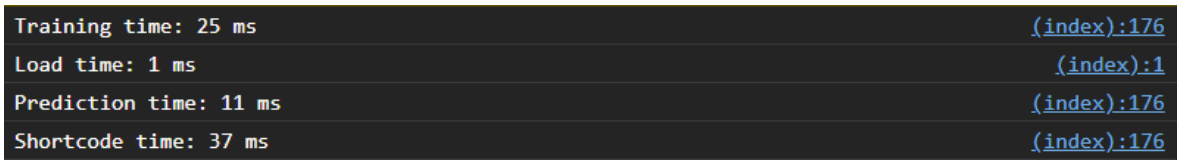
Tuote-ehdotusjärjestelmän suorituskykyä testattiin kolmessa vaiheessa. Ensimmäisessä testausvaiheessa tietokannassa oli 18 tuotetta ja 25 käyttäjää. Koneoppimismallin luomassa tuote-käyttäjä-matriisissa oli tässä vaiheessa nollasta poikkeavia arvoja noin 12 prosenttia kaikkien arvojen lukumäärästä. Tämän vuoksi matriisin ulottuvuuksien määrä voitiin pitää vähäisenä. Ulottuvuuksia määritettiin olevan 3 kappaletta.

Toisessa vaiheessa tuote-ehdotusjärjestelmää testattiin suuremmalla käyttäjien ja tuotteiden määrällä. Tällöin tietokantaan oli tallennettu 100 tuotetta ja 2000 käyttäjää. Koneoppimismallin luomassa tuote-käyttäjä-matriisissa oli nollasta poikkeavia arvoja 2,0 prosenttia kaikkien arvojen lukumäärästä. Niinpä matriisin ulottuvuuksien määrä voitiin edelleen pitää melko vähäisenä. Tässä vaiheessa ulottuvuuksien määrä nostettiin 30 kappaleeseen.

Kolmannessa vaiheessa tuote-ehdotusjärjestelmä testattiin vielä selvästi suuremmalla käyttäjien ja tuotteiden määrällä. Tässä kolmannessa testissä tietokantaan oli tallennettu 200 tuotetta ja 5000 käyttäjää. Mallin luomassa tuote-käyttäjämatriisissa oli nollasta poikkeavia arvoja 1,2 prosenttia kaikkien arvojen lukumäärästä. Koska käyttäjien määrä ja tuote-käyttäjä-matriisin nollasta poikkeavien arvojen suhteellinen osuus kasvoi merkittävästi, ulottuvuuksien määrä nostettiin 100 kappaleeseen.

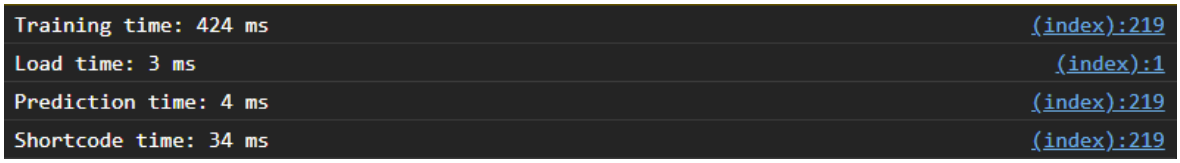
Kuvista 24, 25 ja 26 nähdään tuote-ehdotusjärjestelmän eri prosesseihin kuluneet ajat millisekunteina testausvaiheittain eriteltyinä. Jokaisen testausvaiheen kukin prosessi ajettiin yhteensä 10 kertaa, joiden perusteella laskettiin keskiarvo kunkin prosessin kestolle. Kuviin valittiin niiden ajokertojen mittaustulokset, joissa prosessien kesto oli lähimpänä testausvaiheen keskiarvoja. Jokaisen testausvaiheen yksittäisessä tuote-ehdotusprosessissa tehtiin 6 tuote-ehdotusta.

Kuva 24 sisältää ensimmäisen testausvaiheen testitulokset. Koneoppimismallin opetusprosessiin kulunut aika oli 25 ms. Tuote-ehdotusprosessin alussa ladataan opetettu malli järjestelmän käyttöön, jotta tuote-ehdotuksia voidaan tehdä. Tähän kului aikaa 1 ms. Varsinaiseen tuote-ehdotusprosessiin käytettiin 11 ms ja tuote-ehdotusten näkyviin saamiseen kului kokonaisuudessaan 37 ms. Mainittu 37 ms sisältää sekä tuote-ehdotusten tekemisprosessiin, että HTML-koodin luomiseen kuluneet ajat.



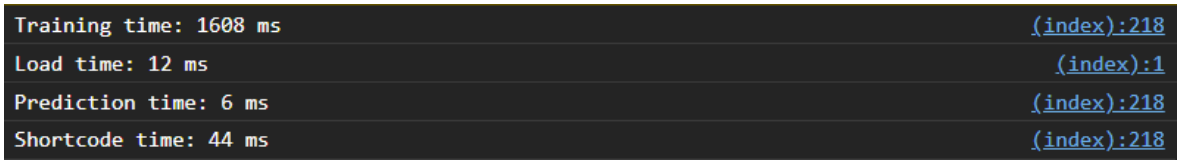
Kuva 24. Eri prosesseihin kuluneet ajat millisekunteina ensimmäisessä vaiheessa

Kuvassa 25 nähdään, että toisessa testausvaiheessa koneoppimismallin opettamiseen kului noin 424 ms. Tämä on noin 17 kertaa enemmän kuin ensimmäisessä testausvaiheessa vastaavaan prosessiin käytetty aika. Muihin prosesseihin ei kulunut merkittävästi enempää aikaa kuin aiemmin, ja tuote-ehdotusten tekemiseen ja näytteille asettamiseen käytettiin jopa hieman vähemmän aikaa. Ensimmäisen ja toisen vaiheen erot olivat kuitenkin opetusprosessia lukuun ottamatta melko vähäisiä.



Kuva 25. Eri prosesseihin kuluneet ajat millisekunteina toisessa vaiheessa

Kuvassa 26 nähdään, että kolmannessa testausvaiheessa koneoppimismallin opettamiseen kului aikaa 1608 ms, mikä on noin 4 kertaa enemmän kuin toisessa vaiheessa. Koneoppimismallin lataamiseenkin käytettiin noin 10 ms enemmän aikaa kuin aiemmissa vaiheissa, joten kyseisen prosessin kesto kasvaa tuotteiden ja käyttäjien määrän kasvun myötä. Tuote-ehdotusten tekemisen kestossa taas ei edelleenkään ollut nähtävissä merkittävää kasvua.



Kuva 26. Eri prosesseihin kuluneet ajat millisekunteina kolmannessa vaiheessa

Tuloksista nähdään, että opetusprossin kesto hidastuu selvästi, kun käsiteltävän datan määrä kasvaa. Sama voidaan huomata koneoppimismallin lataamisprosessin osalta, vaikka siihen käytetty aika kasvoikin melko maltillisesti. Tähän olisi kuitenkin hyvä kehittää jokin ratkaisu, sillä mallin lataamiseen käytettävän ajan osuus tuote-ehdotusten näytteille asettamisen kokonaiskestosta vaikuttaa olevan merkittävä etenkin suuremmissa verkkokaupoissa.

Tulokset osoittavat myös, että tuote-ehdotusjärjestelmä skaalautuu melko hyvin, sillä tuote-ehdotusten tekemiseen kuluneessa ajassa ei ollut nähtävissä merkittävää kasvua. Ehdotettu koneoppimismalli saattaa siis soveltua selvästi suurempiinkin verkkokauppoihin, joissa käyttäjien ja tuotteiden määrä on huomattavasti testauksessa käytettyä suurempi. Tämän varmistamiseksi tarvitaan kuitenkin lisää testejä suuremmalla määrällä dataa.

6 Yhteenveto ja pohdinta

Opinnäytetyön tavoitteena oli luoda tuote-ehdotusjärjestelmä, joka voitaisiin ottaa käyttöön missä tahansa WooCommerce-verkkokaupassa. Järjestelmältä edellytettiin mukautumiskykyä eri verkkokauppojen tarpeisiin. Lisäksi järjestelmän käyttöönoton tuli olla mahdollisimman helppoa ja nopeaa, ja sen suorituskykyyn oli kiinnitettävä huomiota.

Tuote-ehdotusjärjestelmästä tehtiin WordPress-lisäosa, jotta sen asentaminen eri WooCommerce-projekteihin olisi mahdollisimman helppoa. Lisäksi järjestelmään luotiin asetussivut, joiden kautta voidaan muun muassa vaihtaa käytössä olevaa koneoppimismallia ja hienosäätää koneoppimismallien algoritmien toimintaa. Näin järjestelmästä saatiin mukautumiskykyinen, mikä auttaa tuote-ehdotusprosessin optimoinnissa eri verkkokauppoihin.

Työssä esitellyssä koneoppimismallissa järjestelmän suorituskyky pyrittiin huomioimaan vähentämällä mallin opettamiseen ja tuote-ehdotusten tekemiseen käytettävän tuote-käyttäjä-matriisin ulottuvuuksia RSVD-menetelmällä. Näin tuote-ehdotukset voitiin laskea vähemmästä määrästä dataa. Opettamisen aikana tiivistetyn datan pohjalta luotiin binääripuu, josta tuote-ehdotuksia kyettiin hakemaan tehokkaasti. Binääripuun hallinnointi tehtiin KDT-algoritmin avulla, ja tuote-ehdotusten sopivuus laskettiin tuote-ehdotusprosessin aikana CS-menetelmällä.

Tuote-ehdotusjärjestelmää testattiin lopuksi WordPress-testiympäristössä. Ensin selvitettiin, ovatko tuote-ehdotukset ylipäätään tarkoituksenmukaisia. Tämä testattiin vertaamalla koneoppimismallin tekemiä tuote-ehdotuksia tuotteisiin, joita tietokantaan tallennettuja ostotapahtumia loogisesti analysoimalla pidettiin hyvinä ehdotuksina. Järjestelmän todistettiin kykenevän tekemään järkeviä tuote-ehdotuksia. Niiden tiedetään kuitenkin riippuvan melko paljon käytetyn koneoppimismallin algoritmien hienosäädöistä ja niiden suhteesta esimerkiksi tuote-käyttäjä-matriisin kokoon. Niinpä tuote-ehdotusten laatu voi vaihdella eri verkkokauppojen välillä paljonkin.

Lisäksi testattiin, kuinka paljon tuote-ehdotusjärjestelmä ja työssä esitelty koneoppimismalli vaikuttavat WordPress-sivuston suorituskykyyn. Testeissä kyseisen mallin tuote-ehdotusprosessin suorituskyvyn huomattiin pysyvän varsin hyvänä, vaikka tuotteiden ja käyttäjien määrä kasvoikin huomattavasti. Ehdotettua ratkaisua voidaan siis pitää kohtuullisen skaalautuvana. Lisäksi suorituskykytesteissä ilmeni, että opetetun koneoppimismallin lataaminen tiedostosta järjestelmän käyttöön vie huomattavan paljon aikaa suhteessa tuote-ehdotusprosessin kokonaiskestoon.

Testitulosten perusteella tuote-ehdotusjärjestelmän voidaan todeta olevan hyvä pohja jatkokehitykselle, sillä kaikki suunnitellut ominaisuudet saatiin toimimaan, ja uusien

koneoppimismallien lisääminen järjestelmään on pyritty tekemään helpoksi. Testivaiheessa järjestelmän toiminnassa kuitenkin ilmeni kehityskohde, joka on syytä ratkaista ennen sen käyttöönottoa. Tällä hetkellä tuote-ehdotusjärjestelmä voidaan kuitenkin katsoa soveltuvaksi ainakin pieniin verkkokauppaprojekteihin, joissa tuote-ehdotusprosessin kesto osoitettiin melko vähäiseksi. Selvästi suuremmissa verkkokaupoissa tuote-ehdotusjärjestelmä kuitenkin hidastaa verkkosivun lataamisprosessia enemmän, joten järjestelmän suorituskykyä kannattaa vielä kehittää käyttökokemuksen ohella.

Suorituskykyä voitaisiin parantaa poistamalla käytössä olevan koneoppimismallin lataamisen tarve tiedostosta jokaisen tuote-ehdotusprosessin yhteydessä. Malli voitaisiin vaikkapa jakaa rajapinnan kautta jollakin palvelinohjelmistolla heti mallin opetuksen jälkeen. Tällöin mallia voitaisiin käyttää tuote-ehdotusten tekemiseen, kunnes malli jouduttaisiin opettamaan uudestaan. Opetuksen jälkeen vanha versio mallista poistettaisiin ja päivitetty versio jaettaisiin järjestelmän käyttöön.

Tuote-ehdotusjärjestelmän käyttökokemusta taas voitaisiin kehittää suorituskyvyn ohella hakemalla tuote-ehdotukset asynkronisesti vasta sen jälkeen, kun verkkosivu olisi muuten ladattu. Näin käyttäjät pääsisivät käyttämään sivustoa nopeasti, vaikka tuote-ehdotusprosessi kestäisikin hetken. Järjestelmään voitaisiin myös tehdä erilaisia koneoppimismalleja, joista kukin olisi optimoitu erilaisiin verkkokauppoihin, mikä osaltaan voisi parantaa järjestelmän suorituskykyä ja käyttökokemusta.

Lähteet

Aljunid, M. & Huchaiah, M. 2021. An efficient hybrid recommendation model based on collaborative filtering recommender systems. CAAI Transactions on Intelligence Technology. Vol. 6 (4), 480–492. Viitattu 10.2.2022. Saatavissa https://doi.org/10.1049/cit2.12048

Automattic. 2022. Build exactly the eCommerce website you want. Viitattu 5.3.2022. Saatavissa https://woocommerce.com/

Bentley, J. 1975. Multidimensional Binary Search Trees Used for Associative Searching. Communications of the ACM. Vol. 18 (9), 509–517. Viitattu 15.2.2022. Saatavissa https://www.academia.edu/1811956/Multidimensional_binary_search_trees_used_for_associative_searching

Bobadilla, J., Ortega, F., Hernando, A. & Gutiérrez, A. 2013. Recommender systems survey. Knowledge-Based Systems. Vol. 46, 109–132. Viitattu 28.1.2022. Saatavissa https://doi.org/10.1016/j.knosys.2013.03.012

Composer. a. Introduction. MIT-lisenssi. Viitattu 22.2.2022. Saatavissa https://getcomposer.org/doc/00-intro.md#system-requirements

Composer. b. Command-line interface / Commands. MIT-lisenssi. Viitattu 23.2.2022. Saatavissa https://getcomposer.org/doc/03-cli.md

Composer. c. Basic usage. MIT-lisenssi. Viitattu 22.2.2022. Saatavissa https://getcomposer.org/doc/01-basic-usage.md

Deerwester, S., Dumais, S., Furnas, G., Landauer, T. & Harshman, R. 1990. Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science. Vol. 41 (6), 391–407. Viitattu 8.2.2022. Saatavissa https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/1hujjmv/cdi_crossref_primary_10_1002__SICI_1097_4571_199009_41_6_391__AID_ASI1_3_0_CO_2_9

Friedman J., Bentley, J. & Finkel, R. 1977. An Algorithm for Finding Best Matches in Logarithmic Expected Time. ACM Transactions on Mathematical Software. 1–29. Viitattu 15.2.2022. Saatavissa https://www.academia.edu/19017931/An_Algorithm_for_Finding_Best_Matches_in_Logarithmic_Expected_Time

GitHub. 2021. Scien-ide/Tensor. CC 4.0 -lisenssi. Viitattu 22.2.2022. Saatavissa https://github.com/Scien-ide/Tensor

Google. 2020. Matrix Factorization. CC 4.0 -lisenssi. Viitattu 8.2.2022. Saatavissa https://developers.google.com/machine-learning/recommendation/collaborative/matrix

Google. 2021. Collaborative Filtering. CC 4.0 -lisenssi. Viitattu 3.2.2022. Saatavissa https://developers.google.com/machine-learning/recommendation/collaborative/basics

Hidasi, B., Karatzoglou, A., Baltrunas, L. & Tikk, D. 2016. Session-Based Recommendations with Recurrent Neural Networks. Konferenssijulkaisu. 1–10. Viitattu 3.4.2022. Saatavissa https://doi.org/10.48550/arXiv.1511.06939

Li, H., Zhang, S. & Wang, X. 2013. A personalization recommendation algorithm for e-commerce. Journal of Software. Vol. 8 (1), 176–183. Viitattu 28.1.2022. Saatavissa http://www.jsoftware.us/vol8/jsw0801-23.pdf

Li, L., Zhang, Z., Zhang, S. & Jiang, Y. 2021. Hybrid Algorithm Based on Content and Collaborative Filtering in Recommendation System Optimization and Simulation. Scientific programming. Vol. 2021, 7427409 (artikkelin ID), 1–11. Viitattu 2.2.2022. Saatavissa https://doi.org/10.1155/2021/7427409

Lipschutz, S. & Lipson, M. 2009. Schaum's Outline of Linear Algebra Fourth Edition. E-kirja. McGraw Hill Professional. Viitattu 12.3.2022. Saatavissa https://books.google.fi/books?id=RsPqtjiW50YC&printsec=frontcover&dq=isbn:9780071543521&hl=fi&sa=X&redir_esc=y#v=onepage&q=dot%20product&f=false

PHP Group. a. General Information. Viitattu 21.2.2022. Saatavissa https://www.php.net/manual/en/faq.general.php

PHP Group. b. What is PHP?. Viitattu 21.2.2022. Saatavissa https://www.php.net/manual/en/intro-whatis.php

PHP Group. c. What can PHP do?. Viitattu 21.2.2022. Saatavissa https://www.php.net/manual/en/intro-whatcando.php

PHP Group. d. Install Requirements. Viitattu 21.2.2022. Saatavissa https://www.php.net/manual/en/install.windows.requirements.php

PHP Group. e. Downloads. Viitattu 21.2.2022. Saatavissa https://windows.php.net/download/

PHP Group. f. Downloading PECL extensions. Viitattu 23.2.2022. Saatavissa https://www.php.net/manual/en/install.pecl.downloads.php

PHP Group. g. Getting and installing the PEAR package manager. Viitattu 23.2.2022. Saatavissa https://pear.php.net/manual/en/installation.getting.php

PHP Group. h. Installing a PHP extension on Windows. Viitattu 23.2.2022. Saatavissa https://www.php.net/manual/en/install.pecl.windows.php

PHP Group. 2020. What is PECL?. Viitattu 23.2.2022. Saatavissa https://pecl.php.net/

Ren, L., Gu, J. & Xia, W. 2010. An Item-Based Collaborative Filtering Algorithm Utilizing the Average Rating for Items. Teoksessa Kim, T., Pal, S., Grosky,W., Pissinou, N., Shih, T. & Ślęzak, D. (toim.) Signal Processing and Multimedia. Communications in Computer and Information Science. E-kirja. Berlin: Springer, 175–183. Viitattu 28.1.2022. Saatavissa https://doi.org/10.1007/978-3-642-17641-8_22

Rubix ML. 2021a. Cosine. CC 4.0 -lisenssi. Viitattu 18.2.2022. Saatavissa https://docs.rubixml.com/1.0/kernels/distance/cosine.html

Rubix ML. 2021b. Rubix ML. CC 4.0 -lisenssi. Viitattu 22.2.2022. Saatavissa https://docs.rubixml.com/1.0/index.html

Rubix ML. 2021c. Installation. CC 4.0 -lisenssi. Viitattu 22.2.2022. Saatavissa https://docs.rubixml.com/1.0/installation.html

Rubix ML. 2021d. TruncatedSVD. CC 4.0 -lisenssi. Viitattu 3.4.2022. Saatavissa https://docs.rubixml.com/1.0/transformers/truncated-svd.html#fnref:1

Rubix ML. 2021e. K-d neighbors. CC 4.0 -lisenssi. Viitattu 3.4.2022. Saatavissa https://docs.rubixml.com/1.0/classifiers/kd-neighbors.html

Rubix ML. 2021f. K-d Tree. CC 4.0 -lisenssi. Viitattu 4.3.2022. Saatavissa https://docs.rubixml.com/1.0/graph/trees/k-d-tree.html

Rubix ML. 2021g. PersistentModel. CC 4.0 -lisenssi. Viitattu 4.3.2022. Saatavissa https://docs.rubixml.com/1.0/persistent-model.html

Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. 2000. Application of Dimensionality Reduction in Recommender System – A Case Study. Minnesota Univ Minneapolis Dept of Computer Science. Tekninen raportti. 1–15. Viitattu 10.2.2022. Saatavissa https://apps.dtic.mil/sti/citations/ADA439541

Wang, X., Chen, J. & Yu, J. 2017. Optimised quantisation method for approximate nearest neighbour search. Electronics letters. Vol. 53 (3), 156–158. Viitattu 10.2.2022. Saatavissa https://doi.org/10.1049/el.2016.2810

WordPress. a. Democratize Publishing. Viitattu 5.3.2022. Saatavissa https://wordpress.org/about/

WordPress. b. Features. Viitattu 5.3.2022. Saatavissa https://wordpress.org/about/features/

WordPress. c. Get WordPress. Viitattu 21.2.2022. Saatavissa https://wordpress.org/download/

WordPress. d. How to install WordPress. Viitattu 5.3.2022. Saatavissa https://wordpress.org/support/article/how-to-install-wordpress/

WordPress. e. Hooks. Viitattu 3.3.2022. Saatavissa https://developer.wordpress.org/plugins/hooks/

WordPress. f. Cron. Viitattu 2.3.2022. Saatavissa https://developer.wordpress.org/plugins/cron/#:~:text=What%20is%20WP-Cron%20%23%20What%20is%20WP-Cron.%20WP-Cron,scheduling%20system%20that%20is%20available%20on%20UNIX%20systems

Zhang, F., Sun, S. & Yi, H. 2015. Robust collaborative recommendation algorithm based on kernel function and Welsch reweighted M-estimator. IET information security. Vol. 9 (5), 257–265. Viitattu 28.1.2022. Saatavissa https://doi.org/10.1049/iet-ifs.2014.0488