CHALMERS

Spelrekommendationssystem med maskininlärning

Kollaborativ filtrering

Examensarbete inom Data- och Informationsteknik

Robert Felczak
Rami Jabor

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2018

EXAMENSARBETE

Spelrekommendationssystem med maskininlärning

Kollaborativ filtrering

Robert Felczak
Rami Jabor

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg 2018

Spelrekommendationssystem med maskininlärning
Kollaborativ filtrering
Robert Felczak
Rami Jabor

Examinator: Jonas Duregård

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telefon: 031-772 1000

Institutionen för Data- och Informationsteknik
Göteborg 2018

SAMMANFATTNING

Vid ett tidsskede där människor översvämmas av överflöd av information, ökar behovet av rekommendationssystem/filtrering varje dag. Detta gäller även tv-spel där nya spel släpps varje dag och det blir svårare att hitta spel som är till ens intressen. Det finns många olika filtreringstekniker och en av dem är kollaborativ filtrering (Collaborative Filtering) som just detta examensarbetet fokuserar på. Syftet är att utforska möjligheterna med kollaborativ filtrering för företaget *8 Dudes in A Garage AB*s rekommendationssystem, som läggs in på hemsidan IGDB.com (Internet Game DataBase). Flera tester har genomförts som består av att träna upp ett antal modeller från ramverket Graphlab och sedan utvärdera hur bra rekommendationer modellerna ger. Datamängderna som testerna görs på är framtagen från IGDB och Amazon. Målet var att testa den optimala rekommendationssystemsmodellen med användare på hemsidan, men det gjordes inte inom den givna tidsramen. Resultaten visar att modellen ItemSimilarity gav bäst rekommendationer på båda dataseten. Dock gav testerna i överlag svaga värden i evalueringen, vilket kan bero på en rad problem med dataseten som är typiska för den kollaborativa filtreringstekniken. Rapporten går även igenom problem som inte visades i evalueringen och potentiella lösningar på dessa problem som företaget kan implementera. Sammanfattningsvis visar rapporten

1. På vilka sätt företaget skulle gynnas av kollaborativ filtrering.
2. Att med hjälp av ramverket Graphlab, skulle det inte vara svårt att implementera ett rekommendationssystem till hemsidan.
3. Att ett hybridsystem skulle kunna ses som en lösning till att få ett optimalt rekommendationssystem.

Nyckelord: videospel, kollaborativ filtering, rekommendationssystem, maskininlärning

ABSTRACT

In an age of information overflow, the demand for filtering increases every day. This is also the case for videogames, where new games are released everyday. There are many filtering techniques and one of them is Collaborative Filtering which this thesis focuses on. The purpose is to explore the possibilities of collaborative filtering for the company *8 Dudes in A Garage AB*s Recommendation System. The tests performed consist of training a amount of recommendation system models with the Turi Graphlab Framework and evaluate how good the recommendations are for each model. The Datasets that the tests are performed on are taken from the company and Amazons Videogame Category. The goal was to also test the optimal model on the company's website with the community users, but it was not able to be done due to the projects timelimit. The results show that the ItemSimilarity model gave the best values in evaluations, even though the results gave low values on average for all models. The report goes through the problems these results indicate and other problems that were not shown in the results. Suggested solutions that the company could implement in future endeavours with their recommendation system are included.
In conclusion the report shows:

1. In what ways the company would benefit from Collaborative Filtering.
2. That with the Framework Graphlab these changes wouldn’t be hard to implement to the company website.
3. That a hybridsystem would be a solution towards an optimal Recommendation System.

Keywords: Videogames, Collaborative Filtering, Recommendation Systems, Machine Learning,

FÖRORD

Denna rapport är vårt examensarbete på högskoleingenjörsprogrammet Datateknik (180 hp) vid Chalmers Tekniska Högskola, som utfördes under 20 veckor under vårterminen 2018 på halvtid.

Vi vill tacka vår handledare Erland Holmström för hans goda råd och tips.
Vi tackar företaget *8 Dudes in A Garage* för att ha gett oss möjligheten att utföra detta projekt som vårt examensarbete.

Innehållsförteckning

1.Inledning

Videospelsindustrin växer ständigt där den årliga omsättningen idag är vid 108.9 miljarder dollar, vilket är dubbelt så stort som filmindustrin [1][2][3]. Med hundratals nya spel som släpps ut varje månad, så blir det svårt för användare att hitta spel som passar deras intressen [4]. För att undvika att bli överbelastad med information, så åtgärdas detta genom att få rekommendationer. För att bättre rekommendera spel behövs ett automatiserat rekommendationssystem som baserat på en stor mängd data kan se trender, mönster och likheter mellan egenskaper och preferenser.
Dagens moderna rekommendationssystem använder sig av maskininlärning för att bättre anpassa sina rekommendationer utefter användarens återkoppling och analys av stora mängder data. Många företag som Netflix, Amazon och Youtube har med maskininlärning utvecklat avancerade rekommendationssystem som är specialanpassade åt deras tjänster [6]. Detta examensarbetet fokuserar på rekommendationssystem för videospel.

1.1 Bakgrund

Examensarbetet är ett projekt hos företaget 8 Dudes in A Garage AB som har en hemsida, Internet Game DataBase (IGDB). IGDB är ett web community där användare kan utforska, diskutera och recensera bland över 90,000 videospel som finns tillgängliga i deras databas [7]. Företaget vill uppdatera sin rekommendationstjänst på hemsidan, för att ge användare mer passande rekommendationer på spel och lättare tillgång till att utforska nya spel. De har provat att implementera sitt rekommendationssystem med olika metoder, men en metod som inte har testats är kollaborativ filtrering som bl.a. Youtube, Amazon och Netflix använder för sina rekommendationssystem [6]. Företaget är intresserade om tekniken går att implementera, men de spekulerar att deras data/datamängd är otillräcklig för att kunna göra bra rekommendationer. Företaget ser en lösning på problemet genom att använda större datamängd med användarrecensioner från andra företag för att rekommendera spel i deras databas.

1.2 Syfte

Syftet med projektet är att utforska om kollaborativ filtrering är en bra filtreringsteknik för företagets platform, som potentiellt kan förbättra rekommendationer på företagets hemsida. Fokuset ligger i att hitta rekommendationer till spel, med hjälp av filtreringstekniker, baserat på trender i användarnas spelpreferenser/recensioner.

1.3 Mål

Målet med detta examensarbete är att utveckla en prototyp av en rekommendationstjänst för företagets hemsida IGDB.com. Prototypen ska kunna rekommendera ett spel som liknar ett annat spel m.h.a användarnas spelpreferenser, implementerad med datamängd från företagets databas och ett tredje parts bibliotek. Andra dataset kommer att utforskas för att se om rekommendationerna blir bättre än enbart med företagets egna data och därmed kan användas i företagets rekommendationssystem. Prototypen ska kunna användas på företagets hemsida och ska vara lätthanterlig, så att företaget enkelt kan vidareutveckla systemet. Systemet ska slutligen testas med ett urval av användare från IGDB:s hemsida, som bedömer hur bra rekommendationerna är i mån av tid.

1.4 Avgränsningar

Avgränsningarna kommer att utgå ifrån att författarna inte har haft tidigare kunskaper om områdena maskininlärning och rekommendationssystem. För att kunna utföra arbetet och få fram ett resultat inom arbetets tidsram, har följande avgränsningar satts upp:

1. Inlärning av området maskininlärning och rekommendationssystem kommer att göras i samband med exempel i områdena och inlärning av ramverk som finns tillgängliga. Arbetet kommer inte fördjupa sig in i algoritmer inom rekommendationssystem.
2. Användning av ett redan färdigutvecklat ramverk/bibliotek för implementering av rekommendationssystem, där utvecklare inte behöver stora kunskaper om maskininlärning.
3. Användning av befintliga datamängder som rekommendationssystemen ska tränas upp med. Motivering till det är att existerande datamängd är redan strukturerade för att användas, medan arbetet med extrahering, analys och strukturering av en datamängd skulle påverka den planerade tidsramen negativt.
4. Användargränssnitt eller presentation av rekommendationssysten kommer inte behandlas i detta arbetet.
5. Rekommendationer sker baserat på spelet man tittar på nu och ska kunna ges till användare som inte är inloggade. Därför ska rekommendationerna inte skilja sig från användare till användare.

2.Teori

Detta kapitel tar upp tidigare försök för konstruktion av rekommendationstjänsten. Beskrivning av rekommendationssystem som filtreringstekniker, problem som kan uppstå och evaluering av rekommendationssystem och dess betydelse tas också upp.

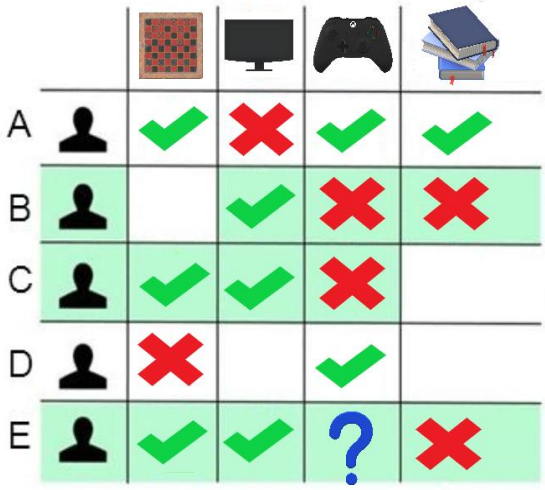
2.1 Rekommendationssystem (RS)

Rekommendationssystem är mjukvara som rekommenderar objekt, som har en stor sannolikhet att vara av intresse för en specifik användare. Ett RS fokuserar på en specifik typ av objekt som t.ex. ett spel, filmer, varor, men kan även ha blandade typer av varor och objekt. Baserat på likheter mellan spelegenskaper, användarnas preferenser i spel eller ytterligare kontextuell data, har spelen en rankning i sannolikheten att bli rekommenderade. De tekniker som används mest inom RS är kollaborativ filtrering och innehållsbaserad filtrering [8].

För att RS ska kunna generera rekommendationer behöver den lära sig mönster från ett dataset. Ett dataset ska först förberedas innan d.v.s. restriktioner och regler av indatan ska struktureras för modellen. Datasetet delas sedan upp till två delar, ett träningsset och ett valideringsset. Träningssetet används till att "träna" modellen med olika algoritmer för att kunna se mönster mellan värden och parametrar. Valideringssetet används för att testa hur pricksäker modellens förutsägelser av rekommendationer är, där utdatan av modellen evalueras [8].

Collaborative filtering

Collaborative filtering är en filtreringsteknik som används inom rekommendationssystem, som rekommenderar objekt till en användare baserat på data från andra användares intressen. Det finns två olika metoder inom tekniken, User-based och Item-based.

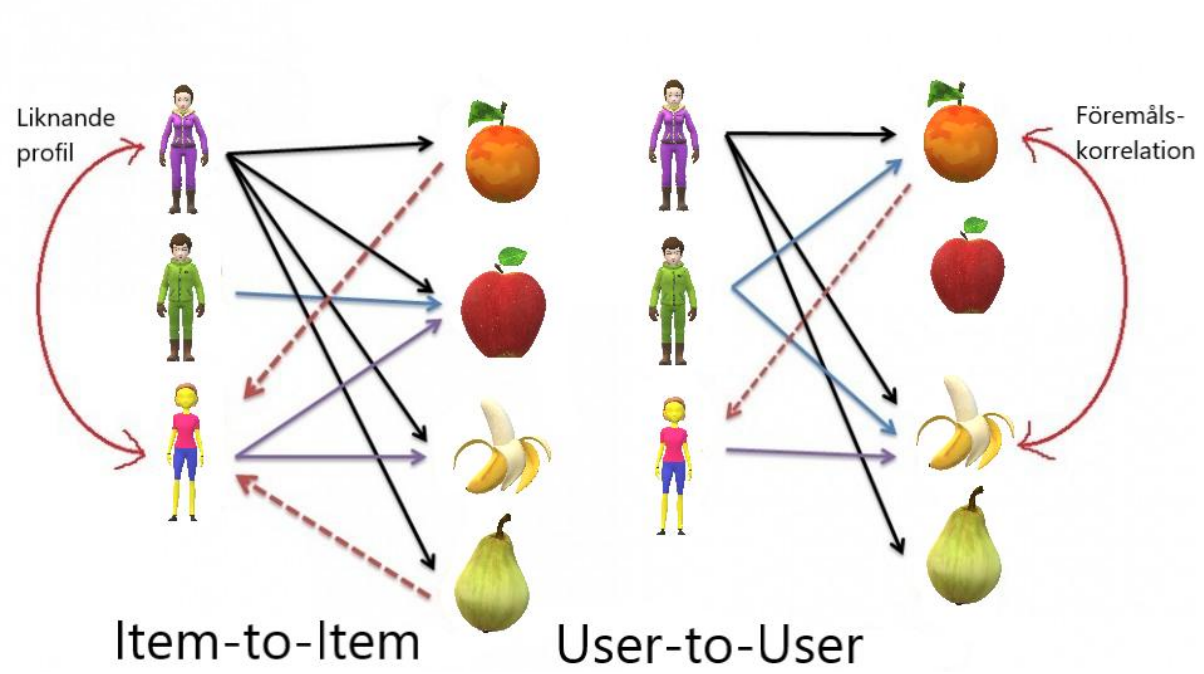


Figur: Exempel på user-based filtrering. E har liknande profil som B & C, där frågetecknet är objekt som förutses. RS:et kommer att markera objektet med kryss(inte rekommenderad), baserat på användare B & C.

Likheter i preferenser bedöms utifrån hur lika olika användares bedömningar varit för ett antal objekt (User-to-User). Ju mer användare det finns med liknande recensionprofiler som har recenserat ett spel, desto högre värde kan det spelet få som en rekommendation. Detta värdet kan sedan rankas med andra spel, där spel med högst värde rankas först. Om ett spel har många positiva recensioner från användare, som även recenserat ett annat spel positivt, så har en av dessa två spelen stor chans att bli rekommenderade till användare beroende på vilket spel man tittar på vid ett tillfälle.

Rankingen för rekommendationer bestäms på antalet användare med liknande preferenser som har bedömt fokus-objektet och rekommendationer-objektet och betygen. Det är denna tekniken som arbetet är fokuserat på och den kallas även “Nearest Neighbor method”.

Istället för att rekommendera baserat på liknande recension profiler, använder Amazon, en av världens största nätbutiker, en annorlunda teknik (Item-to-Item) för kollaborativ filtrering. Tekniken kollar korrelationer mellan objekt som har köpts tillsammans vid ett tillfälle, där det ena objektet rekommenderas till det andra. Fördelen med tekniken är att korrelationerna tar mindre tid att räkna ut, då alla användare inte behöver jämföras med varandra. En modell som använder tekniken behöver inte uppdateras lika ofta eftersom dessa trender inte ändras så signifikant [9][10].



Figur: Hela linjer markerar vad användaren har gillat, böjda linjer markerar vad systemet har bedömt är lika och streckade linjer markerar rekommendationer baserat på likheten. Till vänster bedömer ett User-baserat filtreringssystem att två användare har liknande preferenser och gör rekommendationer baserat på det. Till höger bedömer ett Item-baserat filteringsystem att två föremål har en korrelation, då andra användare som gillat ett av föremålet har gillat det andra med. Baserat på denna liknelsen gör systemet en rekommendation.

Content based filtering

Content Based Filtering (CBF), som också kallas för Innehålls-baserad filtrering, är en teknik som rekommenderar objekt utifrån likheter på egenskaper och attribut till andra objekt. CBF analyserar spelattribut och kräver djup kunskap samt mycket information om spelet. CBF behöver inte information från användarprofiler då all filtrering sker genom spelens attribut. T.ex. om en användare sökt på spel som har genre “racing”, så kan rekommendationssystemet

lära sig att rekommendera andra spel som har samma genre. Stora nackdelen med CBF är kravet på tillräckligt information om spelen och deras attribut [19].

2.2 Vanliga problem med rekommendationssystem

Ett rekommendationssystem kan bete sig olika beroende på datamängden som används och vilken teknik som används. Olika problem och begränsningar kan uppstå med rekommendationssystem som kan ge oväntade konsekvenser som irrelevanta rekommendationer. Problemen som nämns i kapitlet kommer att tas upp i Diskussion, om vilken påverkan det kan ha på resultatet och hur problemen uppstår.

Brist på data

Bland de största problemen i rekommendationssystem är kravet på stor mängd data, främst i kollaborativ filtrering. Rekommendationssystem kräver stora mängder data på både recensioner och spelegenskaper för att göra precisa rekommendationer. Brist på data kan leda till att det blir svårt för systemen att hitta liknande profiler med kollaborativ filtreringsteknik. Brist på data med content based filtering tekniken kan leda till irrelevanta kopplingar i egenskaper/attribut [13].

En ideal datamängd för collaborative filtering innehåller mer än tre recensioner från varenda användare, där alla spel har minst tre recensioner var. Anledningen till varför det skulle vara idealt med minst 3 recensioner på varje spel är att enstaka recensioner inte väger över rekommendationer och att rekommendationer inte sker helt och hållet baserat på en recension.

En ideal datamängd för content based filtering innehåller många relevanta egenskaper varav en del egenskaper är sammankopplade i likhet. Sammankoppling i likhet skulle kunna vara att egenskapen “utspelar sig i Italien” är kopplat till egenskapen “Utspelar sig i Rom” eller “Innehåller Italiensk mat”.

För mycket data

Å andra sidan kan för mycket data bli ett problem för rekommendationssystem att bearbeta inom en rimlig tid och kan leda till favorisering av gamla spel med många recensioner, se nästa rubrik Cold-start. Storföretag med stor mängd data har löst detta med att samla datamängden i clusters och endast söka i enskilda clusters istället för hela datamängden [13].

Cold-start

Ett vanligt problem med kollaborativ filtrering är att nya spel ofta saknar recensioner då folk inte har recenserat spelet än. Bristen på recensioner gör att spelet inte kan dyka upp i rekommendationerna. Detta problem uppstår alltid då ett nytt spel dyker upp som har för lite interaktionsdata (Recensioner) och försvinner med tiden då data från användare tillkommer. Problemet blir värre om spelet i förhand inte är så populärt, i många fall indiespel som ofta har liten marknadsföring, eftersom detta då leder till att spelet förblir orecenserat en längre tidsperiod. Detta löses ofta med att blanda teknikerna content-based filtering och kollaborativ filtrering till hybridsystem, där content-based filtering tar över vid cold-start

problem och rekommenderar baserat på egenskaper. En annan metod utöver content-based filtering tekniken är att motivera användare att recensera nya, okända spel med att ha dem i en egen sektion t.ex. “Undiscovered”,”New unknown games” och även ge uppmuntringar/virtuella medaljer/”achievement” för att vara bland dem första att recensera ett spel [8].

Sparsity

En datamängd med stort antal spel och litet antal användarinteraktioner (recensioner) gör att det blir svårt för kollaborativ filtrerings system att göra precisa rekommendationer. Om för många användare har för lite recensioner leder det till att det blir svårt att hitta användare med liknande preferenser. Detta i sin tur ger leder till svaga och irrelevanta rekommendationer [13].

Synonymitet

Förekommer när ett spel eller en spelegenskap har flera duplikater fast under olika namn eller konsoltyper. Systemet kan inte identifiera att det är samma spel eller spelegenskap och behandlar dem olika. Detta delar upp recensionerna och bidrar till sparsity-problemet. Exempel på detta skulle kunna vara “comedy game” och “funny” skulle klassas som två olika egenskaper av rekommendationssystem eller att ett spel har flera versioner för olika spelkonsoler. Detta problemet kan mildras med singulärvärdesuppdelning (SVD) där taggar med synonymitets-problemet kan beräknas fram att vara väldigt lika. T.ex. taggarna "funny" och "hilarious" som är väldigt lika skulle m.h.a. SVD tolkas som snarlika då spelen med dessa taggarna skulle ofta hamna i samma sammanhang. Dessa sammanhang skulle kunna vara att spel med dessa taggarna får recensioner att de varit roliga eller att spelen hamnar ofta i samma kategori "komedi"[13].

Grey Sheep

Användare som har preferenser som avviker från majoriteten av en grupp får liten nytta av kollaborativ filtrerings system. Detta kan lösas genom content-based filtering då rekommendationer kan ske på användarens preferenser i spelegenskaper. Identifiering av så kallade grå får kan göras med clustering [8]. Clustering är en teknik som innebär att gruppera en datamängd till delmängder, clusters, av data med likheter. Detta möjliggör sedan för att se trender, grupperingar och avvikelser i datamängden. Det underlättar även arbetet för system då de istället för att gå igenom hela datamängden kan bearbeta delmängder [11].

Shilling-attacker/Fake Reviews

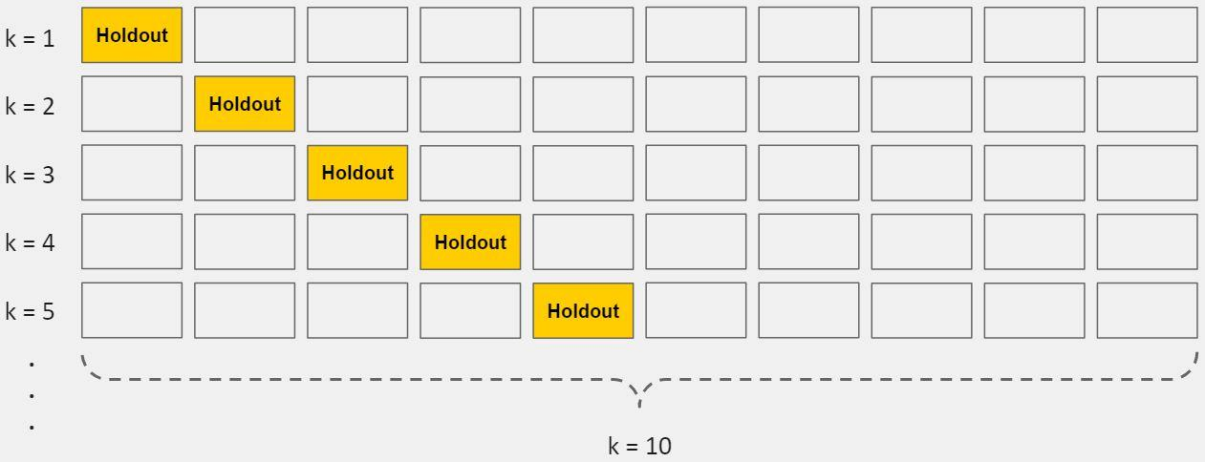
Shilling-attacker är attacker på spel i form av negativa recensioner i större skala. Attackerna kan utföras av en grupp människor med mobbmentalitet eller med ett stort antal elakartade användar-bots som skapar automatiserade recensioner. Syftet är då att skada försäljningen och populariteten av ett visst spel och dess spelföretag [13][14][16]. Det finns även fall där företag försökt öka sitt spels popularitet genom att skriva falska positiva recensioner [15].

Lösning är att moderera bort och bestraffa sådan manipulation samt aktivt spåra plötsliga negativa/positiva trender i recensioner. En lösning av Steam, ett digitalt distributions platform

för bl.a. spel, har varit att sätta histogram på recensioner så användare själva kan se plötsliga negativa trender [17]. Samt att ge möjligheten till spelutvecklare att rapportera recensioner som bryter mot Steams riktlinjer.

Overfitting

Ett problem som kan dyka upp under upplärningen av datamodellen är att modellen kan överanalysera sitt upplärningsdataset d.v.s. träningssetet, fokusera för mycket på störningar och bli för anpassat på träningssetet. Detta leder till att modellen sedan inte kan förutse andra dataset lika bra. En vanlig lösning till detta är användning av K-Fold korsvalidering där datamängden delas upp till *k* antal subset där *k* antal träningar görs med alla subset som träningsset i tur och ordning [18].



Figur: Holdout representerar valideringsdatan och de andra boxarna representerar träningsdatan. Genom att variera valideringsdatan och träningsdatan så hindrar man överanpassning.

2.3 Evaluering av rekommendationssystem

Ett rekommendationssystem (RS) är användbart om ett system kan rekommendera ett spel relevant för en användare vilket skulle ge en hög trovärdighet för systemet. Men ett RS som ger dåliga rekommendationer kan få en användare uppleva RS:et som opålitligt [8]. En dålig rekommendation kan ofta vara sämre än ingen rekommendation alls, då konsekvensen blir ett minskat förtroende för systemet. En evaluering av systemet krävs för att veta om systemet är tillräckligt pricksäker med sina rekommendationer eller inte.

RS går att evaluera genom mätvärden, bl.a. RMSE (root-mean-square-error), precision, recall (återkallelse) och F1-score. Mätvärdena används inte alltid samtidigt utan beror på vilken typ av filtreringsteknik som används och vilket syfte rekommendationssystem har [19].

RMSE

Enligt Ricci et. al. [8] så är RMSE troligtvis det mest använda mätvärdet för att mäta ett systems pricksäkerhet, där systemet förutser vilka betyg som varje spel får. Mätvärdet används när man är intresserad att få veta hur systemet förutser vilket genomsnittsbetyg ett spel skulle få av en användare.

$$RMSE = \sqrt{\frac{1}{|\tau|}\sum_{(u,i)\in\tau}(\widehat{r_{ui}} - r_{ui})^2}$$

Formel: Värdet visar hur stor felmarginal det är mellan systemets förutsägelse av ett spels betyg och originalbetyget för spelet. Systemet genererar de förutsedda betygen $\widehat{r_{ui}}$ för ett test set τ av användar-objekt paren (u,i) för vilket de sanna betygen r_{ui} är kända. Ju lägre RMSE-värde (ända ner till noll), desto närmare har rekommendationssystemet förutsett ett betyg till originalbetyget i träningssetet[8].

Precision, recall & F1-score

Enligt Ricci et. al. [8] är precision och recall några av de vanligare klassifikations måtten för att evaluera en rekommendationssystems modell. Måtten används främst för att ta reda på hur pricksäkert ett rekommendationssystem förutser att en användare kommer att interagera med ett annat spel, utifrån ett spel som användaren tittar på vid tillfället. De representerar antalet relevanta objekt som har hämtas i förhållande till totala mängden relevanta objekt [8][19].

$$Precision = \frac{True\ Positive}{(True\ Positive + False\ Positive)}$$

$$Recall = \frac{True\ Positive}{(True\ Positive + False\ Negative)}$$

*Formel: **True Positive** är värdet på antal spel som systemet har hittat som är relevanta.*
***False Positive** är värdet på antal spel som systemet har hittat, men som är inte relevanta.*
***False Negative** är värdet på antal spel som systemet inte har kunnat hitta, men som är relevanta.*

Precision visar hur många relevanta spel ett system kan hitta på ett antal gissningar. Recall visar hur många relevanta spel som systemet lyckats hitta av alla möjliga relevanta spel. Om ett system hittar 4 av 10 spel och har gissat 6 gånger, blir precisions värdet *4/6* och recall värdet *4/10*.

$$F1_{score} = \frac{2 * Recall * Precision}{Recall + Precision}$$

F1-score är ett mått för modellens pricksäkerhet, som är det viktade värdet av precision och recall. Måttet används för att förenkla precision och recall till ett och samma mått och underlättar jämförelserna mellan olika modeller [15].

Måtten är binära, vilket betyder att de kan mäta om ett spel är relevant eller inte. Måtten kan inte tolka vad som är relevant om en betygskala på 1-5, eftersom skalan är inte binärt. För att måtten ska kunna bedöma ett spel med numeriska värden, så behövs det sättas upp en gräns på betygsskalen. Värdet över gränsen skulle tolkas som ett relevant spel för måtten, där t.ex. en gräns på 3 på en skala mellan 1-5 skulle tolka ett spel som relevant om betyget på spelet är 3 eller högre.

2.4 Tidigare försök av företaget

Nedan redogörs vad som testats innan av företaget och vad för rekommendationssystem som används på deras hemsida, IGDB.com [20].

NEO4J

Företaget har tidigare testat med NEO4J, ett grafdatabassystem som har en inbyggd rekommendationstjänst, baserad på content-based filtering. Enligt NEO4Js manual så är allt i NEO4Js datastruktur uppbyggt i form av antingen en nod, en nodkoppling eller ett attribut. Attribut kan vara kopplade till en nod eller en nod-koppling [21]. Planerna var då att hitta rekommendationer till användarna varje vecka baserat på deras recensionsprofil med NEO4J. Problemet blev då att system var för långsamt och tog för lång tid för beräkningar.

Jaccard index

Efter NEO4J skapade företaget ett content-based filtrerings rekommendationssystem som filtrerar med Jaccard Index och är företagets nuvarande system. Jaccard Index är en statistisk metod som jämför likheterna mellan elementen i två olika datamängder. Ju mer element som de olika datamängderna delar, desto större likhet har datamängderna till varandra [8].
I systemet används egenskaperna genre, tema och nyckelord på spel för att jämföra likheterna mellan olika spel och rekommendationerna rankas baserat på antal liknelser [20]. Systemet har enligt företaget varit bristande i resultat, då de tyckt att rekommendationerna inte alltid har varit relevanta eller att det finns bättre rekommendationer som inte kommer med.

Kinrate

Senaste uppdateringen till hemsidan IGDB.com rekommendationssystem är Kinrate, ett tredjeparts Application Programming Interface (API). Kinrate integreras till hemsidan om tjänsten Kinrate har med spelet i fråga i sitt API, där IGDB.com ger rekommendationer baserat på Kinrates API [23]. Kinrate är ett finskt universitetsprojekt som baserar sina rekommendationer på 18 personlighetsfrågor och hänvisar till en rapport om personlighetspreferenser i videospel [24]. Deras tjänst fungerar bättre om en användare svarar på personlighetsfrågorna och tillför favoritspel. Men på hemsidan ska inte frågorna ställas till användare utan bara rekommendera spel som är lika till andra spel. Det gör att Kinrate inte rekommenderar liknande spel utan mer utifrån vilka personlighetskategorier spelet hamnar inom, vilket inte är anpassat för hemsidans rekommendationssystem.

2.5 Ramverk

Graphlab
Graphlab är ett ramverk med öppen källkod skriven i C++ och är skapad av företaget Turi. Ramverket går att användas till att lätt och smidigt skapa maskininlärningsmodeller från grunden, utan att förkunskaper om maskininlärning krävs. Ramverket har även tillgång till hjälpmedel som beskriver stegvis hur ett rekommendationssytem implementeras. I Graphlabs API finns ett bibliotek med metoder som kan bl.a. manipulera datamängder, träna upp rekommendationsystem och evaluera systemen. API:et använder sig av Python, som är ett öppet, objekt-orienterad programmeringsspråk.

Graphlab använder olika modeller för att kunna skapa ett rekommendationssystem. De modeller som används för kollaborativ filtrering är ItemSimilarityRecommender, RankingFactorizationRecommender och Popularity.

ItemSimilarityRecommender jämför likheter mellan olika spel och rangordnar utifrån spelets likhet till andra spel för en användare. T.ex. om två användare har recenserat två olika spel lika högt eller lågt, så är spelen mer lika än om spelen hade recenserats olika. Ett spels likhet till ett annat beror på hur lika betygen är och likheten mäts med en skala från -1 till 1, där 1 indikerar på att spelen är exakt identiska. För att modellen ska kunna jämföra spel, behöver ett spel ha recenserats av minst 2 användare. Modellen kan använda tre olika likhetsmätvärden som används för att bestämma hur modellen ska tränas. Mätvärdena är jaccard, cosine och pearson. Jaccard är användbar att använda om spelen har blivit bedömda binärt d.v.s. en person har gillat det eller inte, eller om bedömningsskalan (t.ex. 1-5) är ointressant. Om bedömningarna med en icke-binär skala för spelen är intressant, så används Cosine eller Pearson. Skillnaden mellan dom två är att Cosine inte väger in skillnaden i medelvärde och variansen av bedömningarna som gjorts mellan två spel [22].

RankingFactorizationRecommender använder viktvärde och faktor för att rangordna rekommendationerna. Viktvärdena och faktorerna skapas utifrån de bedömningar som gjorts av användare för ett spel. Modellen använder sedan viktvärdena och faktorerna för att ge ett värde för varje rekommendation mellan användare och spel, där värdet används för att rangordna rekommendationerna med högsta värdet först. Ett viktvärde kan ge ett spel eller användare högre eller lägre värde, beroende på om hur ofta ett spel har bedömts högt eller lågt. Faktorn är värdet på interaktionerna mellan användare och spel. Om t.ex. en användare gillar pusselspel och ogillar sportspel, så försöker faktorn fånga det och ger ett högre värde för pusselspel och lägre värde för sportspel [25].

PopularityRecommender rangordnar spel utifrån hur populära de är. Popularitetsvärdet tolkas som hur många gånger ett spel har recenserats, där fler recensioner ger högre värde. Om det specificeras att ta hänsyn till bedömningarna i modellen, beräknas medelvärdet av betygen från recensionerna för varje spel och rangordnar spelen utefter det [26].

Andra ramverk

Andra ramverk som Microsoft Azure, Google Tensorflow och Apache Spark testades. Tensorflow saknade bra exempel på Rekommendationssystem att följa. Azure kan endast användas online som en betalversion där man har en regelbunden månadsavgift beroende på hur mycket data man använder. Slutligen ansågs Apache Spark vara lämplig till arbetet till skillnad till de andra ramverken. Men Graphlab hade bättre visualiseringsverktyg för rekommendationssystemet och gjorde det enklare att presentera för företaget, där Apache Spark valdes också bort.

3. Metod

Detta kapitel beskriver vilka verktyg och metoder som använts för att utföra arbetet.

3.1 Ramverk- och modellval

Ramverket som valdes för implementeringen av rekommendationssystemet är Graphlab, där motiveringen har förklarats i 2.5 Ramverk.

I Graphlab finns det olika modeller för att konstruera ett rekommendationssystem, där en modell väljs utifrån vilken datamängd som används. De kollaborativa filtrerings modeller som finns i Graphlab, som också kommer att användas i testning av deras pricksäkerhet, är ItemSimilarityRecommender, RankedFactorizationRecommender och PopularityRecommender. Testerna har gjorts genom att se hur bra modellerna presterar med dataseten som nämns i 3.2 Dataset och modellernas resultat jämförs med varandra. Modellerna har jämförts genom evaluering av systemen, som beskrivs mer i 3.4 Evaluering.

3.2 Dataset

I arbetet har Amazons och företagets dataset valts eftersom datan är relaterad till spel, som är vad företaget vill fokusera på i sina rekommendationer. Dataseten har uppdelats till två delar, Produktdata och Recensionsdata. Produktdata är extrainformation för en produkt, som används som hjälpmedel för att tolka datan. Recensionsdata används för att träna upp modellen, som innehåller recensioner från användare till varje spel. Tabeller för respektive dataset finns att se i Bilaga 3.

Amazon dataset

Datasetet innehåller 50,953 olika spelprodukter med 1,324,753 recensioner från 800,571 användare, från Maj 1996 till Juli 2014. Med spelprodukter menas produkter som är relaterade till videospel men som inte måste vara videospel (som t.ex. hårdvara). Datasetet hämtades från en hemsida [27][28][29].

IGDB dataset

Datasetet innehåller 94,234 olika spel med 162,974 recensioner från 6,475 användare. Datasetet skapades och tillhandahölls av företaget.

Steam dataset

Steam har tillgång till ca 11 miljoner recensioner på minst 22,000 spel [35][36]. Steam var av intresse för arbetet, men det fanns inget färdig strukturerat dataset som kunde ha använts. Företaget *8 Dudes in A Garage* har tillgång till en datamängd från Steam och har rättigheter att använda det på hemsidan. Denna datamängden kunde dock inte användas på grund av dess komplexitet och invecklade struktur. Försök till att få fram eget dataset misslyckades både på grund av tidsbrist och oväntade problem med hämtningen av datan. Datan som har hämtats genom webcrawling, en metod för bottar att skrapa fram data från källkoden till hemsidor,

kom ofta tillbaka inkomplett av okänd anledning. Om problemet hade fixats skulle det fortfarande kvarstå ett problem med tidsbrist, då denna metod av datahämtning skulle uppskattningsvis ta mellan 2-3 veckor att få fram och analysera. Försök till att få fram datan gjordes trots avgränsning nr 3 i 1.4 Avgränsningar på grund av det stora potentiella värdet av en datamängd från Steam. Webcrawling är lagligt då användandet av datamängden inte sker i kommersiellt användande och bottarna inte överbelastar företagets hemsida. Mer om detta står i kapitel 7. Förslag till Fortsatt Arbete.

3.3 Databehandling

Ibland kan det finnas så stora variationer i datamängden att det kan påverka rekommendationssystemet och ge oväntade resultat. Samtidigt finns det många spel som har recenserats enbart av ett fåtal användare, vilket gör det svårt för rekommendationssystem att lära sig och se mönster i datan. Därmed behöver datan filtreras så att systemet optimeras och blir effektiv i sina förutsägelser. Det är dock lika viktigt att ta hänsyn till hur mycket som ska filtreras bort som att ha tillräckligt för att kunna träna upp ett rimligt bra rekommendationssystem. I Amazon datasetet har många kategorier filtrerats bort (26st i arbetet) som t.ex. Hårdvara, Accessoarer m.m. eftersom de är oväsentliga för arbetets rekommendationssystem, som har fokus på enbart spel. Konsekvensen av filtreringen blev dock att vissa spel som har klassificerats i en borttagen kategori också filtrerades bort. Filtrering av spel med en gräns på antal recensioner och filtrering av användare med endast 1 recension, har också påverkat stort. Nedan visas hur många spel som är kvar utifrån en gräns på hur många recensioner ett spel fått och gjorts av användare:

IGDB:
Ingen recension gräns per spel: 13336 spel kvar
Minst 3st: 4524 spel
Minst 4st: 3854 spel
Minst 5st: 3405 spel
Minst 5st recensioner per spel och minst 2st recensioner per användare: 3401 spel

Amazon:
Inga recension gräns per spel: 50208 spel kvar (37115*)
Minst 3st: 29019 spel (22877*)
Minst 4st: 26108 spel (20730*)
Minst 5st: 23866 spel (19035*)
Minst 5st recensioner per spel och minst 2st recensioner per användare:(18,245*)
* = Siffrorna i parentes visar hur många spel som är kvar med filtreringen av både recensioner och kategorier.

I båda fallen försvinner många spel med filtreringen. Däremot möjliggör det att ett rekommendationssystem bättre kan urskilja likheterna mellan spel och ge bättre rekommendationer, ju fler recensioner ett spel har. I arbetet har ett recensionsgräns på 5st per

spel och 2st per användare satts upp, då det ansågs fortfarande finnas tillräckligt mycket data som rekommendationssystemet kan använda för att träna med.

3.4 Evaluering

Efter att modell prototyperna har skapats, så har de evaluerats på hur pricksäkra de är på att ge rekommendationer. Evalueringen gjordes med att använda mätvärdena precision, recall och F1-score. Motiveringen till valet är att mätvärdena används till att mäta hur pricksäkra systemen kan förutse att en användare kommer interagera med ett spel, vilket nämndes i 2.3 Evaluering av rekommendationssystem. Modellernas F1-score har beräknats utifrån formeln från kapitel 2.3, med medelvärdet på precision och recall för top 5 rekommenderade spel.

Det system som presterar bäst ska också testas genom att låta användare från IGDB.com prova det och bedöma hur korrekta rekommendationerna är. Testet ger i praktiken ett mått från ett verklighetsperspektiv på hur tillförlitligt systemet rekommenderar spel till en användare. Återkoppling från användare kan stärka eller försvaga validiteten av systemets rekommendationer. Testet kommer att skötas av företaget och kommer att låta användare ge återkoppling av hur bra rekommendationerna är, genom att trycka på en gilla/ogilla knapp.

3.5 Genomförande

Skelettet av kodstrukturen för rekommendationssystemet är baserad på ett exempel från Graphlab. Den är uppdelad i olika steg beroende på vilken uppgift som varje steg gör. Koden har modifierats till att skapa ett rekommendationssystem utifrån arbetets mål och krav.

- **Ladda datan**

Dataset filen hämtas och läggs in i två variabler, *actions* och *items*, som är av datatypen SFrame. I *actions* läggs alla recensioner och i *items* läggs metadatan för de spel som recenserats. IGDB datasetet är strukturerad i csv format och Amazon i json format. Några av filformaten som går att läsas av Graphlabs metoder är csv, json, json lines och sql.

- **Förbereda datan**

I båda dataseten manipulerades datan för att optimera rekommendationssystemet. Koden som implementerades räknar antalet recensioner per spel och filtrerar bort alla spel som har mindre än 5 recensioner. Koden räknar även antalet recensioner per användare och tar bort användare som har bedömt endast en gång. Samma resultat erhålls oavsett vilken turordning man har på filtreringen. Extra kod gjordes enbart för Amazon datasetet, där många kategorier filtrerades bort som t.ex. Hårdvara, Accessoarer m.m. Anledningen var att det fanns produkter som inte var spel i datasetet som tillhörde de borttagna kategorier, som är oväsentliga då arbetet fokuserar enbart på rekommendationer av spel. Spel som felaktigt tillhörde kategorierna som togs bort, filtrerades också bort som konsekvens.

- **Träna RS modellen**

För att testa ett systems pricksäkerhet delas en datamängd upp till två separata datamängder, ett träningsset och ett valideringsset som delas upp 80% respektive 20%. Uppdelningen av

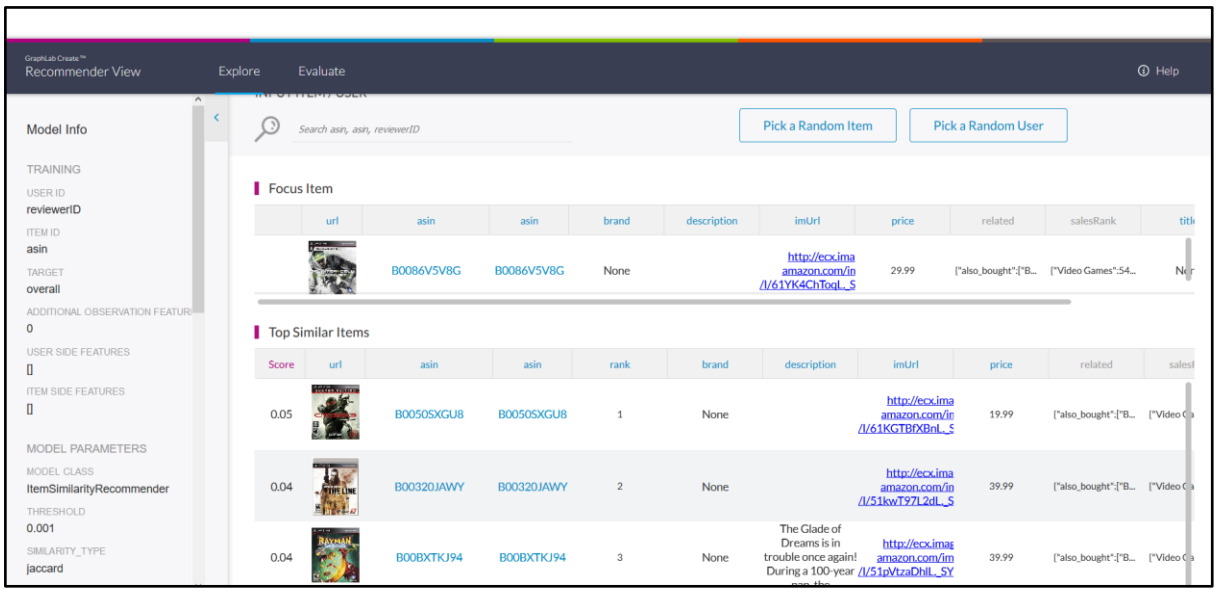
datamängderna kommer att göras slumpmässigt m.h.a. en av Graphlabs egna metoder. Efteråt skapas ett rekommendationssystem utifrån en modell och tränas med träningssetet. Parametrarna för användare, spel och bedömningar är för vissa modeller valbara och används för att specificera vad varje parameter har för information i datasetet.

I Graphlab skapas ett system med ItemSimilarityRecommender modell, som använder "similarity_type" för att välja vilken likhetstyp som ska användas. I testerna skapas olika modeller där parametrarna "jaccard", "cosine" och "pearson" används i olika modeller. Det görs för att jämföra resultaten på modellernas pricksäkerhet. En annan modell som testas är RankingFactorizationRecommender modellen, som har en extra parameter "max_iterations". Parametern bestämmer hur många gånger modellen ska träna sig med datasetet. Ett antal tester med olika värden på "max_iterations" för modellen görs för att se hur stor påverkan och betydelse antalet iterationer har för pricksäkerheten för systemet. Antalet iterationer som används i testerna är 25(default), 100, 200, 500, 1000 och 2000. Den sista modellen som testas är PopularityRecommender, för att se hur bra rekommendationerna är i jämförelse med de andra modellerna.

- **Evaluera modellen**

```
model1.evaluate(validation_data)
```

Denna metod evaluerar modellen med testdatasetet (som finns i vatiabeln *validation_data*), och skriver ut resultaten av mätvärdena precision, recall och RMSE. Värdena av precision och recall kommer att användas i F1-score formeln från 2.3.2 för att få fram F1-score värdet och ge ett slutresultat för systemets pricksäkerhet.



Figur 2: "Recommender View"fönster som visar rekommendationer för ett valt spel.

Ett fönster på en lokal host via en webbläsare kommer att skapas, där det går att interagera och se systemets rekommendationer för både spel och användare. Spel rangordnas utifrån dess likhet till det valda spelet. Skalan är 0 till 1 och bestäms av olika faktorer beroende på modellen. Det finns även en fönster flik som visar ett systems precision/recall graf, där

graferna av testerna hämtas och läggs in i Resultat. Precision och recall har samma värde som till precision/recall grafen från Figur 2 och används för att bekräfta resultatet i grafen (som visas i Bilaga2).

4. Resultat

En rekommendationssystems prototyp skapades där systemet är implementerad med företagets dataset och ett tredje-parts bibliotek, och kan rekommendera spel till användare. Andra dataset utforskades, där ett av dataseten implementerades i de tester som genomfördes. Dock tog vissa delar av arbetet längre tid än beräknat, vilket gjorde att prototypen inte implementerades i hemsidan IGDB.com och ett test med användare från hemsidan kunde inte genomföras i tid.

4.1 Evaluering av modellerna

Testning och evaluering av olika modellers pricksäkerhet har uppfyllts utifrån arbetets mål och visas i tabellen nedan. Värdena utgår från de top 5 rekommenderade spelen för varje användare. Skalan går från 0 till 1, där 1 är det bästa möjliga resultat. Siffror på hur många recensioner (observations), användare (users) och spel (items) som blev kvar efter filtrering visas också för varje dataset. Precision/Recall grafer för varje modell för respektive dataset visas i Bilaga2.

IGDB dataset:

Number of observations **134,167** Number of users **3,617** Number of items **3,401**

modell/mätvärde	mean_precision	mean_recall	F1-score
ItemSimilarity, jaccard	0.194292237443	0.179273158289	0.186480779
ItemSimilarity, cosine	0.185388127854	0.172767450714	0.178855424
ItemSimilarity, pearson	0.00182648401826	0.00111888388606	0.001387686
RankingFactorization, 25 iterationer	0.13401826484	0.0962786298533	0.112056178
RankingFactorization, 100 iterationer	0.135616438356	0.0987307000099	0.114270701
RankingFactorization, 200 iterationer	0.137671232877	0.10911081042	0.121738353
RankingFactorization, 500 iterationer	0.108904109589	0.0814363147602	0.09318829016
RankingFactorization,	0.0842465753425	0.0664150294813	0.074275577

1000 iterationer			
RankingFactorization, 2000 iterationer	0.0748858447489	0.0668028204536	0.070613773
Popularity	0.00182648401826	0.00111888388606	0.001387686

Tabell: Evalueringsvärde med IGDB dataset för varje modell och genomsnittsvärde för top 5 rekommenderade spel. Ju högre värdena är(maxgräns till 1), desto bättre är modellen.

ItemSimilarityRecommender modellen med jaccard eller cosine gav bäst rekommendationer av modellerna, medan pearson och PopularityRecommender gav sämst. RankingFactorizationRecommender fick varierande resultat, med värden som ligger högre än genomsnittet mellan den högsta och lägsta värdet av modellernas F1-score.

Amazon dataset:

Number of observations **360,096** Number of users **76,265** Number of items **18,245**

modell/mätvärde	mean_precision	mean_recall	F1-score
ItemSimilarity, jaccard	0.0251798561151	0.0756738764832	0.03778655
ItemSimilarity, cosine	0.0190647482014	0.0584076894329	0.028746427
ItemSimilarity, pearson	0.000359712230216	0.00179856115108	0.00059952
RankingFactorization, 25 iterationer	0.00359712230216	0.00945607150643	0.000068029
RankingFactorization, 100 iterationer	0.00575539568345	0.0173997165904	0.008649688
RankingFactorization, 200 iterationer	0.00467625899281	0.01176558753	0.006692549
RankingFactorization, 500 iterationer	0.00359712230216	0.0112546326575	0.005451785
RankingFactorization, 1000 iterationer	0.00395683453237	0.0124536734249	0.006005542
RankingFactorization, 2000 iterationer	0.00503597122302	0.0179427886262	0.007864599
Popularity	0.0	0.0	0.0

Tabell: Evalueringsvärde med Amazon dataset för varje modell och genomsnittsvärde för top 5 rekommenderade spel. högre värdena är(maxgräns till 1), desto bättre är modellen.

ItemSimilarity modellen gav bäst rekommendationer med jaccard och cosine medan pearson fick ett sämre resultat i värde, knappt före Popularity. Varierande resultat gavs med olika iterationer för RankingFactorization, där värdet växlade ju fler iterationer som gjordes. Enligt tabellen så gav RankingFactorization med 100 iterationer bäst resultat och rekommendationerna blev sämre med fler iterationer. Men med 1000 iterationer ökade värdet igen på F1-score och indikerades att fortfarande öka efter 2000 iterationer.

5. Diskussion

Utifrån från våra mål så har en prototyp av ett rekommendationssystem gjorts i Graphlab, där både företagets och ett tredje parts dataset testats och evaluerats. Ramverket Graphlab har bedömts som användarvänligt och ger möjligheten till företaget att vidareutveckla systemet. Evalueringen av de olika modellerna har kunnat ge vägledning till vad som passar företaget bäst. Dessvärre gjordes inte testningen av den bäst resulterande rekommendationssystemet med hemsidan IGDBs användare inom arbetets tidsram. Nedan diskuteras resultatet från evalueringen, problem med dataseten, problem med rekommendationssystem och förslag på lösningar.

5.1 Modell

Testerna tyder på att ItemSimilarity modellen ger de bästa rekommendationer för dataseten, där likhetstypen jaccard gav högst värde i båda dataseten. Det var ingen stor klyfta i F1-score värdet mellan ItemSimilarity och RankingFactorization, vilket möjliggör den senare modellen som alternativ för skapandet av företagets rekommendationssystem. Antalet iterationer för RankingFactorization hade betydelse för systemets pricksäkerhet, där olika antal gav varierande värde. För båda dataseten blev F1-score högst vid 100-200 iterationer och lägst vid 500 iterationer. Men dataseten skiljer sedan åt vid 1000 iterationer, där värdet i IGDBs dataset sjunker medan Amazon ökar. Tolkningen av resultatet är att testning av olika värden för modellerna och korsvalidering är obligatoriska för att kunna optimera systemet. Valet av modell har också betydelse som beror på datasetets struktur, där en modell är bättre anpassad för ett dataset än ett annat [30].

5.2 Dataset

IGDBs dataset fick i genomsnitt högre värde i sina tester än Amazon. En faktor till det kan bero på datasetets storlek och struktur. Amazons dataset har mycket större andel data med recensioner, användare och spel än IGDBs. Det gör att systemet har mycket mer data att analysera och se mönster, vilket ökar komplexiteten och gör det svårare för systemet att förutse lämpliga rekommendationer. En annan stor faktor till det kan vara skillnader i användarnas recensionsgrad, där användare i IGDBs dataset hade i genomsnitt mer recensioner. Detta leder till att mer precisa rekommendationer kan göras. Amazon hade i genomsnitt ca 2 recensioner per användare jämfört med IGDBs ca 25 recensioner per användare i genomsnitt före datamanipuleringen. Efter datamanipuleringen ökade recensioner per användare för Amazon till ca 5 medan IGDBs ökade till ca 37 recensioner per användare. Efter datamanipuleringen föll även ~44% och ~87% av alla användare bort från i IGDBs respektive Amazons dataset, när en begränsning på minst 2 recensioner per användare gjordes. Detta indikerar på att båda dataseten har ett problem med sparsity, i synnerlighet Amazon där en stor andel av användare bara har recenserat en gång. I fallet med Amazon kan detta ses som logiskt, eftersom en större del av deras kundbas endast handlar ett fåtal gånger och uppmuntran till recensering är låg.

Ett kollaborativ filtrerings rekommendationssystem behöver många recensioner från flera användare och spel för att det ska ge relevanta rekommendationer. Ett exempel på problem som kan uppstå är om en användare har recenserat ett spel bara en gång och en annan användare recenserar samma spel likadant som förra användaren. Eftersom den förra inte har recenserat mer än en gång, kan inte systemet rekommendera några spel till den andra. Systemet skulle inte kunna tolka datan på ett effektivt sätt och ha större risk att ge irrelevanta rekommendationer. Detta kan även ses som en stor orsak till varför Amazon har en specialbyggd teknik, item-to-item filtreringsteknik, i sitt rekommendationssystem och varför filtrering av dataseten behövdes göras för arbetet.

Dataseten som valdes var lämpligt för arbetet, där det finns tillräckligt med recensioner att jobba med för att kunna få varierande och relevanta rekommendationer. Dock fick mycket av data filtreras bort eftersom en stor andel av spelen hade mindre än fem recensioner. Det gällde samma för användare som hade enbart recenserat en gång. Många kategorier som t.ex. hårdvara fick också filtreras bort från Amazon datasetet eftersom fokus ligger på enbart videospel för arbetet. Andra problem med Amazon datasetet, som också är ett förekommande problem för den kollaborativa filtrerings tekniken, är Cold-start problemet och Synonymitet. Med Cold-start problemet kommer inte alla spel att rekommenderas för att de har fått för lite eller inga recensioner, vilket leder till att de har mycket låg till obefintlig sannolikhet att rekommenderas. Eftersom Amazon datasetet är föråldrat från 2014 så finns det många spel som är släppta år 2013-2014 som saknar tillräckligt med recensioner. Synonymitet förekommer i Amazon då samma spel är uppdelade i olika konsoler och versioner av spelet. Datan blir redundant och risken blir att ett system rekommenderar samma spel eller att spelen saknar tillräckligt med recensioner på enskilda spelkonsoler (Sparsity).

5.3 Evaluering

Evalueringen som valdes är precision, recall och F1-score för att den mäter rekommendationssystemets förmåga att rekommendera relevanta spel, utifrån ett spel som en användare tittar på vid ett tillfälle. Root-Mean-Square-Error värdet visar på systemets förmåga att förutse vilket medelbetyg ett spel får utifrån betygen av användare, vilket inte faller i samband med arbetets mål och anses inte som ett lämpligt mätvärde. Mätvärdena som användes i arbetet fungerar som en estimering av vad en användare hade tyckt varit ett bra rekommendation i en labbmiljö. Däremot räcker det inte att enbart använda mätvärdena, eftersom det inte reflekterar verkligheten. Även om ett system ger höga värden i F1-score, skulle det fortfarande vara möjligt att användaren väljer att inte använda systemet [8].

Några orsaker till att användare väljer att inte använda ett rekommendationssystem kan bero på problem med contextuality och serier. Med contextuality menas att olika spel kan ha starka korrelationer med likheter till varandra, men kontextuellt inte är en lämplig rekommendation som t.ex. åldersgränser och kvalitetsskillnad. En användare som gillar spel från samma serie kommer troligtvis att recensera spelen likvärdigt. Det gör att spelen tolkas som lika av systemet och kommer med stor sannolikhet att hamna högre i rekommendationerna. Rekommendationer av spel i samma serie skulle vara en bra

rekommendation för någon som börjat gilla ett av spelen i serien eller som har missat ett spel från sin favoritserie. Men rekommendationerna blir inte intressant för personer som vill testa nya spel, eftersom möjligheten att utforska spel från olika serier är låg. En lösning på problemet vore att skapa två delar av fönster med spelrekommendationer. Ena delen visar enbart spel från samma serie som den valda spelet (som inte behöver komma från rekommendationssystemet) och den andra som visar rekommendationer från rekommendationssystemet, men rekommendationerna är från olika serier.

I resultatet fick arbetets rekommendationssystem med högst F1-score ett genomsnittsvärde på ca 0,15 och max 0,20 (med en skala mellan 0 till 1). Värdet anses vara så lågt att det inte skulle kunna betraktas som ett bra rekommendationssystem. Dock skulle en bekräftelse behövts göra genom att jämföra arbetets rekommendationssystem med andra rekommendationssystem (som används av olika företag). Företagens systemen hade satt upp en genomsnittsgräns på vart måttvärdena på ett rekommendationssystem bör ligga, för att ett system ska anses som ett bra system. Men en jämförelse kunde inte göras eftersom att i vissa fall användes RMSE istället för precision, recall & F1-score för bedömning av företagens system och att det inte fanns tillgång till värdena för måtten.

Ett test med användare hade fångat brister eller styrkor med ett rekommendationssystem, där det inte går med enbart evaluering av mätvärden. Användartestning hade gett en mer verklighetsbaserad utvärdering av rekommendationerna då rekommendationer är subjektiva.

5.4 Etik

Ett av de stora frågorna kring rekommendationssystem är om dess trovärdighet och om den kan missbrukas. Spel behöver recensioner för att ett kollaborativ filtrerings system ska kunna tränas och rekommenderas. Men det är också viktigt att tolka recensioner på hur det kan påverka rekommendationssystemet, eftersom det finns risk på att det kan missbrukas. T.ex. kan ett företag med vinstdrivande motiv, skapa falska konton som recenserar sina egna spel högt och sina konkurrenter lågt. Konsekvensen blir att rekommendationerna av rekommendationssystemet missbrukas, där rekommendationerna t.o.m. inte delar likheter mellan kategorier som genre, tema m.m.

Recensionsdata från användare är viktigt för utvecklingen av rekommendationssystem där mycket data förbättrar precisionen av rekommendationerna med kollaborativa filtreringstekniken. Men datan är personlig och innehåller känslig information, där det finns risk att en person utanför nätet identifieras till datan. Ett exempel på en sådan händelse skedde när företaget Netflix hade en öppen tävling med mål att få fram den bästa möjliga rekommendationssystemet. Netflix gav ut data till tävlande där ett par forskare kunde identifiera personer genom att koppla Netflixs data med filmrecensioner från Internet Movie DataBase, och Netflix blev stämda för att ha brutit mot USA:s “fairtrade” lagar och Video Privacy Protection Act [8][31]. För att använda datan så behöver företag och forskare följa de lagar och regler som bevarar användares uppgifter, som kan vara känslig eller går att koppla

deras recensioner till dom. Lagen som täcker området är Dataskyddsförordningen (GDPR), som ersätter den nuvarande Personuppgiftslagen [32][33].

6. Förslag till fortsatt arbete

Detta kapitlet diskuterar framtida möjligheter att fortsätta arbetet både för företaget och för studenter.

- Kombinera datasets

Ett förslag till fortsatt arbete är att kombinera dataset på användarrecensioner från Steam och Amazon till ett gemensamt dataset för att få en större datamängd. Utmaningarna med ett sådant arbete skulle vara ett matcha ihop spelen till varandra i de olika dataseten då de har olika id i de olika datasystemen samt att matcha ihop deras olika betygsystem. I Steam har spelen ett unikt id som då finns med kopplat till IGDBs unika spel-id i deras databas. Amazons unika identifikationnummer, ASIN, finns dessvärre inte med i IGDBs Application Programming Interface (API) och måste därför matchas på speltitel, vilket kan vara en osäker metod då ett spel kan ha olika speltitlar eller sakna speltitel helt som det gjorde i testdatasetet från Amazon. Tillgång till datan kan fås genom antingen genom Steamworks och AWS, Steams respektive Amazons API:n eller genom Webscraping/Crawling. Webscraping/Crawling är en metod som använder sig av en bot t.ex. Scrapy eller BeatifulSoup, som läser av relevant info i spelens hemsidor HTML kod och samlar ihop det till en datamängd.

- Hybrid system

Fortsättningsvis skulle man även kunna använda content-based filtering tillsammans med kollaborativ filtrering i arbetets prototyp och göra ett hybridsystem som viktar in egenskaper som temat, genre och Jaccard index på keywords. Hybridsystem är system där man kombinerat filtreringsteknikerna med antingen hög eller låg samverkan, där ena tekniken kan ha större prioritering än den andra. Då företaget redan har ett content-based filtering rekommendationssystem kan prototypen ta rekommendationer från deras system, då rekommendationer från ett kollaborativ filtrerings rekommendationssystem skulle ha för låga värden/för lite recensioner. Därmed kan problem som den kollaborativa filtrerings tekniken har med Sparsity åtgärdas.

- Feature engineering

“

Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering.

”

— Andrew Ng, *Machine Learning and AI via Brain simulations* [34]

Feature engineering är att räkna in variabler/aspekter av datamängden in i maskininlärning. Exempel på detta i just detta fallet skulle vara att räkna in åldersgränserna på spelen så att spel med låg åldersgräns, barnspel, inte matchas med spel med hög åldersgräns, vuxenspel. Detta behöver inte vara en strikt gräns då många spel med låg åldersgräns inte är barnspel, dock är det ett problem som tas upp inom etiken för rekommendationssystem. Andra funktioner som kan läggas till är att inte ha med rekommendationer från samma

serie/franchise då detta minskar på utforskningsmöjligheterna och ger väldigt förutsägbara rekommendationer. Filtrera bort rekommendationer som inte matchar samma spelkonsol kan läggas till som en valbarhet.

- Testa med användare från hemsidan

Förslag från företaget har varit att testa rekommendationssystemet med användare från hemsidan och få feedback från IGDB.com communityt. Feedbacken skulle kunna vara att under varje rekommendation ha en “tummen upp” och “tummen ner” knapp för att få respons om användaren tyckte rekommendationen var bra och även fråga allmänt i community kanalerna om rekommendationssytemet. Baserat på feedbacken skulle man kunna göra en analys av rekommendationssystemet och göra förbättringar. Denna informationen skulle även ge större insikt till vad användarnas önskemål och vilka problem med rekommendationssystemet skulle kunna vara.

Referenser

[1] E. McDonald, “The Global Games Market Will Reach $108.9 Billion in 2017 With Mobile Taking 42%”, 2017. [Online]. Tillgänglig: https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/, hämtad: 2018-02-07.

[2] B. Lang, “Global Box Office Hits Record $38.6 Billion in 2016 Even as China Slows Down”, 2017. [Online]. Tillgänglig: https://variety.com/2017/film/news/box-office-record-china-1202013961/, hämtad: 2018-02-08.

[3] “Theatrical Market Statistics 2016”, Washington D.C., USA: Motion Picture Association of America, 2017. [Online]. Tillgänglig: https://www.mpaa.org/wp-content/uploads/2017/03/2016-Theatrical-Market-Statistics-Report-2.pdf, hämtad: 2018-02-08.

[4] S. Galyonkin, “Monthly summaries - SteamSpy - All the data and stats about Steam games“, 2018. [Online]. Tillänglig: http://steamspy.com/year/, hämtad: 2018-02-10.

[6] C. Underwood, “Use Cases of Recommendation Systems in Business - Current Applications and Methods“, 2017. [Online]. Tillänglig: https://www.techemergence.com/use-cases-recommendation-systems/, hämtad: 2018-02-12.

[7] 8 Dudes in a Garage AB, “What is IGDB.com“, 2018. [Online]. Tillänglig: https://www.igdb.com/about, hämtad 2018-02-17.

[8] F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor, *Recommender Systems Handbook*, New York, USA: Springer, 2011. [Online]. Tillgänglig: https://link.springer.com/book/10.1007%2F978-0-387-85820-3, hämtad: 2018-03-14.

[9] B. Sarwar, G. Karypis, J. Konstan & J. Riedl, ”Item-Based Collaborative Filtering Recommendation Algorithms”, i *WWW '01 Proceedings of the 10th international conference on World Wide Web*, Hong Kong, Hong Kong, 2001, ss. 285-295. [Online]. Tillänglig: http://files.grouplens.org/papers/www10_sarwar.pdf, hämtad: 2018-04-20.

[10]G. Linden, B. Smith & J. York, "Amazon.com recommendations: item-to-item collaborative filtering", in *IEEE Internet Computing*, vol.7, nr.1, ss.76-80, Jan/Feb. 2003 [Online]. Tillänglig: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1167344&isnumber=26323, hämtad: 2018-04-21.

[11] K. Bailey, “Typologies and Taxonomies: An Introduction to Classification Techniques ”, Los Angeles, USA: Sage Publications, 1994. [Online]. Tillgänglig: https://www.researchgate.net/profile/Costas_Drossos/post/What_is_the_difference_between_a_typology_and_a_taxonomy/attachment/59d63753c49f478072ea4bf1/AS%3A273684877512704%401442262968252/download/%5BKenneth_D._Bailey%5D_Typologies_and_Taxonomies_An_Introduction_to_Classification_Techniques.pdf, hämtad: 2018-04-22.

[12]S. Jain, A. Grover, P. S. Thakur & S. K. Choudhary, "Trends, problems and solutions of recommender system", i *International Conference on Computing, Communication & Automation*, Noida, India, 2015, ss. 955-958. Tillgänglig: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7148534&isnumber=7148334, hämtad: 2018-03-15.

[13] S. Khusro, Z. Ali & I. Ullah, ”Recommender Systems: Issues, Challenges, and Research Opportunities”, i *Information Science and Applications (ICISA) 2016*, Singapore, 2016, ss.1179-1189. [Online]. Tillgänglig: https://www.researchgate.net/publication/294860665_Recommender_Systems_Issues_Challenges_and_Research_Opportunities, hämtad: 2018-03-15.

[14] T. Lappas, "Fake Reviews: The Malicious Perspective", i *Natural Language Processing and Information Systems - 17th International Conference on Applications of Natural Language to Information Systems*, Groningen, Nederländerna, 2012, ss. 23-24. [Online]. Tillgänglig: https://link.springer.com/chapter/10.1007/978-3-642-31178-9_3, hämtad: 2018-05-12.

[15] L.Plunkett, “Developer Kicked Off Steam For Posting Fake Reviews“, 2018. [Online]. Tillgänglig: https://kotaku.com/developer-kicked-off-steam-for-posting-fake-reviews-1825939184, hämtad: 2018-05-13.

[16] N.Grayson, "Steam 'Review Bombing' Is A Problem", 2015. [Online]. Tillänglig:https://steamed.kotaku.com/steam-review-bombing-is-a-problem-1701088582, hämtad: 2018-05-04.

[17] B. Sinclair, "Steam adds histograms to address review bombing", 2017. [Online]. Tillänglig:https://www.gamesindustry.biz/articles/2017-09-19-steam-adds-histograms-to-address-review-bombing, hämtad: 2018-05-06.

[18] I. V. Tetko, D. J. Livingstone & A. I. Luik, "Neural network studies. 1. Comparison of Overfitting and Overtraining", *Journal of Chemical Information and Computer Sciences*, vol. 35, nr.5, ss. 826–833, Sep. 1995. [Online]. Tillgänglig: https://pubs.acs.org/doi/abs/10.1021/ci00027a006, hämtad:2018-04-23.

[19] F.O.Isinkaye, Y.O.Folajimi & B.A.Ojokoh, “Recommendation systems: Principles, methods and evaluation”, *Egyptian Informatics Journal*, vol.16, nr.3, ss.261-273, Nov. 2015. [Online]. Tillgänglig: https://www.sciencedirect.com/science/article/pii/S1110866515000341?via%3Dihub, hämtad: 2018-04-23.

[20] J. Innala, Producent, "IGDB.com Insight - Episode 3", *Youtube*, 2016. [Video]. Tillgänglig: https://www.youtube.com/watch?v=Fp3UMOEAlWk, hämtad: 2018-04-26.

[21] Neo4j Inc., "What Is a Graph Database?", 2018. [Online]. Tillänglig: https://neo4j.com/developer/graph-database/, hämtad: 2018-04-26.

[22]Turi,“graphlab.recommender.item_similarity_recommender.ItemSimilarityRecommender — GraphLab Create API 1.10 documentation”, 2018. [Online]. Tillänglig: https://turi.com/products/create/docs/generated/graphlab.recommender.item_similarity_recommender.ItemSimilarityRecommender.html, hämtad: 2018-05-03.

[23] Kinrate.com, "Kinrate games", 2018. [Online]. Tillänglig: https://kinrate.com/about, hämtad: 2018-05-03.

[24] J. Vahlo , J.K. Kaakinen, S.K. Holm & A. Koponen, "Digital Game Dynamics Preferences and Player Types", *Journal of Computer-Mediated Communication*, vol.22, nr.2, ss. 88-103, Mar, 2017. [Online]. Tillänglig: https://academic.oup.com/jcmc/article/22/2/88/4161797, hämtad: 2018-04-20.

[25]Turi, "graphlab.recommender.factorization_recommender.RankingFactorizationRecommender — GraphLab Create API 1.10 documentation", 2018. [Online]. Tillänglig: https://turi.com/products/create/docs/generated/graphlab.recommender.ranking_factorization_recommender.RankingFactorizationRecommender.html#graphlab.recommender.ranking_factorization_recommender.RankingFactorizationRecommender, hämtad: 2018-04-17.

[26]Turi, "graphlab.recommender.popularity_recommender.PopularityRecommender — GraphLab Create API 1.10 documentation", 2018. [Online]. Tillänglig: https://turi.com/products/create/docs/generated/graphlab.recommender.popularity_recommender.create.html, hämtad: 2018-04-17.

[27] J. McAuley, "Amazon review data", 2018. [Online]. Tillänglig: http://jmcauley.ucsd.edu/data/amazon/, hämtad: 2018-03-28.

[28] R. He & J. McAuley, “Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering”, i *WWW '2016 - Proceedings of the 25th International Conference on World Wide Web*, Montréal, Québec, Canada, 2016, ss. 507-517. [Online].

Tillänglig: http://cseweb.ucsd.edu/~jmcauley/pdfs/www16a.pdf, hämtad: 2018-03-28.

[29] J. McAuley, C. Targett, J. Shi & A. van den Hengel, "Image-based recommendations on styles and substitutes", i *SIGIR ´15 - Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Santiago, Chile, 2015, ss. 43-52. [Online]. Tillgänglig: http://cseweb.ucsd.edu/~jmcauley/pdfs/sigir15.pdf, hämtad: 2018-03-28.

[30] Turi, "Choosing a Model", 2018. [Online] Tillgänglig:https://turi.com/learn/userguide/recommender/choosing-a-model.html, hämtad: 2018-03-27.

[31]A.Narayanan & V. Shamitikov, "Robust De-anonymization of Large Sparse Datasets ", i *2008 IEEE Symposium on Security and Privacy (sp 2008)*, Oakland, CA, USA, 2008,ss. 111-125. [Online]. Tillgänglig: https://ieeexplore.ieee.org/document/4531148/, hämtad: 2018-05-14.

[32]Datainspektionen, "*Personuppgiftslagen*",2001. [Online]. Tillgänglig: https://www.datainspektionen.se/lagar-och-regler/personuppgiftslagen/, hämtad: 2018-05-15.

[33]SFS 2010:1969. *Personuppgiftslag.* [Online]. Tillgänlig: http://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/personuppgiftslag-1998204_sfs-1998-204, hämtad: 2018-05-17.

[34] A. NG, Föreläsare, "Machine Learning and AI via Brain simulations", *Forum Stanford*, 2011. [Slides] Tillgänglig: https://forum.stanford.edu/events/2011/2011slides/plenary/2011plenaryNg.pdf, hämtad: 2018-05-13.

[35] L.Plunkett, "Big study of 10 million Steam reviews is absolutely fascinating", 2018. [Online]. Tillgänglig: https://kotaku.com/deep-study-of-10-million-steam-reviews-is-absolutely-fa-1825908713, hämtad: 2018-05-13.

[36]Steam, "Steam games", 2018. [Online]. Tillgänglig: https://store.steampowered.com/search/?sort_by=Released_DESC&category1=998, hämtad: 2018-05-13.

Bilaga1

Koden för konstruktionen av rekommendationssystemet med Amazon datasetet:

```
from os import path
import graphlab as gl
from datetime import datetime
# Path to the dataset directory
data_dir = './dataset/ml-20m'

# Table of reviews
actions = gl.SFrame.read_json(path.join(data_dir,
'reviews_Video_Games.json'), orient='lines')
# Table of meta data
items = gl.SFrame.read_json(path.join(data_dir,
'meta_Video_Games.json'), orient='lines')
actions = actions.filter_by(items['asin'], 'asin')

Prepare the data by removing items that are rare
rare_items = actions.groupby('asin',
gl.aggregate.COUNT).sort('Count')
rare_items = rare_items[rare_items['Count'] <= 5]
items = items.filter_by(rare_items['asin'], 'asin', exclude=True)
actions = actions.filter_by(rare_items['asin'], 'asin',
exclude=True)

rare_items = actions.groupby('reviewerID',
gl.aggregate.COUNT).sort('Count')
rare_items = rare_items[rare_items['Count'] <= 2]
#items = items.filter_by(rare_items['user_id'], 'user_id',
exclude=True)
actions = actions.filter_by(rare_items['reviewerID'],'reviewerID',
exclude=True)

items = items.unpack('categories')
categories_filter
=['Accessories','Hardware','Controllers','Headsets','Chargers','Dr
ums','Cables & Adapters','Cables','Guitars','Cases &
Protectors','Accessory Kits','Subscription
Cards','Electronics','Battery Packs','Gamepads','Racing
Wheels','Fire TV','Remotes','Memory Cards','AC
Adapters','Joysticks','Screen Protectors','Microphones','Gaming
```

```
Keyboards']
items.filter_by(categories_filter,'categories.0', exclude=True)
items.remove_columns(['categories.1','categories.2','categories.3'
,'categories.4'])
items.rename({'categories.0':'categories'})
training_data, validation_data =
gl.recommender.util.random_split_by_user(actions,'reviewerID',
'asin')

model = gl.item_similarity_recommender.create(training_data,
'reviewerID', 'asin','overall','pearson')

view = model5.views.overview(observation_data=training_data,
                             validation_set=validation_data,
                             item_data=items,
                             item_name_column='asin',
                             item_url_column='url'
                             )
#item_url_column='url'
view.show()

model5.evaluate(validation_data)
```

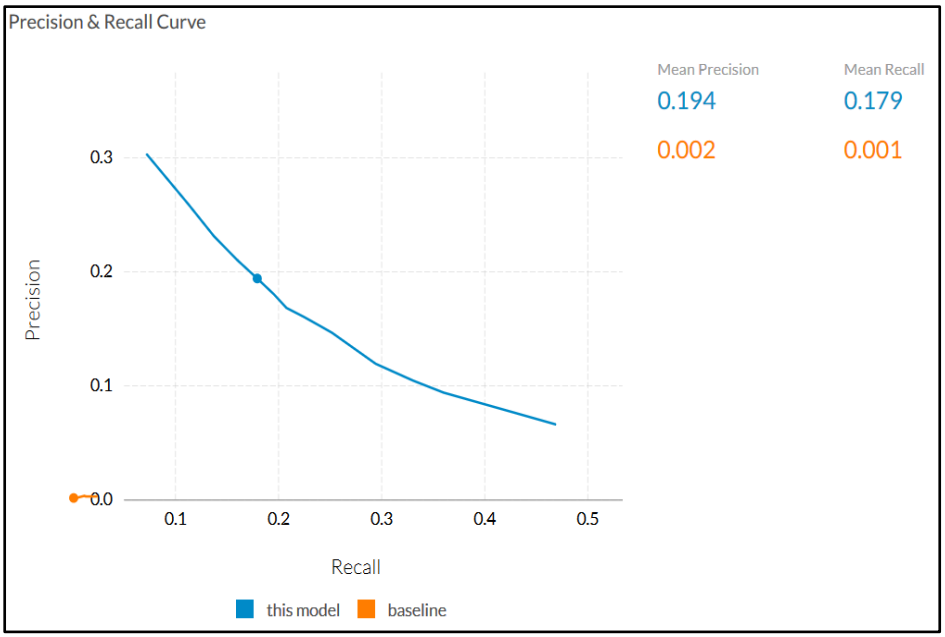
Bilaga2

Precision/Recall grafer för varje modell, där punkten motsvarar värdet för de top 5 rekommendationer, som visas i de övre högra hörnet.

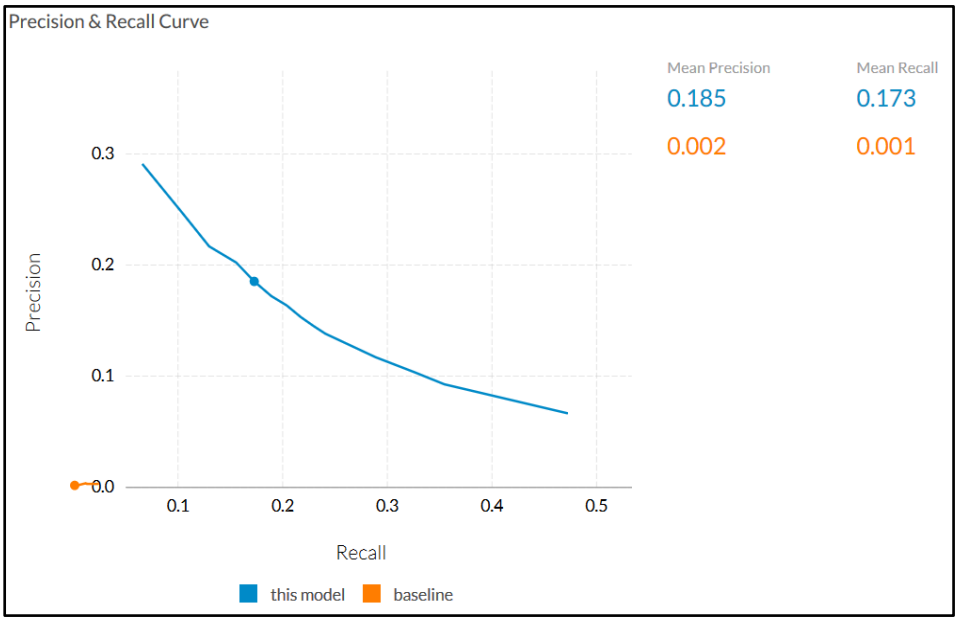
IGDB dataset:

ItemSimilarity

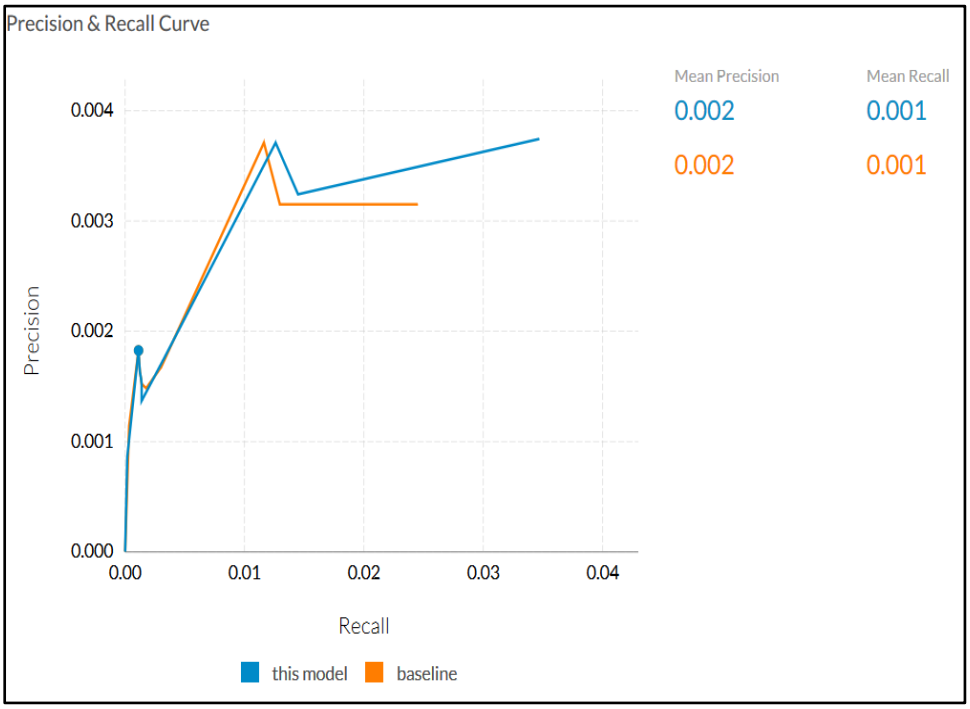
- **Jaccard**



- **Cosine**

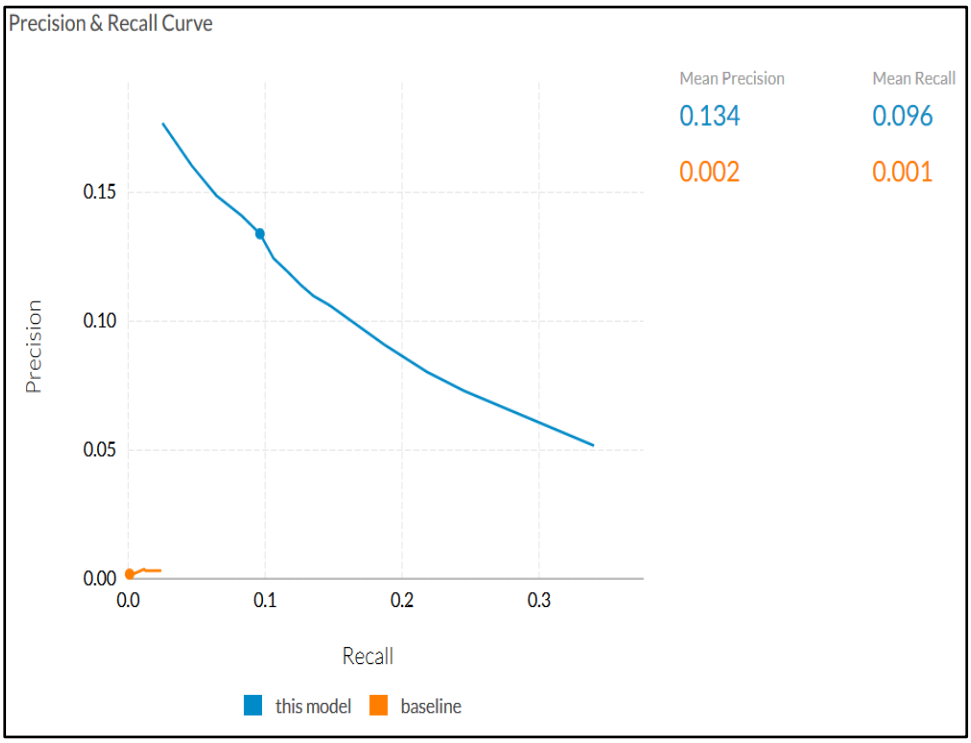


- **Pearson**

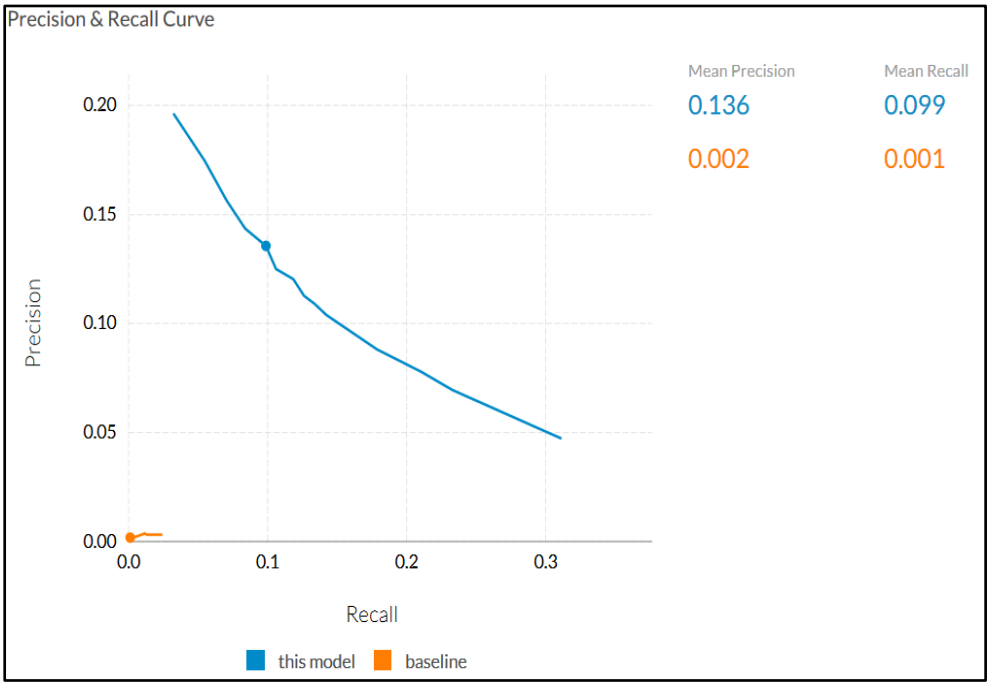


RankingFactorization

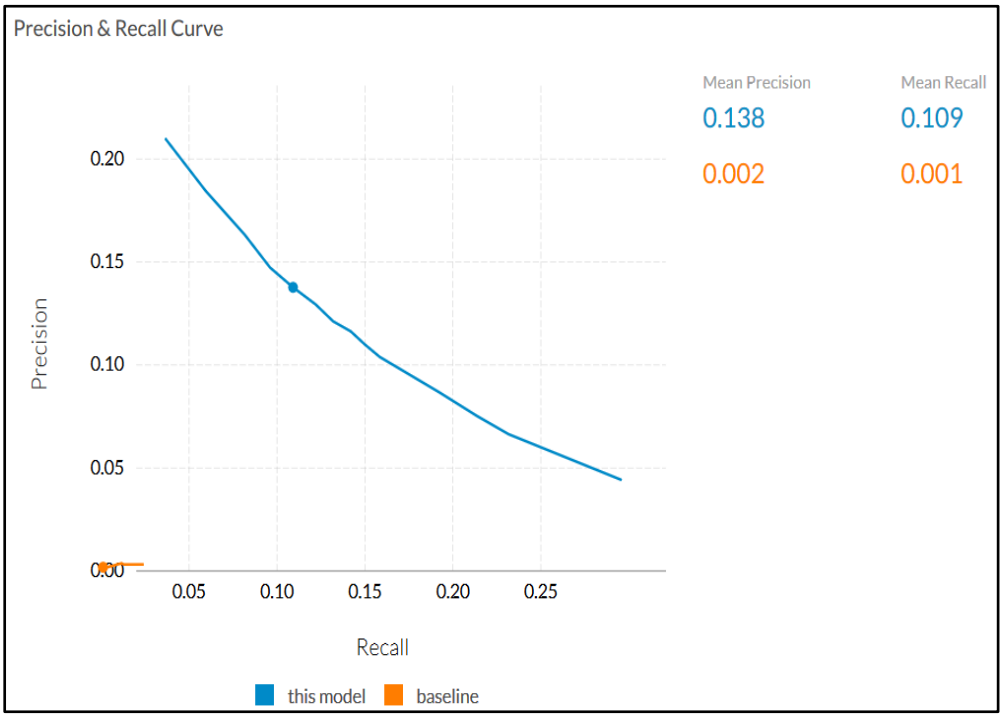
- **25 Iteratione 4.04s Training time**



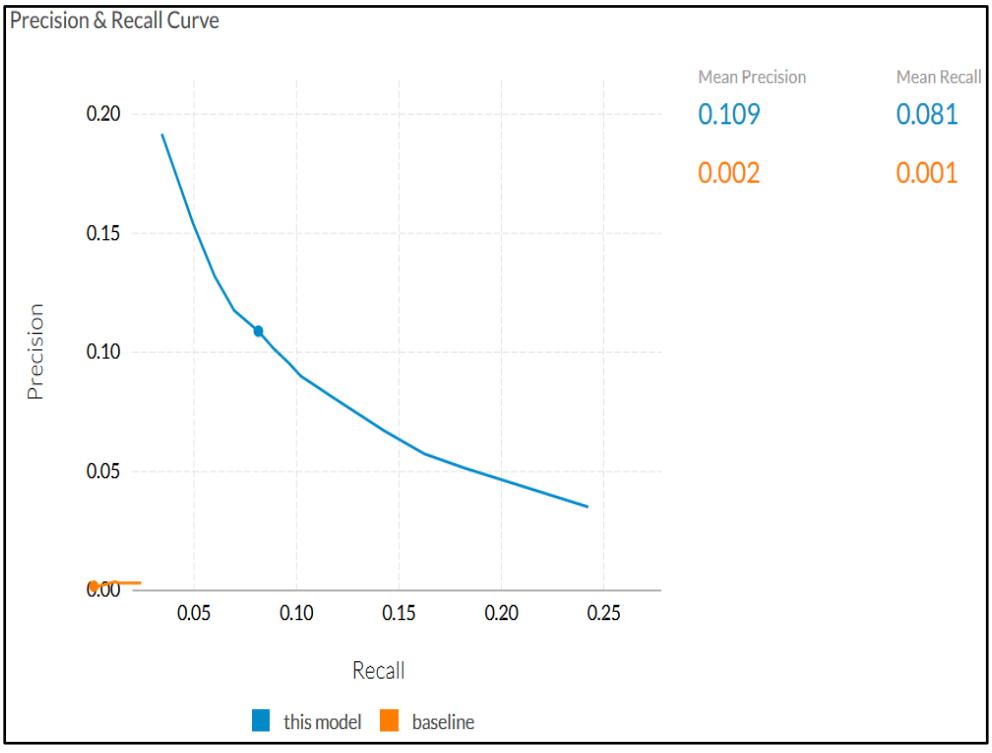
- **100 Iteratione 13.81s Training time**



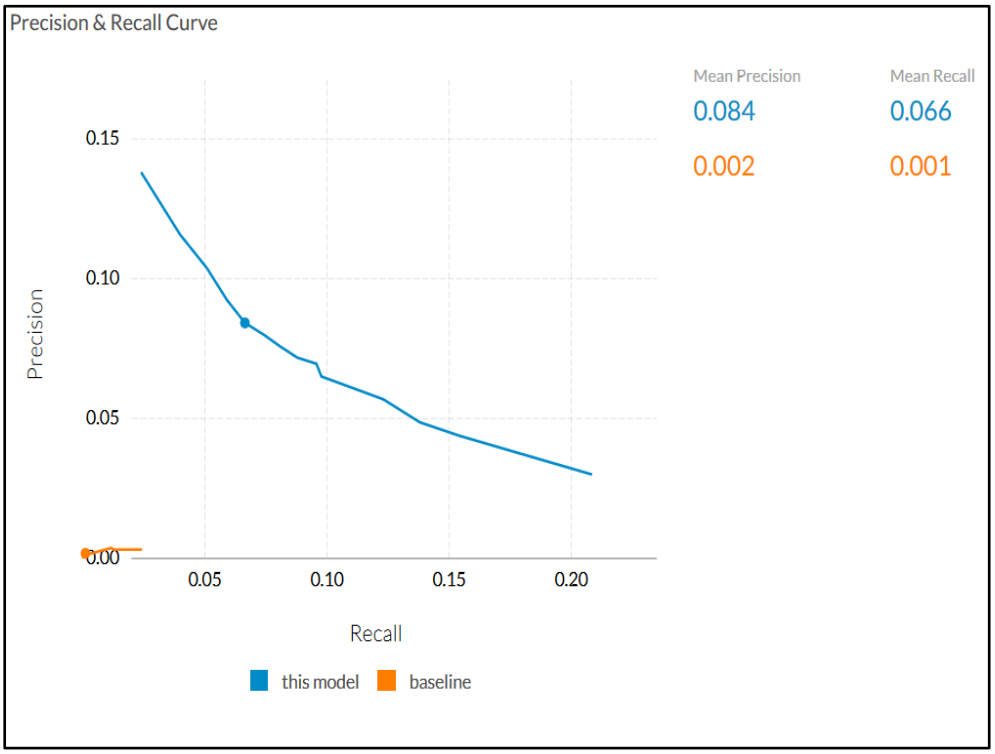
- **200 iterationer 24.12s Training time**



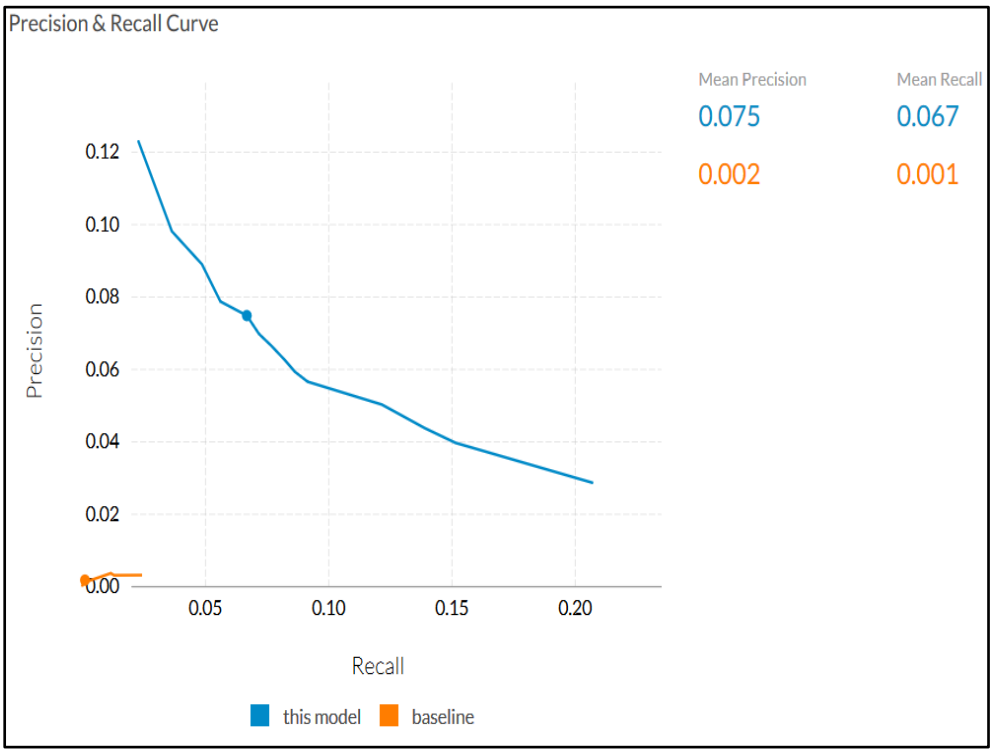
- **500 iterationer 1m 5s**



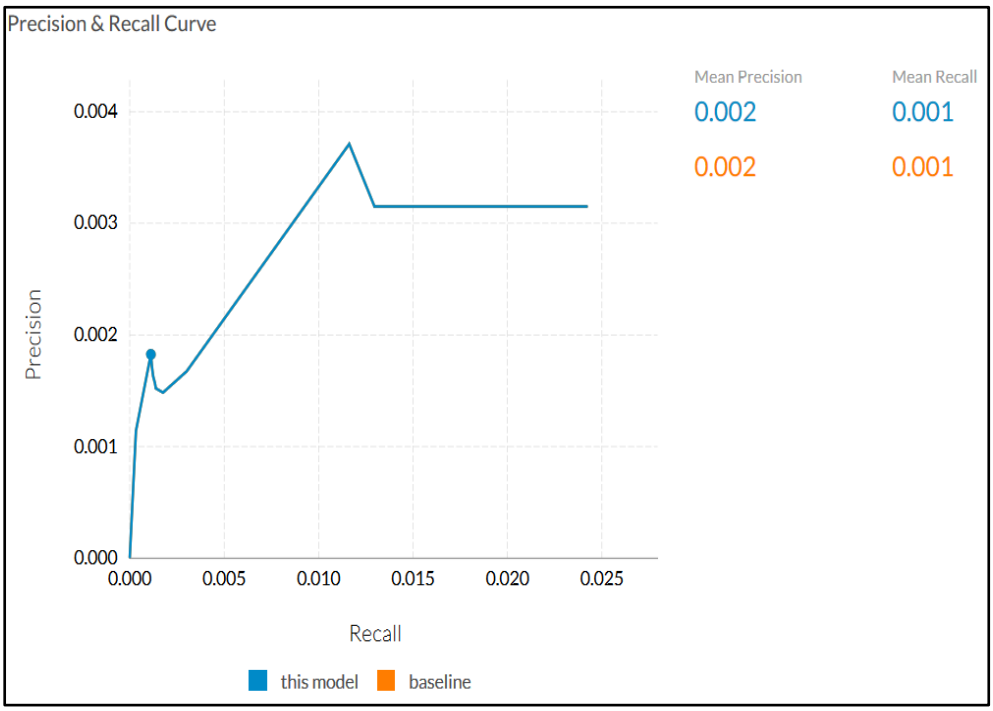
- **1000 Iterationer 2m 3s Training time**



- **2000 Iterationer 4m 6s Training time**



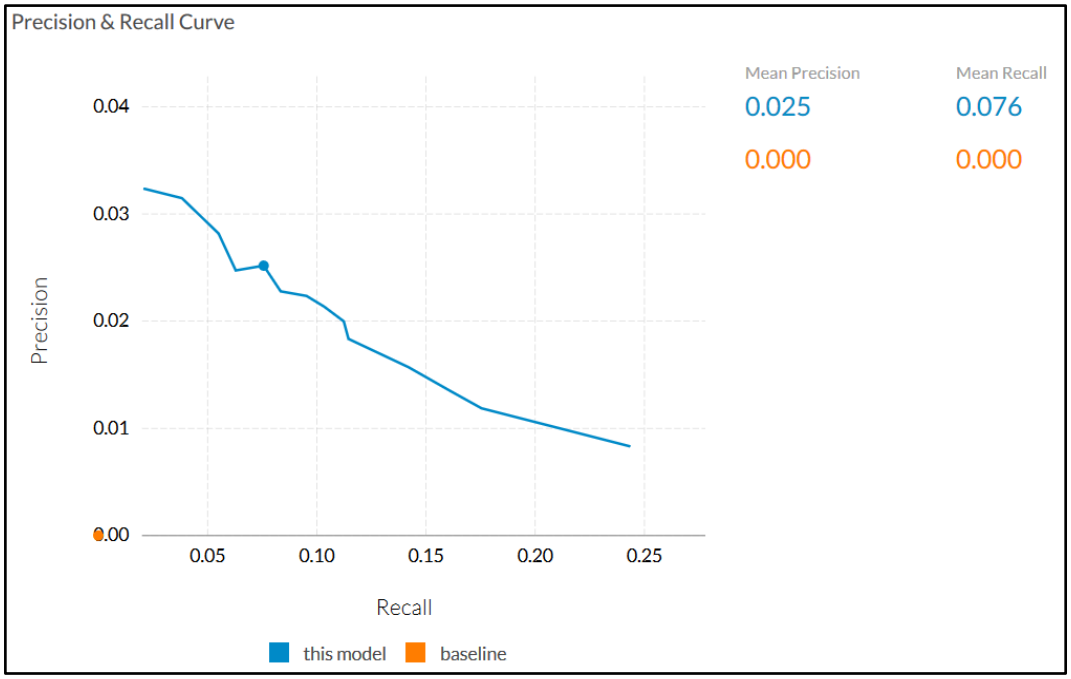
Popularity:



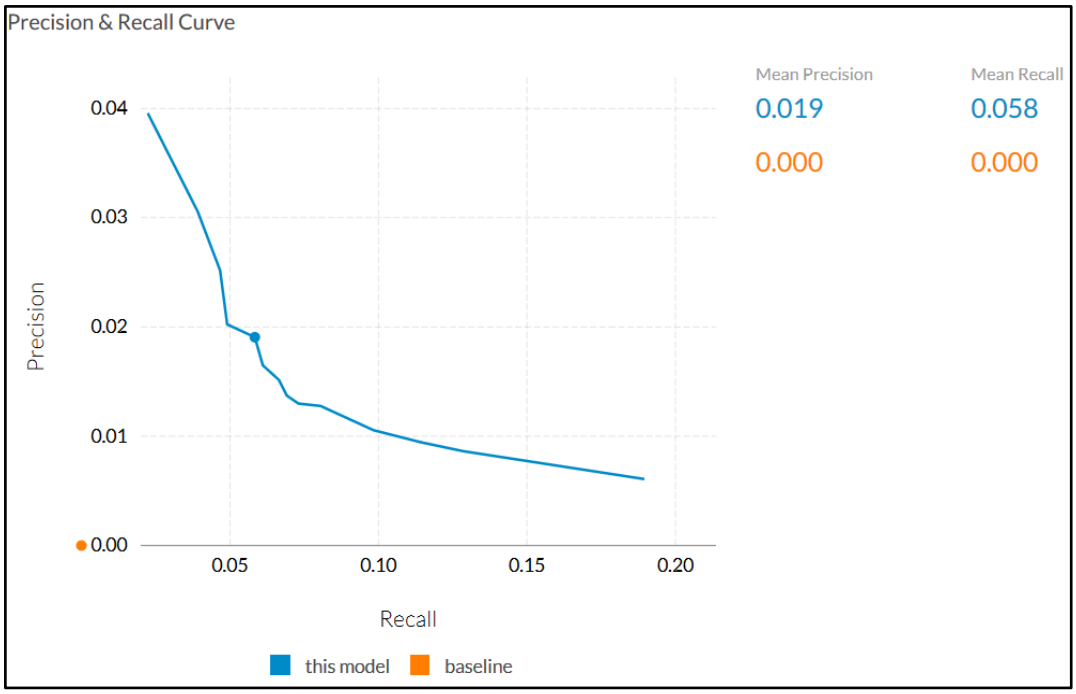
Amazon dataset:

ItemSimilarity

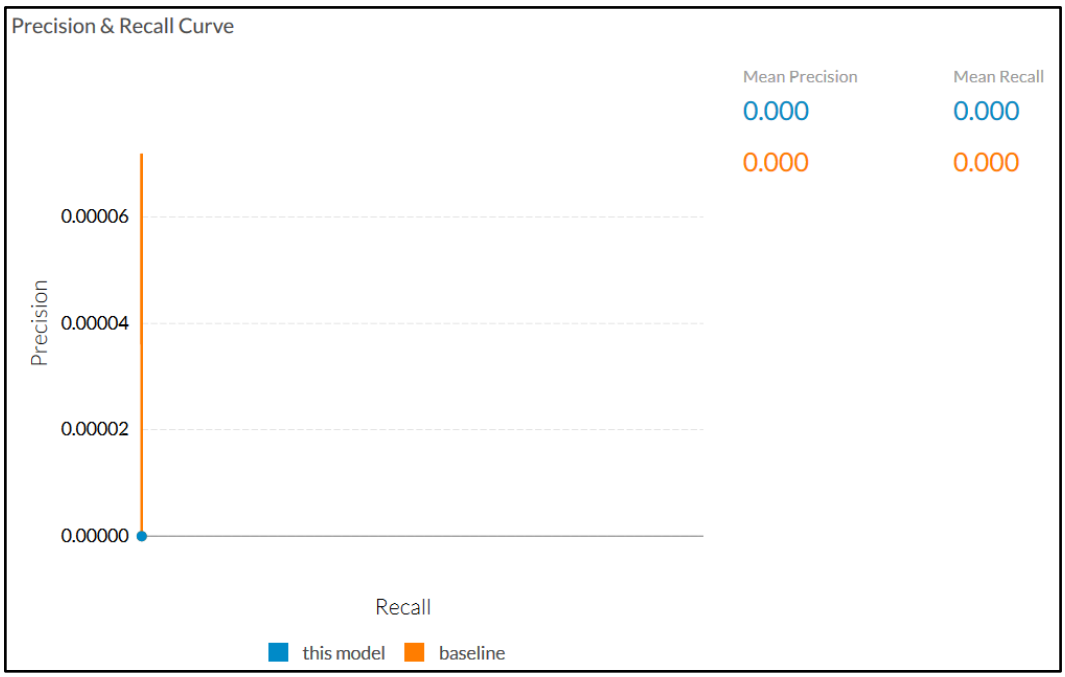
- **Jaccard**



- **Cosine**

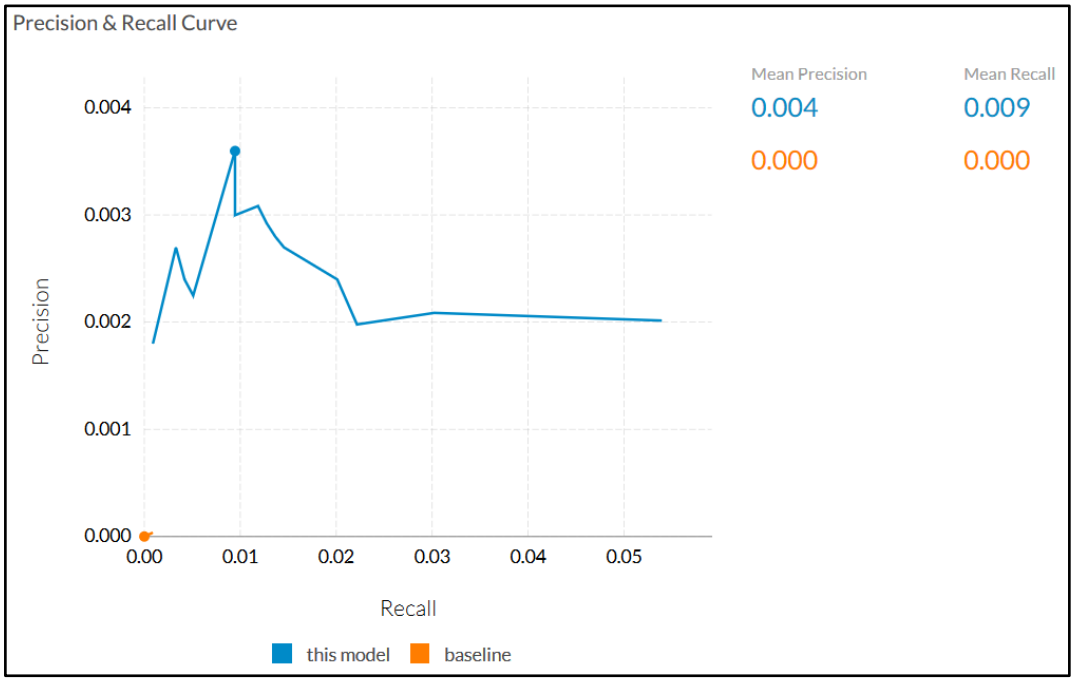


- **Pearson**

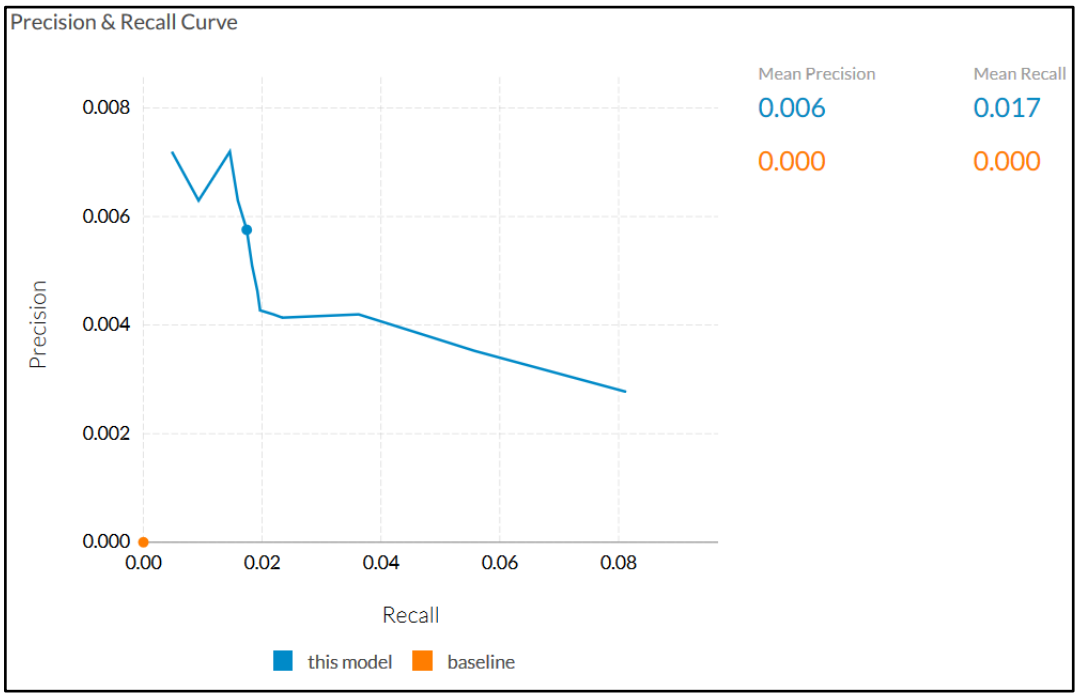


RankingFactorization

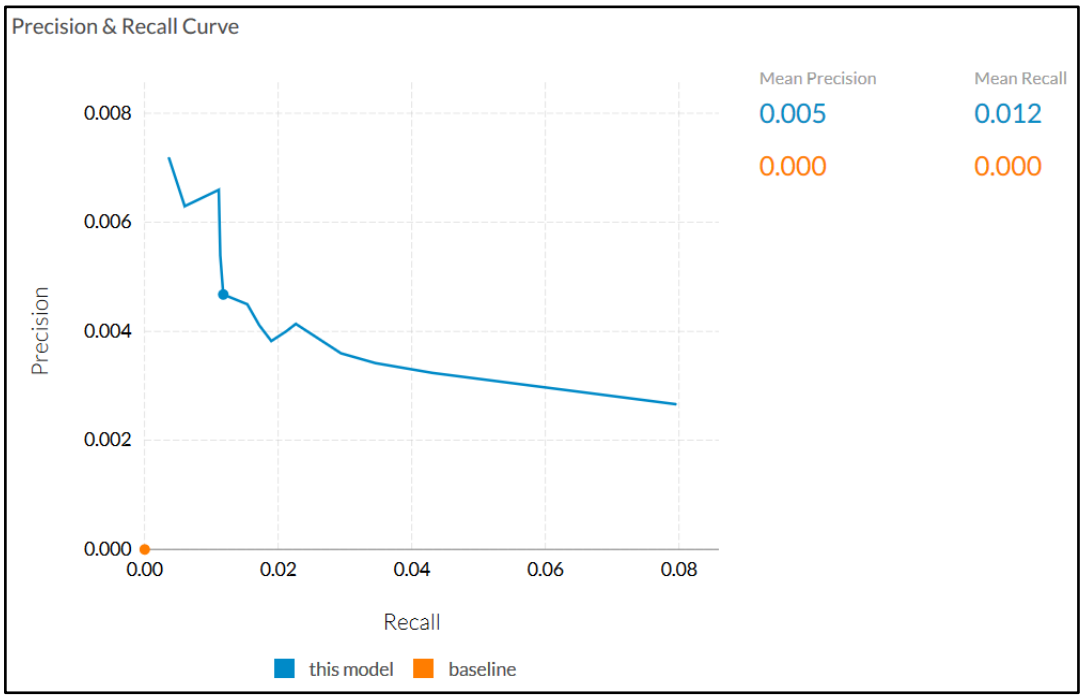
- **25 Iterationer 20.37s**



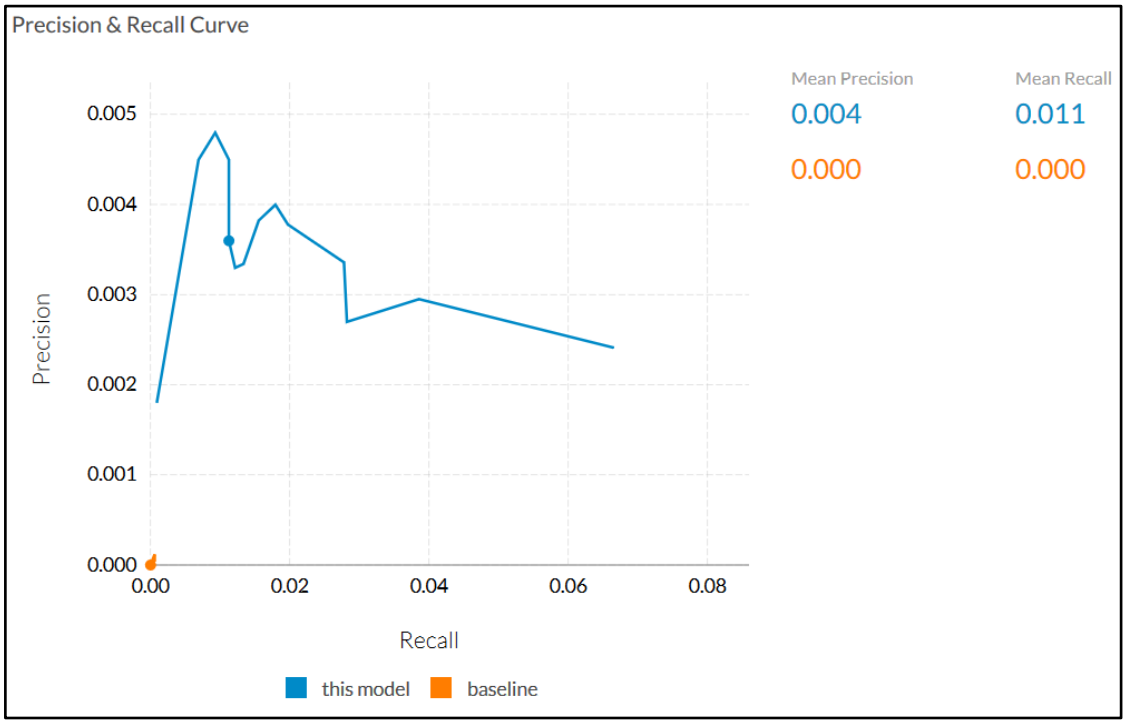
- **100 Iterationer 1m 13s**



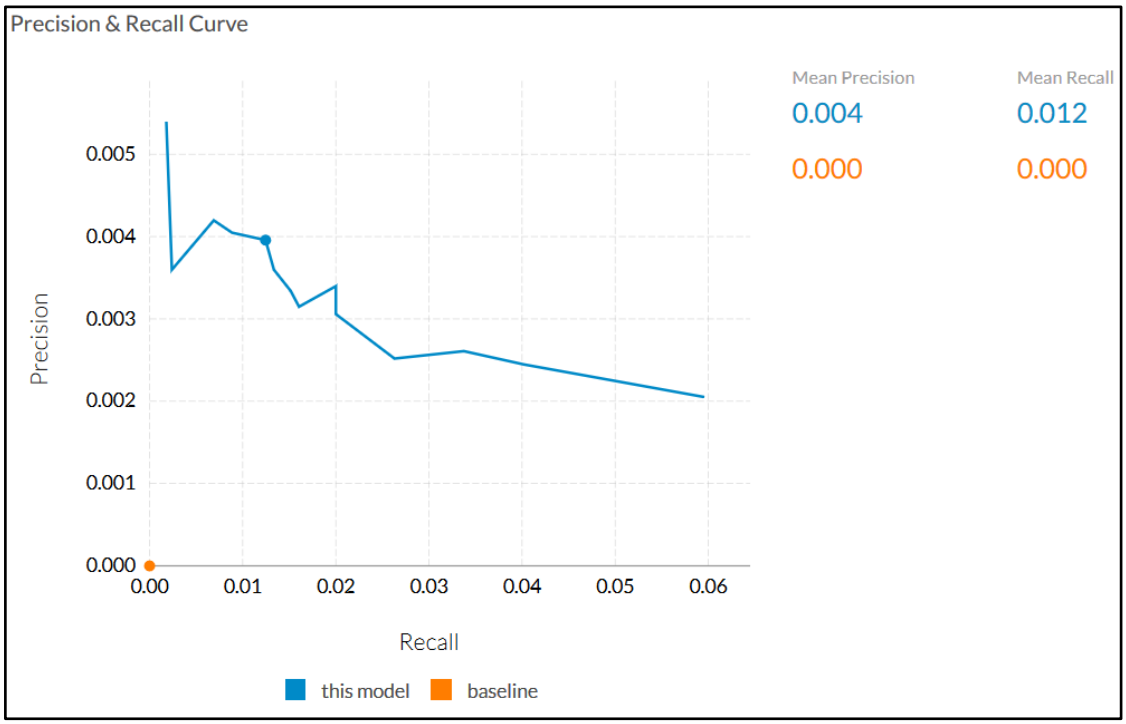
- **200 Iterationer 2m 35s**



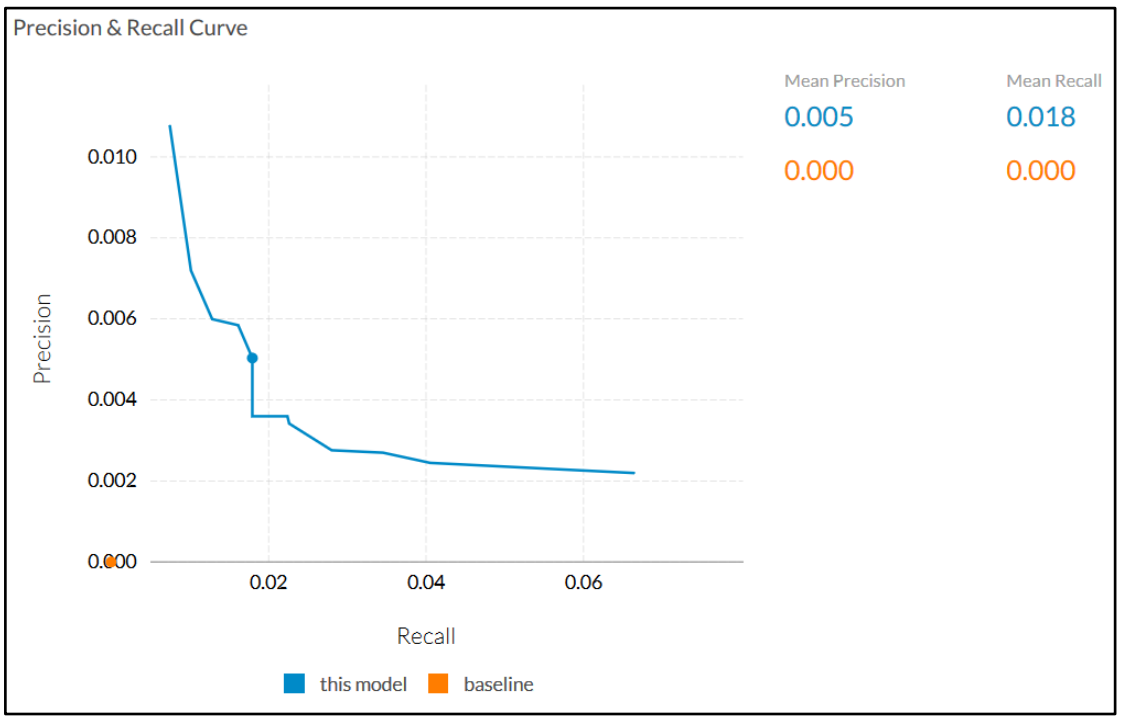
- **500 Iterationer 6m 12s**



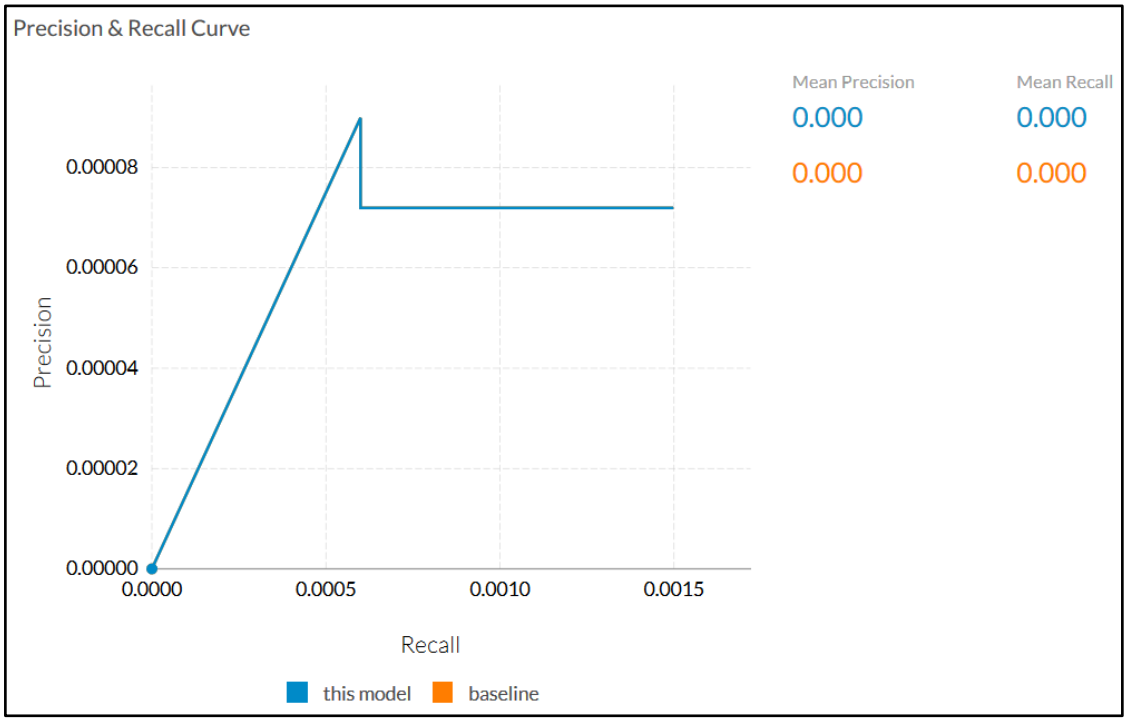
- **1000 Iterationer 11m 26s**



- **2000 Iterationer 21m 1s**



Popularity


Precision & Recall Curve
Mean Precision
0.000
0.000
Mean Recall
0.000
0.000
0.00008
0.00006
0.00004
0.00002
0.00000
Precision
0.0000
0.0005
0.0010
0.0015
Recall
this model
baseline

Bilaga 3

Datasetens variabler med förklaring visas här.

Amazon dataset:

Produktdata

Variabel	**Förklaring**
ASIN	(Amazon Standard Identification Number), Unikt 10 tecken ID för Amazonvaror.
Brand	Varumärket för varan.
Categories	Varans tillhörande kategorier. Exempel:”Xbox 360”,”Accessories”,“Kids & Toys”.
Description	Kort beskrivning på varan.
Price	Pris i Dollar
Related	Listor med relaterade varor i ASIN format i former av Buy_after_viewing(Lista), also_bought(Lista), bought_together(Lista), also_viewed(Lista).
SalesRank	Huvudkategori. Nästan alltid “Videogames” i detta arbetet.
Titel	Titel på varan. Saknas i 48687(~96%) av alla spelprodukter.

Recensionsdata

reviewerID	Recensentens användar-ID.
ASIN	Finns med i båda för att koppla recensioner till spel.
reviewerName	Användarens alias namn.
helpful	En tvåtals array med hur många som tyckt recensionen varit hjälpsam.
reviewText	Recensionstexten skriven av användaren.
Overall	Betyget för varan. Skala 1.0-5.0 med Increment på 0.5 steg.
Summary	Textsammanfatting/titel på recensionen.
ReviewTime	Klockslaget när recensionen gjordes. Finns både i Unix och Raw.

IGDB dataset:

Speldata

Game-ID	ID för spelet.
Name	Namn/Titel för spel.
Genres	Spelgenrer Exempel:Racing, Shooter, Adventure, Strategy, Platform 30265(~32%) av spel saknar genrer.
Themes	Temat för spelet Exempel: Action, Fantasy, Science Fiction, Sandbox, Stealth 53594(~57%) spel saknar tema.
Keywords	Taggar/Nyckelord som användare själva skriver in. 44989(~48%) saknar taggar.

Recensionsdata

User-ID	Användar-ID (Anonymiseras för att behålla användarnas information privat på önskemål av företaget).
Game-ID	ID för spelet.
Rating	Betygskala på 1-10 increment 1,0. Med undantag till 1828 recensioner som har decimal 0,1 increment.
Created_at	Timestamp UTC i format YYYY-MM-DD hh-mm-ss.