TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karl Martin Uustalu 179132IACB

MUUSIKASOOVITUSSÜSTEEMI REALISEERIMINE K-LÄHIMA NAABRI ALGORITMIGA

Bakalaureusetöö

Juhendaja: Kadri Umbleja
PhD

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Karl Martin Uustalu

18.05.2020

Annotatsioon

Antud bakalaureusetöö eesmärgiks on uurida soovitussüsteemide ajalugu ja tööpõhimõtteid, vaadelda nende rakendamist muusikarakendustes ning luua primitiivne sisupõhine soovitussüsteem autori enda muusikaliste eelistuste põhjal.

Töö teoreetilises osas luuakse lühitutvustus soovitussüsteemide ajaloo kohta, antakse ülevaade soovitussüsteemide metoodikatest nagu sisupõhine filtreerimine ja kaasfiltreerimine ning vaadeldakse soovitussüsteemide rakendamist kolmes tänapäeval laialdaselt levinud muusikarakenduses.

Töö praktilises osas kogutakse autori muusikaeelistuste kohta andmeid kasutades Spotify API-t. Seejärel töödeldakse andmed Pythoni programmeerimiskeelt ja Anaconda keskkonda kasutades ning luuakse k-lähima naabri algoritmi baasil soovitussüsteem. Sisupõhisel filtreerimisel üles ehitatud soovitussüsteem soovitab autori andmete põhjal uusi laule. Soovitussüsteemi realiseerimisel vaadeldakse kolme erinevat lahendust.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 24 leheküljel, 9 peatükki, 8 joonist, 2 tabelit.

Abstract

IMPLEMENTATION OF A MUSIC RECOMMENDATION SYSTEM USING K-NEAREST NEIGHBORS ALGORITHM

The aim of this thesis is to give an overview of recommendation systems through their history and implementations, as well as design a basic recommendation system based on the data on musical preferences of the author.

The theoretical part of the thesis will present a short introduction to recommendation systems, looking at their history and implementation methods such as content-based filtering and collaborative filtering. The thesis will also touch on the usage of recommendation systems in three different modern musical platforms.

The practical part of the thesis consists of gathering data on the musical preferences of the author, using the Spotify API and Python programming language. The data will be processed using the Anaconda framework and a basic content-based recommendation system will be built, to make recommendations based on the processed data. The system will use the k-nearest neighbors algorithm. Three different approaches will be analyzed when implementing the recommendation system.

The thesis is in Estonian and contains 24 pages of text, 9 chapters, 8 figures, 2 tables.

Lühendite ja mõistete sõnastik

tf-idf	*term frequency-inverse document frequency*, statistiline mõõt, mida kasutatakse infootsingus sõnade tähtsuse hindamiseks
otsustuspuu	Puu tüüpi diagramm, mis kirjeldab otsuseid ja tagajärgi
tehisnärvivõrk	Bioloogilise aju närvivõrkude eeskujul ehitatud arvutuslik arhitektuur
Naiivne Bayesi klassifikaator	Tõenäosusmudel, milles rakendadakse Bayesi teoreemi tunnuste vahel
konvolutsiooniline närvivõrk	Mitmekihiline tehisnärvivõrkude klass
Billboard 200	Ajakirja Billboard poolt välja antav iganädalane muusikaedetabel, kus on USA 200 populaarseimat stuudioalbumit ja EP-d
API	*Application Programming Interface*, rakendusliides
SQLite	Vabavaraline manustatud andmebaasihaldur
db	*Database*, andmebaasi failivorming
csv	*Comma Separated Values*, komaeraldusega väärtused ehk failivorming, kus andmebaasikirjed on eraldatud komadega
quicksort	Kiirsortimine, võrdlussortimise algoritm

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Infoajastu, ning sellega kaasnenud tehnoloogiline revolutsioon, on avanud inimestele võimaluse produkte ja teenuseid tarbida läbi interneti. Kui tavalises kaupluses saab teenindaja kauplejaga otse suhelda ning vastavalt tema eelistustele anda nõu ja teha soovitusi, siis internetipoodide ja -teenuste üheks probleemiks on olnud sarnase personaliseeritus taseme puudumine. Selle probleemi lahenduseks on välja käidud soovitussüsteemide kontsept, mis tänapäeval leiab laialdast rakendust üle kogu interneti.

Mida enam on kasvamas nii andmete mahud kui ka andmete mitmekesisus, seda rohkem tuleb esile vajadus andmeid filtreerida, et leida nende hulgast kõige tähtsamad ning sobivamad. Seega, võib prognoosida, et soovitussüsteemide kasutus tulevikus ainult kasvab ja nende tähtsuse osakaal, e-kommertsi valdkonnas, suureneb veel rohkem.

Antud bakalaureusetöö autor on otsustanud uurida soovitussüsteeme ning täpsemalt nende rakendamist muusikarakendustes. Töö kontseptsioon kujunes välja autori ideest ühendada tema huvid andmetöötluse ja muusika vastu. Töö põhiliseks osaks on, rakendades omandatud teadmisi, primitiivse muusikasoovitussüsteemi realiseerimine.

Töö teises peatükis annab autor ülevaate soovitussüsteemide kontseptsioonist ning ajaloost. Kolmandas peatükis uurib autor lähemalt soovitussüsteemide tööpõhimõtteid ja metoodikaid, vaadeldes sisupõhist filtreerimist, kaasfiltreerimist ning hübriidsüsteeme. Neljandas peatükis vaatleb autor soovitussüsteemide rakendamist muusikaplatvormides nagu Last.fm, Spotify ja Pandora Radio. Viiendas peatükis kirjeldab autor muusikasoovitussüsteemi realiseerimist enda kuulamisharjumuste baasil, sealhulgas andmete kogumise protsessi ning soovitussüsteemi realiseerimisel k-lähima naabri algoritmi kasutust. Seitsmendas peatükis analüüsib autor saadud tulemusi.

2 Soovitussüsteemid

Soovitussüsteemide tööpõhimõte on pakkuda kasutajale soovitusi mingite toodete või teenuste kohta, mis võiksid talle huvi pakkuda. Populaarsemad soovitusüsteemid on näiteks Netflixi soovitussüsteem, mis genereerib kasutajale varasemalt vaadatud filmide/sarjade põhjal, mis talle meeldisid, uusi soovitusi või Spotify Discover Weekly soovitussüsteem, mis iga nädal kasutajale, tema kuulatud laulude põhjal, uue esitusloendi genereerib. Soovitussüsteemid näevad tänapäeval väga laialdast kasutust erinevates veebiplatvormides, e-poodides, suhtlusvõrgustikes, jne.

Teenusepakkujate jaoks, sisaldavad soovitussüsteemid erinevaid kasutegureid, nagu kasutajate parem mõistmine, rahulolu kasv pakutava teenusega ning nendega kaasnev müüginumbrite ja püsiklientide arvu tõus [1].

2.1 Soovitussüsteemide ajalugu

Juba 90ndate alguses, kui interneti populaarsus kogus hoogu, tekkis inimestel, informatsiooni kasvava mahu tõttu, vajadus dokumente ja e-kirju filtreerida. Xeroxi Palo Alto Uurimiskeskuses (PARC) otsustas grupp teadlasi moodustada uurimisrühma, eesmärgiga arendada eksperimentaalne e-posti süsteemi nimega Tapestry. Lisaks tüüpilisele e-posti hoiustamisele oli Tapestry üheks põhifunktsiooniks ka informatsiooni filtreerimine. Tapestry filtreeris esmalt sissetulevad dokumendid vastavalt kasutaja eelistustele ning seejärel organiseeris ja prioritiseeris valitud dokumendid nii, et esmatähtsus oli kõrge prioriteediga dokumentidel [2].

Tapestry oli esimene filtreerimissüsteem, kus kasutati kaasfiltreerimist (*collaborative filtering*). Kasutajad, kes lugesid dokumente, said neid ka annoteerida ning nende kasulikkust hinnata. Seejärel võimaldas Tapestry kasutajatel teha dokumendipäringuid, mitte ainult sisu alusel, vaid ka teiste kasutajate tagasiside alusel. Teadlased jõudsid järeldusele, et informatsiooni filtreerimine on efektiivsem siis, kui inimesed on ise protsessi kaasatud [3]. Tapestry kasutus piirdus küll vaid PARC-i teadlastega, kuid kaasfiltreerimist kasutab tänapäeval suur osa populaarsemaid soovitussüsteeme.

Interneti ja e-kaubanduse kasvuga tõusis huvi ka isikupärastatud soovituste vastu ning Minnesota Ülikooli teadlaste poolt asutatud soovitussüsteemidele suunatud teadusosakonnast GroupLens arenes välja 90ndate lõpus ettevõte Net Perceptions, mille tegevusalaks oli erinevate isikupärastatud süsteemide arendus. Net Perceptionsi üheks esimeseks kliendiks oli sel ajal internetis raamatute müügiga tegelev ettevõte Amazon. Net Perceptionsi poolt arendatud GroupLensi tarkvara võimaldas Amazoni klientidel, kaasfiltreerimist kasutades, raamatuid hinnata ning nende hinnangute järgi uusi soovitusi leida [4].

GroupLensi teadlased tegelesid lisaks Net Perceptionsi loomisele uue filmindusele suunatud soovitussüsteemi MovieLens arendamisega. MovieLens kasutas erinevaid soovitusalgoritme, et kasutajatele soovitada filme tema varasemate hinnangute põhjal. Tänaseks on nii MovieLens platvorm kui ka MovieLens-i andmehulkade kasutus osutunud väga populaarseks [5].

Üheks suurimaks soovitussüsteemide arengu seemneks 21. sajandil oli Netflixi poolt korraldatud Netflix Prize võistlus. Netflix Prize ülesandeks oli arendada Netflix Cinematch soovitussüsteemist vähemalt 10% täpsem soovitussüsteem. Võistluse peaauhinnaks oli miljon dollarit. Võistlus algas 2006. aastal ning kestis pea kolm aastat. Netflixi poolt anti osalistele kasutada massiivne andmehulk, mis koosnes 100-st miljonist hinnangust 17 770-le erinevale filmile 480 189 kasutaja poolt. Esitatud algoritmide hindamiseks kasutati standardhälvet. Iga aasta andis Netflix ka 50 000 dollari suuruse auhinna meeskonnale, kelle algoritm oli kõige rohkem arenenud võrreldes eelmise aastaga [6].

Võistlus võideti 2009. aastal, kui meeskonna BellKor Pragmatic Chaos, mida juhtisid teadlased AT&T Labs Uurimiskeskusest, algoritm ületas 10% künnise 0,06% võrra ning saavutas standardhälve väärtusega 0,8567. Meeskond nimega The Ensemble saavutas küll täpselt sama hea tulemuse, kuid esitas oma algoritmi 20 minutit hiljem kui BellKor Pragmatic Chaos ning selle tõttu pidi leppima teise kohaga [6] [7].

3 Soovitussüsteemi metoodikad

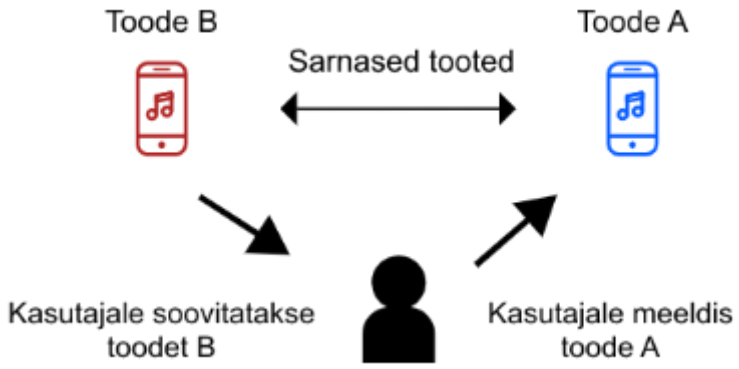
Soovitussüsteem teeb valikuid ennustades kasutaja eelistusi tema kohta varasemalt kogutud andmete põhjal. Kaks kõige levinumat metoodikat, mis soovitussüsteemides rakendust leiavad on sisupõhine filtreerimine (*content-based filtering*) ja kaasfiltreerimine [8].

3.1 Sisupõhine filtreerimine

Sisupõhine filtreerimine vaatleb mingi objekti kindlaid omadusi ning juhul kui see objekt kasutajale meeldis soovitab uusi objekte, millel on kasutajale meeldinud objektiga sarnased omadused või karakteristikud. Näiteks, kui kasutajale meeldis mõni film, võib sisupõhine filtreerimine soovitada kasutajale uut filmi, millel on sama žanr, sama peanäitleja, sama režissöör või muu ühine iseloomujoon [8].

Üldjuhul on sellise süsteemi tugevuseks see, et ei ole vaja andmeid teiste kasutajate kohta ning valikuid saab teha kasutaja enda spetsiifiliste eelistuste põhjal. Nõrkuseks see, et kõikidel objektidel peavad olema kindlalt määratud karakteristikud ning kui kasutaja pole varasemalt teenust kasutanud ei ole võimalik teha ennustusi [9]. Seda nimetatakse *cold-start* probleemiks.

Sisupõhisele filtreerimisele võib ühest küljest läheneda kui infootsingu ülesandele, kus objekte kirjeldavad omadused teisendatakse *tf–idf* sõnavektoriteks ja esitatakse vektorruumis. Vektorite koosinussarnasuse põhjal soovitatakse uus objekt. Alternatiivselt saab sisupõhist filtreerimist vaadelda kui klassifikatsiooni ülesannet ning rakendada klassifikatsioonialgoritme, nagu näiteks k-lähima naabri algoritmi, otsustuspuid või tehisnärvivõrke [10]. Sisupõhist filtreerimist illustreeriv näide on toodud Joonises 1.



Joonis 1. Sisupõhise filtreerimise näide

3.2 Kaasfiltreerimine

Kaasfiltreerimine vaatleb mitme eri kasutaja hinnanguid objektidele ning leides nende vahel sarnasusi genereerib kasutajale soovitusi. Kasutajate poolt antud hinnangud jagunevad otsesteks ja kaudseteks. Otsesed hinnangud on kasutaja reaalsed hinnangud objektile, näiteks numbrilisel skaalal. Kaudsed hinnangud on andmed, mis on kogutud kasutaja interaktsioonist objektiga, nagu näiteks kui palju kasutaja objektile klõpsanud on, või kui kaua ta objekti vaadanud on [11].

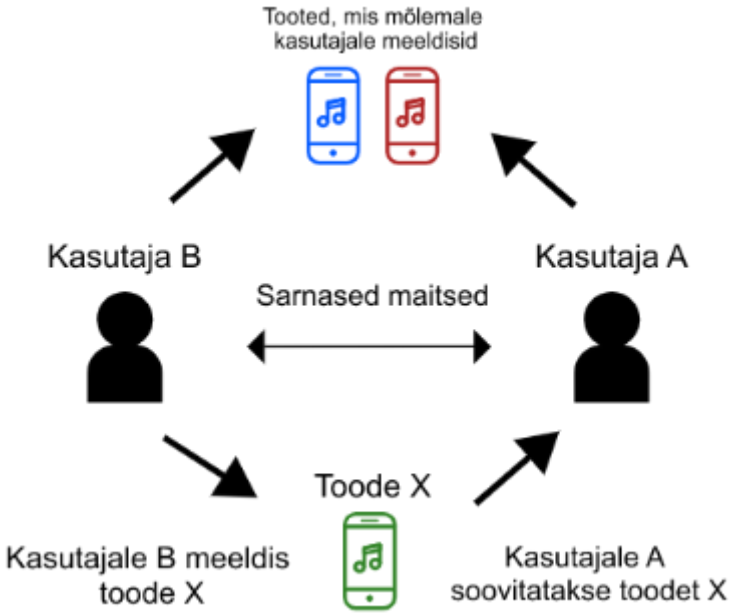
Kaasfiltreerimine jaguneb omakorda kaheks: naabruskonnapõhine meetod ja mudelpõhine meetod. Naabruskonnapõhine meetod seisneb selles, et võetakse grupp kasutajaid, ning määratakse neile kaal, mingi vaadeldava kasutaja eelistustega sobivuse suhtes. Sellest grupist valitakse omakorda osa kasutajaid, kellel on kõige suuremad sarnasused vaadeldava kasutaja eelistustega, ehk moodustatakse naabruskond. Naabruskonna liikmete eelistuste kombinatsiooni järgi tehakse soovitus vaadeldavale kasutajale [10].

Kuna naabruskonnameetod ei ole hea skaleeritavusega, algoritmide keerukuse tõttu, kasutatakse suuremahuliste andmetega töötlemisel erinevaid laiendusi. Üheks neist on näiteks objektpõhine kaasfiltreerimine, kus sarnaste kasutajate leidmise asemel leitakse kasutaja poolt hinnatud objektidele teisi sarnaseid objekte [10].

Mudelpõhised meetodid üritavad ennustada kasutaja hinnangut objektile, kasutades masinõppe ja andmekaeve algoritmidel üles ehitatud mudeleid. Üheks populaarseimaks mudelpõhiseks meetodiks on maatriksi faktoriseerimise mudel (*matrix factorization*) [10].

Maatriksi faktoriseerimise käigus luuakse maatriks, kus on kaardistatud objektid ja kasutajad vektorite kaudu, mille elemendid on objektide hindamismustritest järeldatud faktorid. Kasutajate ja objektide faktorite kõrge vastavuse alusel tehakse soovitusi. Selline meetod on hea skaleeritavuse, täpsuse ja paindlikusega. Maatriksi faktoriseerimine on kasutusel näiteks Netflixi filmide ja seriaalide soovitamisel [12].

Kaasfiltreerimise tugevuseks on see, et ei ole tingimata vaja eelnevaid andmeid objekti kohta ning tänu kollaboratsioonile, saab selline süsteem hästi soovitada täiesti uusi objekte, millel ei ole varasemaid seoseid kasutajaga. Nõrkuseks see, et mudel ei suuda soovitada täiesti uusi objekte kui neil pole veel ühtegi hinnangut. Kaasfiltreerimist illustreeriv näide on toodud Joonises 2.



Joonis 2. Kaasfiltreerimise näide

3.3 Hübriidsed lähenemised

Nii sisupõhise kui ka kaasfiltreerimise tugevuste ära kasutamiseks on loodud hübriidsüsteemid, kus kasutatakse aspekte mõlemast meetodist. Üheks lihtsamaks näiteks on mõlema meetodi rakendamine eraldi ning lõpuks kõikide saadud soovituste kokku kogumine. Keerulisem süsteem on näiteks Prem Melville poolt pakutud kaasfiltreerimine sisutoetusega, kus sisufiltreerimise tulemusel saadud soovituste baasil teisendatakse hõre kasutajamaatriks hinnangute maatriksiks ning seejärel kasutatakse kaasfiltreerimisel Naiivset Bayesi klassifikaatorit lõppsoovituste tegemiseks.

Paljud hübriidsed süsteemid lähenevad soovituste tegemisele kui klassifitseerimisülesandele ning mõned üritavad otse ühendada sisupõhise ja kaasfiltreerimise sama raamistikku. Hübriidsed süsteemid aitavad lisaks täpsuse suurendamisele parandada ka eelmainitud *cold-start* probleemi. Tänapäeval leiavad erinevad hübriidsed meetodid laialdast kasutust soovitussüsteemide implementeerimisel [10].

4 Soovitussüsteemid muusikarakendustes

Tänapäeval tarbitakse muusikat peamiselt voogedastusrakenduste kaudu nagu Spotify, Apple Music, Pandora Radio, jne. Lisaks tavalisele muusikakuulamise võimalusele on voogedastusrakenduste üheks ülesandeks ka kasutajale uue muusika soovitamine. Selle tõttu on valdavalt igas voogedastusrakenduses kasutusel soovitussüsteemid. Muusika soovitamiseks kasutatakse rakendustes nii kaasfiltreerimisel põhinevaid süsteeme, sisupõhiseid filtreerimissüsteeme ning ka hübriide mõlemast.

Peamiselt on siiski muusikasoovitussüsteemid kas hübriidsed või sisupõhise filtreerimisega, sest voogedastusrakendustes on pigem kättesaadavad laulude kindlad karakteristikud, kui kasutajate hinnangud. Laulude karakteristikuteks võivad olla kas informatsioon, mis on kogutud laulu helisignaalist, või laulu metaandmed [1]. Metaandmeteks võivad olla näiteks annotatsioonid nagu žanr, artist, albumi pealkiri, plaadifirma ja muu tooteinfo, kuid annotatsioonide hulka võivad kuuluda ka muusikat kirjeldavad omadused nagu näiteks helilaad ja meeleolu. Annotatsioonid on kättesaadavad muusikaekspertide loodud andmebaasidest. Lisaks ekspertide loodud annotatsioonidele võivad metaandmeteks olla ka kasutajate poolt laulule antud märksõnad [1].

Metaandmetele lisaks võimaldab helisignaali analüüs koguda informatsiooni laulu akustiliste tunnuste kohta, mis võivad olla näiteks laulu tämbrit, tonaalsust, tempot või helivaljusust kirjeldavad karakteristikud [1].

4.1 Last.fm soovitussüsteem

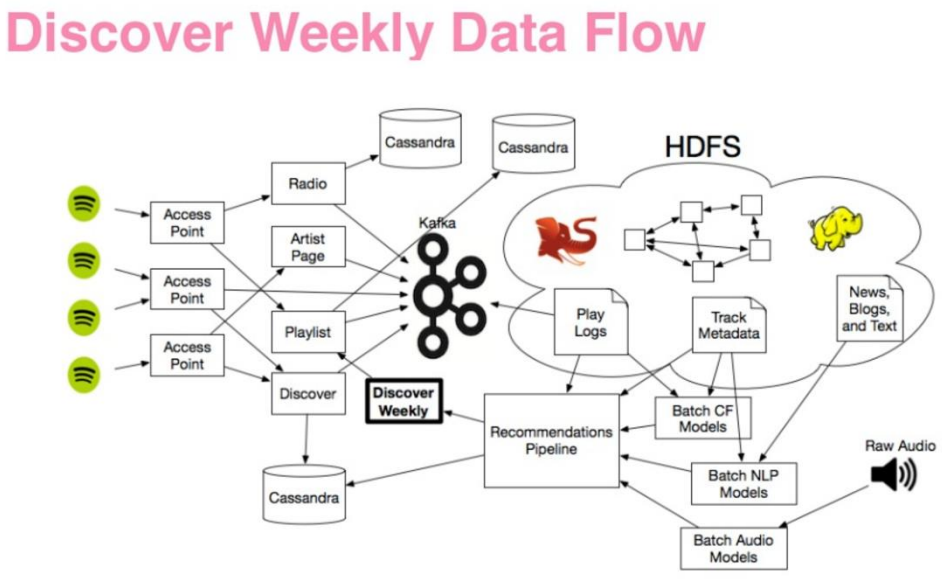
Last.fm on veebiplatvorm, mida võib kirjeldada muusikale suunatud sotsiaalvõrgustikuna. Last.fm salvestab kasutaja poolt, läbi erinevate muusikaplatvormide, kuulatud laulud ning kuvab kasutaja muusika kuulamisharjumused tema profiilil [13].

Last.fm põhineb kaasfiltreerimisel, ehk soovitusi tehakse kasutajate põhjal, kellel on sarnased kuulamisharjumused. Üheks Last.fm-i funktsiooniks on ka kasutajate

omavahelise sobivuse kuvamine kuulatud artistide põhjal. Lisades selle kõigele juurde ka sotsiaalvõrgustiku aspekti on tegemist vägagi võimsa ja efektiivse soovitussüsteemiga. Soovitussüsteemi tööpõhimõtet võib lihtsustatult kujutada kui triaadilist sulgemist – kui kasutajal A on kasutajaga B sarnased muusikamaitsed on nende vahel tugev side. Olgu näiteks, et kasutaja B kuulab väga palju artisti C, ehk ka nende vahel on tugev side. Kasutaja A ei ole artisti C kunagi kuulanud, kuid tugevatest sidemetest kasutaja A ja B ning kasutaja B ja artisti C vahel, võib järeldada, et tõenäoliselt meeldiks artist C ka kasutajale A [13].

4.2 Spotify soovitussüsteem

Spotify on üks populaarsemaid ja enimkasutatumaid muusika voogedastusplatvorme maailmas. Spotify soovitussüsteemi lipulaevaks on Discover Weekly. Discover Weekly on igale kasutajale personaalselt loodud esitusloend, kus kasutaja kuulamisharjumuste baasil genereeritakse iga nädal 30 uut laulu, mis talle meeldida võiksid. Spotify soovitussüsteem on hübriid erinevatest meetoditest. Kasutusel on kaasfiltreerimine, loomuliku keele töötlus ning ka helisignaali analüüs [14]. Spotify Discover Weekly andmevoo diagramm on esitatud Joonisega 3.



Joonis 3. Spotify Discover Weekly esitusloendi andmevoog [14].

Spotify kaasfiltreerimine põhineb kaudsetel hinnangutel, sest rakendus ei võimalda kasutajatel muusikat otseselt hinnata. Seega vaatab Spotify kui palju kasutaja mingit

laulu kuulanud on või kas kasutaja on laulu esitusloendisse lisanud. See võimaldab leida kasutajate kuulamisharjumuste vahel sarnasusi ning rakendada soovituste leidmiseks maatriksi faktoriseerimist [15].

Lisaks kaasfiltreerimisele on Spotify soovitussüsteemis kasutusel ka NLP (*Natural Language Processing*) ehk loomuliku keele töötlus. NLP kasutus Spotify-s seisneb selles, et analüüsitakse ja töödeldakse laule kirjeldavaid omadussõnu, karakteristikuid, metaandmeid ja muud teksti mida laulu kohta internetis leidub [15].

Kolmandaks on Spotify soovituste täpsuse suurendamiseks kasutusele võtnud helisignaali analüüsi. Helisignaali analüüsi tugevuseks on see, et ei vajata muud informatsiooni peale laulu enda. Konvolutsioonilisi närvivõrke kasutades saadakse laulu helisignaalist kätte erinevad karakteristikud nagu taktimõõt, helistik, helilaad, tempo ja helivaljus [15].

4.3 Pandora Radio soovitussüsteem

Pandora Radio oli üks esimesi muusika voogedastusplatvorme, mis oma unikaalsuse tõttu, on tänapäeval saanud suurimaks internetiraadio laadseks teenuseks Ameerika Ühendriikides. Pandora põhimõte seisneb selles, et kasutajale tehakse raadiojaamad, kus mängitakse laule kasutaja eelistuste põhjal. Igas raadiojaamas saab kasutaja anda tagasisidet laulu kohta, lisades selle meeldivaks või mitte meeldivaks. Selle informatsiooni põhjal otsustab Pandora algoritm milliseid laule kasutajale edaspidi mängida ja milliseid mitte [16].

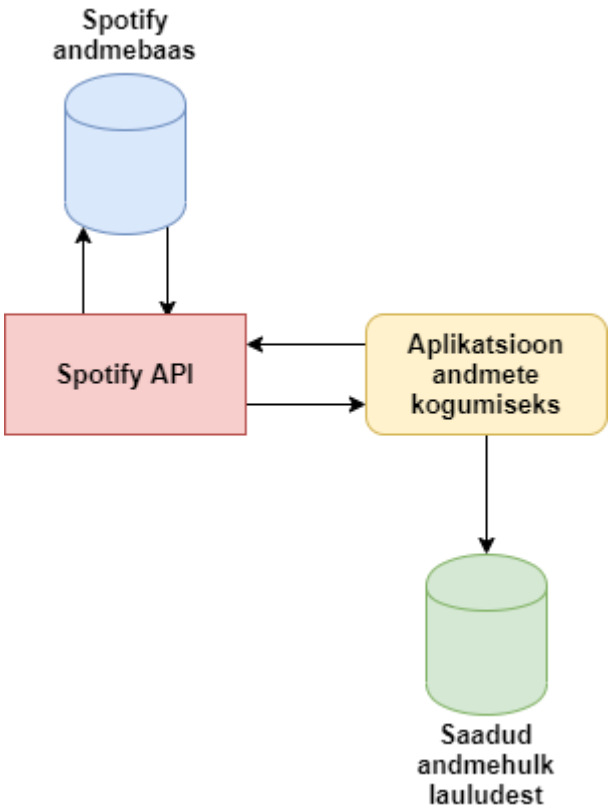
Pandora Radio põhineb tehnoloogial, mis töötati välja Pandora asutajate poolt Music Genome Projecti raames. Music Genome Projecti andmebaasis olevatele lauludele annavad kirjeldusi grupp muusikaanalüütikuid. Igal laulu on kuni 450 eri karakteristikut, mille põhjal algoritmid laule organiseerivad [17].
Music Genome Projecti kaudu, laulude kohta saada olevate metaandmete ja helitunnuste põhjal, koostatakse sõnavektorid ning võrreldakse vektorite koosinussarnasusi soovitussüsteemis rakendamiseks. Pandora soovitussüsteemis on kasutusel 70 eri algoritmi soovituste tegemiseks: 10 neist sisu analüüsimiseks, 40 kollaboratsiooni protsessimiseks ja ülejäänud 30 tegelevad isikustatud filtreerimisega [18].

5 Soovitussüsteemi realiseerimine

5.1 Soovitussüsteemi mudel

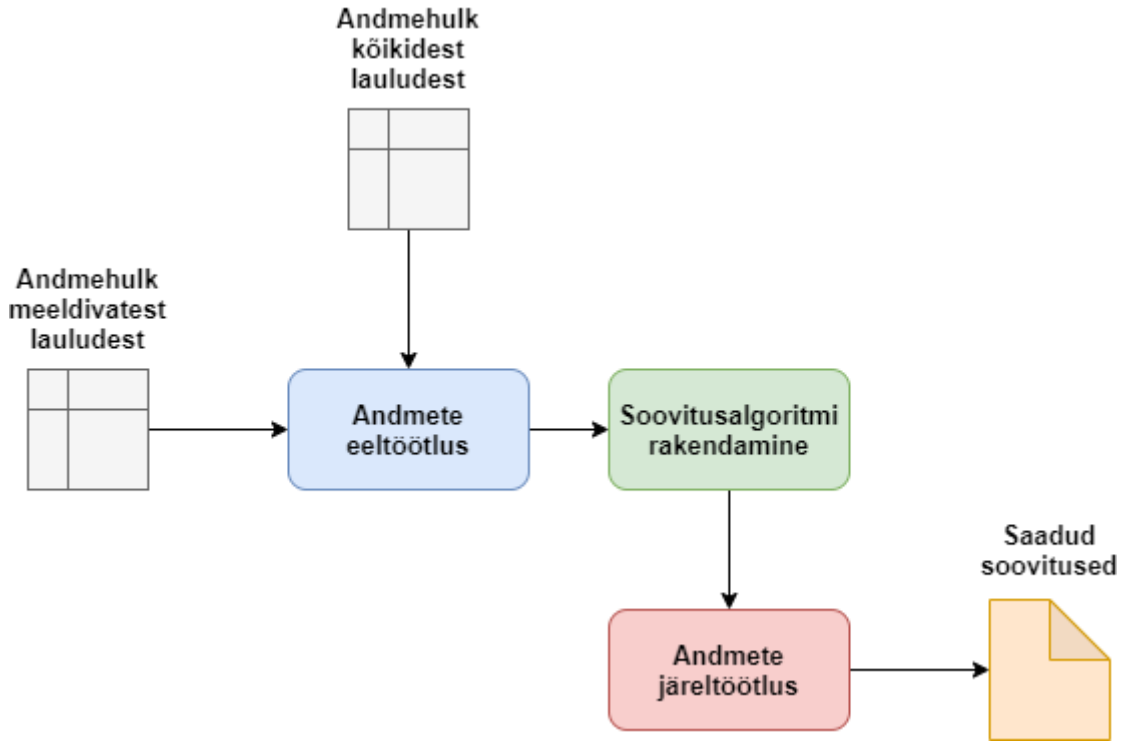
Lõputöö praktilise osa eesmärgiks on realiseerida primitiivne sisupõhine soovitussüsteem. Soovitussüsteem realiseeritakse Spotify API-t ning Pythoni andmetöötlus ja masinõppe teeke kasutades. Töö autor otsustas kasutada sisupõhist filtreerimist, sest soovitussüsteem realiseeritakse autori enda kuulamiseelistuste baasil ning soovituste tegemiseks analüüsitakse autorile meeldivate laulude sisulisi karakteristikuid. Soovitussüsteemi realiseerimise võib jagada kaheks osaks: andmete kogumine ning süsteemi realiseerimine.

Andmete kogumiseks kasutab autor Spotify API-t, mille kaudu on võimalik hankida talle meeldivad laulud, nende metaandmed ja helikarakteristikud (Joonis 4).



Joonis 4. Andmehulga kogumise diagramm

Soovitussüsteemi enda tööpõhimõte seisneb selles, et süsteemi antakse kaks andmehulka – andmehulk, kus on suur kogus laule, mille seast tehakse soovitusi ning andmehulk, kus on autorile meeldivad laulud. Ideaalis oleksid mõlemad andmehulgad sisult samas formaadis, ehk sisaldaksid samu tüüpi karakteristikuid. Saadud andmetega sooritatakse eeltöötlus ning valitakse tunnused, mille alusel soovitusi tegema hakatakse. Soovitussüsteemis kasutatakse sarnasuste leidmiseks laulude vahel k-lähima naabri algoritmi, analüüsides laulude vastavaid tunnuseid. Seejärel sooritatakse saadud andmetega järeltöötlus ning kuvatakse kasutajale saadud laulud ja neid esitavad artistid (Joonis 5).



Joonis 5. Soovitussüsteemi mudeli diagramm

5.2 Andmehulkade valimine ja kogumine

5.2.1 Andmehulkade valikud

Soovitussüsteemi realiseerimiseks on esmalt vaja võimalikult efektiivset andmehulka lauludest koos metaandmete ning helitunnustega. Sellest andmehulgast hakatakse tegema soovitusi.

Üheks võimaluseks on kasutada veebilehte, nimega Million Songs Dataset, kus on vabalt saadaval andmehulk, mis koosneb miljoni laulu helitunnustest ja metaandmetest. Suureks miinuseks on see, et andmehulk koostati aastal 2011 ning seega pole andmehulgas ühtegi laulu, mis on väljastatud viimase 9 aasta jooksul. Märkimisväärne on ka andmehulga tohutu suurus – 280 GB [19].

Teiseks variandiks leidis töö autor veebilehe, kus on saadaval Spotify-st kogutud andmehulk, mis sisaldab 1963. kuni 2019. aastal, Billboard 200 edetabelisse tõusnud albumitel olevate laulude helitunnuseid ja metaandmeid. Kokku on andmehulgas 340 000 erinevat laulu [20].

Kolmandaks võimaluseks on, Spotify API-t kasutades, ehitada üles ise sobiv andmehulk, kuid selle projekti raames oleks see lisatöö ning efektiivsem oleks kasutada olemasolevat andmehulka.

Lisaks üldisele andmehulgale on vaja ka kasutaja andmeid, mille põhjal soovitusi tegema hakatakse. Selle lõputöö raames võtab autor kasutusele andmehulga, mis põhineb ta enda kuulamisharjumustel.

Üheks sellise andmehulga hankimise võimaluseks oleks kasutada Spotify API-t, kuid API võimaldab eksportida vaid kasutaja viimased 50 kuulatud laulu. Kuulamisajaloo asemel on võimalik võtta kasutusele kasutaja enda tehtud esitusloendid, mille kättesaamist Spotify API võimaldab. Töö autor on teinud nii 2018 kui ka 2019 aasta kohta esitusloendid, kuhu on salvestanud oma lemmik lood, mida nendel aastatel kuulas. Esitusloendite peale kokku on salvestatud 350 lugu.

Üheks alternatiivseks võimaluseks, mida töö autor kaalus oma päris kuulamisajaloo kätte saamiseks, on veebikeskkond Last.fm. Töö autor on juba mitu aastat oma kuulamisajaloo salvestamiseks Last.fm-i kasutanud ning Last.fm pakub tasuta vabalt kättesaadavat API-t, mis võimaldaks kuulamisajaloo kohta andmehulga koostada [21]. Töö autori Last.fm profiilil on tema viimase seitsme aasta kuulamisajalugu, kus on kokku üle 40 000 kuulamise. Töö autor otsustas seda varianti mitte kasutada, sest ilmselt ei tuleks soovitussüsteem väga täpne kui kasutusel oleks terve kuulamisajalugu. Aastatega on autori muusikamaitse muutunud ning paratamatult sisaldab kuulamisajalugu ka laule, mis autorile tegelikult ei meeldinud.

5.2.2 Andmehulkade kogumine

Töö autor valis esimeseks andmehulgaks Billboard 200 edetabeli laulud [20]. Põhjuseks see, et andmete maht tundus sobiva suurusega ning andmed olid töötluseks kohaselt formuleeritud. Veebileht võimaldas alla laadida andmebaasi, milles sisaldus tabel *acoustic_features*, kus on 340 000 laulu metaandmed ja erinevad helitunnused. SQLite käsurea abil sai tabeli kergelt db (*database*) failist eksportida andmete töötluse jaoks sobilikku csv (*comma separated values*) faili.

Teiseks andmehulgaks otsustas autor võtta oma 2018 ja 2019 aastatel Spotify-s loodud esitusloendites olevad laulud. Nende kättesaamine osutus natuke keerukamaks kui esimese andmehulga puhul. Töö autor avastas Pythoni teegi nimega Spotipy [22], mis võimaldab Spotify Web API funktsioone kasutada Pythoni programmeerimiskeelega. Autori jaoks oli see esimene tõsisem kokkupuude päris API-ga.

Esmalt tuli autoril teegi kasutamiseks luua Spotify for Developers veebilehel aplikatsioon, mis võimaldab hankida vajalikud muutujad Spotify API-ga ühendamiseks. Seejärel lõi autor Pythoni skripti, milles kasutas Spotipy poolt pakutavaid funktsioone, mis võimaldasid kasutaja etteantud esitusloendist välja lugeda laulud, nende metaandmed ja erinevad helitunnused ning salvestada need esimese andmehulga kujul csv faili. Metaandmed ja helitunnused, mis iga laulu kohta salvestati on järgmised: laulu identifikaator, laulu nimi, albumi nimi, artisti nimi, akustilisus, tantsitavus, laulu pikkus, energia, instrumentaalsus, helistik, elujõulisus, helilaad, kõnemeelsus, tempo, taktimõõt, valents, albumi identifikaator ja albumi väljalaske kuupäev.

Skripti saab tööle panna käsurealt andes argumentideks kaasa oma Spotify kasutaja identifikaatori ning esitusloendi identifikaatori, milles olevaid laule tahetakse kätte saada.

5.3 Andmehulkade eeltöötlus

Soovitussüsteemi mudeli realiseerimiseks kasutas töö autor Anaconda raamistikku, mis on spetsiaalselt loodud Pythonis andmetöötluse ja masinõppe eesmärgil. Andmete töötlemiseks kasutas autor Pandas ja NumPy teeke. Pandas on andmetöötluse teek, mis on mõeldud lihtsustamaks andmete manipulatsiooni ja analüüsi Pythonis. Numpy on

teaduslikeks arvutusteks mõeldud teek, mis lihtsustab Pythonis andmemassiivide ja maatriksite kasutamist.

Autori loodud skript loeb esimesena sisse mõlemad csv failid, kasutades Pandas *read_csv* funktsiooni ning salvestab need eraldi *dataframe* tüüpi muutujatesse *songs_df* ja *playlist_df*. Kasutades Pandas *dropna* funktsiooni loeb skript mõlemad andmehulgad üle, ning juhul, kui kummagi andmetabeli mõnes reas on tühi väärtus, eemaldab funktsioon terve rea tabelist.

Järgmiseks teisendab skript *dataframe* kujul olevad andmetabelid NumPy *array* tüüpi andmemassiivideks. Tunnuste jaoks, mille põhjal hakatakse tegema soovitusi, luuakse samuti kaks eraldi NumPy andmemassiivi. Nendesse massiividesse ei võeta tekstilisi tunnuseid nagu laulu identifitseerimiskood, albumi identifitseerimiskood, laulu nimi, artisti nimi, albumi nimi ja väljalaske kuupäev. Numbrilistest tunnustest jäetakse võtmata ka laulu pikkus, helistik ning helilaad, sest töö autor hindas, et on tegu liialt üldiste tunnustega. Seega tunnused, mille põhjal soovitusi tegema hakatakse on laulu akustilisus, tantsitavus, energia, instrumentaalsus, elujõulisus, kõnemeelsus, tempo, taktimõõt ja valents.

Eraldi andmemassiiv luuakse ka laulu ja laulule vastava artisti nime hoidmiseks.

6 K-lähima naabri algoritmi rakendus

6.1.1 K-lähima naabri algoritm

Soovitussüsteemis rakendatava algoritmina otsustas töö autor kasutada k-lähima naabri algoritmi. K-lähima naabri algoritmi kasutuse põhjus seisneb selles, et algoritm on hea valik kahe objekti vahelise sarnasuse leidmiseks, sellel on relatiivselt lihtne tööpõhimõte ning algoritmi kasutatakse laialdaselt ka reaalsetes soovitussüsteemides.

K-lähima naabri (k-NN) algoritmi printsiip seisneb selles, et mingi andmehulga X igale objektile leitakse teisest andmehulgast Y k-arv lähimaid naabreid ehk objekte, mis on sellega mingite tunnuste alusel kõige sarnasemad. Objektide sarnasuse, ehk kauguse väärtusena kasutatakse tavaliselt harilikku eukleidilist kaugust. Eukleidiline kaugus avaldub valemiga (1).

$$D(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \quad (1)$$

$D(q,p)$ – Eukleidiline kaugus objekti kohta

q – esimese objekti tunnus

p – teise objekti tunnus

n – tunnuste arv

6.1.2 K-lähima naabri algoritmi näide

Antud töö raames rakendatakse k-lähima naabri algoritmi laulude kauguste leidmiseks. Seega, olgu näitena valitud kasutajale meeldivate laulude andmehulgast üks laul tähisega A. Laulu kirjeldavaid tunnuseid sisaldava tabeli näide on toodud Tabelina 1.

Tabel 1. Laulu A helitunnused

Tähis	**Akustilisus**	**Energia**	**Instrumentaalsus**	**Elujõulisus**	**Tempo**	**Valents**
A	0,189	0,533	0,000049	0,344	116,231	0,527

Olgu teisest andmehulgast, kust hakatakse soovitusi otsima, valitud näitena kolm laulu tähistega B, C ja E. Nende tunnused avalduvad Tabeli 2 kaudu.

Tabel 2. Laulude B, C ja E helitunnused

Tähis	**Akustilisus**	**Energia**	**Instrumentaalsus**	**Elujõulisus**	**Tempo**	**Valents**
B	0,924	0,372	0,898	0,113	80,848	0,260
C	0,300	0,750	0,000000	0,234	126,010	0,365
E	0,0855	0,944	0,145	0,116	119,983	0,676

Selleks, et kõik tunnused oleksid võrdse kaaluga, on vaja enne k-lähima naabri algoritmi rakendamist andmed skaleerida. Praeguse näite puhul võib jätta, lihtsuse mõttes, andmete skaleerimise vahele.

K-lähima naabri algoritmi rakendamise põhimõte seisneb meeldiva laulu eukleidilise kauguse leidmises igast teises andmehulgas olevast laulust. Rakendades eukleidilise kauguse valemit (1), leiame peale arvutuste sooritamist järgmised kaugused:

$D(B)$ = 35,404
$D(C)$ = 9,783
$D(E)$ = 3,788

Eukleidiliste kauguste arvutustest võib järeldada, et laulule A lähim naaber on laul E, sest nende vaheline kaugus on kõige väiksem. Teiseks lähimaks naabriks on laul C ning kolmandaks on laul B.

Seega, võib järeldada, et tunnuste järgi laulule A on kõige sarnasem laul E, ning seda võib kasutajale soovitada.

6.1.3 K-lähima naabri algoritmi rakendamine

K-lähima naabri algoritmi rakendamiseks skriptis kasutab töö autor teeki nimega Scikit-learn. Scikit-learn on Pythoni masinõppe teek, mis pakub kasutajale erinevaid andmetöötluse tööriistu ning võimaldab kasutada masinõppe algoritme.

K-lähima naabri algoritm nõuab esmalt andmete skaleerimist, et eukleidilise kauguse arvutuse puhul objekti kõik tunnused oleksid võrdse kaaluga. Andmed skaleeritakse

mõlemas andmemassiivis scikit-learn *StandarScaler* klassi *fit_transform* funktsiooni kasutades.

Scikit-learni moodul *NearestNeighbors* teeb algoritmi rakendamise üsna lihtsaks. Kasutades *NearestNeighbors* klassi funktsiooni *fit* saame sobitada mudeli andmemassiivile ning kasutades funktsiooni *kneighbors* salvestame ennikusse *result* nende laulude indeksid, mis on lähimad naabrid esitusloendi andmemassiivis olevatele lauludele. Lisaks sellele salvestame ka naabrite vahelise kauguse.

Skript leiab igale esitusloendi andmemassiivis olevale laulule ühe naabri. Töö autori esitusloendis on 350 laulu, seega teeb skript ka 350 soovitust. Soovituste arvu vähendamiseks ja täpsuse tõstmiseks käib skript läbi, soovitatud laulude indeksitest ja kaugustest koosneva massiivi, ning sorteerib massiivi kauguste veeru põhjal, rakendades NumPy funktsiooni *argsort*, mis sorteerimiseks kasutab vaikimisi *quicksort* algoritmi. Seejärel valitakse sorteeritud massiivist 10% kõige väiksematele kaugustele vastavat laulude indeksit ning lisatakse need uude listi *songs*. Võib juhtuda, et listis leidub duplikaate, sest sama laul võib olla lähimaks naabriks mitmele teisele laulule. Skript käib listi üle ning eemaldab kõik duplikaadid.

Viimaseks käiakse üle eelnevalt loodud massiiv *song_names*, kus on iga laulu ja seda esitava artisti nimi. Kui skript leiab, et laulu indeks selles massiivis ühtib mõne laulu indeksiga listis *songs*, väljastab skript nende laulude ja neid esitavate artistide nimed kasutajale.

6.1.4 K-lähima naabri algoritmi alternatiivsed implementatsioonid

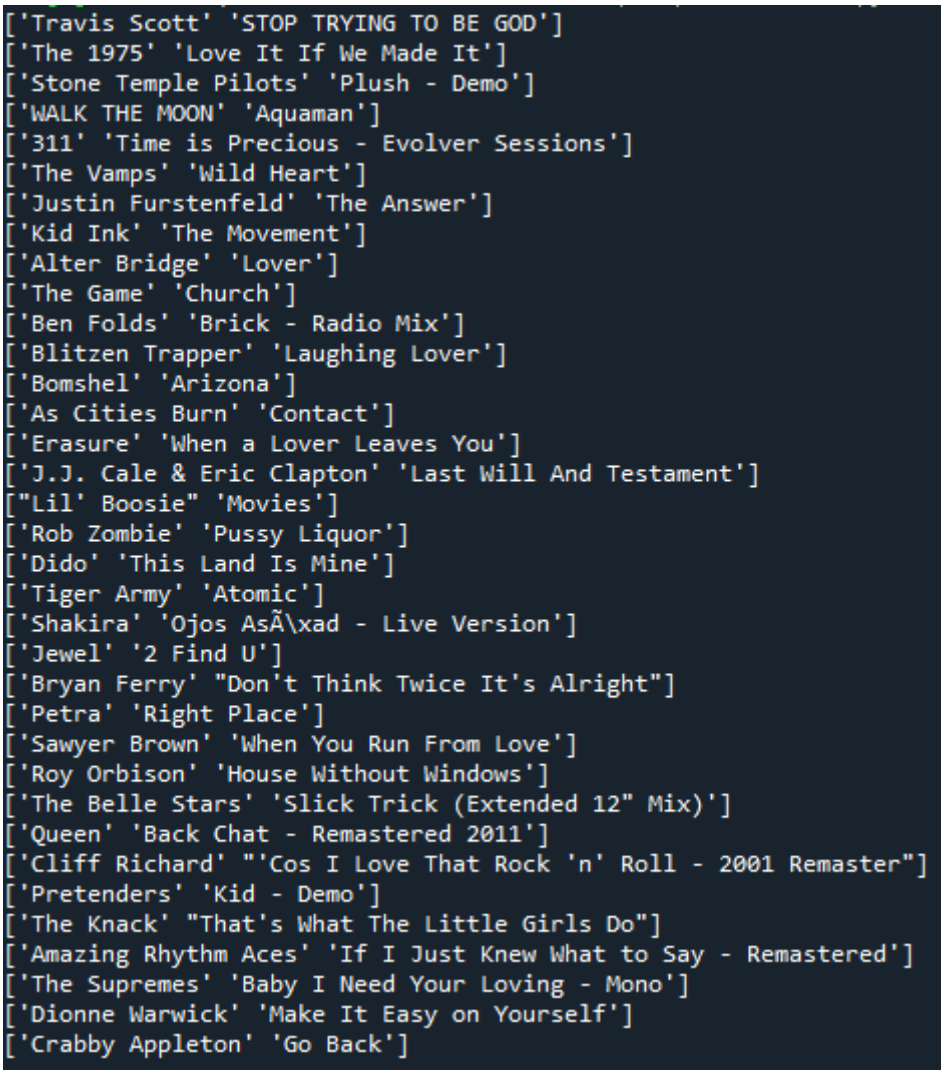
K-lähima naabri algoritmi võib rakendada mitmel eri moel. Töö autor otsustas võrdluseks läheneda algoritmi implementeerimisele alternatiivselt. Erinevalt eelnevast meetodist, ei leita igale esitusloendi laulule ühte lähimat naabrit, vaid leitakse kõikide andmebaasis olevate laulude kohta iga esitusloendis oleva laulu eukleidiline kaugus. Seejärel liidetakse kõik kaugused NumPy *sum* funktsiooni kasutades kokku ning saadakse esitusloendis olevate laulude eukleidiliste kauguste summa iga andmebaasis oleva laulu kohta. Moodustatakse massiiv, kus on eukleidiliste kauguste summa ning andmebaasi laulude indeksid. Seejärel rakendatakse massiivile eelmainitud NumPy funktsiooni *argsort* ja sorteeritakse massiiv kauguste summa järgi. Sorteeritud massiivist võetakse 30 esimest, ehk 30 kõige väiksema eukleidiliste kauguste summaga

laulu indeksit ning salvestatakse need uude listi *songs*. Laulude indeksite järgi väljastatakse, sarnaselt esimesele versioonile, indeksitele vastavate laulude ja artistide nimed ning kuvatakse soovitused kasutajale.

Töö autor otsustas proovida ka kolmandat lähenemist, mis on eelnevast kahest veidi teistsuguse funktsionaalsusega. Nimelt leitakse k-lähima naabri algoritmiga, igale esitusloendis olevale laulule, 30 lähimat naabrit ning salvestatakse nende indeksid massiivi. Loodud massiivist valitakse, Pythoni *random* teegi funktsiooni *randrange* kasutades, ühe suvalise esitusloendis oleva laulu indeks ning saadud indeksi alusel salvestatakse sellele laulule vastava 30 lähima naabri indeksid listi *songs*. Viimaseks kuvatakse kasutajale laul, mille alusel soovitusi tegema hakatakse ning seejärel *songs* listis olevate laulude ja neid esitavate artistide nimed, sarnaselt kahele eelnevale lahendusele.

7 Tulemuste võrdlus

Kõik kolm implementatsiooni on sisult erinevad ning annavad ka erinevaid tulemusi. Esimene lahendus leiab ainult ühe lähima naabri igale esitusloendi laulule ning seejärel, tulemuste arvu vähendamiseks, kuvab kasutajale kõikide leitud naabrite seast vaid 10% kõige lähedasemat laulu. Töö autori esitusloendi põhjal, kus oli 350 laulu, tehakse seega 35 soovitust. Saadud soovitustest meeldis autorile 20 laulu, 9 laulu kohta jäi autor ükskõikseks ning 6 laulu ei meeldinud autorile üldse. Esimese lahenduse tulemused on näidatud Joonisega 6.



Joonis 6. Soovitussüsteemi esimese lahenduse tulemused

Teine lahendus leiab iga soovitusandmebaasis oleva laulu kohta kõikide esitusloendis olevate laulude eukleidilised kaugused, liidab need eukleidilised kaugused iga andmebaasi laulu kohta kokku, võtab sorteeritud massiivist 30 kõige väiksema eukleidilise summaga laulu ja soovitab need kasutajale. Seega tehakse olenemata esitusloendi suurusest kasutajale alati 30 soovitust. Saadud soovitustest meeldis autorile 17 laulu, 10 laulu kohta jäi autor ükskõikseks ning 3 laulu ei meeldinud autorile üldse. Teise lahenduse tulemused on näidatud Joonisega 7.

```
['Trans-Siberian Orchestra' 'Who I Am']
['Belle And Sebastian' 'Ever Had A Little Faith?']
['Kina Grannis' 'The Fire']
['Milo Greene' "Don't You Give Up On Me"]
['The Cure' 'No Heart - Band Rehearsal Instrumental - 06/88']
['Tori Amos' 'Abnormally Attracted To Sin']
['Smokie Norful' 'I Understand']
['Bonnie Raitt' 'Spit Of Love']
['Dave Matthews Band' 'Pig']
['Sara Evans' "Fool, I'm a Woman"]
['Alison Moyet' 'So Am I - 2016 Remastered']
['Buddy Guy' 'Trouble Man']
['Bruce Springsteen' 'Souls of the Departed']
['Van Morrison' 'Daring Night']
['The Everly Brothers' 'More Than I Can Handle']
['Aretha Franklin' 'Just My Daydream - 12" Version']
['707' 'Eagle One']
['Jefferson Starship' 'Just the Same']
['Blue Oyster Cult' 'In Thee']
['Electric Light Orchestra' 'So Fine']
['Rare Earth' 'Generation (Light Up The Sky)']
['Utopia' 'Jealousy']
['David Laflamme' 'This Man']
['Bachman-Turner Overdrive' "Lookin' Out For #1"]
['Edgar Winter' "Hello Mellow Feelin'"]
['Peter Frampton' 'Underhand']
['Small Faces' 'Rene (Mono Version) (2018 Remaster)']
['Sonny & Cher' "But You're Mine - LP / Single Version"]
['The Beach Boys' 'This Whole World - Remastered 2009']
['Bob Crewe Generation' 'Melancholy Serenade']
```

Joonis 7. Soovitussüsteemi teise lahenduse tulemused

Kolmas lahendus erineb eelnevatest kõige rohkem, sest selle meetodi puhul ei tehta soovitusi kogu esitusloendi põhjal, vaid valitakse esitusloendist suvaline laul ning kuvatakse 30 soovitust ainult selle laulu põhjal. Antud meetodi eelis seisneb selles, et ühe esitusloendi põhjal, saab teha nii palju kordi erinevaid soovitusi, kui palju on esitusloendis erinevaid laule. Teisi meetodeid mitu korda sama esitusloendi peal rakendades jääb tulemus samuti alati samaks. Töö autori esitusloendist ühe suvalise laulu põhjal tehtud soovitustest meeldis autorile 14 laulu, 9 laulu kohta jäi autor ükskõikseks ning 7 laulu ei meeldinud autorile üldse. Kolmanda lahenduse tulemused on näidatud Joonisega 8.

```
Recommending songs similar to  ['Tyler, The Creator' "IGOR'S THEME"]
['Drake' 'Lord Knows']
['Death Grips' 'Flies']
['Rick Ross' 'Maybach Music V']
['Boondox' 'Born in Fire']
['ZZ Top' 'Tush - Live']
['Meek Mill' 'Lord Knows (feat. Tory Lanez)']
['Boosie BadAzz' 'No Surrender No Retreat']
['Boosie BadAzz' 'Streets On Fire']
['Rick Ross' 'Rich Is Gangsta']
['The Used' 'Force Without Violence']
['Freddie Gibbs & Madlib'
 'PiÃ‡ata (feat. Domo Genesis, G-Wiz, Casey Veggies, Sulaiman, Meechy Darko, Mac Miller)']
['Various Artists'
 'Earthquake (DJ Fresh vs. Diplo) (feat. Dominique Young Unique) - U.S Ultra Dance 15']
['Bun-B' 'All A Dream (feat. Letoya Luckett)']
['Papoose' '6am']
['The Game' 'Jesus Piece']
['Tech N9ne' 'URALYA']
['Bloc Party' 'Team A']
['Idina Menzel' 'Everybody Knows - Live']
['Switchfoot' 'Hello Hurricane - Live [Bonus Track]']
['The Avett Brothers' 'Talk On Indolence']
['Trip Lee' "Show's Over"]
['tobyMac' 'The Slam - Live']
['K-Paz De La Sierra'
 'PresentaciÃ³n - En Vivo Desde MÃ©xico Auditorio Nacional/2008']
['The Clark Sisters' 'Pray For The USA (Reprise) - Live']
['dead prez' 'Psychology']
['Methods Of Mayhem' 'Metamorphosis - Album Version (Edited)']
['Sam Cooke' 'Soul Twist / Introduction - Live']
['Creedence Clearwater Revival' "Travelin' Band - Live"]
['Mott The Hoople' 'Sweet Angeline - Live Version']
['Rory Gallagher' "Messin' With The Kid - Live From Ulster Hall"]
```

Joonis 8. Soovitussüsteemi kolmanda lahenduse tulemused

8 Edasiarenduse suunad

Soovitussüsteem on üsna primitiivne ning kindlasti on võimalik süsteemi täiendada ning selle täpsust suurendada. Üheks täpsuse suurendamise võimaluseks oleks teha uurimus, koos katsetega, selle kohta, millised laulude karakteristikud on soovituste tegemisel kõige tähtsamad ning mudelis kasutada vaid neid. Võimalik oleks ka lisada juurde uusi karakteristikuid nagu näiteks laulu žanr või plaadifirma nimi. Töömahukam võimalus oleks katsetada erinevaid algoritme, k-lähima naabri algoritmi asemel, ning vaadelda, milline algoritm annab kõige parema tulemuse.

Täpsuse kontrollimine on samuti aspekt, mida potentsiaalselt oleks võimalik kas automatiseerida või muul moel täiustada. Hetkel piirdub täpsuse kontrollimine kõikide soovitatud laulude läbi kuulamisega ning seejärel otsustamisega, millised laulud meeldisid ja millised mitte.

Praeguses vormis on kogu töövoog, alates andmete kogumisest kuni soovituste tegemiseni, küll lihtne, ja selle töö raames sobilik, kuid tavakasutajale üsna kohmakas. Üheks edasiarenduse suunaks on süsteemile graafilise kasutajaliidese loomine ning arendada see edasi, kas tavaliseks rakenduseks, või veebirakenduseks, mis teeks tavakasutajale soovitussüsteemi kasutamise lihtsamaks. Lisaks graafilisele liidesele on üheks soovitussüsteemi kasutajasõbralikumaks muutmise võimaluseks ka automaatselt soovituste põhjal kasutajale Spotify esitusloendi koostamine.

9 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli saada ülevaade soovitussüsteemidest, nende tööpõhimõtetest, rakendustest ning omandatud teadmiste põhjal arendada välja primitiivne muusikasoovitussüsteem, töö autori muusikaliste eelistuste põhjal.

Eesmärkide saavutamiseks uuriti soovitussüsteemide ajaloo kohta, analüüsiti metoodikaid, täpsemalt sisupõhist filtreerimist, kaasfiltreerimist ja hübriidseid lahendusi ning vaadeldi muusikasoovitussüsteemide rakendamist Spotify, Last.fm ja Pandora Radio näitel.

Töö praktilise osa raames määrati soovitussüsteemi mudel ning koguti vajalikud andmed. Autori muusikaeelistuste kätte saamiseks loodi Pythonit kasutades Spotify API rakendus, mis võimaldab kasutajal csv faili salvestada etteantud esitusloendis olevad laulud, nende metaandmed ning helitunnused.

Andmete töötluseks ning soovitussüsteemi mudeli realiseerimiseks kasutati Anaconda raamistikku ning NumPy, Pandas ja Scikit-learn teeke Pythonis. Soovituste leidmiseks kasutati k-lähima naabri algoritmi. Töö käigus andis autor ülevaate k-lähima naabri algoritmi tööpõhimõttest ning tõi algoritmi kohta ka näite laulude vahelise sarnasuse leidmisel. Seejärel kirjeldati k-lähima naabri algoritmi realiseerimist. Lisaks sellele toodi välja ka erinevaid lahendusi, kuidas soovituste tegemisele alternatiivselt läheneda, ning implementeeriti k-lähima naabri algoritm eri viisidel.

Töö tulemusena valmis soovitussüsteemist kolm erinevat versiooni, mis kõik andsid ka erinevaid tulemusi. Töö autori hinnangul oli kõige täpsem soovitussüsteemi teine versioon, kuid üldiselt oli töö autor kõigi kolme versiooni tulemustega rahul.

Kasutatud kirjandus

[1] F. Ricci, L. Rokach and B. Sharpia, Recommender Systems Handbook, New York: Springer Science+Business Media, 2011.

[2] D. B. Terry, A Tour Through Tapestry, Palo Alto: Xerox Corporation, 1993.

[3] D. Goldberg, D. Nichols, B. Oki and D. Terry, Using Collaborative Filtering to Weave an Information Tapestry, Palo Alto: Xerox Corporation, 1992.

[4] Amazon, "Amazon Press Center," Amazon, 10 02 1997. [Võrgumaterjal]. Available: https://press.aboutamazon.com/news-releases/news-release-details/net-perceptions-transforms-worlds-largest-bookstore-unique/. [Kasutatud 03 05 2020].

[5] F. M. Harper and J. A. Konstan, "The MovieLens Datasets: History and Context," *ACM Transactions on Interactive Intelligent Systems, Vol. V, No. N, Article XXXX,* p. 20, 2015.

[6] Netflix, "Netflix Prize," Netflix, 2009. [Võrgumaterjal]. Available: https://www.netflixprize.com/. [Kasutatud 03 05 2020].

[7] D. Jackson, "The Netflix Prize: How a $1 Million Contest Changed Binge-Watching Forever," Thrillist, 07 07 2017. [Võrgumaterjal]. Available: https://www.thrillist.com/entertainment/nation/the-netflix-prize. [Kasutatud 03 05 2020].

[8] S. Luo, "Introduction to Recommender System," Towards Data Science, 10 12 2018. [Võrgumaterjal]. Available: https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26. [Kasutatud 08 03 2020].

[9] Google, "Content-based Filtering Advantages & Disadvantages," Google, 30 11 2018. [Võrgumaterjal]. Available: https://developers.google.com/machine-learning/recommendation/content-based/summary. [Kasutatud 08 03 2020].

[10] P. Melville and V. Sindhwani, "Recommender Systems," in *Encyclopedia of Machine Learning*, Springer, 2010.

[11] Google, "Collaborative Filtering," Google, 10 02 2020. [Võrgumaterjal]. Available: https://developers.google.com/machine-learning/recommendation/collaborative/basics. [Kasutatud 08 03 2020].

[12] Y. Koren, R. Bell and V. Christ, "Matrix Factorization Techniques For Recommender Systems," IEEE Computer Society, Washington, 2009.

[13] C. University, "Networks - Course blog for INFO 2040/CS 2850/Econ 2040/SOC 2090," Cornell University, 20 9 2012. [Võrgumaterjal]. Available: https://blogs.cornell.edu/info2040/2012/09/20/last-fm-music-reccomendation-incorporating-social-network-ties-and-collaborative-filtering/. [Kasutatud 21 03 2020].

[14] Galvanize, "Ever Wonder How Spotify Discover Weekly Works? Data Science," Galvanize, 22 08 2016. [Võrgumaterjal]. Available: https://blog.galvanize.com/spotify-discover-weekly-data-science/. [Kasutatud 08 04 2020].

[15] S. Ciocca, "How Does Spotify Know You So Well?," Medium, 10 10 2017. [Võrgumaterjal]. Available: https://medium.com/s/story/spotifys-discover-weekly-how-machine-learning-finds-your-new-music-19a41ab76efe. [Kasutatud 08 04 2020].

[16] I. Ramesar, "Pandora Recommender Systems," 10 04 2019. [Võrgumaterjal]. Available: https://rstudio-pubs-static.s3.amazonaws.com/485366_936f85c099ac449aa1058e1d82826968.html. [Kasutatud 27 04 2020].

[17] R. Pegoraro, "Pandora's "Music Genome Project" explores the cold hard facts of how we interact with music," Boing BOing, 24 05 2014. [Võrgumaterjal]. Available: https://boingboing.net/2014/05/24/pandoras-music-genome-proj.html. [Kasutatud 27 04 2020].

[18] G. Lawton, "How Pandora built a better recommendation engine," TheServerSide, 09 08 2017. [Võrgumaterjal]. Available: https://www.theserverside.com/feature/How-Pandora-built-a-better-recommendation-engine. [Kasutatud 29 04 2020].

[19] D. P. E. B. W. P. L. Thierry Bertin-Mahieux, "The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference," ISMIR, 2011. [Võrgumaterjal]. Available: http://millionsongdataset.com/. [Kasutatud 08 04 2020].

[20] A. Thompson, "Acoustic and meta features of albums and songs on the Billboard 200," 07 04 2019. [Võrgumaterjal]. Available: https://components.one/datasets/billboard-200/. [Kasutatud 08 04 2020].

[21] Last.fm, "Last.fm Web Services," Last.fm, 2019. [Võrgumaterjal]. Available: https://www.last.fm/api/. [Kasutatud 08 04 2020].

[22] P. Lamere, "Spotipy," Spotipy, 2014. [Võrgumaterjal]. Available: https://spotipy.readthedocs.io/en/2.11.2/#. [Kasutatud 20 04 2020].

Lisa 1 – Programmikoodi hoidla link

Andmete hankimise ja soovitussüsteemi programmikoodi leiab töö autori GitHub lehelt:

https://github.com/karluust/thesis