Sustavi preporuka glazbe prema sadržaju

Klarin, Danijel

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

DIPLOMSKI RAD

SUSTAVI PREPORUKA GLAZBE PREMA SADRŽAJU

Danijel Klarin

Split, travanj 2024.

Temeljna dokumentacijska kartica

Diplomski rad

Sveučilište u Splitu

Prirodoslovno-matematički fakultet

Odjel za informatiku

Ruđera Boškovića 33, 21000 Split, Hrvatska

SUSTAVI PREPORUKA GLAZBE PREMA SADRŽAJU

Danijel Klarin

SAŽETAK

U ovom diplomskom radu korišteni su algoritmi strojnog učenja – k-NN, K-Means i DBSCAN u sustavu preporuka prema sadržaju na primjerima pjesama. Korišten je skup podataka istrugan s vlastito kreiranog popisa pjesama sa glazbenog servisa Spotify. Korištene su tehnike pri obradi podataka: min-max skaliranje i dimenzioniranje uz pomoć algoritama PCA i t-SNE. Kod k-NN provedeni su testovi uz različite udaljenosti, algoritme pretraživanja i vremensku komponentu, dok su kod K-Means i DBSCAN algoritma korišteni testovi svojstveni za nenadzirano učenje. Sve kako bi se ocijenilo koji algoritmi, tehnike i parametri dovode do najboljih rezultata preporuke.

Ključne riječi: sustav preporuke prema sadržaju, strojno učenje, glazba, Spotify, struganje podataka, k-NN, K-Means, DBSCAN, PCA, t-SNE, min-max

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 51 stranicu, 10 grafičkih prikaza, 14 tablica i 34 literaturna navoda. Izvornik je na hrvatskom jeziku.

Mentor: **Dr. sc. Saša Mladenović,** *redoviti profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Ocjenjivači: **Dr. sc. Saša Mladenović,** *redoviti profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Dr. sc. Goran Zaharija, *docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Dr. sc. Divna Krpan, *docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Rad prihvaćen: **travanj 2024.**

Basic documentation card

Thesis

University of Split

Faculty of Science

Department of computer science

Ruđera Boškovića 33, 21000 Split, Croatia

MUSIC CONTENT-BASED RECOMMENDATION SYSTEMS

Danijel Klarin

ABSTRACT

In this master's thesis, machine learning algorithms - k-NN, K-Means, and DBSCAN - were used in a content-based recommendation system using examples of songs. The dataset was extracted from a self-created list of songs from the Spotify music service. Data processing techniques included min-max scaling and dimensionality reduction using PCA and t-SNE algorithms. For k-NN, tests were conducted with different distances, search algorithms, and a time component, while for K-Means and DBSCAN algorithms, tests specific to unsupervised learning were used. All in order to evaluate which algorithms, techniques, and parameters lead to the best recommendation results.

Key words: recommendation system, content-based, machine learning, music, Spotify, data scraping, k-NN, K-Means, DBSCAN, PCA, t-SNE, min-max

Thesis deposited in library of Faculty of science, University of Split

Thesis consists of: 51 pages, 10 figures, 14 tables and 34 references

Original language: Croatian

Mentor: **Saša Mladenović, Ph.D.** *Full Professor of Faculty of Science, University of Split*

Reviewers: **Saša Mladenović, Ph.D.** *Full Professor of Faculty of Science, University of Split*

Goran Zaharija, Ph.D. *Assistant Professor of Faculty of Science, University of Split*

Divna Krpan, Ph.D. *Assistant Professor of Faculty of Science, University of Split*

Thesis accepted: **April 2024.**

IZJAVA

Kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam diplomski rad s naslovom SUSTAVI PREPORUKA GLAZBE PREMA SADRŽAJU izradio samostalno pod voditeljstvom prof. dr. sc. Saše Mladenovića. U radu sam primijenio metodologiju znanstvenoistraživačkog rada i koristio literaturu koja je navedena na kraju diplomskog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo u diplomskom radu na uobičajen, standardan način citirao sam i povezao s fusnotama s korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskog jezika.

Student

DANIJEL KLARIN

Sadržaj

Uvod 1

1. Općenito o sustavima preporuka 2

1.1. Povijest sustava za preporuke 2

1.2. Problematika preporuka 4

1.3. Pregled istraženosti 4

1.4. Vrste sustava preporuka 6

1.4.1. Suradničko ili kolaborativno filtriranje 7

1.4.2. Filtriranje po sadržaju 7

1.4.3. Hibridno filtriranje 7

1.5. Algoritmi za sustave preporuke temeljeni na strojnom učenju 8

1.5.1. Nadzirano učenje 8

1.5.2. Nenadzirano učenje 10

1.5.3. Metrike 12

1.6. Slučajevi korištenja 14

2. Sustavi preporuke u glazbi 17

2.1. Odabir tehnologijske platforme za usporedbu algoritama 17

2.1.1. Google Colaboratory 18

2.1.2. Odabir algoritama za usporedbu 19

2.1.3. Odabir metrika 19

2.1.4. Odabir skupa podataka 20

2.1.5. Analiza podataka 23

2.1.6. Priprema podataka 27

2.1.7. Realizacija prototipa 32

2.1.8. Usporedba rezultata 38

3. Zaključak 50

Literatura 52

Popis slika i tablica 55

Uvod

U današnjem digitalnom dobu, ekspanzija glazbenog sadržaja na različitim platformama pruža korisnicima široku lepezu glazbenih brojeva. S obzirom na obilje glazbe, postavlja se pitanje: kako korisnicima omogućiti iskustvo otkrivanja nove glazbe? Diplomski rad „Sustavi preporuka glazbe prema sadržaju“ istražuje upravo tu problematiku, držeći fokus na sustave preporuka prema sadržaju u kontekstu glazbenih platformi.

Cilj ovog rada je bolje razumijevanje mehanizama i izazova koji stoje iza razvoja sustava preporuka u glazbenom svijetu. Proučavajući relevantnu literaturu i analizirajući postojeće modele, rad će identificirati ključne komponente koje utječu na učinkovitost sustava preporuka glazbe prema sadržaju, uzimajući u obzir raznolike glazbene preferencije i njihovu promjenjivost kroz vrijeme.

Kroz analizu algoritama preporuka te proučavanje njihovih prednosti i nedostataka, rad će pridonijeti razumijevanju optimalnih pristupa u izgradnji glazbenih preporuka. Osim toga, istraživanje će se usredotočiti na tehničke aspekte sustava preporuka, naglašavajući važnost (ne)preciznosti preporuka.

Kroz primjenu teorijskih spoznaja na konkretne primjere glazbenih platformi kakav je Spotify, rad će analizirati primjene sustava preporuka. Konačno, cilj ovog rada je ne samo doprinijeti teorijskom okviru sustava preporuka prema sadržaju, već i ponuditi praktične smjernice za razvoj inovativnih i učinkovitih sustava koji će obogatiti glazbeno iskustvo korisnika u digitalnom dobu čiji će prototip biti prikazan u Jupyter bilježnicama.

1. Općenito o sustavima preporuka

Sustav preporuka u kontekstu informacijske tehnologije je informacijski sustav koji je dio većeg sustava za filtriranje informacija, a sadrži metode koje služe kako bi dale preporuke korisnicima na temelju njihovih preferencija tj. „ukusa“, prijašnjeg ponašanja ili povijesti interakcija [1]. S druge strane, definicija same riječi „preporuka“ znači povoljno mišljenje, bilo usmeno ili pismeno, povoljna ocjena o svojstvima nekog entiteta ili ono što se daje kao savjet [2]. Prethodno spomenuti „veći“ sustav za filtriranje informacija sastoji se od dvije vrste filtriranja informacija, to su već spomenuti sustav preporuka u kojoj informacija, proizvod ili stavka pronalazi korisnika, te pretraga u kojoj korisnik sam traži određenu stavku.

1.1. Povijest sustava za preporuke

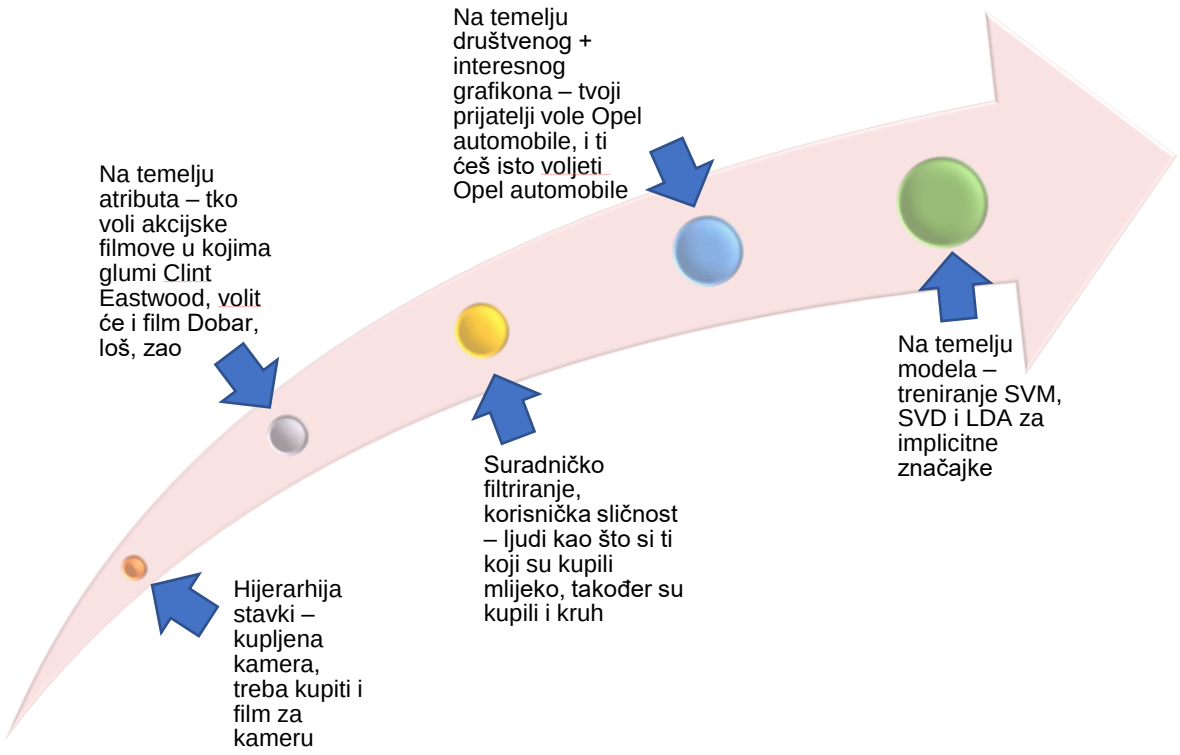
Prvi sustav preporuka stvoren je 1979. godine pod nazivom *Grundy* od strane američke računalne znanstvenice Elaine Rich, a razvijen je za knjižnicu [3]. Njezina ideja bila je sustav koji je postavljao pitanja korisniku knjižnice, te je na taj način davao korisniku određene oznake i zatim ga smještao u određenu kategoriju. Na ovaj način se korisniku mogla preporučiti određena knjiga. Nakon toga, još jedan rani sustav preporuka pod nazivom *Digital Bookshelf* opisan je u tehničkom izvještaju 1990. godine, autora Jussi Karlegrena s Columbia sveučilišta [4].

Tijekom 90-tih. godina, sustavi preporuka su još uvijek bili u svojim začecima, ali počeli su postajati sve značajniji, te se počinju primjenjivati u različitim područjima [5]. Počinju se razvijati algoritmi za preporuku sadržaja na web stranicama i portalima. Javlja se ideja suradničkog ili kolaborativnog filtriranja [6], iako još uvijek ne automatizirano, u kojem se preporuke temelje na sličnosti između korisnika. Tako je 1992. godine predložen sustav pod nazivom *Tapestry* koji je bio prvi sustav preporuka temeljen na kolaborativnom filtriranju [1]. Počinju se razvijati i metrike za procjenu učinkovitosti sustava preporuka, kao što su preciznost (engl. *precision*) [7], odziv (engl. *recall*) [7] i greška apsolutne sredine (engl. *mean absolute error*) [8].

Amazon je već u počecima svojeg poslovanja usvojio sustave preporuka. Ključan je bio razvoj algoritma "item-to-item collaborative filtering", omogućivši personalizaciju *online* trgovine za svakog korisnika. Taj je algoritam doprinio poboljšanju korisničkog

iskustva i efikasnosti oglašavanja putem weba i e-pošte [9]. Razvoj i stalno poboljšavanje algoritma mehanizma za preporuke Amazona, s pristupom ogromnoj količini potrošačkih podataka, ono je što čini Amazon internet trgovinom broj 1. Prema procjenama, 35% kupnje potječe od strane sustava za preporuku [10].

Netflix započinje avanturu sa sustavom preporuka 2000. godine, a 2006. godine objavljeno je natjecanje pod nazivom „Netflix Prize“ [11]. Cilj natjecanja bio je poboljšati sustav preporuke za 10%. Dobitnik nagrade bio je BellKor Pragmatic Chaos tim koji je uspio poboljšati postojeći Netflixov algoritam Cinematech za 10,06%. Iako je skup podataka sa kojim se radilo bio anonimiziran, jedan tim je naknadno uspio povezati podatke sa IMDB-ovim identifikatorima korisnika, što je stvorilo potencijalne probleme [12].



Slika 1 Evolucija sustava za preporuke [13]

S obzirom da su sustavi preporuka zaživjeli tijekom devedesetih, rano se prepoznalo kako *web* pruža odlične prilike za personalizaciju, koje nisu bile dostupne u drugim kanalima.

1.2. Problematika preporuka

Primarni cilj sustava preporuka je porast prodaje proizvoda ili povećanje konzumacije sadržaja, što se tiče trgovina ili pružatelja usluga. S druge strane korisniku je važno otkrivanje sličnih sadržaja (engl. *items*) ili sadržaja koji bi mu se mogli svidjeti. Dobri sustavi preporuka ne predlažu samo slične sadržaje, nego i sadržaje koji bi mogli nadopuniti već odabrani sadržaj, kao što je npr. kod odabira dijelova računala u internetskoj trgovini. Dobri sustavi preporuka mogu ponuditi i nešto s potpuno suprotnog spektra proizvoda, jer se može događati da sustavi preporuka predlažu samo najpopularnije proizvode ili stavke.

Problemi kod sustava preporuka su razni, počevši od tzv. hladnog početka (engl. *cold start*), a znači nedostatak informacija što se tiče preferencija korisnika, samim time sustav preporuka ne zna što bi preporučio korisniku. Filter mjehura (eng. *filter bubble*) znači da se korisnik izolira od cijelog skupa podataka, pa mu sustav preporučuje uvijek samo slične stvari. Dakle, korisnik zapravo nikad ne dobije priliku izaći iz mjehura i dobiti priliku poslušati novi žanr pjesme ili pogledati neki drugi žanr filma, iako je „nikad“ malo preteška riječ. Jedan od problema je i promjena ponašanja, tako korisnik može promijeniti svoj ukus „preko noći“, a sustav preporuke može imati problema u praćenju izmjena. Još jedan problem kod sustava preporuka je i pristranost (engl. *bias*), a znači češću preporuku popularnijih sadržaja, dok se manje popularni sadržaji općenito rjeđe predlažu.

Problem kod izrade sustava preporuka je i vrednovanje, odnosno na koji točno način mjeriti efikasnost sustava preporuke. Jedan problem koji se ne tiče samih sustava preporuka, ali je neizravno povezan s njima je problem prenaučenosti (engl. *overfitting*), a povezuje se s algoritmima dimenzioniranja [14]. Naime, u radu se koriste algoritmi dimenzioniranja, a velik broj dimenzija tj. značajki može pridonijeti prenaučenosti.

1.3. Pregled istraženosti

U ovom odjeljku opisujemo trenutno stanje istraživanja u području sustava preporuka s fokusom na filtriranje prema sadržaju i primjenu strojnog učenja. Moja teza se temelji na primjeni strojnog učenja u ovom kontekstu, stoga je naglasak na pregledu literature koji uključuje upotrebu metoda strojnog učenja kao ključnu komponentu. Na temelju provedene pretraživačke strategije, pronađeno je ukupno 265 radova u *online* bazama podataka koji se bave ovim temama u razdoblju od 2019. do 2023. godine. Kao

smjernica za provedbu pretraživanja literature, korišteni su preporučeni PRISMA standardi [15]. Primjeri ključnih riječi korištenih pri pretraživanju u online bazama podataka mogu se vidjeti u tablici 1.

Tablica 1 Primjeri korištenja ključnih riječi i prekidača

Ključne riječi				
recommendation system	AND	content-based	AND	machine learning
recommendation system	AND	content-based	AND NOT	collaborative
recommendation system	AND	content-based	AND	k-nn
recommendation system	AND	content-based	AND	k-means

Baze korištene u pregledu istraženosti:

- Scopus,
- IEEEXplore,
- Google Scholar i
- Web of Science.

Ovi radovi pružaju temelj za razumijevanje postojećih pristupa, algoritama i tehnika u području sustava preporuka s filtriranjem prema sadržaju, te njihovu primjenu u poboljšanju performansi sustava preporuka.

Modularni pristup u ovom pregledu literature omogućuje detaljan uvid u različite algoritme, metode i modele koji se koriste u sustavima preporuka s filtriranjem prema sadržaju te u primjeni strojnog učenja u tim sustavima. To omogućuje prilagodbu analize ovisno o specifičnim potrebama istraživanja.

Ključni korak u istraživanju je odabir relevantnih članaka koji se bave ovim temama te praćenje njihove literature kao osnovu za pregled najnovijih dostignuća. Ovim pristupom osiguravamo da naš pregled reflektira najnovije trendove i napredak u području sustava preporuka s filtriranjem prema sadržaju i primjenom strojnog učenja.

Kriteriji za isključenje određeni su kao: kolaborativno filtriranje, sustavi učenja i algoritmi za učenje. S tim kriterijima isključeno je još 111 članaka, a 10 članaka je pronađeno duplim. Ulaskom u srž članaka pregledom sažetka i glave pronađeno je 20 relevantnih članaka, a izbor je na kraju sužen na 6 najvažnijih članaka korištenih za usporedbu.

Postoji zanimljiva analiza koja se bavi stvaranjem sustava preporuka za mobilne aplikacije u Google Play trgovini [16]. Ovaj pristup, koji datira iz 2020. godine, koristi se preporukom prema sadržaju, uz korištenje algoritma k-NN i kosinusne udaljenosti, koristeći podatke iz Google Play trgovine. Evaluacija ovog sustava obuhvaća tri studije slučaja, svaka fokusirana na različite aplikacije. Rezultati tih studija pružaju uvid u udaljenosti i ključne faktore koji utječu na preporuke aplikacija.

U istraživanju koje se fokusira na automatsko generiranje glazbenih popisa prema raspoloženju putem grupiranja audio značajki [17], primjenjuje se preporuka prema sadržaju korištenjem Euklidske udaljenosti i grupiranja. Rad koristi algoritme poput DBSCAN-a i OPTICS-a, dok su podaci prikupljeni korištenjem Spotify-jevog API-ja. Iako nije navedeno, metrike korištene u istraživanju nisu eksplicitno objavljene.

U istraživanju objavljenom 2022. godine, koje se fokusira na korištenje društvenih mreža za automatsko generiranje YouTube popisa pjesama [18], pristupa se grupiranju koristeći pet različitih algoritama nenadziranog učenja: K-Means++, EMGMM, Mean-Shift, DBSCAN i OPTICS. Iako je u radu korištena tehnika obrade prirodnog jezika (NLP) koja nije dio ovog diplomskog rada, istraživanje je zanimljivo jer koristi iste metrike kao i ovaj rad, kao što su koeficijent siluete i Calinski-Harabasz koeficijent, a rezultati se mogu iskoristiti za usporedbu.

1.4. Vrste sustava preporuka

Sustavi za preporuke mogu se podijeliti prema načinu filtriranja na suradničko filtriranje (engl. *collaborative filtering*) i na sustave preporuke temeljene na sadržaju (engl. *content-based filtering*). Suradničko ili kolaborativno filtriranje ima svojstvo raditi preporuke na temelju usporedbe danog korisnika s korisnikom koji mu je sličan po ukusu. Kod sustava preporuka temeljenog na sadržaju preporuka se daje usporedbom danog sadržaja (proizvod, film, pjesma itd.) sa sadržajem koji je sličan po svojstvima. Postoji još jedan često korišten tip filtriranja – hibridno filtriranje, koji predstavlja kombinaciju

kolaborativnog filtriranja i filtriranja po sadržaju. Hibridnim načinom filtriranja nastoji se izbjeći nedostatke obaju sustava preporuka, te od svakog uzeti samo dobra svojstva.

1.4.1. Suradničko ili kolaborativno filtriranje

Suradničko filtriranje je tehnika koja se koristi u sustavima preporuka za izradu predviđanja interesa korisnika na temelju prethodnih ocjena, preferencija ili drugih informacija. Pretpostavka polazi od toga da dvije osobe mogu imati slično mišljenje, pa je veća vjerojatnost da će i o nekom drugom pitanju imati slično mišljenje u odnosu s nekim trećim osobama. Prednosti suradničkog filtriranja su kreiranje personalizirane preporuke za svakog korisnika i korištenje informacija od korisnika koji imaju slične povijesti interakcije za preciznije predviđanje. Dvije su glavne vrste suradničkog filtriranja: zasnovano na memoriji što znači da koristi pohranjene podatke o preferencijama korisnika i zasnovano na modelu što znači da koristi modele strojnog učenja za predviđanje preferencija. Nedostatci suradničkog filtriranja su velika količina podataka o korisnicima koja je potrebna, osjetljivost na „hladan početak“ kod novih korisnika i moguća pristranost ako se dobro ne namjeste svi potrebni parametri.

1.4.2. Filtriranje po sadržaju

Filtriranje informacija može se definirati kao proces selektivnog prikupljanja, analize i obrade informacija kako bi se izdvojile važne informacije, dok se istodobno eliminiraju ili smanjuju informacije koje nisu bitne za određeni cilj ili potrebu. Filtriranje informacija također se može smatrati zadatkom klasifikacije, odnosno dijeljenjem informacija u određene klase. Na temelju podataka za učenje stvara se model koji omogućuje filtracijskom sustavu klasificirati buduće podatke u kategorije koje su korisne za korisnika i podataka koji nisu korisni za korisnika. Stavka koja je opisana nekim svojstvima, te se stavka može prikazati kao vektor $x = (x_1, x_2, \ldots, x_n)$ od n komponenti.

1.4.3. Hibridno filtriranje

Hibridni sustavi [19] mogu pružiti bolje rezultate kombinacijom suradničkog filtriranja i filtriranja po sadržaju iako su moguće i druge kombinacije, kao npr. kombinacija različitih tehnika iste vrste. Suradničko filtriranje i filtriranje po sadržaju smatraju se komplementarnim. Suradničko filtriranje fokusira se na preporuke već ocijenjenih stavki, dok preporuke temeljene na sadržaju mogu predložiti nove stavke koje korisnik još nije

procijenio. Osim toga, sadržaj preporuke može biti neprikladan ili teško izdvojiv. Hibridno filtriranje koristi prednosti ova dva načina filtriranja i odbacuje njihove slabosti te se oslanja na više izvora kako bi se iskoristile najprikladnije metode [19].

1.5. Algoritmi za sustave preporuke temeljeni na strojnom učenju

U radu se koriste dvije vrste algoritama. S jedne strane to je algoritam k-NN, tj. algoritam k-najbližih susjeda (engl. *k-nearest neighbors*) koji je algoritam nadziranog učenja. S druge strane koriste se još i algoritmi K-Means i DBSCAN koji su algoritmi nenadziranog učenja. Nadzirano učenje se temelji na korištenju označenih skupova podataka, dok je nenadzirano učenje usmjereno na grupiranje neoznačenih skupova podataka [20].

1.5.1. Nadzirano učenje

Algoritmi strojnog učenja koji spadaju pod nadzirano učenje (za razliku od algoritama nenadziranog učenja) su oni kojima su osnova označeni ulazni podaci, kako bi model naučio određeni izlaz s ciljem određivanja izlaznih rezultata kada dobije nove neoznačene podatke. Drugim riječima, novim podatcima dodijeliti oznake putem analize klasifikacije ili regresije.

Nadzirano učenje se može pojednostavljeno prikazati kao učenik (npr. u osnovnoj školi) tj. računalo, kojim upravlja neki određeni nadzornik (npr. učitelj) tj. programer, te se zbog tog nadzornika i naziva nadzirano učenje. U nadziranom učenju se učenika želi npr. naučiti kako izgleda slon. Učeniku se prikazuju različite slike, među kojima ima slika slonova i slika ostalih životinja. Učenik će nakon toga kada dobije neke nove slike, znati razlučiti što je slon, a što nije slon.

Algoritam k-najbližih susjeda je algoritam strojnog učenja koji spada pod nadzirano učenje. Lako se uči i implementira, a može poslužiti i u slučajevima klasifikacije i u slučajevima regresije. Jedan je od najjednostavnijih i najčešće korištenih tehnika za klasifikaciju, te je otporan na šum. k-NN algoritam pretpostavlja da slični podaci postoje u neposrednoj blizini. k-NN se temelji na ideji sličnosti (može biti riječ o udaljenosti ili o blizini) koja se izračunava uz pomoć matematičkih formula. Kako se povećava količina podataka koju k-NN algoritam koristi, primjetan je porast vremena potrebnog za obradu, što

je jedna od njegovih značajnijih mana. Prednosti algoritma k-NN su jednostavnost, lagan je za implementaciju, nema potrebe za izgradnjom modela, podešavanjem parametara ili postavljanjem dodatnih pretpostavki. Algoritam je svestran, a osim što se može koristiti za klasifikaciju i regresiju može se koristiti i za pretraživanje koje se koristi u sustavima preporuka.

Algoritam k-najbližih susjeda može koristiti različite udaljenosti koje se kao parametar u algoritmu nazivaju metrika (engl. *metric*) za izračunavanje udaljenosti između podataka u prostoru. Ova mjera udaljenosti ključna je za određivanje „bliskosti“ ili „sličnosti“ između podataka tijekom izračuna algoritmom k-najbližih susjeda. U praksi, različite metrike mogu biti prikladne ovisno o vrsti podataka i problemu s kojim se hvatamo u koštac. Jedna od najčešće korištenih je Euklidska (engl. *Euclidean*), koja mjeri „ravnu“ udaljenost između točaka. Formula za izračun putem Euklidske udaljenosti nalazi se pod brojem (1). Ako su podatci predstavljeni kao vektori u prostoru većeg broja dimenzija, Euklidska udaljenost je odličan izbor [21].

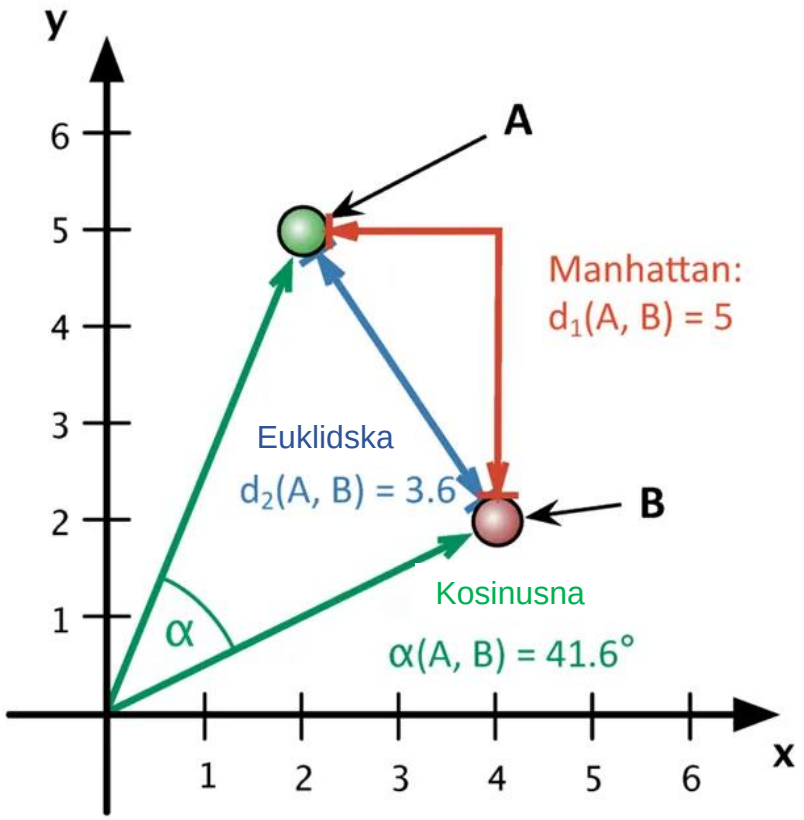
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{1}$$

Često korištena udaljenost je i Manhattan metrika koja mjeri udaljenost „po ulicama u gradu“ i računa zbroj apsolutnih razlika između koordinata [21]. Manhattan udaljenost je korisna ako podatci imaju različite skale ili ako su osi prostora nejednako važne. Formula za izračun putem Manhattan udaljenosti nalazi se pod brojem (2).

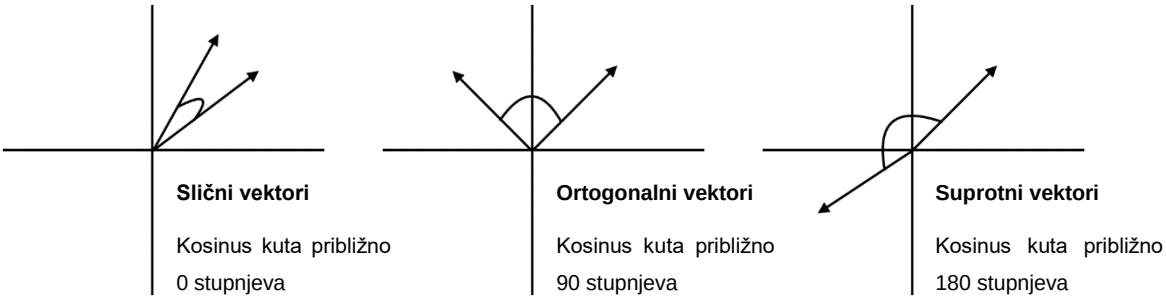
$$d = |x_2 - x_1| + |y_2 - y_1| \tag{2}$$

Kosinusna (engl. *cosine*) udaljenost koristi se za izračunavanje sličnosti između dva vektora [22]. Mjeri se kosinusom kuta između dva vektora i određuje da li dva vektora pokazuju u istom smjeru. Kada se koristi s algoritmom k-NN, ova mjera daje novu perspektivu problemu i omogućuje pronalazak nekih skrivenih informacija u podacima koje nisu bile moguće uz Euklidsku i Manhattan udaljenost. Udaljenosti prikazane u prostoru mogu se vidjeti na slici 2. Također se može reći kako kosinusna „radi na drugačiji način“ od prethodne dvije udaljenosti. Koristi se često u analitici teksta za pronalaženje sličnosti između dva dokumenta prema broju pojavljivanja određenog skupa riječi u njemu. Korištenjem ove udaljenosti dobivamo vrijednosti između 0 i 1, gdje 0 znači da su vektori 100% slični jedan drugome, a 1 znači da uopće nisu slični [22]. Primjeri kosinusne sličnosti prikazani su na slici 3.

$$cos\theta = \frac{A * B}{||A||\ ||B||} \tag{3}$$



Slika 2 Usporedba udaljenosti prikazane u prostoru [23]



Slika 3 Primjeri kosinusne sličnosti [24]

1.5.2. Nenadzirano učenje

Nenadzirano učenje je vrsta učenja koje uči iz neoznačenih podataka [25], što implicira nepostojanje postojećih oznaka ili kategorija kod podataka. Pojednostavljeno,

postoje ulazni podatci, dok algoritam sam određuje izlazne podatke. Cilj nenadziranog učenja je otkriti obrasce i odnose u podacima bez ikakvih uputa. Algoritmi nenadziranog učenja sami moraju pronaći veze među podatcima grupiranjem prema sličnostima, obrascima i razlikama bez prethodnog treniranja na oznakama, kao kod nadziranog učenja.

Nenadzirano učenje klasificira se u dvije kategorije. Prva kategorija je grupiranje (engl. *clustering*), a podrazumijeva pridruživanje podataka grupama prema svojstvima sličnosti. Druga kategorija je udruživanje (engl. *association*), a znači da algoritam otkriva pravila koja opisuju podatke. U ovom radu u praktičnom dijelu, što se tiče nenadziranog učenja koristi se grupiranje.

K-Means (ili K srednjih vrijednosti), algoritam je nenadziranog učenja koji grupira neoznačeni skup podataka u različite grupe [26]. Cilj K-Means algoritma je dijeljenje podatkovnih točaka unutar svake grupe koje su međusobno slične, a različite od drugih podatkovnih točaka u drugim grupama. K-Means je metoda particioniranja koja dijeli skup podataka u K grupa, koje se ne preklapaju i koje imaju temelj na sličnosti koja pokušava minimizirati varijancu (grešku) unutar svake grupe. Iterativni je algoritam, iako broj grupa K može odrediti programer, najbolji način određivanja broja K je uz pomoć metode lakta.

Kod algoritma DBSCAN (*Density-Based Spatial Clustering Of Applications With Noise*) grupe sadrže gusta područja u prostoru podataka, koje su odvojene područjima niže gustoće točaka. DBSCAN algoritam temelji se na ideji u kojoj svaka točka grupe ima susjedstvo zadanog radijusa, te mora sadržavati barem minimalan broj točaka.

DBSCAN ima prednost u odnosu na K-Means jer ne zahtijeva precizno odvajanje grupa, prednost je i manji utjecaj šuma, kao i odstupanje u podatcima. Kod DBSCAN algoritma ne određuje se broj grupa unaprijed, već algoritam sam određuje broj grupa pri treniranju.

Iako sljedeća dva algoritma ne služe za sustave preporuka, oni će se spomenuti u ovom podnaslovu iz razloga što spadaju pod algoritme nenadziranog učenja.

PCA (*Principal Component Analysis*) se temelji na pretpostavci u kojoj se podaci u prostoru veće dimenzionalnosti preslikavaju u prostor niže dimenzionalnosti, a varijanca podataka u prostoru niže dimenzionalnosti treba biti maksimalna [27]. PCA je statistički postupak koji koristi ortogonalnu transformaciju kako bi skup koreliranih varijabli pretvorio u skup nekoreliranih varijabli. PCA je najčešće korišten alat u istraživačkoj analizi podataka i u strojnom učenju za prediktivne modele, te je tehnika nenadziranog učenja koja se koristi

za ispitivanje međusobnih odnosa između skupa varijabli. Glavna svrha PCA je smanjiti dimenzionalnost skupa podataka i pritom sačuvati najvažnije obrasce ili odnose između varijabli i zadržati veći dio informacija iz uzorka. Još jedno svojstvo koje PCA omogućuje je vizualizacija, jer dvije ili tri dimenzije je moguće nacrtati, dok više od tri nije moguće ili je vrlo teško moguće.

t-SNE (*t-distributed Stochastic Neighbor Embedding*) je tehnika nelinearnog smanjenja dimenzionalnosti podataka koja se koristi za istraživanje podataka i vizualizaciju podataka visoke dimenzionalnosti [28]. Nelinearno smanjenje dimenzionalnosti znači da algoritam omogućuje razdvajanje podataka koji se ne mogu razdvojiti ravnom linijom. Često se koristi za vizualizaciju visokodimenzioniranih podataka u dvije i tri dimenzije, što omogućuje bolje razumijevanje osnovnih uzoraka i odnosa u podacima.

Razlika između PCA i t-SNE se očituje u čuvanju varijance u podacima kod PCA, dok t-SNE čuva odnose između podatkovnih točaka u niždimenzionalnom prostoru.

1.5.3. Metrike

Metrike ili mjerila izvedbe u strojnom učenju imaju ključnu ulogu u procjeni učinkovitosti i pouzdanosti modela. Metrike omogućuju programerima finu prilagodbu algoritama i poboljšanje rezultata. Metrike se mogu kategorizirati u dvije glavne vrste, to su regresijske metrike i klasifikacijske metrike.

Metrike za regresiju su:

- srednja apsolutna pogreška (engl. *mean absolute error*),
- srednja kvadratna pogreška (engl. *mean squared error*),
- korijen srednje kvadratne pogreške (engl. *root mean squared error*) i
- koeficijent determinacije.

Metrike za klasifikaciju su:

- matrica konfuzije (engl. *confusion matrix*),
- točnost (engl. *accuracy*),
- preciznost (engl. *precision*),
- odziv (ili opoziv) (engl. *recall*),
- harmonijska sredina preciznosti i odziva (engl. *F1 score*) i

- područje ispod krivulje radne karakteristike prijamnika ili AU-ROC (engl. *area under the receiver operating characteristic curve*) [29].

Iako se algoritam k-NN koristi i za probleme klasifikacije i regresije, u sustavima preporuka ne koristi se nijedno ni drugo. Kod sustava preporuka koriste se udaljenosti (ili sličnosti) između tražene stavke i svih ostalih stavki, tako da nije moguće koristiti metrike niti za klasifikaciju niti za regresiju. No, međutim udaljenosti se ipak mogu shvatiti kao svojevrsna regresija, jer zapravo imamo neki numerički niz koji linearno raste. Tako da je moguće zapravo napraviti MAE, MSE i RMSE tako što ćemo napraviti predviđanja udaljenosti za cijeli skup podataka i samo za jednu pjesmu za koju tražimo udaljenosti tj. preporuke.

Kod k-NN algoritma za sustave preporuka moguće je napraviti i vremenski test za skalirane podatke putem min-max načina skaliranja, i/ili podatke s manjim brojem dimenzija putem algoritama dimenzioniranja PCA ili t-SNE. Još jedan mogući test je usporedba tražene pjesme s predloženim pjesmama kako bi se procijenilo ima li ta lista ikakvog smisla, no za to je potrebno dublje razumijevanje skupa podataka. S obzirom da je skup podataka istrugan s mojeg Spotify računa, usudio bih se reći kako mogu dobro analizirati primjere koji se nalaze u ovom skupu podataka.

S obzirom da su algoritmi K-Means i DBSCAN algoritmi nenadziranog učenja, i da ovi algoritmi stvaraju grupe podataka, njihove metrike su nešto drugačije. Metrike kod algoritama nenadziranog učenja su:

- koeficijent siluete (engl. *silhouette coefficient*),
- indeks Calinski-Harabasz (engl. *Calinski-Harabasz index*),
- indeks Davies-Bouldin (engl. *Davies-Bouldin index*),
- Dunnov indeks (engl. *Dunn index*),
- Xie-Beni rezultat (engl. *Xie-Beni score*) [21],
- indeks Ball-Hall (engl. *Ball-Hall index*) [30].

Koeficijent siluete je tehnika mjerenja sličnosti podataka unutar grupe u usporedbi s drugom grupom. Rezultat koeficijenta siluete je numerička vrijednost između -1 i 1, uključujući i te brojeve. Vrijednost 1 znači da je svaka grupa potpuno odvojena od svih ostalih i to je najbolji rezultat, dok vrijednost -1 ima značenje da su svi podaci dodijeljeni pogrešnim grupama. Vrijednost 0 znači da nije moguće stvoriti smislene grupe iz podataka.

Indeks Calinski-Harabasz ili kriterij omjera varijance koristi se za procjenu kvalitete grupa mjerenjem omjera disperzije između grupa i disperzije unutar grupe. Osnova je izmjera razlike između zbroja kvadrata udaljenosti podataka između grupa i podataka unutar grupe. Najbolji rezultat je što veći i to znači da su grupe dobro odvojene. Mali problem je što ne postoji definirani raspon, tj. gornja i donja granica, pa je ova metrika dobra za odabir najboljeg broja k. Rezultat ovog indeksa osjetljiv je na broj grupa i veći broj grupa može dovesti do većeg rezultata. Preporuka je da se uz ovu metriku koriste i druge metrike za potvrdu rezultata.

Indeks Davies-Bouldin se mjeri izračunom prosječne sličnosti između svake grupe i njene najsličnije. Omjer udaljenosti unutar grupe i udaljenosti između grupa izračunava sličnost, to znači da će udaljenije i manje raspršene grupe dovesti do boljih rezultata. Indeks Davies-Bouldin ima za cilj imati što je moguće manji rezultat. Što je rezultat niži, to je svaka grupa više (bolje) odvojena.

Dunnov indeks je odnos najmanje udaljenosti između podataka iz različitih grupa i najveće udaljenosti između grupa, što je veći odnos to je rezultat bolji. Xie-Beni indeks je omjer čiji je brojnik procjena razine zbijenosti podataka unutar iste grupe, a čiji je nazivnik procjena razine odvajanja podataka između različitih grupa, bolji je manji rezultat. Ball-Hall indeks je mjera raspršenosti temeljena na kvadratnim udaljenostima točaka grupa u odnosu na njihov centroid. Manji indeks ukazuje na manju raspršenost podataka unutar grupe, što se smatra poželjnim u grupiranju. Stoga, cilj je minimizirati Ball-Hall indeks kako bi se postigao bolji rezultat.

1.6. Slučajevi korištenja

Često dok čovjek „surfa“ bespućima interneta može naići na poruke tipa: „proizvodi koji vam se mogu svidjeti“, „ljudi također traže“ ili „kupci koji su ovo kupili također su kupili“. Sve prethodne navedene poruke dio su *user-friendly* poruka koje se obično nalaze uz stavke ili proizvode koje sustavi preporuka imaju za predložiti. Jedna od svrha sustava preporuka je i daljnje zadržavanje na trenutno posjećenoj web stranici, a kruna sustava preporuka je odabir ponuđenog ili ponuđenih proizvoda ili stavki. Sustavi preporuka se najčešće koriste u e-trgovini kao što je maloprodaja, u medijima i zabavi, društvenim platformama, platformama za pretraživanje sadržaja, bankarstvu i financijama itd.

Sustavi preporuke u e-trgovini i maloprodaji su vrlo važni alati za poboljšanje korisničkog iskustva, povećanje prodaje i povećanje lojalnosti kupaca. Oni analiziraju ponašanje i povijest kupovine korisnika, te na temelju tih podataka predlažu proizvode (ili usluge) koji bi kupcu mogli biti zanimljivi i/ili korisni. Za primjer se može uzeti trgovina informatičkom opremom Links u kojoj kupac kupi npr. matičnu ploču i najčešće negdje u blizini (često ispod artikla) sadržaja pojavi se preporuka koji bi još dijelovi odgovarali toj matičnoj ploči. Ova preporuka je korisna i za kupca i za prodavatelja, za kupca jer zna koji su dijelovi kompatibilni (poznato je da uvijek postoje u istom razdoblju tri ili četiri generacije procesora, memorija, matičnih ploča itd.) i što odabrati, a za prodavača naravno daljnja zarada. Jedan od primjera e-trgovine je Amazon[1] koji u svom dućanu na internetu samo za američko tržište ima preko 400 milijuna proizvoda, a bez sustava preporuke teško bi bio jedan od najvećih svjetskih e-trgovina. Prema McKinseyu [10], 35% prihoda Amazona ostvaruje se zahvaljujući njegovom statusu preporuka.

Što se tiče glazbene industrije, također postoji obilje glazbe pa postaje izazovno sa strane slušatelja odabrati što slušati i kako doći do nove glazbe, a s druge strane kod samih servisa ponuditi dobar sustav preporuka i novu glazbu. Sustavi preporuka za glazbu puno su složeniji [31] od drugih sustava za preporuke zbog nekoliko karakteristika (značajki). Prvi je trajanje glazbenog zapisa, obično između 3 do 5 minuta, te s obzirom da je kraći nego filmski zapis, čini ga lakše zaboravljivim. Dakle, ta kratkotrajnost glazbenih pjesama doprinosi njihovoj relativnoj prolaznosti u pamćenju slušatelja u usporedbi s drugim umjetničkim oblicima poput filmova (ili knjiga) koji pružaju dublje i dulje iskustvo koje ostaje u pamćenju. Sljedeća karakteristika je glazbeni katalog koji je mnogo širi, pjesme se broje u milijunima u odnosu na filmove, što može činiti skalabilnost sustava jako izazovno. Kontinuiran pristup izvođenju pjesama također je izazov jer je glazba češće konzumirana, pravilan raspored je nužan za dobro funkcioniranje sustava, korisnici često slušaju jednu te istu pjesmu za redom, što nije slučaj kod filmova. Ponašanje pri slušanju je takvo da konzumenti sadržaja često uopće ne obraćaju pažnju na pjesmu, točnije važan im je samo osjećaj koji pjesma izaziva. Neki od glazbenih sustava su Tidal[2], Deezer[3], YouTube[4] i

[1] https://www.amazon.com/
[2] https://tidal.com/
[3] https://www.deezer.com/
[4] https://www.youtube.com/

Spotify[5] čiji skup podataka koristim u ovom radu, koji je istrugan s moje Spotify liste sa sljedeće *web* poveznice:

- https://open.spotify.com/playlist/12w1a30iLX7c8csQRHweMt.

[5] https://open.spotify.com/

2. Sustavi preporuke u glazbi

2.1. Odabir tehnologijske platforme za usporedbu algoritama

Što se tiče odabira tehnologijske platforme, razmatrane su dvije platforme: Googleov alat Colaboratory ili skraćeno Colab[6] i Microsoft Visual Studio Code[7].

Visual Studio Code je proširiv kroz dodatke, što omogućuje prilagodbu okruženja prema potrebama. Postoji veliki broj dodataka koji podržavaju rad s raznim bibliotekama i alatima koji podržavaju strojno učenje kao što su TensorFlow[8], PyTorch[9], scikit-learn[10] itd. Također ima bogato integrirano okruženje za razvoj što uključuje sintaksno označavanje, automatsko dovršavanje kôda, refaktoriranje kôda i upravljanje projektima. Podržava funkcije za ispravke (engl. *debug*). Podržava kontrolu verzija poput Git-a, što omogućuje programerima da upravljaju svojim kôdom i surađuju s drugima na projektima.

Google Colab omogućuje korisnicima izradu, izvršavanje i dijeljenje bilježnica (engl. *notebook*) s Python[11] kôdom u pregledniku, koristeći resurse u oblaku. Colab pruža besplatnu upotrebu grafičkih procesora (GPU) i jedinice za obradu tenzora (engl. *Tensor Processing Unit)* (TPU) za ubrzavanje izvršavanja kôdova. Colab dolazi s već instaliranim popularnim paketima i bibliotekama za strojno učenje kao što su Pandas[12], NumPy[13], TensorFlow, PyTorch, scikit-learn itd.

Ukratko, Visual Studio Code pruža bogato integrirano okruženje za razvoj i proširivost, dok Google Colab nudi besplatno okruženje u oblaku s GPU i TPU podrškom, zajedno s predinstaliranim paketima za strojno učenje i mogućnost interaktivnog rada u bilježnicama. Oba alata imaju svoje prednosti i često se koriste zajedno u radu na projektima strojnog učenja.

[6] https://colab.research.google.com/
[7] https://code.visualstudio.com/
[8] https://www.tensorflow.org/
[9] https://pytorch.org/
[10] https://scikit-learn.org/stable/
[11] https://www.python.org/
[12] https://pandas.pydata.org/
[13] https://numpy.org/

Nakon svega ovoga, iako i jedna i druga platforma imaju svoje prednosti i rijetke nedostatke, odlučio sam se za Google Colab iz jednog važnog razloga. Taj razlog je njegova jednostavna prenosivost na bilo koju platformu s obzirom da radi u bilo kojem pregledniku. Tako se može koristiti primjerice na računalu s Windows 10[14], na laptopu s Linux Ubuntu[15] te čak na pametnom telefonu s Android[16] operativnim sustavom, dakle nema ograničenja s platformom niti preglednikom. Visual Studio Code također radi na mnogim operativnim sustavima, ali treba više instaliranja i prilagodbe. Prednost Visual Studio Code u odnosu na Colab je da nema ograničenja koliko Jupyter bilježnica može biti istodobno pokrenuto, dok je kod Colab osnovne verzije ograničenje svedeno na tri istodobno pokrenute bilježnice.

2.1.1. Google Colaboratory

U radu se kao platforma za testiranje algoritama koristi Google Colab. Google Colab je besplatna platforma za izvođenje Python kôda u oblaku. Google Colab omogućuje korisnicima pisati, dijeliti i izvršavati Python skripte i Jupyter bilježnice putem internetskog preglednika, preduvjet korištenja Google Colaba je prethodna izrada Google računa. Google Colab je platforma u kojoj je već instaliran Python, te velik broj biblioteka za Python. Dodatne biblioteke se mogu instalirati uz pomoć naredbe `!pip install <ime biblioteke>`. Colab je posebno popularan među osobama koje se bave analizom podataka, strojnim učenjem, dubokim učenjem te raznim drugim istraživanjima.

Korisnici mogu izvršavati Python kôd na virtualnim strojevima u Google-ovom oblaku bez potrebe za kupnjom skupih računalnih resursa. Međutim, iako je Colab besplatan ipak postoje određena ograničenja, pa pri radu Colab može prestati raditi zbog nedostatka memorije. Ograničenja su osnovni CPU, GPU i nedovoljno radne memorije. Postoje proširene i bolje verzije Colaba koje treba platiti, a jedna od značajki je i automatski završetak pisanja kôda uz pomoć umjetne inteligencije, što je u trenutku pisanja ovog rada dostupno samo u SAD-u.

Google Colab ima podršku za Jupyter bilježnice, što omogućuje interaktivno pisanje kôda, uključivanje tekstualnih opisa i vizualizacija te dijeljenje bilježnica s drugima. Više korisnika može surađivati na istoj bilježnici u stvarnom vremenu, što uključuje zajednički

[14] https://en.wikipedia.org/wiki/Windows_10
[15] https://ubuntu.com/
[16] https://www.android.com/

rad i dijeljenje ideja. Colab bilježnica najjednostavnije se može podijeliti putem internetske poveznice preko preglednika. Google Colab omogućuje pristup grafičkim procesorima (GPU) i tenzornim procesorima (TPU) za ubrzanje obrade podataka, što je posebno korisno za zadatke strojnog učenja i dubokog učenja. Colab je integriran s Google Drive-om, što olakšava spremanje i dijeljenje bilježnica pohranjenih u oblaku.

2.1.2. Odabir algoritama za usporedbu

S obzirom da se ovaj rad bavi sustavom preporuka prema sadržaju (engl. *content-based filtering*), korištene su dvije vrste algoritama. U radu je korišten algoritam nadziranog učenja k-NN, koji u sustavu preporuka postaje algoritam nenadziranog učenja, te s druge strane čiste algoritme nenadziranog učenja. Korišteni algoritmi su K-Means i DBSCAN. Algoritmi k-NN i K-Means uzeti su za istraživanje iz razloga njihove popularnosti što se tiče sustava preporuka, dok algoritam DBSCAN više služi za testiranje i usporedbu s algoritmom K-Means. Što se tiče k-NN algoritma, on će se testirati putem vremenske komponente te putem različito skaliranih i dimenzioniranih podataka, a još jedna vrsta testa su MSE, RMSE i MAE.

2.1.3. Odabir metrika

Već je navedeno da se metrike razlikuju za nadzirano i za nenadzirano učenje. Za k-NN se tako koriste različiti podatci, točnije rezimirati ću kako preprocesirani podatci daju najbolje rezultate. Kombinacije koje se koriste su čisto min-max skaliranje, min-max skaliranje uz algoritam dimenzioniranja PCA na dvije i tri dimenzije, te min-max skaliranje uz algoritam dimenzioniranja t-SNE uz pet različitih stopa učenja, o čemu će poslije još biti riječi. Još jedan parametar je uzet a to je k, odnosno korišten je k = 10, i k koji iznosi duljinu cijelog skupa podataka, što je moguće zapravo samo u manjim skupovima podataka. Koriste se još MAE, MSE i RMSE, na sljedeći način, pravi se izračun udaljenosti za cijeli skup podataka i samo za pjesmu za koju se traži preporuka, dakle na taj način možemo dobiti stvarne vrijednosti i predviđene vrijednosti. Da bih dobio vrijednost MSE od stvarnih vrijednosti se oduzimaju procijenjene vrijednosti nakon čega se vrši kvadriranje i zatim se pronalazi prosječna vrijednost, formula je pod brojem 4.

$$mse = mean((stvarne\ vrijednosti - predviđene\ vrijednosti)^2) \qquad (4)$$

Što se tiče vrijednosti RMSE, ona nije ništa drugo nego drugi korijen od MSE, formula pod brojem 5.

$$rmse = \sqrt{mse} \tag{5}$$

Posljednja mjera koja se koristi je MAE, da bi se dobila vrijednost MAE od stvarnih vrijednosti se oduzimaju predviđene vrijednosti što sve skupa mora ići u apsolutne zagrade, a zatim pronalazimo prosjek, što je formula pod brojem 6.

$$mae = mean|stvarne\ vrijednosti - predviđene\ vrijednosti| \tag{6}$$

Kod nenadziranog učenja također se koriste podatci kao i kod algoritma k-NN te se koriste dobro poznate metrike, metoda lakta, koeficijent siluete i Calinski-Harabasz koeficijent koji su opisani u podnaslovu „Metrike“.

2.1.4. Odabir skupa podataka

Iako skupova podataka koji se sastoje od značajki koje opisuju pjesmu/e ima dostupnih na otvorenim spremištima podataka kao što je Kaggle[17], odlučio sam se ipak za struganje podataka sa Spotify API-a. Struganje znači spremanje informacija o pjesmama uzimanjem poveznice nekog određenog popisa pjesama, u ovom slučaju jednog od mojih popisa pjesama koji sam stvorio isključivo u svrhu ovog diplomskog rada. Razlog izrade vlastitog skupa podataka je dobro poznavanje pjesama koje se nalaze u tom popisu pjesama. Na taj način također mogu ocijeniti koliko dobro rade odabrani algoritmi. Atributi od kojih se sastoji određena pjesma su: *track_uri*, *track_name*, *artist*, *track_popularity*, *genres*, *track_explicit*, *release_date*, *danceability*, *valence*, *energy*, *mode*, *tempo*, *loudness*, *speechiness*, *instrumentalness*, *liveness*, *acousticness*, *key* i *duration_ms*. U tablici 2 dani su prijevodi atributa kao i njihovi opisi.

Tablica 2 Opis atributa skupa podataka [17]

Track_uri	Jedinstveni identifikator pjesme.
Track_name	Naziv pjesme.

[17] https://www.kaggle.com/

Artist	Izvođač pjesme.
Track_popularity	Popularnost pjesme u trenutku uzimanja podataka. To znači da se popularnost može mijenjati, raspon od 0 do 100.
Genres	Žanrovi pjesme, koji su zapravo žanrovi cijelog albuma.
Track_explicit	Da li pjesma sadrži eksplicitne riječi pjesama ili ne.
Release_date	Datum izdavanja pjesme.
Danceability	Opisuje koliko je pjesma prikladna za ples, koji se oslanja na kombinaciju glazbenih elemenata koji uključuje tempo, stabilnost ritma, snagu udarca i opću pravilnost. Vrijednost 0,0 je najmanje plesno, dok je 1,0 najviše plesno.
Valence	Mjera koja iznosi od 0,0 do 1,0 i koja opisuje glazbenu pozitivnost koju posjeduje pjesma. Glazba s visokom valencom zvuči pozitivnije (npr. sretno, vedro, euforično), dok zapisi s niskom valencom zvuče negativnije (npr. tužno, potišteno, ljuto).
Energy	Energija je mjera od 0,0 do 1,0 i predstavlja percepcijsku mjeru intenziteta i aktivnosti. Tipično, energične glazbene kompozicije doživljavamo kao brze, glasne i bučne. Na primjer, *death metal* ima visoku energiju, dok je Bachov preludij nisko pozicioniran na skali. Percepcijske značajke koje pridonose ovoj karakteristici uključuju dinamički raspon, percipiranu glasnoću, timbre, stopu pojave i opću entropiju.
Mode	Pokazuje da li je pjesma svijetle (engl. *major*) ili tamne (engl. *minor*) boje, gdje je *major* = 1, dok je *minor* = 0. *Major* zvuči svijetlo, veselo i optimistično, osnovna skala *major* tonaliteta sastoji se od 7 nota, raspoređenih prema određenom uzorku polutonova i cijelih tonova. Često se povezuje s pozitivnim emocijama i slavnim, sretnim melodijama. *Minor* zvuči tamno, melankolično i introspektivno, osnovna skala *minor* tonaliteta također se sastoji od 7 nota, ali s drugačijim uzorkom polutonova

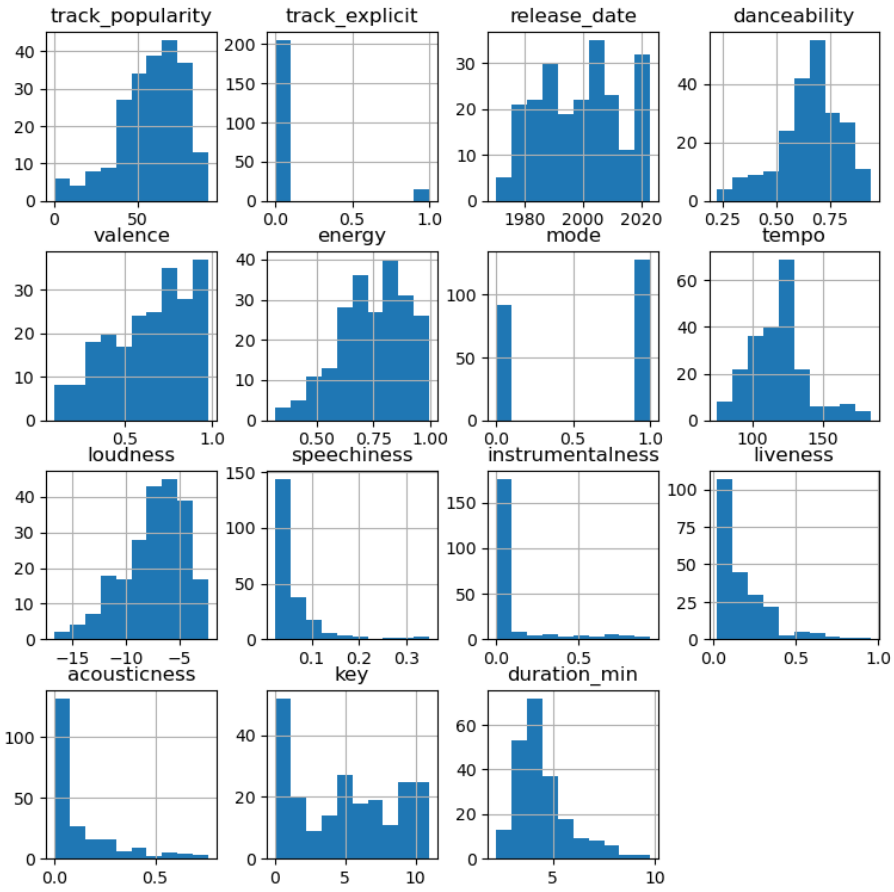
	i cijelih tonova od *major* skale. Često se povezuje s melankolijom, tugom ili ozbiljnim emocijama. Vrlo je važno razumjeti razlike između *major* i *minor* tonaliteta, jer ove razlike imaju veliki utjecaj na doživljaj glazbenog djela. U praksi, skladatelji biraju *major* ili *minor* tonalitet ovisno o željenom izrazu emocija u glazbi.
Tempo	Opća procijenjena brzina glazbene kompozicije izražena u udarcima po minuti (BPM). U glazbenoj terminologiji, tempo označava brzinu ili ritam određenog glazbenog djela i izravno proizlazi iz prosječnog trajanja udarca.
Loudness	Opća glasnoća glazbene kompozicije izražena u decibelima (dB). Vrijednosti glasnoće su prosječne kroz cijelu kompoziciju i korisne su za usporedbu relativne glasnoće različitih kompozicija. Glasnoća predstavlja kvalitetu zvuka koja je glavni psihološki korelat fizičke jačine (amplitude). Tipične vrijednosti kreću se između -60 i 0 dB.
Speechiness	Detektira prisutnost izgovorenih riječi u glazbenoj kompoziciji. Što je snimka više slična govoru (npr. *talk show*, audioknjiga, poezija), to je vrijednost atributa bliža 1,0. Vrijednosti iznad 0,66 opisuju kompozicije koje su vjerojatno potpuno sastavljene od izgovorenih riječi. Vrijednosti između 0,33 i 0,66 opisuju kompozicije koje mogu sadržavati i glazbu i govor, bilo u određenim dijelovima ili slojevima, uključujući primjere poput *rap* glazbe. Vrijednosti ispod 0,33 vjerojatno predstavljaju glazbu i druge kompozicije koje nisu slične govoru.
Instrumentalness	Predviđa sadrži li glazbena kompozicija vokale. Zvukovi poput „ooh“ i „aah“ smatraju se instrumentalima u ovom kontekstu. *Rap* ili govorne kompozicije jasno se smatraju „vokalnima“. Što je vrijednost instrumentalnosti bliža 1,0, to je veća vjerojatnost da je kompozicija instrumentalna.
Liveness	Prepoznaje prisutnost publike u snimci. Veće vrijednosti živosti predstavljaju povećanu vjerojatnost da je glazbeni trag izveden uživo. Vrijednost iznad 0,8 ukazuje na snažnu vjerojatnost da je trag izveden uživo.

Acousticness	Mjera pouzdanosti od 0,0 do 1,0 koja odražava je li glazbeni trag akustičan. Vrijednost 1,0 predstavlja visoku pouzdanost da je trag akustičan.
Key	Procijenjeni opći tonalitet glazbenog traga. Cijeli brojevi se preslikavaju na visine pomoću standardne notacije tonova razreda (engl. *pitch class*). Primjerice, 0 = C, 1 = C♯/D♭, 2 = D i tako dalje. Ako tonalitet nije detektiran, vrijednost je -1.
Duration_ms	Trajanje pjesme u milisekundama.

2.1.5. Analiza podataka

Nakon što su podaci istrugani sa Spotify API-a, slijedi sređivanje tih podataka. Skup podataka koji je skinut uz pomoć Spotify API-a i biblioteke spotipy, zapravo je skup svih pjesama s različitih listi koje imam na svom Spotify računu objedinjene u jednom popisu pjesama. Trenutni skup podataka sastoji se od 228 pjesama tj. redaka i sadrži 19 stupaca. Prvo slijedi uklanjanje duplih pjesama, nakon čega slijedi provjera da li postoje ćelije koje ne sadrže vrijednosti, nakon svega skup spada na 212 pjesama. Što se tiče prilagodbe skupa podataka, vrijednosti u stupcu *release_date* svedene su samo na godinu, a stupac je preimenovan u *release_year*. Vrijednosti u stupcu *duration_ms* su konvertirane iz milisekundi u minute, a stupac je preimenovan u *duration_min*. Vrijednosti u stupcu *track_explicit* su zamijenjene iz True i False u 1 i 0.

Slijedi pregled najvažnijih značajki koje opisuju pjesmu, na slici 4 može se vidjeti distribucija po pojedinom numeričkom atributu, pri čemu bi se jedino za stupac *danceability* moglo reći da naginje normalnoj distribuciji. Vidljivo je kako niti jedan stupac značajki nema normalnu distribuciju. Što se tiče najvećih vrijednosti u cijelom skupu podataka, najbliža godina u kojoj ima pjesama je 2023., najdalja je 1970.. Najveća vrijednost *danceability* je 0,942, najmanja je 0,224, najveća vrijednost za *valence* je 0,979, dok je najmanja 0,0948. Najveća vrijednost za *energy* je 0,996, dok je najmanja 0,318, najbrži *tempo* je od 184 bpm, a najsporiji je od 75 bpm. Nema smisla navoditi baš sve vrijednosti, ali navest ću još kako je najduže trajanje pjesme od 9,76 min, dok je najkraće trajanje pjesme 2,2 min.

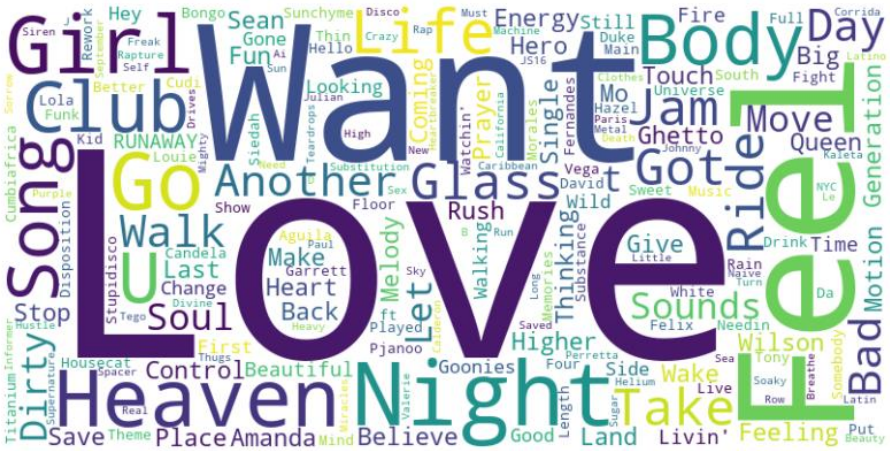


Slika 4 Distribucije podataka

Što se tiče prosječnih vrijednosti, srednja vrijednost popularnosti pjesama je 53,63, najmanja popularnost iznosi 0, dok najveća popularnost iznosi 89. Prosječan tempo svih pjesama je 119,04 udaraca po minuti uz standardno odstupanje od 19,89 udaraca po minuti. Prosječno trajanje pjesama je 4,48 minute uz standardno odstupanje od 1,36 minuta.

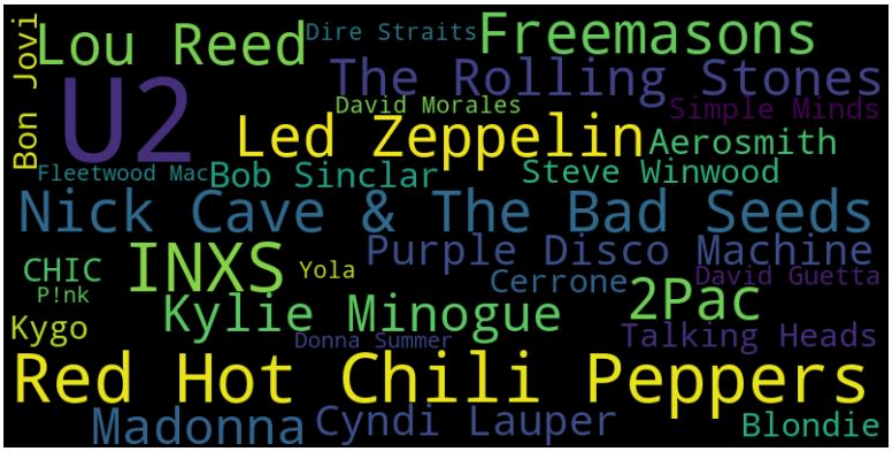
U pogledu broja jedinstvenih izvođača, on iznosi 162 izvođača, vezano iz koliko godina postoje pjesme, taj broj iznosi 51, dakle od 1970. do 2023. iz samo dvije godine ne postoje pjesme. U vezi tipova podataka, stupci *track_uri*, *track_name*, *artist* su stringovi, dok je stupac *genres* također niz znakova (prikazan kao lista), no u Pandas tablici oni se navode kao tip podatka objekt. Zatim su *track_popularity*, *release_date*, *key*, *track_explicit* i *mode* cjelobrojni tip podatka, s tim da dva zadnja poprimaju samo dvije vrijednosti 0 i 1, dakle možemo reći da koriste binarni sustav. Ostatak stupaca poprimaju float odnosno decimalne vrijednosti.

U odnosu na broj najčešćih riječi ili frekvencija u naslovima pjesama, vidljivo je da je najčešća riječ *Love*, zatim *Want*, *Feel*, *Night*, *Heaven* itd. Na slici 5 može se vidjeti *wordcloud* s najčešćim riječima prikazane od najveće i najčešće do najmanje i najrjeđe.



Slika 5 Wordcloud naslova pjesama

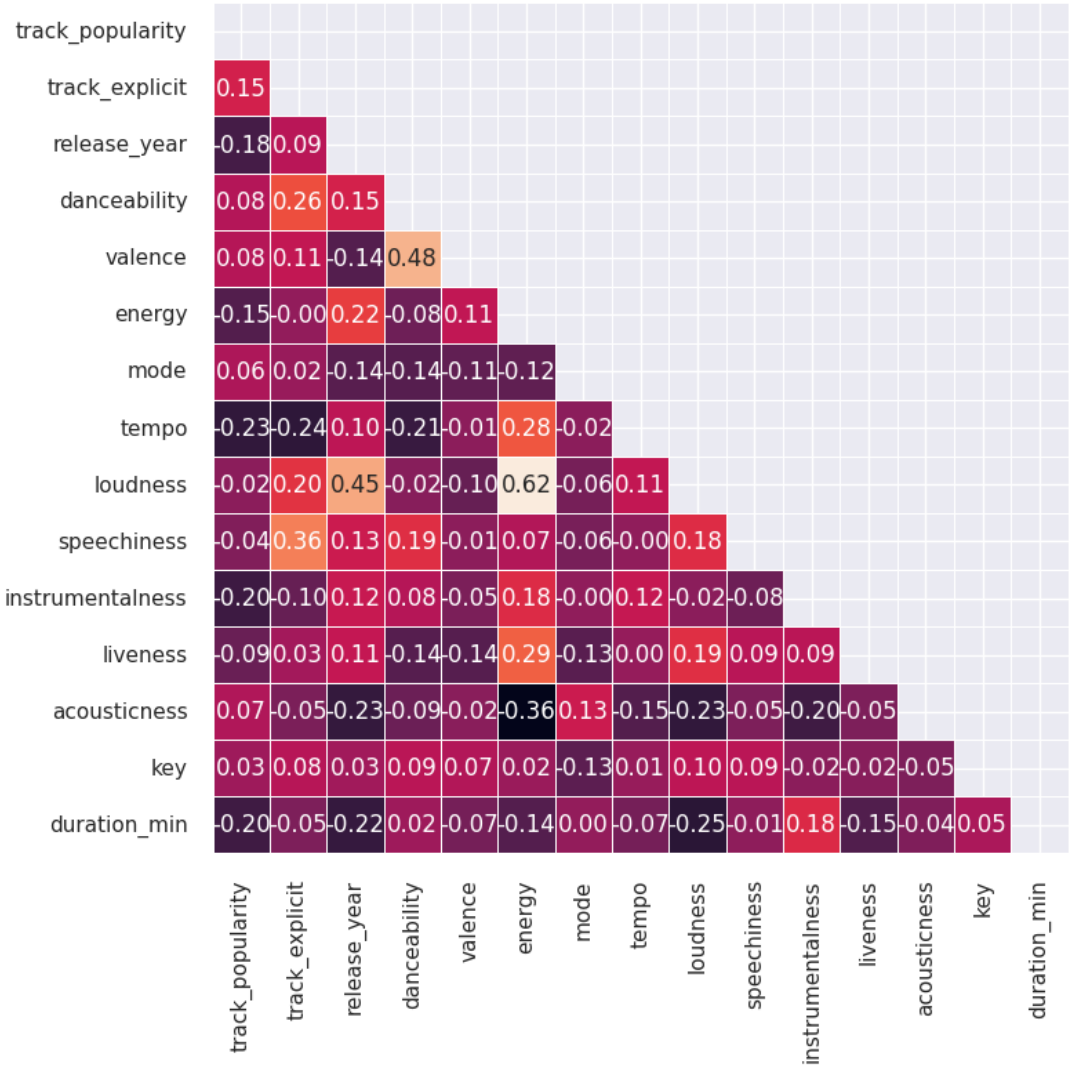
S obzirom na izvođače, napravljen je također prikaz u *wordcloudu*. Najčešći izvođači koji se nalaze u skupu podataka su redom U2, Red Hot Chili Peppers, INXS, Lou Reed itd. Cijeli *wordcloud* može se vidjeti na slici 6.



Slika 6 Wordcloud izvođača

Najpopularnija pjesma u trenutku pisanja ovog rada je pjesma od *Taylor Swift – Anti-Hero* s vrijednošću popularnosti od 89. Slijedi *Bon Jovi* s pjesmom *Livin' On A Prayer* s vrijednošću 86. Najmanje popularna pjesma s vrijednošću 0 je pjesma od *The Charlatans – NYC (There's No Need To Stop)*. Vidljivo je također kako postoji jako malo pjesama sa svojstvom *track_explicit* koje imaju vrijednost 1. Što se tiče vrijednosti *mode*, tj. da li je pjesma pozitivna, vesela i sl. nešto je više pjesama koje su *major* od *minor*, tj. više je veselih pjesama, za otprilike 20%. U pogledu broja pjesama po godinama, najviše pjesama 11 ima iz 2023. godine, zatim 1987. godine. Dok, najmanje pjesama ima iz 1971., 1972., 1973., 1976. i 2015. godine, dok iz godine 1980. ima četiri pjesme.

Za daljnju analizu podataka upotrijebljen je Pearsonov koeficijent korelacije. Nakon čega slijedi izrada matrice korelacija uz pomoć grafičke reprezentacije u kojem su vrijednosti opisane numeričkom vrijednosti i bojom (engl. *heatmap*), slika 7. Na ovaj način se može najjednostavnije primijetiti koje varijable imaju najjaču povezanost između sebe ili drukčije rečeno, kakav utjecaj varijable imaju jedna na drugu. Prvo će biti prikazana mapa topline bez zavisne varijable, a nešto kasnije u ovom istom odlomku vizualizacija sa zavisnom varijablom.



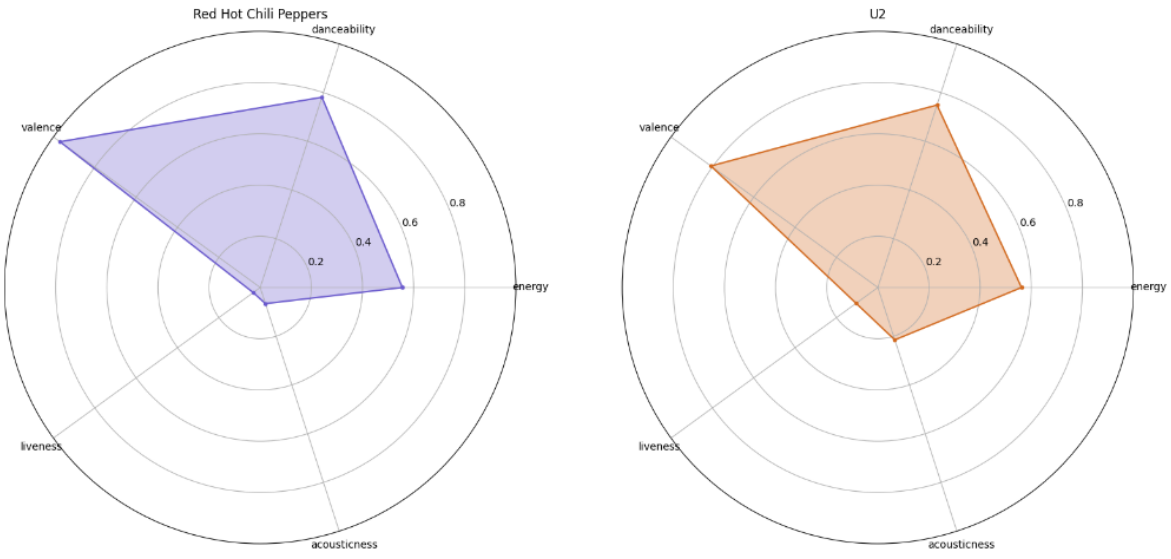
Slika 7 Matrica korelacija bez zavisne varijable

Iz matrice korelacija može se vidjeti kako najveću povezanost imaju varijable *loudness* i *energy* s vrijednošću 0,62, zatim *valence* i *danceability* s vrijednošću 0,48. Među jačim korelacijama još se ističu i *loudness* s *release_year* s vrijednošću 0,45 te *liveness* i *energy* s vrijednošću 0,29. Prethodno nabrojane korelacije su pozitivne korelacije, što znači, kako jedna varijabla raste tako i druga varijabla raste. Još jedna zanimljiva korelacija je

između varijabli *acousticness* i *energy* i iznosi -0,36, što znači kako jedna varijabla raste dok druga varijabla pada.

Da bi mogli vidjeti odnose između dviju varijabli, radimo dijagram raspršenosti (engl. *Scatter-plot*). U dijagramu raspršenosti možemo vidjeti kakav je točan odnos između varijable *energy* i *acousticness*, a vidljivo je dakle kako *energy* raste dok *acousticness* pada, što nismo mogli iščitati iz matrice korelacija. Jedna zanimljivost je i da su pjesme označene sa *track_explicit* imale popularnost najčešće oko vrijednosti 40 (rjeđe) i između 70 i 90 (češće). Može se zaključiti kako su pjesme s eksplicitnim riječima imale veću popularnost. Također se u dijagramu raspršenosti može vidjeti kako su pjesme s eksplicitnim riječima bile izdavane najčešće između 1995. i 2010. godine.

Dalje, ono što možemo vidjeti u kutijastom dijagramu (engl. *Box-plot*) su popularnost najzastupljenijih izvođača, pa tako recimo popularnost *Red Hot Chili Peppers* varira od 40 do 83, medijan iznosi 80,5 dok srednja vrijednost iznosi 69,16. Prvi kvartil iznosi 55,75, treći kvartil iznosi 82,75. Udio *Red Hot Chili Peppers*, *INXS* i *U2* zauzima svaki posebno 27,27% u prva četiri najzastupljenija izvođača, dok je *Lou Reed* zastupljen s 18,18%. Još jedan prikaz koji je interesantan je radarski dijagram (engl. *Radar-plot*), zanimljiv prikaz u kojem se može vidjeti više varijabli istodobno, na slici 8.

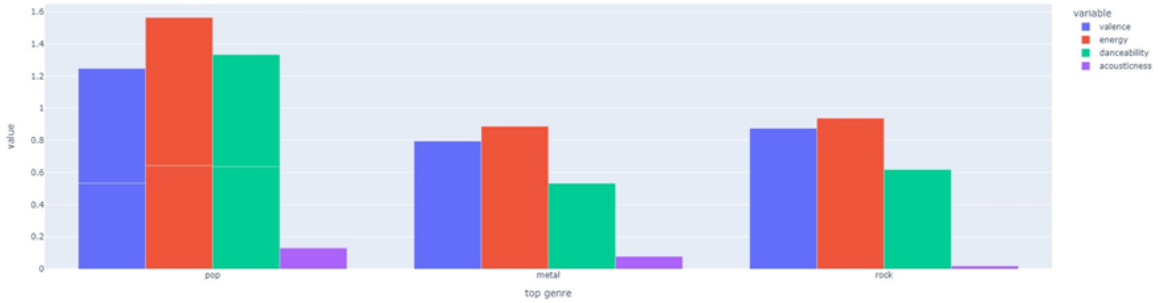


Slika 8 Radarski dijagram

2.1.6. Priprema podataka

Prethodno je već navedeno kako se jedan stupac sastoji od liste žanrova, pa je stoga potrebno svesti taj cijeli niz na samo jedan žanr. Kao što je navedeno ranije ti žanrovi ne

pripadaju samoj pjesmi, već pripadaju cijelom albumu, pa možemo reći kako žanrovi ne odražavaju realno stanje. Ovdje ću svesti žanrove na samo jedan glavni žanr i barem približno svesti pjesmu na jedan žanr, iako se može dogoditi da pjesma ne bude tog žanra. Primjer je album od *The Rolling Stonesa Bridges to Babylon* koji ima zapravo samo jedan žanr i to *rock*, no međutim, pjesma s tog albuma *You Don't Have to Mean it* je zapravo *reggae*. Zanimljivost prije svega što tip podatka *genres* nije prava lista, već samo oponaša listu na način npr. ['rock', 'pop', …], dakle ove uglate zagrade nisu zagrade, već niz znakova (engl. *string*) tj. objekt kod Pandas biblioteke. Zbog svega ovoga ću prvo niz znakova konvertirati u listu (engl. *list*) kao tip podatka. Iz liste izdvojiti prvu vrijednost, jer ta prva vrijednost s najvećom vjerojatnošću opisuje žanr pjesme, što se može provjeriti na servisu Spotify Playlist Analyzer[18]. Jedinstveni broj žanrova koji se nalazi u stupcu *top_genre* je 75, sada treba zamijeniti sve riječi koje sadrže *rock* u sebi, kao npr. *album rock*, sa samo *rock*, riječi koje sadrže samo *pop*, kao što je npr. *new wave pop*, sa samo pop itd. Od svih žanrova na kraju je ostalo samo njih 11, to su: *big beat*, *disco*, *edm* (*electronic dance music*), *folk*, *metal*, *hip hop*, *mellow*, *pop*, *r&b*, *rock* i *soul*. Na slici 9 mogu se vidjeti vrijednosti nekoliko varijabli za određene žanrove.

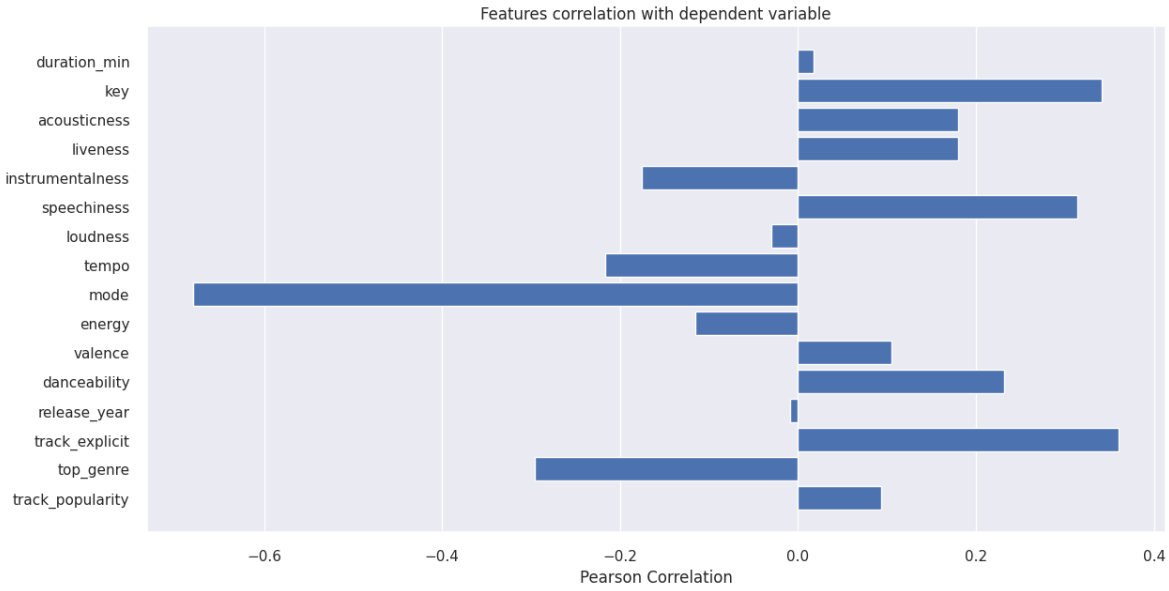


Slika 9 Svojstva *valence*, *energy*, *danceability* i *acousticness* za žanrove *pop*, *metal* i *rock*

Za određivanje najvažnijih prediktora koristi se sam k-NN algoritam na način u kojem on odredi udaljenosti za svaku pojedinu pjesmu, udaljenosti se dodaju u okvir podataka u novi stupac, te na taj način dobivamo zavisnu varijablu. Prije samog treniranja podataka, potrebno je podatke skalirati o čemu će još biti riječi. Najvažniji prediktori mogu se odrediti na više načina. Jedan od njih je klasa `FeatureCorrelation` iz biblioteke `yellowbrick.target`, koji služi za izradu stupčastog dijagrama, te na taj način nam prikazuje najvažnije varijable za predviđanje zavisne varijable uz pomoć Pearsonove korelacije na slici 10. Još jedan način je korištenje matrice korelacije,

[18] https://www.chosic.com/spotify-playlist-analyzer/

slično kao što je već i gore prikazana, samo što sada imamo i dodatnu (zavisnu) varijablu – udaljenost. Još jedan zanimljiv način određivanja važnih značajki je uz pomoć algoritma *Random Forest* [32], koji se je koristio samo radi usporedbe. Opet dobivamo stupčasti dijagram, a razlika između ovog dijagrama i onog iz `yellowbrick` biblioteke je što `yellowbrick` prikazuje varijable i s predznakom minus, dok algoritam *Random Forest* prikazuje samo apsolutne vrijednosti varijabli.



Slika 10 Najvažniji prediktori (biblioteka *yellowbrick.target*)

Za početak treba ostaviti samo one stupce koji imaju brojčane vrijednosti. Iz skupa podataka tako se brišu varijable *track_uri*, *track_name* i *artist*. Kao ekspert za ovo područje smatram da varijabla *duration_min* nema većeg utjecaja na određivanje sličnosti, a isto tako i varijabla *liveness*, te sam stoga obrisao i ova dva stupca. Varijabla *track_uri* je jedinstveni niz znakova za svaku pjesmu sastavljen od velikih i malih slova te brojeva, te se može smatrati identifikatorom, a za uvježbavanje podataka nije potreban. Varijable *track_name* i *artist* su naziv pjesme i izvođač koji su sastavljeni od niza znakova, te se također brišu. Iako bi se varijabla *artist* mogla i ostaviti, a niz znakova zamijeniti s brojevima, no na taj način se može pojaviti pristranost, te sustav može nuditi pjesme istog izvođača češće, što zapravo i nije cilj sustava preporuka.

Isti slučaj može biti i s naslovima pjesama, no tu je izbor ipak malo prevelik za zamjenu s brojevima. Ono što bi se moglo napraviti jest pretvoriti svaku riječ iz naslova pjesme u *token* tj. u broj, no to nije tema ovog rada, ali se može iskoristiti u budućnosti.

Sada ostaju samo stupci čije vrijednosti su sami brojevi, neki imaju samo dvije znamenke kao što su npr. varijable *track_explicit*, *mode*, neki kao što su *release_year* su

godine tj. brojevi od 1970. do 2023. i *key* koji ima brojeve između 1 i 10. Ostale vrijednosti u stupcima su decimalni brojevi od koji su svi pozitivni, a samo je varijabla *loudness* negativna. Dakle, s početnih 19 stupaca skup podataka spao je na 14 stupaca, i s tim značajkama će se nastaviti raditi.

Sljedeći korak koji se radi je skaliranje podataka. Skaliranje služi kako bi se značajke svele na vrijednosti u istom rasponu. Najčešće korišteni načini skaliranja su min-max skaliranje i standardno skaliranje, iako se i drugi načini često koriste, za vrijeme studija najčešće su korištena upravo ova dva navedena. U ovom radu će se koristiti samo min-max način skaliranja. Dao sam prednost korištenju min-max načina skaliranja iz nekoliko razloga. Min-max skaliranje je pogodno kada su vrijednosti podataka većeg raspona i ne podliježu normalnoj distribuciji, osjetljiv je na ekstremne vrijednosti, te zadržava interpretabilnost podataka jer preslikava sve vrijednosti na raspon između 0 i 1. S obzirom da skup podataka korišten u ovom radu ispunjava uvjete za korištenje min-max načina skaliranja, odlučio sam se za isti. Iako je početno zamišljeno korištenje i standardnog načina skaliranja, od tog se odustalo, jer bi sve skupa iziskivalo previše vremena.

Min-max skaliranje se naziva još i normalizacija. Cilj normalizacije je prilagoditi vrijednosti podataka unutar određenog raspona, a vrijednosti su najčešće između 0 i 1, što uključuje i ta dva broja. Tako će npr. brojevi 1, 2 i 3 postati 0, 0,5 i 1. Općeniti razlog zašto se radi skaliranje je što algoritmi bolje rade sa skaliranim vrijednostima. Tehnika korištenja min-max skaliranja često pomaže u poboljšanju performansi algoritama strojnog učenja, a posebno kada su korišteni modeli osjetljivi na različite skale podataka [33]. Postupak min-max skaliranja uključuje transformaciju svake pojedinačne vrijednosti (X) prema formuli 7:

$$X_{minmax} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{7}$$

Gdje je:

- X_{minmax} – skalirana vrijednost (vrijednost s kojom će algoritam nadalje raditi, tj. vršiti izračune),
- X_{min} – najmanja vrijednost u skupu podataka i
- X_{max} – najveća vrijednost u skupu podataka.

Za min-max skaliranje koristi se klasa `MinMaxScaler` koja se nalazi u biblioteci `sklearn.preproccesing`. Za obradu značajki koristi se metoda `fit_transform`. Metoda `fit_transform` se koristi za skaliranje vrijednosti atributa ili značajki (engl. *features*) tako da se oni transformiraju na određeni raspon. Konkretno `MinMaxScaler` koristi transformaciju min-max za skaliranje podataka. Metoda `fit_transform` zapravo radi dvije stvari, fit izračunava minimum i maksimum za svaku značajku iz skupa podataka „X“ i sprema ih unutar „scaler“ objekta. Dok transform skalira stvarne vrijednosti značajki prema izračunatim minimumima i maksimumima, postavljajući ih u određeni raspon. Ova tehnika često pomaže u normalizaciji podataka i poboljšava performanse mnogih algoritama strojnog učenja**,** posebno onih koji su osjetljivi na razmjere ili magnitudu značajki. Nakon što su sve značajke skalirane na potrebne vrijednosti, slijedi treniranje uz pomoć algoritama k-NN, K-Means i DBSCAN.

Velik broj dimenzija može dovesti do problema prenaučenosti, većeg vremena računanja i smanjenja točnosti modela, što je poznato kao „prokletstvo“ dimenzionalnosti koje se javlja prilikom rada s visokodimenzionalnim podacima. Neki algoritmi strojnog učenja mogu biti osjetljivi na broj dimenzija. Zbog ovoga gore nabrojanoga koriste se tehnike inženjeringa značajki, uključujući selekciju i ekstrakciju značajki. Smanjenje dimenzionalnosti vrsta je tehnike ekstrakcije značajki koja za cilj ima smanjiti broj ulaznih značajki zadržavajući što više originalnih informacija [27].

PCA klasa se uvodi iz biblioteke `sklearn.decomposition`. Iako PCA prima nekoliko parametara, ovdje će se koristiti samo `n_components`, a znači koliko želimo napraviti dimenzija. U radu se koriste dvije i tri dimenzije. Upotreba algoritma je jednostavna. Prvo se incijalizira PCA objekt uz korištenje parametra s vrijednošću 2 ili 3. PCA će identificirati dvije glavne komponente koje najbolje zadržavaju varijancu podataka. Zatim se koristi metoda `fit_transform(data)`, gdje su data podaci koje želim dimenzionirati, `fit_transform` primjenjuje PCA transformaciju na ulazne podatke.

TNSE klasa se uvodi iz biblioteke `sklearn.manifold`. Iako se može koristiti veći broj parametara, ovdje se koristi samo parametar `learning_rate` s različitim vrijednostima od 25, 50, 100, 200 i 400. Prvo se vrši inicijalizacija t-SNE objekta s postavljenom brzinom učenja, pri čemu se stvara objekt za t-SNE. Brzina učenja utječe na to kako algoritam prilagođava raspored podataka u niže dimenzije. Brzina učenja (`learning_rate`) je važan hiperparametar u t-SNE algoritmu, različite vrijednosti mogu

dovesti do različitih rezultata vizualizacije. Visoke vrijednosti brzine učenja mogu uzrokovati veće promjene u rasporedu podataka. Metoda `fit_transform` primjenjuje t-SNE transformaciju na originalne podatke.

2.1.7. Realizacija prototipa

Prototip sustava za preporuke sastoji se od dva dijela. Prvi dio sastoji se od sustava preporuka koji vrši filtriranje prema sadržaju i koristi k-NN algoritam koji se kombinira s različitim metrikama (udaljenostima) te s različitim algoritmima pretraživanja. Udaljenosti korištene pri eksperimentiranju su Euklidska, Manhattan i kosinusna, dok su algoritmi pretraživanja brute-force, kd-tree i ball-tree.

Drugi dio prototipa sastoji se od algoritama nenadziranog učenja u što spada K-Means s parametrom `K` koji označava broj grupa (engl. *clusters*) koje algoritam treba naći i parametar `n_init` koji označava broj koliko puta se algoritam pokreće s različitim centroidnim sjemenjem (engl. *seeds*). Sljedeći korišteni algoritma kod nenadziranog učenja je DBSCAN algoritam koji će koristiti dva parametra `eps` koji predstavlja „epsilonski okoliš“ te definira radijus oko svakog podatka i `min_samples` koji označava minimalni broj susjednih točaka potrebnih da bi se određena točka smatrala središnjom točkom grupe.

Praktični dio rada proveden je kroz Google Colab, za koji, kako je već navedeno treba Google račun. Međutim, s obzirom da se s programskim jezicima, a poglavito Python programskim jezikom može automatizirati apsolutno sve, tako je moguće i napisati kôd u Jupyter bilježnici koji razaznaje nalazi li se u Google Colabu ili u nekom drugom uređivaču teksta kao što je npr. Visual Studio Code. Jedan od razloga dodavanja takvog kôda je mogućnost programeru odluke hoće li raditi u jednom ili drugom uređivaču kôda. Ovo se odnosi samo na dio pri učitavanju podataka, dok u ostalim dijelovima nema razlike pri radu s kôdom.

2.1.7.1 k-NN

Kod primjene algoritma k-NN korišteno je nekoliko tehnika. Tehnike koje su korištene prvenstveno se odnose na manipulaciju podacima, a to je skaliranje podataka uz tehniku min-max skaliranja. Zatim je korišteno dimenzioniranje podataka uz algoritme smanjenja dimenzija PCA i t-SNE. Neke tehnike koje su još korištene bit će spomenute kasnije.

k-NN algoritam za određivanje udaljenosti funkcionira na sljedeći način:

1. učitavanje podataka,
2. odabir broja k (broj najbližih susjeda),
3. za svaki primjer u podacima,
    a. iz podataka se izračunava udaljenost između primjera upita i trenutnog primjera,
    b. dodaje se udaljenost i indeks primjera u uređenu zbirku,
4. uređena zbirka složi se od najmanjeg do najvećeg prema udaljenostima, dakle uzlaznim redoslijedom, indeksi prate svaki od podataka u odnosu na udaljenosti,
5. odabire se prvih k unosa iz sortirane zbirke,

Za upotrebu algoritma k najbližih susjeda koristi se klasa `NearestNeighbors` koja se povlači iz biblioteke `sklearn.neighbors`. Klasa `NearestNeighbors` prima nekoliko parametara, a to su `n_neighbors`, koji je zadano bez eksplicitnog navođenja 5, što znači da će prikazati 5 najbližih susjeda, točnije prikazat će pjesmu za koju se traže preporuka s udaljenošću 0 i četiri najbliže pjesme. Zatim prima `radius` koji je zadano 1.0, a koristi se kako bi se definirao radijus unutar kojeg tražimo susjede za određenu točku, no međutim kod metode `kneighbors` radijus se ne koristi, već se koristi kod metode `radius_neighbors` koja pronalazi sve susjede unutar zadanog radijusa, dok metoda `kneighbors` pronalazi k najbližih susjeda. Metoda `radius_neighbors` može također poslužiti za pronalazak najbližih susjeda, samo razlika je što se ne može zadati broj k, već se pronalaze svi najbliži unutar zadanog radijusa.

Sljedeći parametar je `algorithm`, koji je po zadanom svojstvu namješten na „auto“. Svojstvo „auto“ skraćeno je od automatic i znači da će model automatski odabrati najbolji odgovarajući algoritam na temelju podataka koji se obrađuju. Inače, model kod metode može birati između tri postojeća algoritma, to su: brute-force, k-d tree i ball tree. Odabir odgovarajućeg algoritma može ovisiti o broju podataka, dimenzijama prostora i drugim karakteristikama skupa podataka. Korištenje „auto“ omogućava algoritmu da dinamički odabere najprikladniji način izvođenja, što može rezultirati boljim performansama i bržim izvođenjem, posebno kod različitih vrsta podataka ili različitih karakteristika skupa podataka. Brute-force algoritam ima svojstvo linearnog pretraživanja svih podataka. K-d tree (K-dimenzionalno stablo) je binarno stablo koje se koristi za organizaciju prostornih

podataka u višedimenzionalnom prostoru. Ovaj algoritam je često korišten u problemima pretraživanja prostornih podataka, posebno u problemima k-najbližih susjeda. Ideja k-d tree algoritma je organizirati podatke na način da se brzo može identificirati podskup podataka koji su bliski u prostoru. Prednosti k-d tree algoritma uključuju brze upite k-najbližih susjeda u višedimenzionalnom prostoru. Međutim, postoji nekoliko situacija u kojima k-d tree može izgubiti učinkovitost, kao što su visoka dimenzionalnost i neravnomjerna raspodjela podataka. U tim situacijama, drugi algoritmi poput ball tree ili brute-force pretraživanja mogu biti bolji izbor.

Ball tree je još jedan algoritam k-najbližih susjeda koji se koristi za organizaciju prostornih podataka. Ovaj je algoritam posebno koristan kada podaci imaju velik broj dimenzija. Za razliku od k-d tree, ball tree koristi kugle (sferne regije) za predstavljanje skupa podataka. Prednosti ball tree algoritma uključuju visoku učinkovitost u visokodimenzionalnim prostorima, gdje drugi algoritmi poput k-d tree algoritma mogu izgubiti učinkovitost. Međutim, ball tree također može imati svoje nedostatke, posebno u situacijama kada su podaci raspoređeni neravnomjerno ili kada su prostorne razmjere različite u različitim dimenzijama. U takvim situacijama, važno je eksperimentirati s različitim algoritmima i parametrima kako bi se pronašao najučinkovitiji pristup za specifičan skup podataka.

Još jedan parametar koji se koristi je `leaf_size`, te se koristi samo uz algoritme poput k-d tree i ball tree kako bi se kontrolirala veličina listova (terminalnih čvorova) u stablu. Kada broj podataka u čvoru padne ispod vrijednosti `leaf_size`, taj čvor postaje list, a daljnje dijeljenje prostora se zaustavlja. Ovaj parametar utječe na brzinu izgradnje stabla i brzinu upita, pa se njegova vrijednost prilagođava ovisno o specifičnostima skupa podataka i zahtjevima performansi. Zadana vrijednost ovog parametra je 30, i druge vrijednosti se neće koristiti.

Sljedeći parametar je parametar `metric` i odnosi se na mjeru koja se koristi za procjenu udaljenosti između podataka u prostoru. Metrike ili mjere su detaljnije opisane u naslovu koji govori o algoritmu k najbližih susjeda. Zadana metrika je Minkowski, i to je opća metrika koja obuhvaća Euklidsku i Manhattan udaljenost. Parametar `p` kontrolira vrstu norme (kada je p=2, to je Euklidska udaljenost, a p=1 je Manhattan udaljenost). Zadana vrijednost ovog parametra je Minkowski, s p=2 što znači da je zapravo Euklidska udaljenost.

Sljedeći parametar je `metric_params`, koji se slično kao i parametar `radius` koristi samo kod određenih udaljenosti, ne koristi se kod gore spomenutih udaljenosti, već kod npr. Mahalanobisove udaljenosti. Inače zadana vrijednost ovog parametra je `None`.

Zadnji parametar koji se koristi u metodi `NearestNeighbors` je `n_jobs` koji služi za paralelizaciju poslova koji se izvršavaju tijekom treniranja ili predviđanja. Parametar `n_jobs` omogućuje ubrzanje korištenjem više jezgri procesora. Vrijednosti parametra su, `None` koji je i zadan, i znači da će se koristiti samo jedna jezgra tj. rad se odvija serijski. Ako je `n_jobs` postavljen na cijeli broj (npr. 3) tada će se koristiti toliko jezgri koliko je navedeno, ako su dostupne. Ako navedeni broj premašuje ukupan broj jezgri koristit će se sve dostupne. Ako se vrijednost `n_jobs` postavi na -1 koriste se sve dostupne jezgre na računalu. Parametar `n_jobs` najčešće utječe na dijelove algoritma koji mogu paralelno raditi, kao što su treniranje modela, pretraživanje parametara ili predviđanje. Ne podržavaju svi algoritmi paralelan rad, pa parametar `n_jobs` neće uvijek imati učinka. Kod manjih skupova podataka ili jednostavnih modela, paralelizacija može povećati vrijeme izvođenja zbog dodatnih troškova povezanih s paralelnim izvođenjem. Kod velikih skupova podataka i složenijih modela, paralelizacija pomoću `n_jobs` može značajno ubrzati treniranje.

Dakle, za potrebe izrade prvog prototipa modela koji će vršiti preporuku preko algoritma k najbližih susjeda uz pomoć metode `NearestNeighbors` koristiti će se sljedeći parametri:

- broj susjeda (`n_neighbors`) = `10` (i/ili dužina cijelog skupa podataka)
- `radius = 1.0` (ne koristi se)
- `algorithm = auto`
- `leaf_size = 30` (koristi se u ovisnosti o algoritmu koji se koristi)
- `metric = 'euclidean'` ('manhattan' i 'cosine')
- `metric_params = None` (ne koristi se)
- `n_jobs = None` (ne koristi se paralelizacija).

Prvo što se radi je incijalizacija objekta `NearestNeighbors` uz postavljanje parametara. Zatim se koristi inicijalizirani objekt knn za treniranje modela na prethodno skaliranim podacima. Zadnje što se radi je korištenje metode `kneighbors`, koja prima kao parametar pjesmu za koju tražimo preporuke. Iako se podaci za treniranje i tražena pjesma

mogu koristiti u obliku Pandas okvira podataka, bolje je izvući same vrijednosti uz pomoć metode `values`, tako da metode rade s NumPy nizovima.

Dakle, metoda `kneighbors` služi za određivanje udaljenosti i za određivanje indeksa na kojem se nalazi svaka pojedina udaljenost. Metoda kao vrijednosti vraća dvije varijable, (kao što je već rečeno) udaljenosti i pripadajuće indekse u obliku NumPy liste, poredane od najbliže prema udaljenijim vrijednostima. Kada znamo indekse na kojima se nalaze najbliže udaljenosti možemo jednostavno preko metode `iloc[]` saznati koje su uopće to pjesme. Postoji dva načina filtriranja, prvi je za k = 10, u kojem algoritam daje preporuku za prvih 10 najbližih pjesama, te za k = `len(cijeli skup podataka)`, u kojem algoritam provjerava da li žanr pjesme za koju se traži preporuka odgovara žanru preporučene pjesme, ako odgovara onda daje kao preporuku, dok se ne pokaže deset pjesama. Svi parametri korišteni u modelu mogu se saznati preko metode `get_params()`.

Pri treniranju algoritma koristile su se kombinacije Euklidske i Manhattan metrike sa brute, k-d tree i ball tree algoritmima pretraživanja, te kosinusna udaljenost samo sa brute force algoritmom, te u kombinaciji s različito skaliranim podacima i različito dimenzioniranim podatcima, te uz dvije vrijednosti k od 10, i k veličine cijelog skupa podataka.

2.1.7.2 K-Means

Što se tiče skaliranja i dimenzioniranja podataka kod K-Means algoritma potpuno identično se odradilo kao i za algoritam k-NN. Kod algoritma K-Means potrebno je odrediti najbolji K tj. K koji će imati najmanju pogrešku uz pomoć metode lakta [34]. To se radi uz pomoć klase `KElbowVisualizer` koja se nalazi u biblioteci `yellowbrick.cluster`, postoje i neki drugi načini izračuna optimalnog K. Uz `KElbowVisualizer` klasu potrebno je uvesti i klasu `KMeans` iz biblioteke `sklearn.cluster`, jer ta klasa zapravo radi grupiranje, te se koristi skupa s `yellowbrick_cluster` za određivanje optimalnog K. Algoritam određivanja najboljeg K iterira od 1 do određenog broja kojeg sami zadamo, u ovom slučaju iteracija se vrši od 1 do 25, što ne uključuje 25. Metoda lakta odredila je da je najbolji K = 8, uz vrijednost Distortion od 103,127. K = 8, znači i da ćemo ručno pri treniranju podataka zadati algoritmu K-Means treniranje za određivanje 8 grupa. Nakon što su svi podaci uspješno skalirani i dimenzionirani vrši se treniranje odnosno grupiranje podataka.

K-Means prima samo dva parametra, to su broj K, dakle koliko grupa tražimo da kreira te parametar `n_init` koji određuje koliko puta će se algoritam ponoviti s različitim početnim centroidima kako bi se pronašlo optimalno rješenje, najbolje je ipak ostaviti ovu vrijednost na „auto“. U isto vrijeme dok se vrši grupiranje radi se i usputni test koeficijenta siluete kako bi se otkrilo koji će podaci i koja mjera ostvariti najbolje rezultate.

Prikazuje se još i dijagram raspršenosti za različite podatke i metrike, te se zatim provodi i test Calinski-Harabasz, koji je uz test koeficijent siluete još jedan pokazatelj koliko su grupe dobro podijeljene. Kruna svega je prikaz preporuka. Preporuke se odabiru iz originalnog skupa podataka, a odabire se samo naziv izvođača, naziv pjesme i žanr. Žanr pjesme se odabire samo iz razloga da vidimo da li se pjesme uopće uklapaju u traženi žanr. No, kao što je rečeno žanr nam je samo jedan pokazatelj. Tom novom skupu podataka dodaje se novi stupac u kojem su sadržane grupe, odnosno svaki redak dobiti će novu vrijednost koja je zapravo određeni broj koji je algoritam K-Means pri grupiranju odredio. Zatim određujemo pjesmu za koju tražimo preporuke, na način da pronađemo kojoj grupi pripada ta pjesma, u ovisnosti da li želimo 5 ili 10 preporuka ili neki drugi broj, odredimo `head(5)` i na taj način dobijemo preporuke.

2.1.7.3 DBSCAN

Kod DBSCAN algoritma grupiranja koristili su se isti postupci skaliranja i dimenzioniranja podataka. DBSCAN u odnosu na K-Means algoritam ne zahtijeva unaprijed određeni broj grupa, već je sposoban identificirati proizvoljan broj grupa na temelju gustoće podataka, dok je kod K-Means potrebno odrediti unaprijed broj grupa.

Klasa `DBSCAN` uvozi se iz biblioteke `sklearn.cluster`. DBSCAN prima dva parametra, `eps` i `min_samples`. Parametar `eps` označava „epsilonski okoliš“ i predstavlja parametar koji definira radijus oko svakog podatka. Parametar `eps` igra ključnu ulogu u određivanju koje se točke smatraju susjedima unutar gustoće temeljnog grupiranja. Parametar `eps` pomaže definirati gustoću grupa u prostoru podataka, te ga treba pažljivo odabrati kako bi odgovarao specifičnostima problema i prirodi podataka. Ako `eps` bude prevelik, može doći do spajanja različitih grupa, dok premalen `eps` može rezultirati identifikacijom većeg broja manjih grupa ili pojedinih točaka kao šuma (engl. *noise*).

Parametar `min_samples` označava minimalni broj susjednih točaka potrebnih da bi se određena točka smatrala središnjom točkom grupe. Ova vrijednost pomaže u definiranju gustoće grupe i utječe na to kako algoritam klasificira točke u podacima.

Kada algoritam DBSCAN pregledava podatke, on gleda koliko točaka se nalazi unutar epsilonskog okoliša svake pojedine točke. Ako broj točaka unutar tog okoliša (uključujući tu točku) premašuje vrijednost `min_samples`, ta će se točka smatrati središnjom točkom grupe. Točke koje su susjedne središnjoj točki grupe, a da same nisu središnje, dodaju se toj grupi.

Ako točka nema dovoljno susjeda unutar epsilonskog okoliša ili ako je sama središnja točka, ali nema dovoljno susjeda, ta se točka smatra "šumom" ili rubnom točkom, a ne dijelom grupe.

Parametar `min_samples` kontrolira gustoću grupe – veća vrijednost rezultira manjim grupama veće gustoće, dok manja vrijednost može dovesti do većih i manje kompaktnih grupa.

Slično kao i kod K-Means algoritma određuje se najbolji koeficijent siluete kombiniranjem `eps` parametra i parametra `min_samples`. Također je korišten dijagram raspršenosti, a pronalazak preporuka identičan je kao i kod K-Means algoritma.

2.1.8. Usporedba rezultata

Za izradu sustava preporuka korišten je profil svake pojedine pjesme. Profil pjesme sastoji se od različitih atributa. S obzirom da skup podataka sam po sebi nije imao zavisnu varijablu, tj. varijablu koja se predviđa, bilo je potrebno napraviti testno određivanje udaljenosti za svaku pjesmu u okviru podataka, kako bi se zapravo moglo odrediti koji su najvažniji prediktori u okviru podataka, što je i napravljeno pri obradi podataka.

Tako su za predviđanje varijable y, zadužene sve varijable, osim tekstualnih, dok je zavisna varijabla postala varijabla udaljenost. Iz matrice korelacija vidljivo je da najviše utjecaja na udaljenost ima varijabla *mode* s vrijednošću -0,68, zatim varijabla *track_explicit* s vrijednošću 0,36 i kao treća varijabla *key* s vrijednošću 0,34. Najmanjeg utjecaja imaju varijable *release_year*, *duration_min* i *loudness*. Odlučio sam se ipak ostaviti sve varijable, osim varijabli koje nemaju numeričku vrijednost, dakle *track_uri*, *artist* i *track_name*, te sam još izbacio i varijable *liveness* i *duration_min*. Ove dvije varijable koje imaju numeričke vrijednosti izbacio sam iz razloga što zapravo ne želim da mi prisutnost publike kod varijable

liveness ima utjecaja na preporuku, te da mi trajanje pjesme također nema utjecaja, jer smatram irelevantnim za moj sustav preporuka. Neke druge kombinacije s najboljim i najgorim prediktorima mogu se naći u nekim budućim istraživanjima, jer bi sve moguće kombinacije prediktora, podataka, metrika i algoritama pretraživanja izašli iz okvira ovog rada, tj. znatno ga produžili.

2.1.8.1 k-NN

Kod algoritma k-najbližih susjeda korišteno je šesnaest testova s različitim kombinacijama metrika, algoritama pretraživanja i kombinacije u kojem k iznosi 10 i u kojem k iznosi duljinu cijelog skupa podataka.

Korištene su kombinacije Euklidske udaljenosti s algoritmima ball_tree, kd_tree i brute, zatim Manhattan udaljenost s istim algoritmima pretraživanja kao i kod Euklidske udaljenosti, te je još korištena i kosinusna udaljenost uz samo algoritam pretraživanja brute. Napravljene su dvije verzije testiranja metrika s algoritmima. Prva verzija su Euklidska i Manhattan metrike s algoritmima pretrage kd tree i ball tree, zatim su napravljene verzije s Euklidskom, Manhattan i kosinusnom metrikom i algoritmom pretraživanja brute. Kao što je prethodno navedeno, kosinusna udaljenost ne podržava kd_tree i ball_tree algoritme. Za provjeriti koji algoritmi pretraživanja rade uz koju metriku može se koristiti sljedeći kod: `print(sorted(VALID_METRICS['ball_tree']))`, rezultat ovog koda je sortirana lista svih metrika koje podržavaju ball_tree algoritam pretraživanja. Iz rezultata je vidljivo da ne podržava kosinusnu udaljenost. Za izradu preporuka s različitm testovima korištena je pjesma *U2 – I Still Haven't Found What I'm Looking For*. Osim svega toga, napravljeno je testiranje i s različito obrađenim podacima. Provedene su tri različite vrste obrade podataka, uključujući min-max skaliranje, dimenzioniranje pomoću PCA algoritma i dimenzioniranje pomoću t-SNE algoritma. Prvo uvježbavanje provedeno je uz min-max skalirane podatke, nakon čega je primijenjeno min-max skaliranje u kombinaciji s PCA dimenzioniranjem, sa dvije i tri dimenzije. Najzad je korišteno min-max skaliranje uz algoritam t-SNE s različitim stopama učenja. Stope učenja kod algoritma t-SNE iznosile su 25, 50, 100, 200 i 400.

S obzirom da je napravljeno jako puno testiranja što se tiče skaliranja i dimenzioniranja podataka, ovdje će se prikazati koji tip pripreme podataka ima najvišu „isplativost“, te s kojim parametrima su podaci pripremani, kao i koji parametri su korišteni kod metode najbližih susjeda.

Za početak, svi rezultati testiranja su skupljeni u jednu tablicu. Prvo testiranje je obavljeno kako bi se utvrdilo koja kombinacija skaliranja, dimenzioniranja, broja k, metrike (udaljenosti) i algoritma pretraživanja zahtijeva najmanje vremena izvođenja. Najbolji tj. najbrži rezultat pokazao se je model koji je kombinirao min-max skaliranje uz PCA algoritam dimenzioniranja s tri dimenzije i vrijednost k 10. Metrika koja je korištena je Euklidska uz algoritam pretraživanja kd_tree, a vrijeme potrebno za izračun preporuka iznosilo je 1,04 milisekundi. Pjesma za koju se tražila preporuka je pjesma od *U2 – I Still Haven't Found What I'm Looking For*. Prvih pet rezultata mogu se vidjeti u tablici 3.

Tablica 3 Najbolja vremena za različito dimenzioniranje, k, metriku i algoritam pretraživanja

Redni broj	**Skaliranje**	**Dimenzioniranje, PCA (dim) ili t-SNE (stopa učenja)**	**k**	**Metrika**	**Algoritam**	**Vrijeme (ms)**
1.	min-max	PCA, 3	10	Euklidska	kd_tree	1,04
2.	min-max	t-SNE, 400	10	Euklidska	kd_tree	1,08
3.	min-max	t-SNE, 400	10	kosinusna	brute	1,09
4.	min-max	PCA, 3	10	Manhattan	ball_tree	1,09
5.	min-max	t-SNE, 50	10	Manhattan	brute	1,09

Najlošiji rezultat koji se je pokazao bio je model s min-max skaliranjem bez dimenzoniranja i uz k koji je iznosio cijeli skup podataka, te uz kosinusnu metriku i brute algoritam, a vrijeme izračuna preporuka iznosilo je 21,61 milisekundi, što je 20,77 puta gore od prvog rezultata. Vidljivo je kako k koji iznosi cijelu duljinu skupa podataka može biti dovoljno brz da odradi preporuku kao i k od 10, a riječ je o sedmom po redu rezultatu od 1,12 milisekundi uz t-SNE 25, kosinusnu metriku i brute algoritam pretraživanja. Naravno, ipak je riječ o manjem skupu podataka.

S obzirom da nije moguće uspoređivati MSE, RMSE i MAE kod Euklidske i Manhattan udaljenosti s jedne strane i kosinusne udaljenosti s druge strane, napravljene su

dvije nove tablice. U jednoj tablici se nalaze Euklidska i Manhattan metrike, dok je u drugoj tablici kosinusna metrika.

Najbolji rezultat što se tiče MSE, RMSE i MAE ostvario je model min-max uz PCA s dvije dimenzije, k od 10 i uz Euklidsku metriku, srednja udaljenost iznosi 0,0486, MSE iznosi 0,0029, RMSE 0,0545 i MAE iznosi 0,0236. Što se tiče algoritama pretraživanja, oni nemaju utjecaja na ova tri testa, dakle jedino imaju utjecaja na vrijeme pretraživanja. U tablici 4 mogu se vidjeti prvih pet rezultata, podaci, nije navedeno u tablici, skalirani su uz min-max skaliranje.

Tablica 4 MSE, RMSE i MAE mjere kod Euklidske i Manhattan udaljenosti

Redni broj	**Dimenzioniranje, PCA (dim) ili t-SNE (stopa učenja)**	**k**	**Metrika**	**Srednja udaljenost**	**MSE**	**RMSE**	**MAE**
1.	PCA, 2	10	Euklidska	0,0486	0,0029	0,0545	0,0236
2.	PCA, 2	10	Manhattan	0,0607	0,0051	0,0717	0,0304
3.	PCA, 3	10	Euklidska	0,0988	0,0155	0,1248	0,0806
4.	PCA, 2	cijeli skup	Euklidska	0,7813	0,0236	0,1538	0,1114
5.	PCA, 3	cijeli skup	Euklidska	0,8708	0,0291	0,1706	0,1252

Najveće rezultate MSE, RMSE i MAE pokazali su min-max skalirani podatci sa t-SNE dimenzioniranjem i stopom učenja od 400, uz Manhattan metriku. MSE u ovom slučaju iznosi 43,686, RMSE 6,609 i MAE 4,306.

Kod kosinusne udaljenosti najniži MSE, RMSE i MAE pokazao je min-max uz t-SNE s stopom učenja 400 i brute algoritmom pretraživanja uz srednju udaljenost od 0,0001, MSE 0,000005, RMSE 0,00215 i MAE 0,00049. Prvih pet rezultata mogu se vidjeti u tablici 5.

Tablica 5 MSE, RMSE i MAE mjere kod kosinusne udaljenosti

Redni broj	Dimenzioniranje, PCA (dim) ili t-SNE (learning rate)	k	Srednja udaljenost	MSE	RMSE	MAE
1.	t-SNE, 400	10	0,000126	0,000005	0,002150	0,000490
2.	PCA, 2	10	0,000874	0,000034	0,005823	0,001839
3.	t-SNE, 200	10	0,000476	0,000335	0,018305	0,002497
4.	t-SNE, 100	10	0,000071	0,000345	0,018576	0,003039
5.	t-SNE, 25	10	0,000254	0,000647	0,025429	0,003737

Što se tiče prosječnih rezultata ustanovljenih za agregirane podatke, dakle za sve retke u kojima se nalazi pojedina metrika, ustanovljeno je što se tiče prosječnog vremena koje je uzelo stvaranje preporuka, vrijeme potrebno za izračune najmanje je bilo potrebno Manhattan udaljenosti, tablica 6. Što se tiče prosječne pozicije na prvom mjestu nalazi se Euklidska udaljenost.

Tablica 6 Najbolje prosječno vrijeme po metrikama

Redni broj	Metrika	Prosječna pozicija	Ukupna pozicija	Prosječno vrijeme (ms)	Ukupno vrijeme (ms)	Broj testova
1.	Manhattan	55,83	2680	3,161	151,72	48
2.	kosinusna	58,5	936	3,466	55,45	16
3.	Euklidska	54,16	2600	3,846	184,59	48

Prosječna pozicija kumulativan je zbroj indeksa na kojem se nalazi ista metrika, podijeljen s ukupnim brojem u kojem se nalazi ta metrika. Što se tiče samih preporuka, odabrane su preporuke od prvog najboljeg rezultata, a to je min-max s PCA algoritmom

dimenzioniranja s tri dimenzije i k = 10, Euklidskom udaljenošću i kd_tree algoritmom pretraživanja koji je obavio preporuku u 1,04 milisekundi. Preporuke za pjesmu od *U2 – I Still Haven't Found What I'm Looking For*, mogu se vidjeti u tablici 7.

Tablica 7 Preporuka za pjesmu *U2 - I Still Haven't Found What I'm Looking For*

Redni broj	**Izvođač**	**Pjesma**	**Žanr**	**Udaljenost**
1.	Jon Secada	Just Another Day	rock	0,06850
2.	Pink Floyd	Take It Back	rock	0,06872
3.	The Clash	London Calling	rock	0,07298
4.	Donna Summer	I Feel Love	disco	0,07735
5.	U2	Who's Gonna Ride Your Wild Horses	rock	0,09431

2.1.8.2 K-Means

Drugi dio istraživanja posvećen je algoritmima nenadziranog učenja, iako je ovdje i algoritam k-najbližih susjeda obavio posao nenadziranog učenja.

No, međutim ipak je velika razlika u određivanu preporuka, kao i u određivanju mjera koliko dobro radi koji algoritam ili kako se algoritam uči na određenom tipu obrade podataka.

Drugi dio započeo je poučavanjem s K-Means algoritmom. Kao i kod algoritma k-najbližih susjeda tako se i kod K-Means algoritma poučavalo model s različito obrađenim podatcima. Za razliku od k-najbližih susjeda smanjen je obujam različitih podataka tj. korištenih parametara. Što se tiče broja k, kojeg sami određujemo u algoritmu k-najbližih susjeda, ovdje se najoptimalniji K određuje uz pomoć metode lakta. Uostalom k kod algoritma k-najbližih susjeda i kod algoritma K-Means dvije su različite stvari. Kod algoritma k-NN to je broj najbližih susjeda u traženoj preporuci, dok je kod K-Means broj grupa koje će algoritam samostalno odrediti. Optimalni K se utvrđuje iteracijom od 1 do 25,

a iteracija do broja 24 određena je proizvoljno. Optimalni K utvrđuje se uz pomoć metode `KElbowVisualizer` koja se nalazi u biblioteci `yellowbrick.cluster`. Optimalni broj K, kojeg je `KElbowVisualizer` odredio je 8, uz *distortion* rezultat od 103,127.

Podaci se prvo skaliraju uz pomoć MinMaxScalera, nakon čega im se smanjuje broj dimenzija na samo dvije dimenzije uz pomoć algoritma t-SNE. Polaznu točku za izradu dijagrama raspršenosti (engl. *scatterplot*) čine podaci dimenzionirani uz pomoć t-SNE algoritma i to uz stopu učenja od 100, stopa učenja od 100 nije uvjetovana. Dijagram raspršenosti služi kako bi vidjeli kako algoritam K-Means radi grupiranje (engl. *clustering*) i da bi procijenili na oko, koliko dobro radi razdvajanje grupa.

Korišteni su podaci koji su prvo min-mix skalirani, zatim dimenzionirani uz PCA na dvije i tri dimenzije i dimenzionirani uz pomoć t-SNE algoritma uz stope učenja 25, 50, 100, 200 i 400.

Prva metrika, kako je prethodno navedeno dijagram raspršenosti, na kojem se može vidjeti koliko dobro su grupe odvojene, sljedeće metrike su koeficijent siluete te Calinski-Harabasz indeks. Najbolji rezultat dobiven je za min-max podatke dimenzionirane uz PCA algoritam s dvije dimenzije uz koeficijent siluete od 0,496 i Calinski-Harabsz koeficijent 774,666. Prva tri najbolja rezultata mogu se vidjeti u tablici 8.

Tablica 8 Najbolji rezultati koeficijenta siluete i Calinski-Harabasz koeficijenta

Redni broj	**Dimenzioniranje, PCA (dim) ili t-SNE (learning rate)**	**Koeficijent siluete**	**Calinski-Harabasz koeficijent**
1.	PCA, 2	0,496	774,666
2.	t-SNE, 400	0,434	932,094
3.	t-SNE, 25	0,421	869,083

Što se tiče samih preporuka pjesama, ono radi na drukčiji način nego kod algoritma k-najbližih susjeda. Dakle, algoritam ne određuje udaljenosti od pjesme za koju se traži preporuka, nego se sve pjesme podijele u grupe. Zatim se pronađe ona pjesma za koju se

traži preporuka, tj. nađe se broj njene grupe, te na taj način se pronađu pjesme koje se nalaze u istoj grupi, tj. k prvih pjesama, koliko sami odredimo.

Prijedlog preporuka za najbolji rezultat koeficijenta siluete za pjesmu od *U2 - I Still Haven't Found What I'm Looking For* može se vidjeti u tablici 9.

Tablica 9 Prijedlog preporuka za najbolji rezultat koeficijenta siluete za pjesmu *U2 - I Still Haven't Found What I'm Looking For*

Redni broj	**Izvođač**	**Pjesma**	**Žanr**
1.	Talking Heads	This Must Be The Place	rock
2.	Steve Winwood	Valerie	rock
3.	Lionel Richie	You Are	disco
4.	Baltimora	Tarzan Boy	disco
5.	Wham!	Wake Me Up Before You Go-Go	pop

2.1.8.3 DBSCAN

Još jedan algoritam koji je korišten za sustav preporuka je algoritam DBSCAN, DBSCAN je algoritam nenadziranog učenja, koji radi na sličan način kao i K-Means algoritam. Kod algoritma DBSCAN postoje parametri `eps` i `min_samples`, te se uz pomoć iteracije mogu odrediti najbolji parametri.

Napravljen je test prvo samo s min-max skaliranim podacima, a najbolji rezultat pokazao je parametar `eps` od 1,0 uz parametar `min_samples` od 9 i 10, koji je dao svega 3 grupe uz rezultat siluete 0,256487 i Calinski-Harabasz rezultat 48,133. Prvih pet najboljih rezultata mogu se vidjeti u tablici 10.

Tablica 10 Rezultati kod DBSCAN algoritma za min-max podatke

Redni broj	***eps***	***min_samples***	**Broj grupa**	**Koeficijent siluete**	**Calinski-Harabasz koeficijent**
1.	1,0	9 i 10	3	0,256	48,133
2.	1,0	7	4	0,249	35,596
3.	0,9	8	3	0,245	44,194
4.	1,0	8	4	0,243	35,076
5.	0,9	9	3	0,242	43,108

Što se tiče min-max podataka koji su još dodatno dimenzionirani uz pomoć t-SNE algoritma, rezultati su se pokazali čak nešto lošiji u odnosu na podatke koji nisu dimenzionirani. Jedino se stopa učenja od 100 s parametrima `eps` od 0,9 i `min_samples` od 2 pokazala nešto bolja, no međutim stvoreno je čak 49 grupa uz koeficijent siluete 0,26. Prvih pet najboljih rezultata za svaku stopu učenja može se vidjeti u tablici 11. Kod podataka koji su dimenzionirani t-SNE algoritmom, DBSCAN algoritam stvorio je puno veći broj grupa uz manji broj `min_samples`. To nije dobro za skup podataka koji ima manji broj primjera, no moglo bi biti dobro za skup podataka s većim brojem primjera iako ne znamo što bi se zbilja dogodilo. Kod skupa podataka s većim brojem primjera bilo bi dobro kad bi DBSCAN zadržao broj grupa kao i kod skupa podataka s manjim brojem primjera, s kakvim se eksperimentira u ovom radu. No, međutim ako bi stvorio isti omjer grupa (onoliko puta veće za koliko je veći skup podataka) u odnosu na manji skup podataka, onda DBSCAN algoritam ne bi imao previše smisla. Ovo može biti još jedan cilj budućih istraživanja.

Tablica 11 Rezultati za t-SNE dimenzionirane podatke

Redni broj	***learning_rate***	***eps***	***min_samples***	**Broj grupa**	**Koeficijent siluete**	**Calinski-Harabasz koeficijent**
1.	25	0,9	2	54	0,231	15,405
2.	50	0,9	2	50	0,254	16,857
3.	100	0,9	2	49	0,26	24,575
4.	200	1,0	2	34	0,219	33,777
5.	400	0,9	2	50	0,185	18,986

Može se vidjeti kako t-SNE dimenzioniranje ima vrlo malo utjecaja na poboljšanje rezultata u odnosu na rezultate koji su samo skalirani uz pomoć min-max skalera.

Pri korištenju podataka dimenzioniranih PCA algoritmom na dvije i tri dimenzije, ostvareni su nešto bolji rezultati u usporedbi s t-SNE algoritmom dimenzioniranja. Najbolji rezultat kod PCA algoritma dimenzioniranja za dvije dimenzije jest koeficijent siluete od 0,607, no s tim rezultatom siluete algoritam je stvorio najviše dvije grupe pjesama, što se ne može smatrati dobrim rezultatom u kontekstu sustava preporuka. U prethodnoj rečenici se spominje „najviše dvije grupe“ iz razloga što je koeficijent siluete od 0,607 isti za cijeli niz parametra `eps` i isto tako za parametar `min_samples`, korišteni parametri i rezultati mogu se vidjeti u tablici 12. Najveći broj grupa koje je algoritam kreirao je 4, uz koeficijent siluete od 0,557, i to se može smatrati nešto boljim rezultatom u kontekstu sustava preporuka i stvorenih grupa. Prva tri najbolja rezultata u kontekstu koeficijenta siluete i Calinski-Harabasz koeficijenta mogu se vidjeti u tablici 12.

Tablica 12 Rezultati za PCA s dvije dimenzije

Redni broj	***eps***	***min_samples***	**Broj grupa**	**Koeficijent siluete**	**Calinski-Harabasz koeficijent**
1.	0,4 – 0,8	2 - 10	2	0,607	356,621
2.	0,2 i 0,3	2 - 10	3	0,572	196,326
3.	0,1	7	4	0,557	225,114

Kada je riječ o PCA dimenzioniranju s tri dimenzije, najbolji rezultat siluete je 0,483, ali isto kao i kod PCA algoritma s dvije dimenzije, DBSCAN je također stvorio samo dvije grupe. Algoritam DBSCAN je također kreirao cijeli niz istog rezultata siluete, čak njih 36 za različiti parametar `eps` i parametar `min_samples`, što se može vidjeti u tablici 13. Prva tri najbolja rezultata u odnosu na rezultat siluete mogu se vidjeti u tablici 13.

Tablica 13 Rezultati za PCA s tri dimenzije

Redni broj	***eps***	***min_samples***	**Broj grupa**	**Koeficijent siluete**	**Calinski-Harabasz koeficijent**
1.	0,5 – 0,8	2 - 10	2	0.483	207,677
2.	0,3	6	3	0,467	111,152
3.	0,3	3, 4 i 5	3	0.465	111,134

Glede rezultata preporuke, i s obzirom da je DBSCAN algoritam grupiranja, rezultati preporuke su slični kao i kod algoritma K-Means, dakle nije kao kod algoritma k-NN, nego se uzima neki uzorak iz skupa kojem pripada i tražena pjesma. U tablici 14 mogu se vidjeti rezultati DBSCAN algoritma pri PCA dimenzioniranju s dvije dimenzije i 4 grupe, rezultat 3 iz tablice 12 za pjesmu od *U2 – I Still Haven't Found What I'm Looking For*.

Tablica 14 Prijedlog preporuka za najbolji rezultat koeficijenta siluete za pjesmu *U2 - I Still Haven't Found What I'm Looking For*

Redni broj	**Izvođač**	**Pjesma**	**Žanr**
1.	Stardust	Music Sounds Better With You	edm
2.	Robin S	Show Me Love	edm
3.	Junior Jack	Stupidisco	edm
4.	Avicii	My Feelings For You	edm
5.	Dario G	Sunchyme	edm

Vidljivo je kako DBSCAN nije dobro odradio posao preporuke, jer osim dobrog rezultata siluete važan je i broj kreiranih grupa. Vidi se da je DBSCAN predložio pjesme koje nemaju veze sa žanrom pjesme za koju je tražena preporuka.

3. Zaključak

U ovom radu predstavljen je način rada sustava preporuka, točnije samo jedan uži skup sustava preporuka koji koristi filtriranje prema sadržaju. Iako sustavi preporuke krajnjim korisnicima izgledaju jednostavni, bar prema onom što vidimo na nekoj *web* trgovini, mogli bi ipak na koncu svega reći kako u ovom području brojni izazovi. Kao što je rečeno negdje na početku, sustavi preporuka iznimno su važni, jer o njima ovisi velik dio zarade kao što je to kod Amazona.

U radu je prikazana povijest sustava za preporuke, kao i nužnost za razvojem ovakvih sustava, a već je sedamdesetih godina dvadesetog stoljeća razvijena osnovna inačica sustava za preporuke. Također je prikazana osnovna podjela sustava za preporuke, prema načinu filtriranja: suradničko (kolaborativno) filtriranje, filtriranje prema sadržaju te hibridno filtriranje.

U radu je pokazan način funkcioniranja sustava za preporuke prema sadržaju uz pomoć tri različita algoritma k-NN, K-Means i DBSCAN. Posvetila se i velika pažnja obradi podataka, jer podatci uz algoritme čine samu bit sustava za preporuke na primjeru glazbe. Korišten je min-max način skaliranja, te dva različita načina dimenzioniranja podataka PCA s dvije i tri dimenzije, te t-SNE s pet različitih stopa učenja. Prikazani su rezultati dobiveni uz različite udaljenosti, te uz različite algoritme pretraživanja. S obzirom da k-NN algoritam radi drukčije nego algoritmi K-Means i DBSCAN, nije moguća točna usporedba testova ovih algoritama. No, algoritam k-NN sigurno može bolje odrediti koje pjesme odgovaraju jedna drugoj, tj. na kojoj udaljenosti (ili blizini, ovisno kako gledamo) se nalazi koja pjesma. Dok, algoritmi K-Means i DBSCAN stvaraju grupe i u grupe smještaju pjesme, što naravno da i ovo ima svoju svrhu i može se iskoristiti. Vidljivo je također i da su rezultati kod K-Means definitivno bolji što se tiče samih preporuka, a da DBSCAN nije dobro odradio traženi posao. Vidjeli smo i ako se upotrijebe filteri, za pjesme koju su dobivene preporukom, može se napraviti i manipulacija preporukama. Iako je sustav preporuka zanimljiva tema, bilo je ipak izazovno baviti se njim. Kod ovog diplomskog rada kreirano je apsolutno sve od početka, od samog vlastitog skupa, preko analize, pripreme podataka, kombiniranja parametara, izrade modela, izrade preporuka i na kraju evaluacije.

Za prikupljanje podataka koji su se koristili u ovom diplomskom radu korišten je Spotify API. Preduvjeti za struganje podataka sa poveznice na neku od Spotify listi pjesama

je vlastiti Spotify korisnički račun. Prikupljanje se obavljalo uz pomoć spotipy biblioteke za Python. Prikupljanje podataka provedeno je s jednog vlastitog popisa pjesama, kreiranog isključivo u svrhu ovog diplomskog rada. Vlastiti skup podataka omogućio mi je bolji uvid u funkcioniranje značajki, kao i moguće razloge preporuke pjesama za traženu pjesmu. Isključivanje pojedinih značajki i praćenje rezultata, te različito skaliranje i dimenzioniranje podataka, a nakon toga kombiniranje različitih vrijednosti korištenih parametara dali su neke očekivane ali i neočekivane rezultate. Uglavnom, bilo je zanimljivo pratiti cijeli proces i gledati rezultate, i na kraju dobiti sasvim funkcionalni sustav preporuke prema sadržaju na područje glazbe, koji se sada može iskoristiti bilo gdje.

Ono zbog čega sam se uopće odlučio za skup podataka koji se sastoji od pjesama je moja sklonost ka glazbi. Može se reći da svatko voli glazbu, i svako ima svoj određeni „ukus“ za glazbu. Mene je oduvijek zanimala glazba, i uvijek sam sanjao kako bih puštao glazbu za druge, a to sam i uspio ostvariti puštanjem glazbe na nekim događanjima. Želja mi se ostvarila i radom na lokalnoj radio stanici u ulozi jednog čovjeka koji je obavljao puno funkcija tzv. *one-man show*, a obuhvaćalo je voditelja programa, urednika emisija, tehničara i *disk jockey-a* sve u jednom. Svime ovime sam uspio skupiti iskustvo da bih se ekspertno mogao pozabaviti skupom podataka u ovom radu.

Literatura

[1] Z. Dong, Z. Wang, J. Xu, R. Tang, i J. Wen, „A brief history of recommender systems“, *ArXiv Prepr. ArXiv220901860*, 2022.

[2] „Preporuka (značenje i definicija)“. Pristupljeno: 12. ožujak 2024. [Na internetu]. Dostupno na: https://jezikoslovac.com/word/clo4

[3] E. Rich, „User modeling via stereotypes“, *Cogn. Sci.*, sv. 3, izd. 4, str. 329–354, 1979.

[4] V. Gupta i S. R. Pandey, „Recommender systems for digital libraries: a review of concepts and concerns“, *Libr. Philos. Pract.*, sv. 2417, 2019.

[5] „Recommender system“. Pristupljeno: 12. ožujak 2024. [Na internetu]. Dostupno na: https://en.wikipedia.org/wiki/Recommender_system

[6] X. Su i T. M. Khoshgoftaar, „A survey of collaborative filtering techniques“, *Adv. Artif. Intell.*, sv. 2009, 2009

[7] M. Hossin i M. N. Sulaiman, „A review on evaluation metrics for data classification evaluations“, *Int. J. Data Min. Knowl. Manag. Process*, sv. 5, izd. 2, str. 1, 2015.

[8] G. S. Handelman *i ostali*, „Peering into the black box of artificial intelligence: evaluation metrics of machine learning methods“, *Am. J. Roentgenol.*, sv. 212, izd. 1, str. 38–43, 2019.

[9] G. Linden, B. Smith, i J. York, „Amazon. com recommendations: Item-to-item collaborative filtering“, *IEEE Internet Comput.*, sv. 7, izd. 1, str. 76–80, 2003.

[10] „How retailers can keep up with consumers“. Pristupljeno: 17. ožujak 2024. [Na internetu]. Dostupno na: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

[11] J. Bennett, S. Lanning, i others, „The netflix prize“, u *Proceedings of KDD cup and workshop*, New York, 2007, str. 35.

[12] „Data Privacy — The Netflix Prize competition“. Pristupljeno: 12. prosinac 2023. [Na internetu]. Dostupno na: https://medium.com/@EmiLabsTech/data-privacy-the-netflix-prize-competition-84330d01cc34

[13] „Overview of Recommender Systems And Implementations“. Pristupljeno: 05. travanj 2024. [Na internetu]. Dostupno na: https://medium.com/geekculture/overview-of-recommender-systems-and-implementations-cae13088369

[14] X. Ying, „An overview of overfitting and its solutions“, u *Journal of physics: Conference series*, IOP Publishing, 2019, str. 022022.

[15] B. Takkouche i G. Norman, „PRISMA statement“, *Epidemiology*, sv. 22, izd. 1, str. 128, 2011.

[16] A. Fuad, S. Bayoumi, i H. Al-Yahya, „A Recommender System for Mobile Applications of Google Play Store“, *Int. J. Adv. Comput. Sci. Appl.*, sv. 11, izd. 9, 2020

[17] M. Bakhshizadeh, A. Moeini, M. Latifi, i M. T. Mahmoudi, „Automated mood based music playlist generation by clustering the audio features“, u *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*, IEEE, 2019, str. 231–237.

[18] I. Bansal, K. Gupta, B. Sehgal, V. Sharma, i others, „A Social Network Approach for Automated Generation of YouTube Playlists“, u *2022 IEEE International Conference on Data Science and Information System (ICDSIS)*, IEEE, 2022, str. 1–6.

[19] V. Danilova i A. Ponomarev, „Hybrid recommender systems: The review of state-of-the-art research and applications“, u *Proceedings of the 20th Conference of FRUCT Association*, 2017.

[20] „Supervised vs. Unsupervised Learning: What’s the Difference?“ Pristupljeno: 12. veljača 2024. [Na internetu]. Dostupno na: https://www.ibm.com/blog/supervised-vs-unsupervised-learning/

[21] N. E. M. Isa, A. Amir, M. Z. Ilyas, i M. S. Razalli, „The performance analysis of K-nearest neighbors (K-NN) algorithm for motor imagery classification based on EEG signal“, u *MATEC web of conferences*, EDP Sciences, 2017, str. 01024.

[22] „Most Popular Distance Metrics Used in KNN and When to Use Them“. Pristupljeno: 15. veljača 2024. [Na internetu]. Dostupno na: https://www.kdnuggets.com/2020/11/most-popular-distance-metrics-knn.html

[23] „Outliers or Key Profiles? Understanding Distance Measures for Authorship Attribution“. Pristupljeno: 13. prosinac 2023. [Na internetu]. Dostupno na: https://dh2016.adho.org/abstracts/253

[24] „Machine Learning :: Cosine Similarity for Vector Space Models (Part III)“. Pristupljeno: 14. prosinac 2023. [Na internetu]. Dostupno na: https://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/

[25] P. Dayan, M. Sahani, i G. Deback, „Unsupervised learning“, *MIT Encycl. Cogn. Sci.*, str. 857–859, 1999.

[26] M. Ahmed, R. Seraj, i S. M. S. Islam, „The k-means algorithm: A comprehensive survey and performance evaluation“, *Electronics*, sv. 9, izd. 8, str. 1295, 2020.

[27] „Principal Component Analysis (PCA)“. Pristupljeno: 13. veljača 2024. [Na internetu]. Dostupno na: https://www.geeksforgeeks.org/principal-component-analysis-pca/

[28] „Introduction to t-SNE“. Pristupljeno: 13. veljača 2024. [Na internetu]. Dostupno na: https://www.datacamp.com/tutorial/introduction-t-sne

[29] „Top Performance Metrics in Machine Learning: A Comprehensive Guide“. Pristupljeno: 16. ožujak 2024. [Na internetu]. Dostupno na: https://www.v7labs.com/blog/performance-metrics-in-machine-learning

[30] J.-O. Palacio-Niño i F. Berzal, „Evaluation metrics for unsupervised learning algorithms“, *ArXiv Prepr. ArXiv190505667*, 2019.

[31] D. Perera, M. Rajaratne, S. Arunathilake, K. Karunanayaka, i B. Liyanage, „A critical analysis of music recommendation systems and new perspectives“, u *Human Interaction, Emerging Technologies and Future Applications II: Proceedings of the 2nd International Conference on Human Interaction and Emerging Technologies: Future Applications (IHIET–AI 2020), April 23-25, 2020, Lausanne, Switzerland*, Springer, 2020, str. 82–87.

[32] R. Genuer, J.-M. Poggi, i C. Tuleau-Malot, „Variable selection using random forests“, *Pattern Recognit. Lett.*, sv. 31, izd. 14, str. 2225–2236, 2010.

[33] M. M. Ahsan, M. P. Mahmud, P. K. Saha, K. D. Gupta, i Z. Siddique, „Effect of data scaling methods on machine learning algorithms and model performance“, *Technologies*, sv. 9, izd. 3, str. 52, 2021.

[34] „Elbow Method for optimal value of k in KMeans“. Pristupljeno: 04. travanj 2024. [Na internetu]. Dostupno na: https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/

Popis slika i tablica

Slika 1 Evolucija sustava za preporuke [13] 3

Slika 2 Usporedba udaljenosti prikazane u prostoru [23] 10

Slika 3 Primjeri kosinusne sličnosti [24] 10

Slika 4 Distribucije podataka 24

Slika 5 Wordcloud naslova pjesama 25

Slika 6 Wordcloud izvođača 25

Slika 7 Matrica korelacija bez zavisne varijable 26

Slika 8 Radarski dijagram 27

Slika 9 Svojstva *valence*, *energy*, *danceability* i *acousticness* za žanrove *pop*, *metal* i *rock* 28

Slika 10 Najvažniji prediktori (biblioteka *yellowbrick.target*) 29

Tablica 1 Primjeri korištenja ključnih riječi i prekidača 5

Tablica 2 Opis atributa skupa podataka [17] 20

Tablica 3 Najbolja vremena za različito dimenzioniranje, k, metriku i algoritam pretraživanja 40

Tablica 4 MSE, RMSE i MAE mjere kod Euklidske i Manhattan udaljenosti 41

Tablica 5 MSE, RMSE i MAE mjere kod kosinusne udaljenosti 42

Tablica 6 Najbolje prosječno vrijeme po metrikama 42

Tablica 7 Preporuka za pjesmu *U2 - I Still Haven't Found What I'm Looking For* 43

Tablica 8 Najbolji rezultati koeficijenta siluete i Calinski-Harabasz koeficijenta 44

Tablica 9 Prijedlog preporuka za najbolji rezultat koeficijenta siluete za pjesmu *U2 - I Still Haven't Found What I'm Looking For* 45

Tablica 10 Rezultati kod DBSCAN algoritma za min-max podatke 46

Tablica 11 Rezultati za t-SNE dimenzionirane podatke 47

Tablica 12 Rezultati za PCA s dvije dimenzije 48

Tablica 13 Rezultati za PCA s tri dimenzije..... 48

Tablica 14 Prijedlog preporuka za najbolji rezultat koeficijenta siluete za pjesmu *U2 - I Still Haven't Found What I'm Looking For* 49