



Klasifikacija i klasterovanje horala J.S. Bahá

02.06.2022

—
Student: Mihailo Simić

Profesor: prof. dr Nenad Mitić

Asistent: Marija Erić
Matematički fakultet, Univerzitet u Beogradu

Uvod

Bahovi horali se odnose na kolekciju muzičkih kompozicija koje je napisao Johan Sebastijan Bah. Ovi horali su četvoroglasne harmonizacije luteranskih himni i smatraju se značajnim u istoriji zapadne muzike. Bah je komponovao stotine horala koji se široko proučavaju zbog njihove muzičke strukture i bogatstva harmonije.

U kontekstu mašinskog učenja, skup podataka poznat kao "Bach Chorales" se često koristi za različite zadatke povezane sa muzikom, kao što su generisanje muzike, analiza harmonije i transkripcija muzike. Ovaj skup podataka sadrži muzičke partiture Bahovih horala u simboličkom obliku koji se može lako obraditi mašinskim učenjem.

Uobičajeno, skup podataka "Bach Chorales" uključuje sledeće informacije za svaki horal:

- Melodijske linije ili glasovi, kao što su sopran, alt, tenor i bas.
- Note ili visine tonova koje svaki glas svira u različitim vremenskim koracima.
- Odgovarajući muzički simboli, dužine i ritmički obrasci.

Istraživači i programeri koriste skup podataka "Bach Chorales" za obuku modela mašinskog učenja, kao što su rekurentne neuronske mreže (RNN) ili generativni modeli, kako bi naučili obrusce, strukturu i stilove koji se javljaju u Bahovoj muzici. Ovi modeli se zatim mogu koristiti za generisanje novih kompozicija u Bahovom stilu, analizu harmonijskih progresija ili pomoći u komponovanju muzike.

Dostupnost skupa podataka "Bach Chorales" doprinela je napretku u oblasti računarske muzikologije, omogućavajući istraživačima da istraže osnovne principe Bahove muzike i primene tehnike mašinskog učenja za analizu i kreiranje muzike u Bahovom stilu.

Ciljevi

- I. Analiza skupa podataka i opšte karakteristike.
- II. Modeli klasifikacije horala i harmonija.
- III. Modeli za klasterovanje horala u grupacije po harmonijama.

Analiza skupa podataka “Bach Chorales”

Uvod

Skup podataka sadrži informacije o visini tona iz MIDI (eng. Musical Instrument Digital Interface) podataka Bahovih horala.

Atributi skupa podataka su:

1. Choral ID: Identifikator određenog horala (ukupno 62 horala).
2. Event number: ID određenog događaja ili akorda unutar horala.
3. Pojava tona (3-14): Svaka od 12 tonova posebno navedena, označava da li se pojavljuje u određenom akordu (da ili ne).
4. Klasa najnižeg tona (Bass): Klasa tona najnižeg glasa u datom akordu.
5. Važnost akorda (Meter): Mera važnosti akorda, predstavljena na skali od 1 do 5.
6. Naziv akorda (Chord label): Naziv akorda koji se sastoji od klase tona i informacije da li je major (Dur) ili minor (Mol).

Svaka instanca skupa podataka je jedan zabeleženi “akord” iz Bahovih horala.

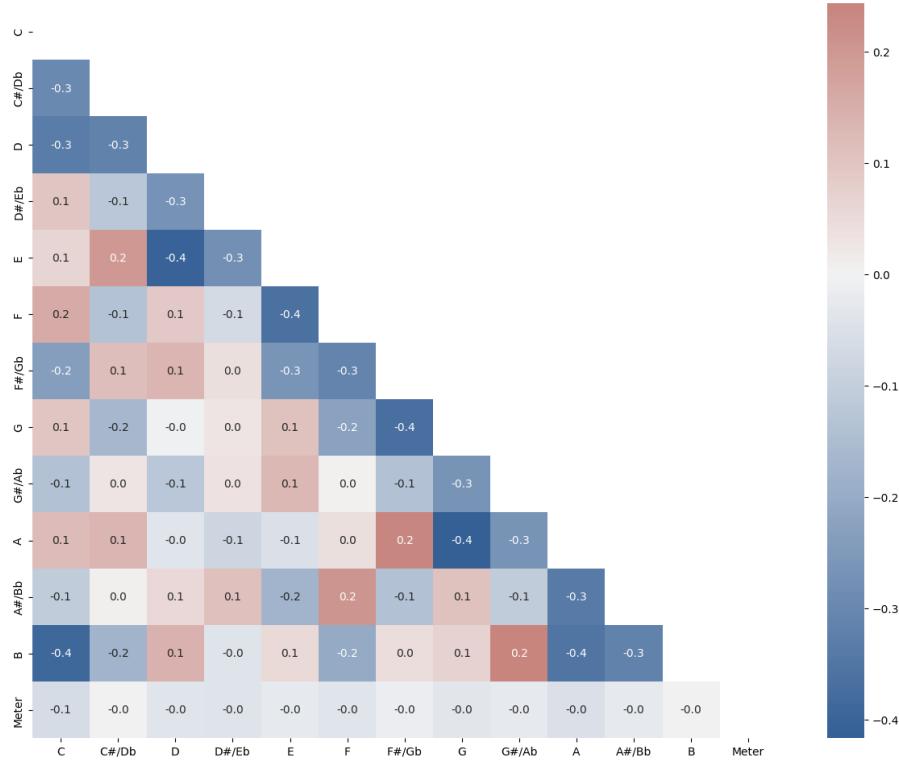
Atribut **Choral ID** je samo identifikator pa nećemo preterano obraćati pažnju na njega u daljoj analizi skupa podataka, kao ni u modelima klasifikacije i klasterovanja.

Atributi 3-14 su binarni atributi, i vrednosti mogu biti “YES” ili “NO”.

Atribut **Bass** je diskretni imenski atribut, dok je atribut **Meter** diskretni redni atribut.

Atribut **Chord label** je imenski atribut i zapravo će na njemu biti fokus tokom klasifikacije.

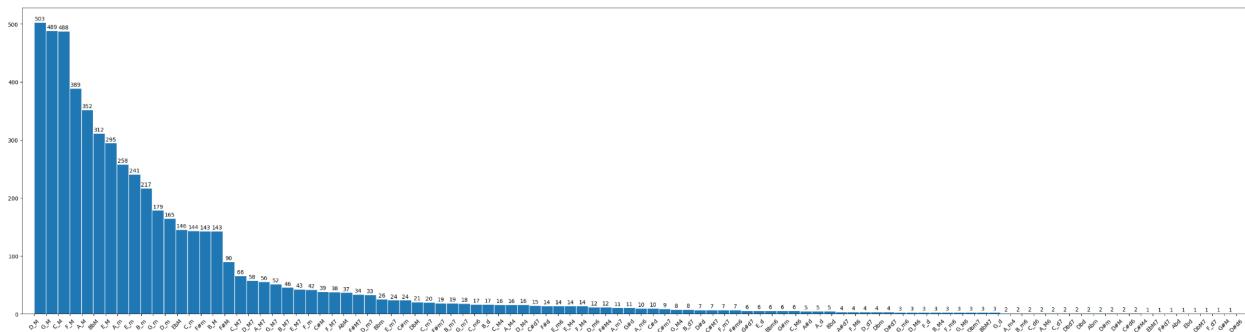
Možemo zaključiti da atributi nisu međusobno korelisani (slika ispod) na neki način koji bi nama bio od značaja u daljem radu i razvijanju modela za klasifikaciju.



Skup podataka sadrži 5665 instanci, nema nedostajućih kao ni nepoznatih vrednosti.

Postoji 102 različita akorda koji su zabeleženi u Bahovim horalima i svaka instanca, odnosno jedna harmonija pripada nekom od tih akorda.

Daljom analizom možemo utvrditi da su učestalosti akorda neuravnotežene, pa samim tim imamo pojavu nebalansiranosti klase, što se može videti na slici ispod.



Zbog same prirode skupa podataka, moraćemo da uradimo preprocesiranje da bismo dobili što je moguće bolje uslove za samo pravljenje modela klasifikacije.

Moguće su razne metode rešavanja ovakvog problema, pa ćemo se u nastavku osvrnuti na nekoliko koje smo probali, i kakve smo rezultate dobili.

Balansiranje na celom skupu

Kako je razlika između najučestalije klase i najmanje učestale klase oko 500 puta, kao i činjenica da neke klase imaju samo po jednu instancu, balansiranje bi drastično povećalo veličinu skupa podataka.

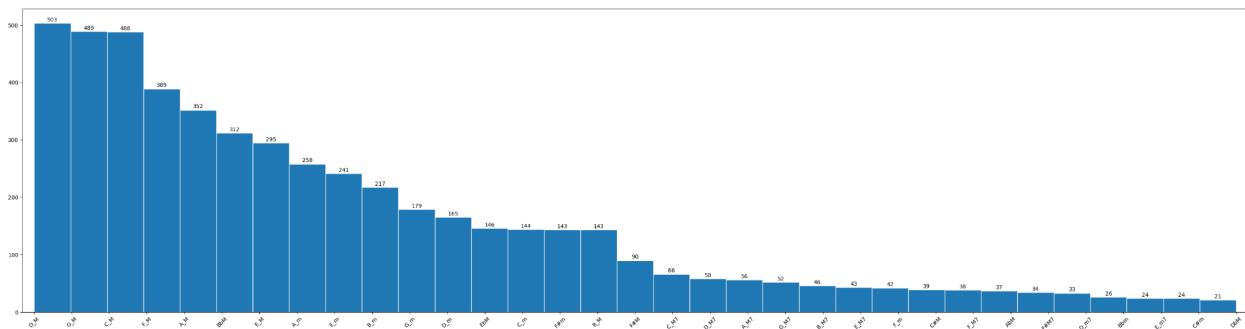
Pošto klasa koje sadrže samo nekoliko instanci ima relativno puno u odnosu na ukupan broj klasa, ovo rešenje ne bi bilo upotrebljivo pre svega zbog neadekvatnog generisanja veštačkih instanci (over-sampling), a onda i zbog veličine samog novogenerisanog skupa podataka koji bi zahtevao mnogo više vremena za treniranje modela klasifikacije.

Odsecanje klasa

Da bismo izbegli poteškoće pravljenja kao i treniranja modela, možemo pokušati da jednostavno "skratimo" skup izbacivanjem instanci koje imaju kao akord klasu koja se pojavljuje veoma malo puta.

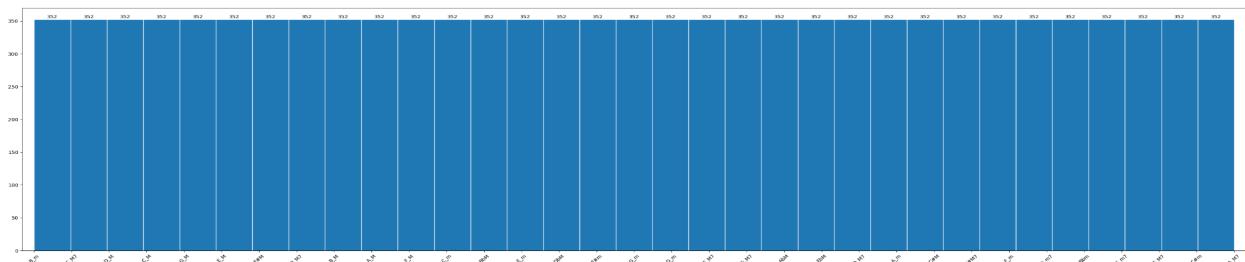
Ovim postižemo ne samo povećanje izbalansiranosti klasa skupa podataka, već i smanjenje samog skupa jer smo neke instance izbacili.

Dobijen je skup koji je sa 102 klase smanjen na oko 40, što se može videti na slici ispod.



Nakon odsecanja odnos najučestalije i najmanje učestale klase u skupu je pao za 10 puta (sa 500 na 50), nakon čega je moguće primeniti neki od metoda balansiranja kao što su **Random Oversampling** (ROS), ili **Synthetic Minority Oversampling Technique** (SMOTE) koji smo koristili u nastavku rada.

Slika ispod predstavlja raspodelu klasa posle primjenjenog SMOTE algoritma balansiranja.



Klasifikacija

Uvod

Cilj ove studije je uporediti performanse različitih modela klasifikacije. Koraci u istraživanju bi bili: analiza podataka, preprocesiranje, izbor parametara za pravljenje modela, korišćenje i evaluacija modela.

Kako smo u uvodu uradili analizu samih podataka kao i jedan deo preprocesiranja skupa podataka, ovde ćemo se fokusirati na specifičnosti samih modela, kao i neke razlike u pripremi podataka ukoliko model to zahteva.

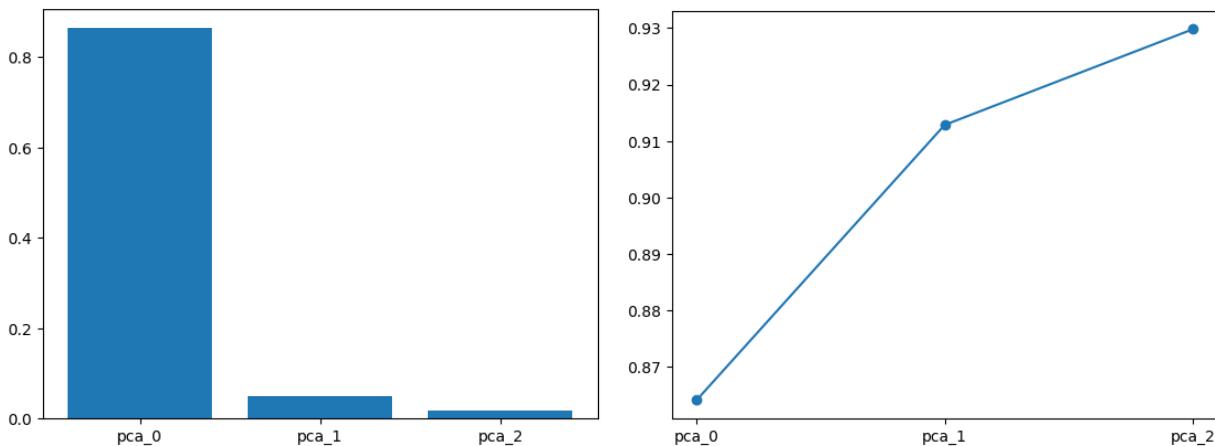
Za potrebe istraživanja ovog skupa podataka, za modele klasifikacije izabrani su modeli:

1. Stablo odlučivanja (Decision Tree)
2. Naivni Bajes (Categorical Naive Bayes)

Pored ovih korišćen je i ansambl Slučajne Šume (Random Forest).

Preprocesiranje

Pored pripreme podataka koje smo videli na početku (balansiranje, odsecanje), moguće je smanjiti dimenzionalnost samog skupa koristeći neku tehniku kao što je **Principal Component Analysis** (PCA).



Na slikama iznad se vidi da je između 2 i 3 dimenzije neznatna razlika (~0.01), pa ćemo skup smanjiti na 2 dimenzije radi lakše vizualizacije prilikom klasterovanja o kome će biti reč kasnije. Naravno pre svega ovoga smo skup podataka podelili na trening i test skupove, a PCA prilagodili trening skupu, i koristimo ga za transformaciju test skupa.

Opis modela

Stabla odlučivanja

Motivacije za odabir metode su:

1. Interpretabilnost: Drvo odlučivanja pruža jasan i intuitivan pregled procesa odlučivanja koji je (relativno) lako razumeti.
2. Mogućnost finog podešavanja modela parametrima.

Za pronalaženje optimalnih hiperparametara koristimo **GridSearch**. Prosleđujemo vrednosti parametara čije će se kombinacije dalje pretraživati. Testiraćemo različite mere (Entropija, Ginijev indeks, Log_loss), dubine stabla i kriterijume za podelu unutrašnjih čvorova i određivanje listova.

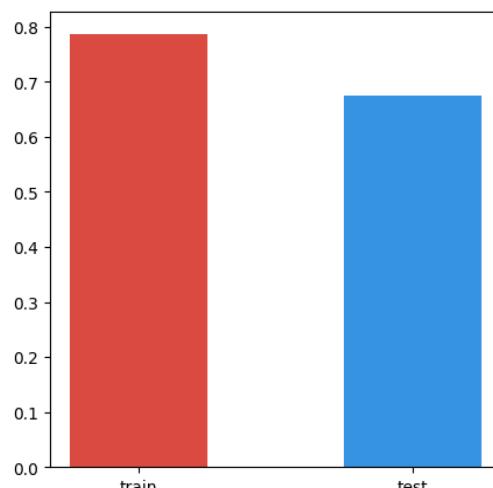
Prilikom implementacije ovog modela susreli smo se sa problemom tipova atributa skupa podataka. Naime, kako su većina atributa binarni, i to u fomratu "YES" i "NO", da bi naš model mogao da radi, skup podatka moramo transformisati. Ovo smo postigli enkodiranjem podataka koristeći klasu **OrdinalEncoder**.

Transformišemo skupove za treniranje i testiranje modela i tako transformisane skupove koristimo u daljem radu.

Sledeći korak je da kreirani model treniramo na transformisanom skupu za trening kako bismo videli koja kombinacija parametara je najbolja i daje najbolje rezultate.

Od svih modela koje smo isprobali (koristeći GridSearch), dobili smo da najbolji model ima najbolji rezultat 72.68%, a kako smo stavili da je način bodovanja tačnost, zaključujemo da je najbolja tačnost svih modela 72.68%.

Na trening skupu smo izračunali da model ima tačnost od 78.74%, dok na test skupu model ostvaruje tačnost od 67.41%, što možemo videti na slici ispod.

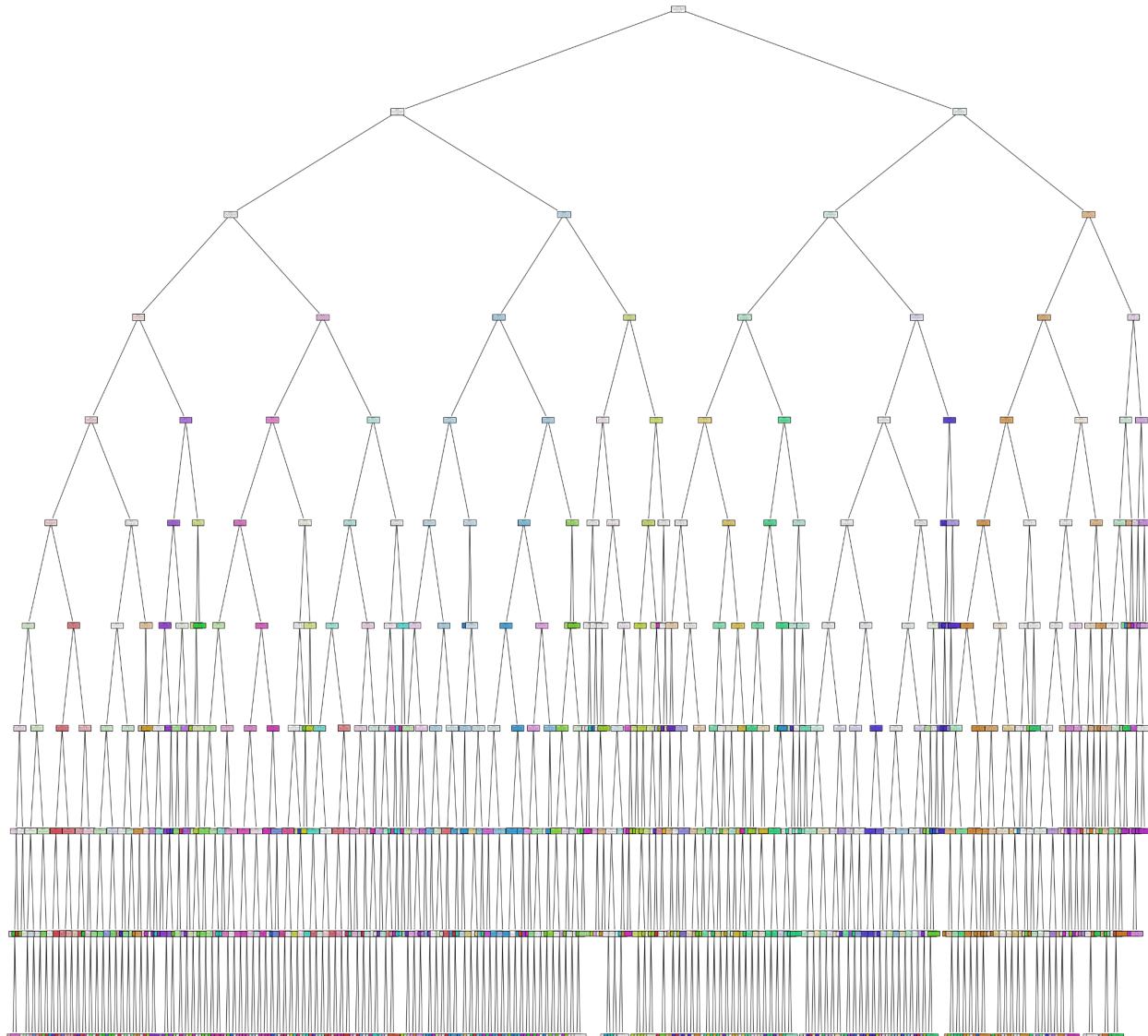


Vizualizacijom matrice konfuzije za test skup možemo videti kako model klasifikuje, gde greši i koliko greši, kao i sa kojim klasama ima problema u klasifikaciji. Sliku matrice konfuzije za test skup možemo videti ispod.



Možemo primetiti da na dijagonali koja bi trebalo da bude najviše obojena, postoje praznine koje su uzrokovane time da neke klase imaju veoma mali broj instanci, ili da pak ne postoje instance date klase u skupu. O rešavanju ovog problema ćemo više pričati kada budemo razmatrali model koji uzima u obzir balansirane instance.

Kao što smo već pominjali motiv za ovaj model je interpretabilnost pa ga možemo vizualizovati iscrtavanjem samog stabla odlučivanja na slici ispod.



Naravno, stablo odlučivanja je veoma veliko i ima velik broj čvorova i grananja zbog same prirode skupa podataka na kome radimo i broju atributa koji skup poseduje, uprkos tome možemo imati neku predstavu o tome koji atributi utiču na odlučivanje i u kojoj meri (bar na prvih nekoliko nivoa stabla).

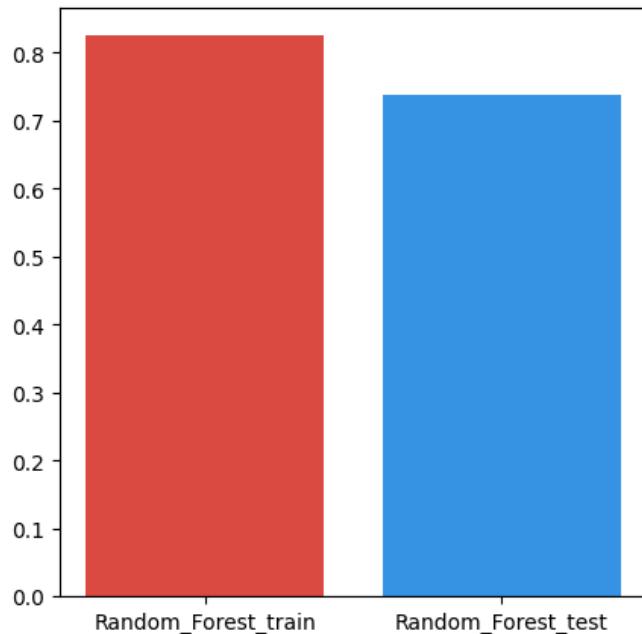
Slučajne šume

Motivacija za odabir modela su kao i kod stabala odlučivanja, sa tim što slučajne šume pripadaju ansablma, odnosno većem broju modela (u ovom slučaju stabala odlučivanja) koji zajednički donose odluke.

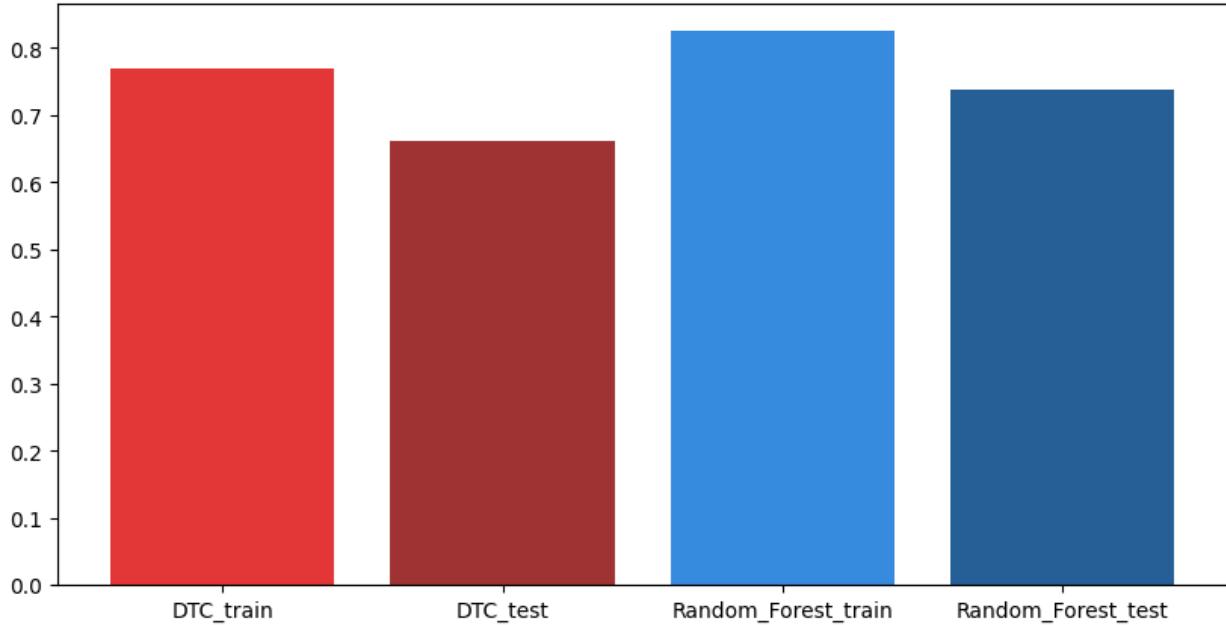
Pripreme podataka za ovaj model su identične kao i kod stabala odlučivanja, pa smo koristili iste transformisane trening i test skupove.

Naglasimo da smo se i ovde odlučili da, iako možda nepotrebno, iskoristimo GridSearch da bismo našli najbolje hiperparametre za model. Bitno je spomenuti da ova odluka u kombinaciji sa samim ansamblom slučajne šume, dovodi do znatno vremenski zahtevnije operacije treniranja modela. Naime, prethodni model stabla odlučivanja o kome smo pričali je na ovom skupu podataka završio treniranje za svega nekoliko sekundi, dok je proces treniranja ansambla slučajne šume trajao čak 16 minuta (korišćen je Google Colab backend).

Ansambl je ostvario najbolju tačnost od 75.71%. Na trening skupu tačnost iznosila 82.45%, dok je na test skupu tačnost bila 73.71%. Vizualizaciju tačnosti ansambla slučajne šume možemo videti na slici ispod.

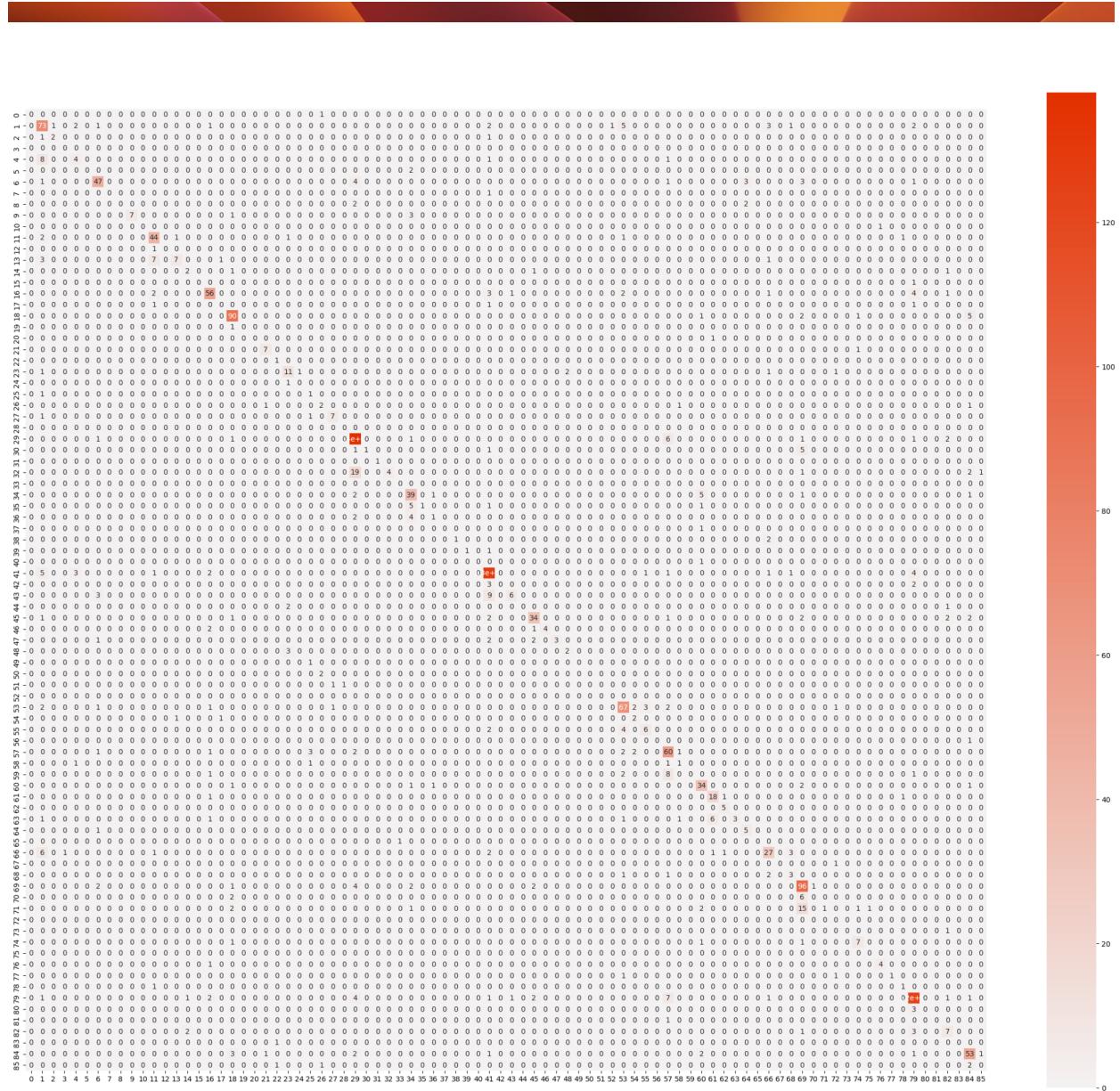


Ako uporedimo slučajne šume i stabla odlučivanja možemo videti povećanje tačnosti od oko 4% na trening skupu, dok povećanje tačnosti na test skupu iznosi oko 6%. Vizualizaciju poređenja možemo videti ispod.



Iako poboljšanje od oko 6% nije zanemarljivo, moramo se zapitati da li je vredno drastičnog povećanja vremena potrebnog za trening sa nekoliko sekundi na više od 10 minuta, i da li bi treniranje na nekom drugom skupu koji ima nekoliko desetina hiljada instanci bilo praktično izvodljivo.

Kao i kod stabala odlučivanja možemo dodatno oceniti model vizualizacijom matrice konfuzije na test skupu.



Naivni Bajes

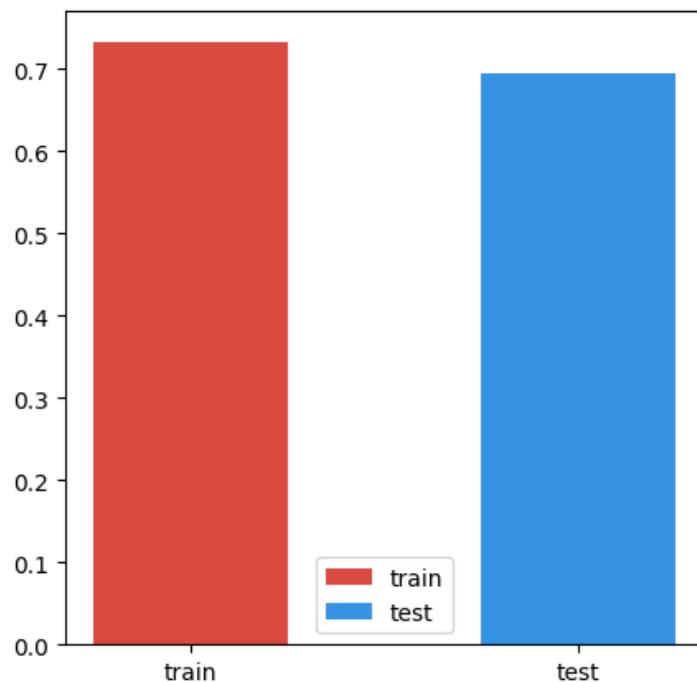
Motiv za izbor modela je to što je, za razliku od stabala odlučivanja, naivni Bajes veoma brz i može se koristiti za predviđanje u realnom vremenu. Nama predviđanje u realnom vremenu nije krucijalno, ali svakako da možemo uvideti benefite u brzini pravljenja pa i samog treniranja modela koji koristi naivni Bajesov algoritam za klasifikaciju. Takođe, naivni Bajes ne zahteva velike trening skupove, što je dobro i potpomaže samoj brzini treniranja.

Priprema podataka je kao i kod prošlih modela veoma slična: podela na trening i test skupove, praćeno enkodiranjem podataka.

I ovde smo koristili GridSearch i unakrsnu validaciju kako bismo pronašli optimalne vrednosti za hiperparametre.

Isprobali smo dve verzije naivnog Bajesa: kategorički naivni Bajes (**Categorical Naive Bayes**) i Bernulijev naivni Bajes (**Bernoulli Naive Bayes**)

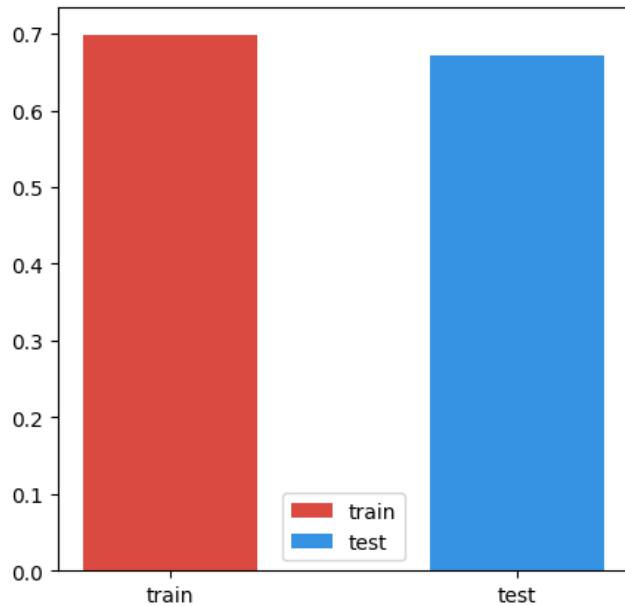
Model kategoričkog naivnog Bajesa sa najboljim hiperparametima je imao najbolji rezultat (tačnost) od oko 71%. Tačnost predviđanja na trening skupu iznosi oko 73%, dok je tačnost na test skupu oko 70%. Možemo na slici ispod pogledati vizualizaciju tačnosti predviđanja kategoričkog naivnog Bajesa na trening i test skupovima.



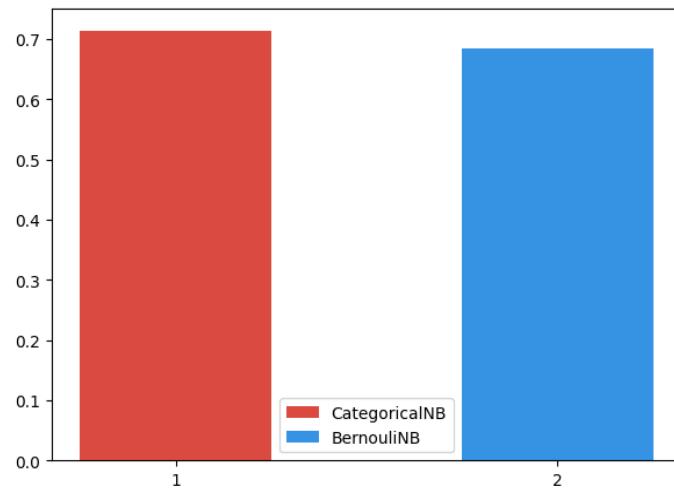
Za Bernulijevog naivnog Bajesa nije bilo potrebno ništa dodatno raditi ili preprocesirati, te smo samo iskoristili iste transformisane trening i test skupove.

Takođe, i za Bernulijev naivni Bajes smo tražili optimalne hiperparametre primenjivanjem unakrsne validacije. Model sa najboljim hiperparametrima je imao najbolji skor (tačnost) oko 68%. Tačnost predviđanja na trening skupu iznosi oko 70%, dok je tačnost na test skupu 67%.

Pogledajmo na slici ispod vizuelan prikaz tačnosti predviđanja Bernulijevog naivnog Bajesa.



Model Bernulijevog naivnog Bajesa je na istom skupu trening završio u proseku za oko 3 sekunde, dok je modelu kategoričkog naivnog Bajesa u proseku trebalo oko 7 sekundi. Pogledajmo poređenje prosečne tačnosti predviđanja ova dva modela naivnog Bajesa.

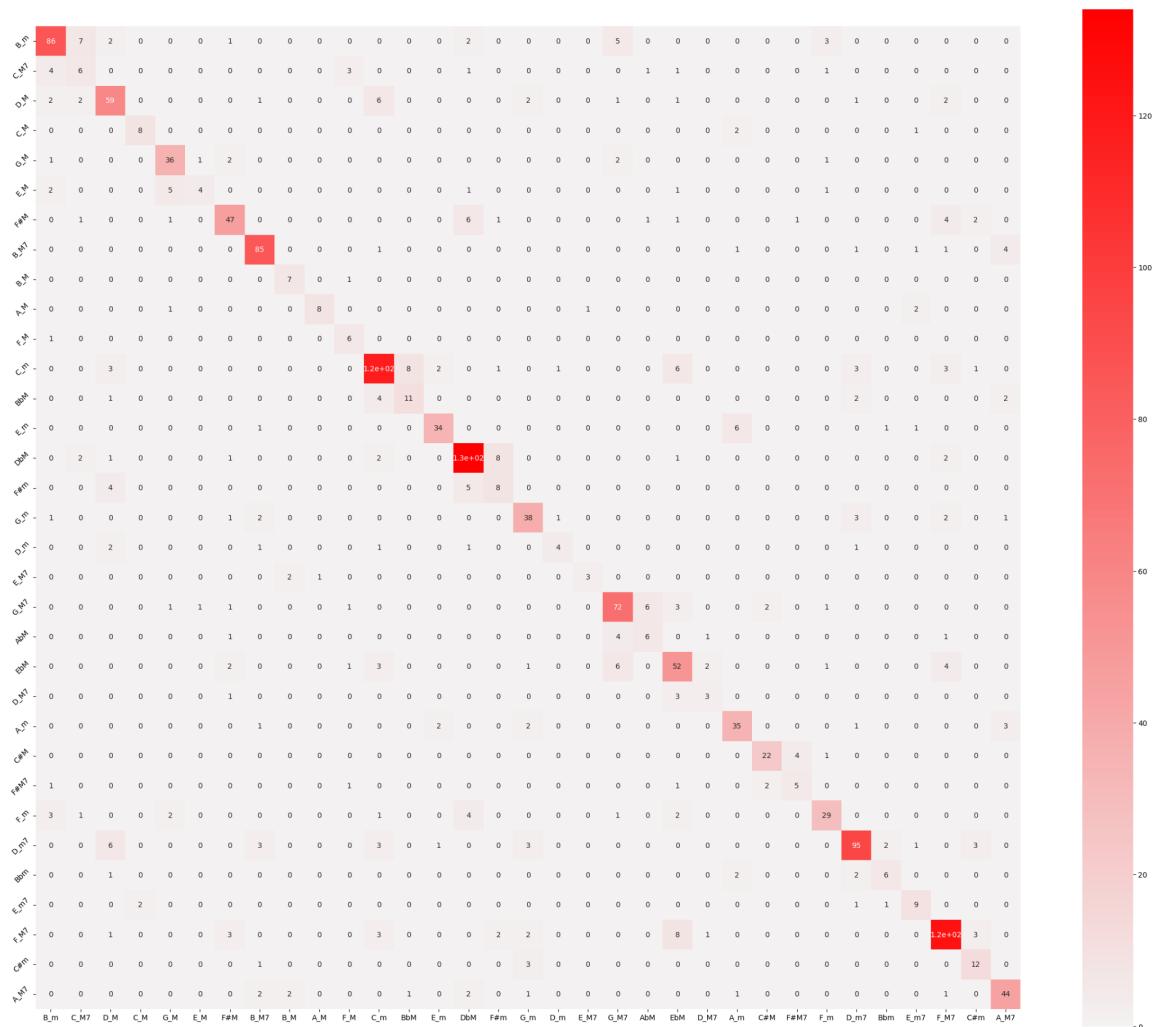


Poboljšanje modela klasifikacije

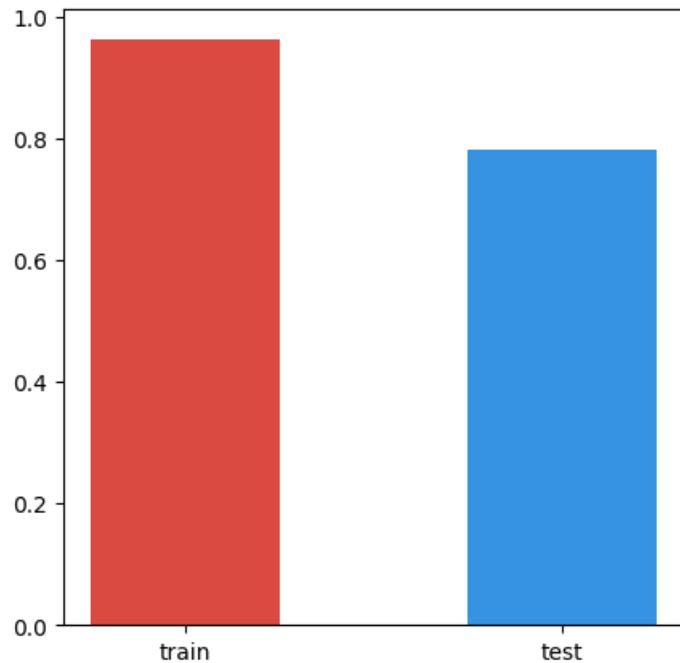
Kao što smo već rekli, ovaj skup podataka i instance koje se u tom skupu nalaze su veoma specifične. Zbog prirode samih harmonija koje se pojavljuju u Bahovim horalima, neki se javljaju veoma malo puta ili se uopšte ne javljaju. Probali smo nekoliko stvari koje imaju potencijal da poboljšaju tačnost klasifikovanja ovog skupa, bez da remetimo prirodnu strukturu samih podataka, kao i ono što ti podaci predstavljaju.

Pre svega smo hteli da vidimo da li problem u kreiranju modela i njegovoj tačnosti predviđanja predstavlja to što postoje klase koje sadrže samo jednu ili dve instance. Iz skupa podataka smo, kao što je pre bilo reči, izbacili te klase i njihove instance. Zatim smo preostale klase balansirali SMOTE algoritmom kao što smo rekli u koraku [Odsecanje klasa](#). Za model klasifikacije kojim bismo pokazali poboljšanje smo izabrali ansambl [slučajne šume](#), obzirom na to da je taj model dao najbolje rezultate i pre.

Model je ovog puta trening završio za samo nekoliko sekundi, a vizuelnu reprezentaciju matrice konfuzije na test skupu možemo pogledati na slici ispod.



Na prvi pogled, matrica konfuzije nam izgleda kao da je relativno uspešno model klasifikovao podatke, a isto nagoveštava i tačnost (*accuracy*) na trening skupu od čak 96%, što je najviše do sad ostvarena. Međutim, na test skupu tačnost predviđanja iznosi 78%.

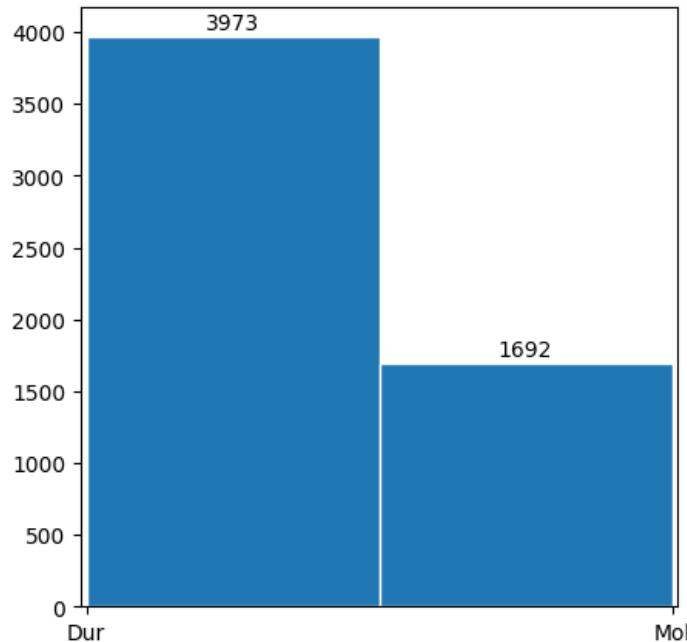


Objašnjenje za ovo ponašanje krije se u tome da smo poboljšanje u vidu balansiranja klasa, naravno, radili samo na trening skupu podataka nadajući se da će kao proizvod toga naš mode biti bolji i kvalitetniji za klasifikaciju na test skupu, pa i bilo kom nepoznatom skupu koji nam posle dođe. Upravo ovo nam pokazuje specifičnost podataka ovog skupa, kao i to da nam (videćemo posle) ovaj model zaista daje malo bolje rezultate, ali ni blizu onoliko koliko smo očekivali da bismo mogli reći da je imalo smisla na ovaj način optimizovati.

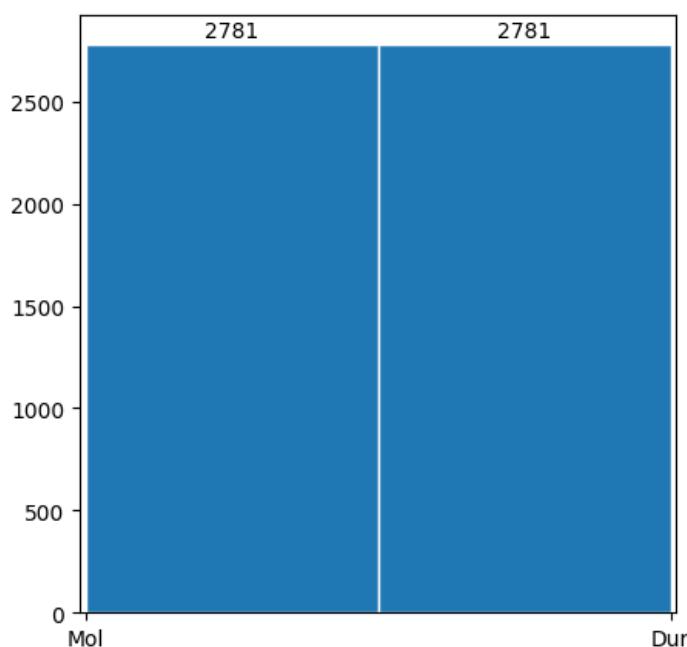
Zaključak je da uprkos balansiranju skupa podataka, nismo bili u mogućnosti da preterano povećamo tačnost klasifikacije modela nad našim skupom podataka.

Sledeći pokušaj optimizacije se svodi na to da promenimo sam skup podataka, ali tako da to ima smisla, i ne remeteći značaj klasa ili ono što one predstavljaju. Naime, klase koje mi želimo da predvidimo su, u muzičkom smislu, harmonije ili akordi. Generalno gledano, u muzici prirodno možemo podeliti sve akorde u dve grupe, dur ili **Major**, i mol ili **Minor**.

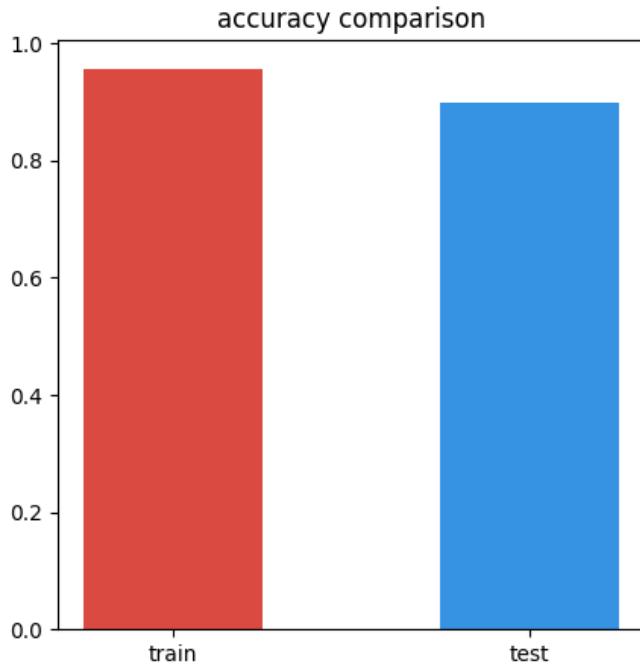
Imajući to u vidu, transformisali smo naš skup podataka i sveli naš problem na klasifikaciju u dve gore pomenute klase. Raspodela instanci po klasama u transformisanom skupu izgleda kao na slici ispod.



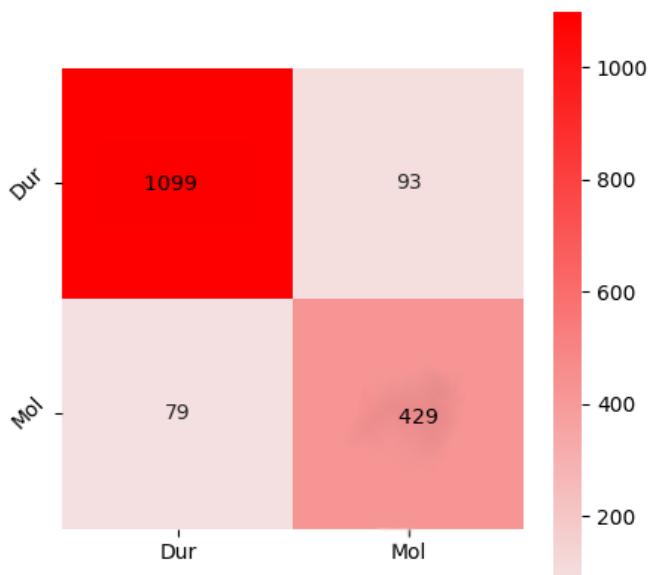
Možemo primetiti da odnos klasa nije drastično veći kao što je to bio slučaj pre kada smo imali mnoštvo klasa. Ipak, zbog maksimalne optimizacije, i ovo ćemo izbalansirati. Balansiranje smo i ovde radili pomoću **SMOTE** algoritma, a novu raspodelu pogledajmo na slici ispod.



I ovde smo koristili ansambl slučajne šume, i trening ovog ansambla na novom skupu podataka je završen za oko 2 sekunde. Ovako istrenirani model klasifikacije na trening skupu ostvaruje tačnost predviđanja od oko 95%, dok je tačnost na test skupu oko 90%.



Možemo primetiti da ovog puta, iako pojednostavljena klasifikacija, i na test skupu tačnost predviđanja modela ne odstupa u toj meri kao pre, što znači da smo transformisanjem skupa podataka na ovaj način, uspeli da povećamo efikasnost klasifikacije a da ne poremetimo neki prirodni odnos podataka i šta oni predstavljaju. Da bismo se u to uverili, možemo pogledati matricu konfuzije klasifikacije na test skupu.



Poređenje modela klasifikacije

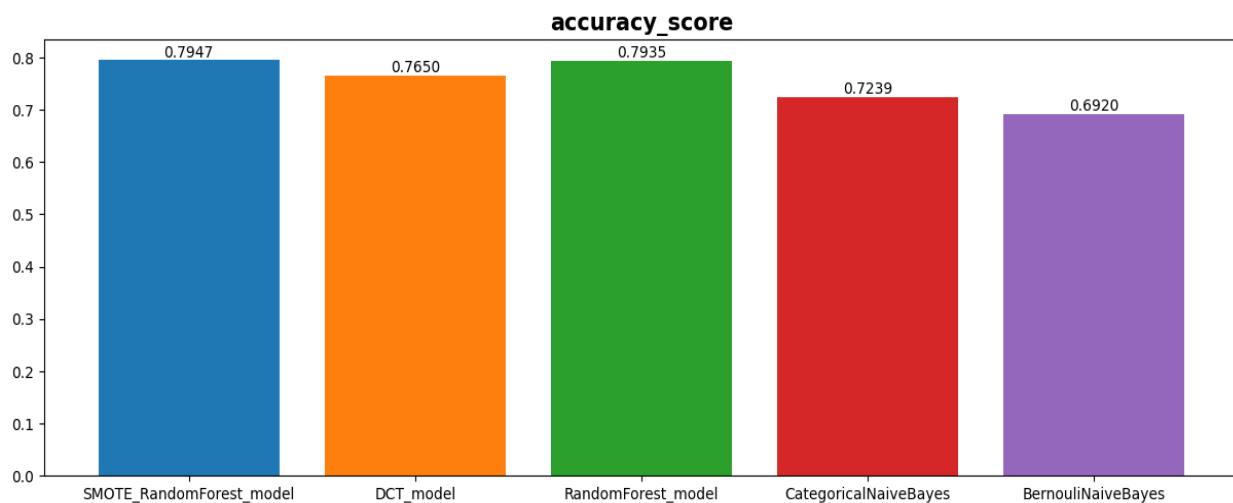
Uvod

Svaki dosadašnji model je imao svoje prednosti i mane, i svaki nam je pružio malo bolje razumevanje samog skupa podataka. Od toga koji su algoritmi klasifikacije pogodniji za naš skup, do toga da smo u svakom imali priliku da vidimo kako parametri samog modela utiču na tačnost predviđanja. U nastavku ćemo na jednom mestu uporediti sve modele koje smo koristili, i vizualizovati njihove rezultate. Sve modele smo učitali i na istom skupu testirali. Da bismo još detaljnije uporedili modele, odlučili smo da poredimo za više različitih metrika. Izabrali smo:

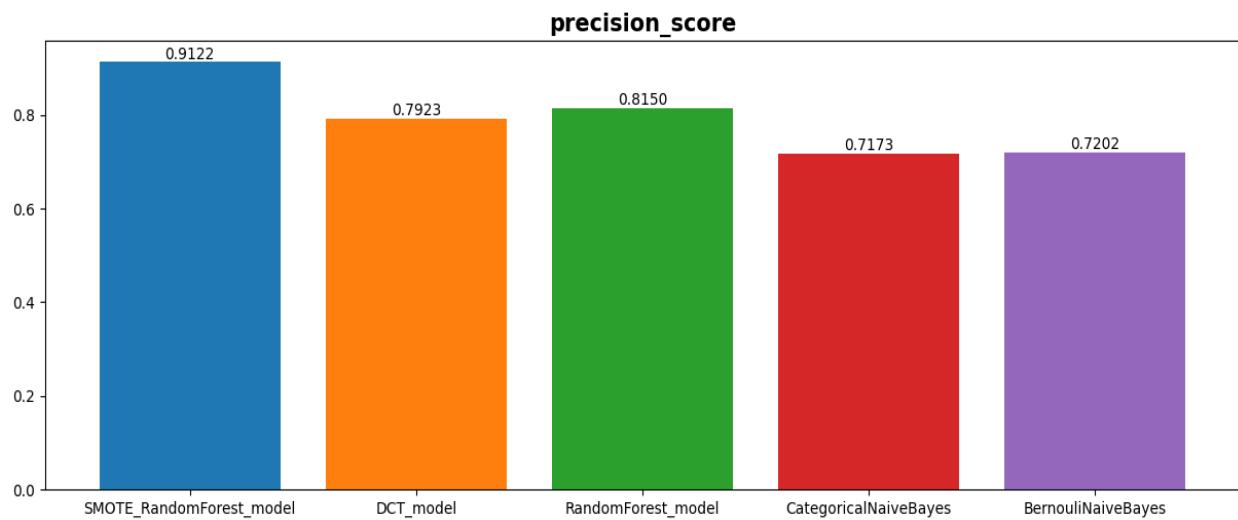
1. Tačnost (**Accuracy Score**)
2. Preciznost (**Precision Score**)
3. Odziv (**Recall Score**)
4. F1 Mera (**F1 Score**)

Za svaki od njih ćemo imati zasebnu vizualizaciju i kratko ih prokomentarisati.

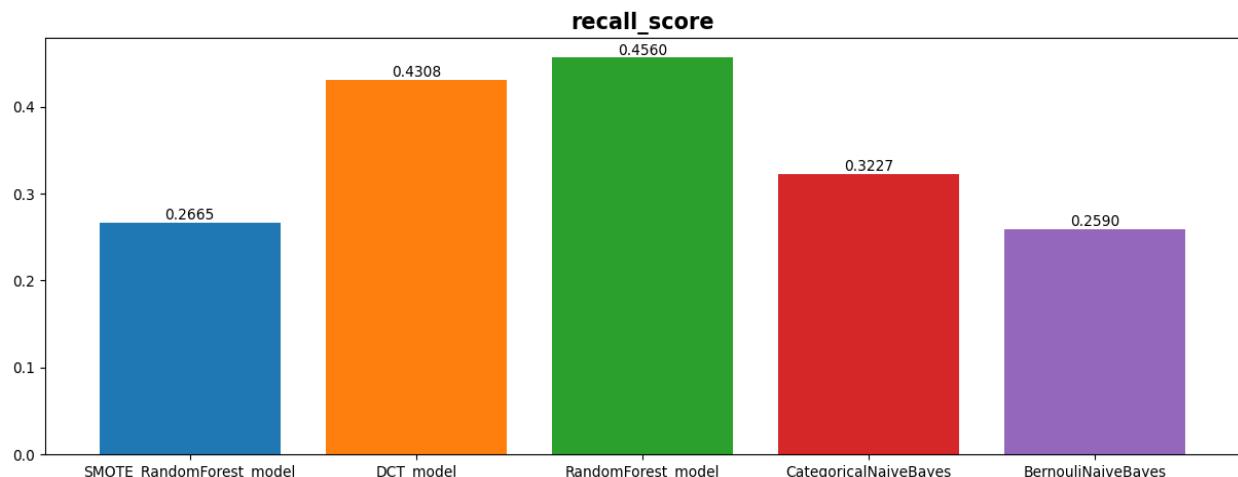
Vizualizacija poređenja modela



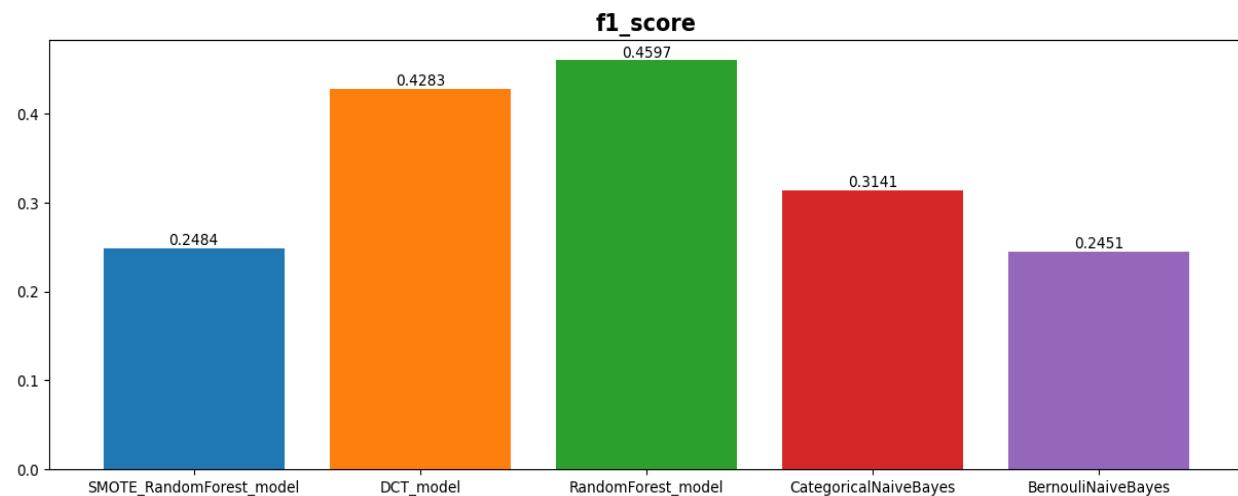
Možemo videti da najbolju tačnost imaju ansambli slučajnih šuma. Iako smo u jednom modelu radili poboljšanja balansiranjem i odsecanjem klasa (plavo), vidimo da poboljšanja u odnosu na običan ansambl (zeleno) zapravo nema.



Kada za metriku koristimo preciznost (Precision Score) vidimo da ansambl koji je treniran na balansiranom i potkraćenom skupu podataka (plavo) daje za oko 10% bolje rezultate. Ovaj model ima dobru sposobnost tačnog identifikovanja pozitivnih slučajeva.



Odziv (Recall Score) predstavlja sposobnost modela da tačno identificuje sve relevantne pozitivne slučajeve prilikom klasifikacije. Ovde vidimo da tu model treniran na balansiranom i skraćenom skupu (plavo), značajno zaostaje za običnim ansamblom slučajne šume. Takođe, primećujemo da se slično ponašaju (bar za odziv) i modeli naivnog Bajesa, sa tim da je kategorični naivni Bajes malo bolji.



F1 mera je metrika koja kombinuje i preciznost (*Precision*) i odziv (*Recall*). Pruža balansiranu ocenu performansi modela uzimajući u obzir i lažno pozitivne i lažno negativne rezultate. F1 rezultat se izračunava kao harmonijska sredina preciznosti i odziva. Nizak F1 rezultat ukazuje na to da model ima poteškoće u postizanju prave ravnoteže između preciznosti i odziva. Imajući u vidu da je ansambl treniran na balansiranom i skraćenom skupu imao veliku razliku između preciznosti i odziva, logično je da stoga ima i lošij rezultat kada je u pitanju F1 mera.

Zaključak

Iz vizualizacija poređenja modela sa različitim metrikama možemo zaključiti nekoliko stvari:

1. Ansambli slučajnih šuma daju najbolje globalne rezultate klasifikovanja.
2. Pokazali smo da uprkos balansiranju i skraćivanju broja klasa nismo dobili nikakav boljši rezultat u klasifikovanju (zbog slične strukture i prirode skupa podataka nad kojim radimo)
3. Pokazali smo da postoje više načina tumačenja skupa podataka (na primer da se gledaju Dur i Mol harmonija umesto sami akordi), i da u tim slučajevima modeli klasifikacije mogu dati prilično zadovoljavajuće rezultate i primenjive u realnosti.

Klasterovanje

Uvod

Cilj klasterovanja podataka je identifikacija inherentnih obrazaca, struktura ili grupa unutar skupa podataka. Klasterovanje je tehnika nenadgledanog mašinskog učenja koja ima za cilj grupisanje sličnih podataka na osnovu njihovih intrinzičkih karakteristika ili blizine u prostoru karakteristika. Osnovni ciljevi klasterovanja su:

1. **Istraživanje podataka:** Klasterovanje pomaže u istraživanju i razumevanju osnovne strukture ili prirodnih grupa prisutnih u podacima. Može dati uvide i pružiti opšti pregled skupa podataka.
2. **Prepoznavanje obrazaca:** Identifikacijom klastera, algoritmi za klasterovanje mogu otkriti obrasce, trendove ili veze u podacima koje možda nisu odmah očigledne. Može pomoći u identifikaciji značajnih segmenata ili podgrupa unutar podataka.
3. **Otkrivanje anomalija:** Klasterovanje može biti korisno u identifikaciji vanrednih vrednosti ili anomalija u podacima. Podaci koji ne pripadaju nijednom klasteru ili su značajno drugačiji od ostalih mogu se smatrati potencijalnim anomalijama koje zahtevaju dalju analizu.

Preprocesiranje

Da bismo započeli proces klasterovanja, moramo izvršiti neki vid preprocesiranja nad skupom podataka. Kao što smo već pričali kod preprocesiranja za klasifikaciju, uradili smo smanjenje dimenzionalnosti skupa podataka. Ovo smo uradili iz nekoliko razloga. Prvi je to što, očigledno, smanjenje broja atributa dovodi do povećanja brzine treniranja modela klasterovanja i njegovog izvršavanja. Drugi razlog, a možda nama i bitniji, je to što smanjivanjem dimenzionalnosti skupa na samo 2¹ dimenzije a da pritom što bolje objasnimo varijansu podataka. Ovim postižemo veoma laku vizualizaciju klasterovanja koje budemo radili.

Pored ovoga, podatke smo skalirali tako da nam budu pogodniji za vizualizaciju. Iako ovo možda nije bilo potrebno obzirom na to kakve smo podatke dobili posle smanjivanja dimenzionalnosti skupa podataka, ipak smo skalirali podatke klasom **MinMaxScaler**.

Sa ovako transformisanim skupom podataka možemo dalje nastaviti kreiranje modela za klasterovanje.

¹ Mogli smo i 3 dimenzije ali nije bilo nikavog bitnog poboljšanja, a pritom nam je bilo pogodnije da vizualizaciju radimo u 2D nego u 3D.

Opis modela

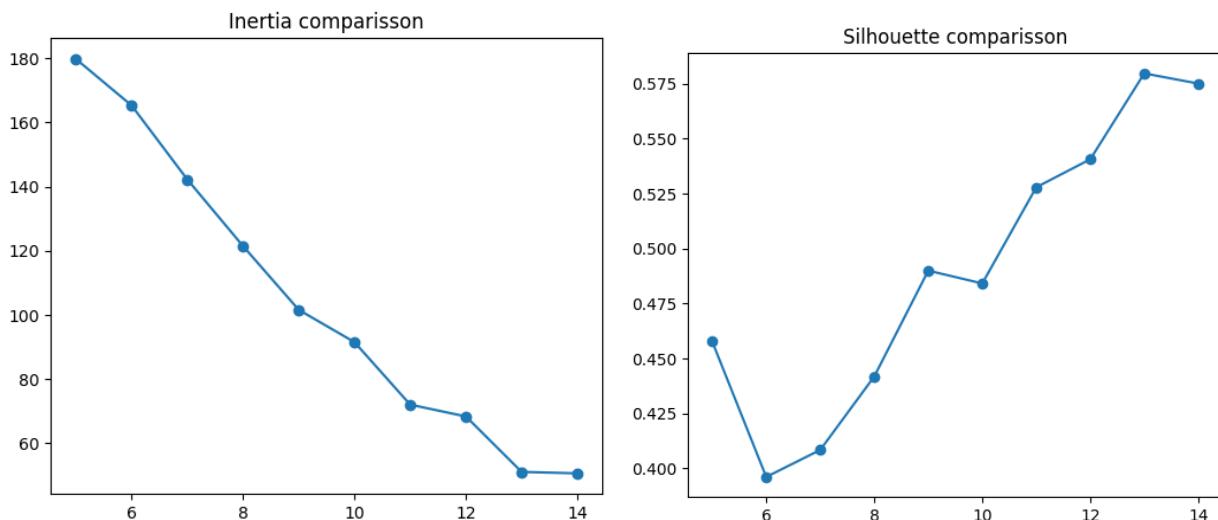
Za modele klasterovanja smo se odlučili za:

1. K-Sredina (*K-Means*)
2. Hjernarhijsko klasterovanje (*Agglomerative Clustering*)
3. DBSCAN (*Density-Based Spacial Clustering of Applications with Noise*)

K-Sredina

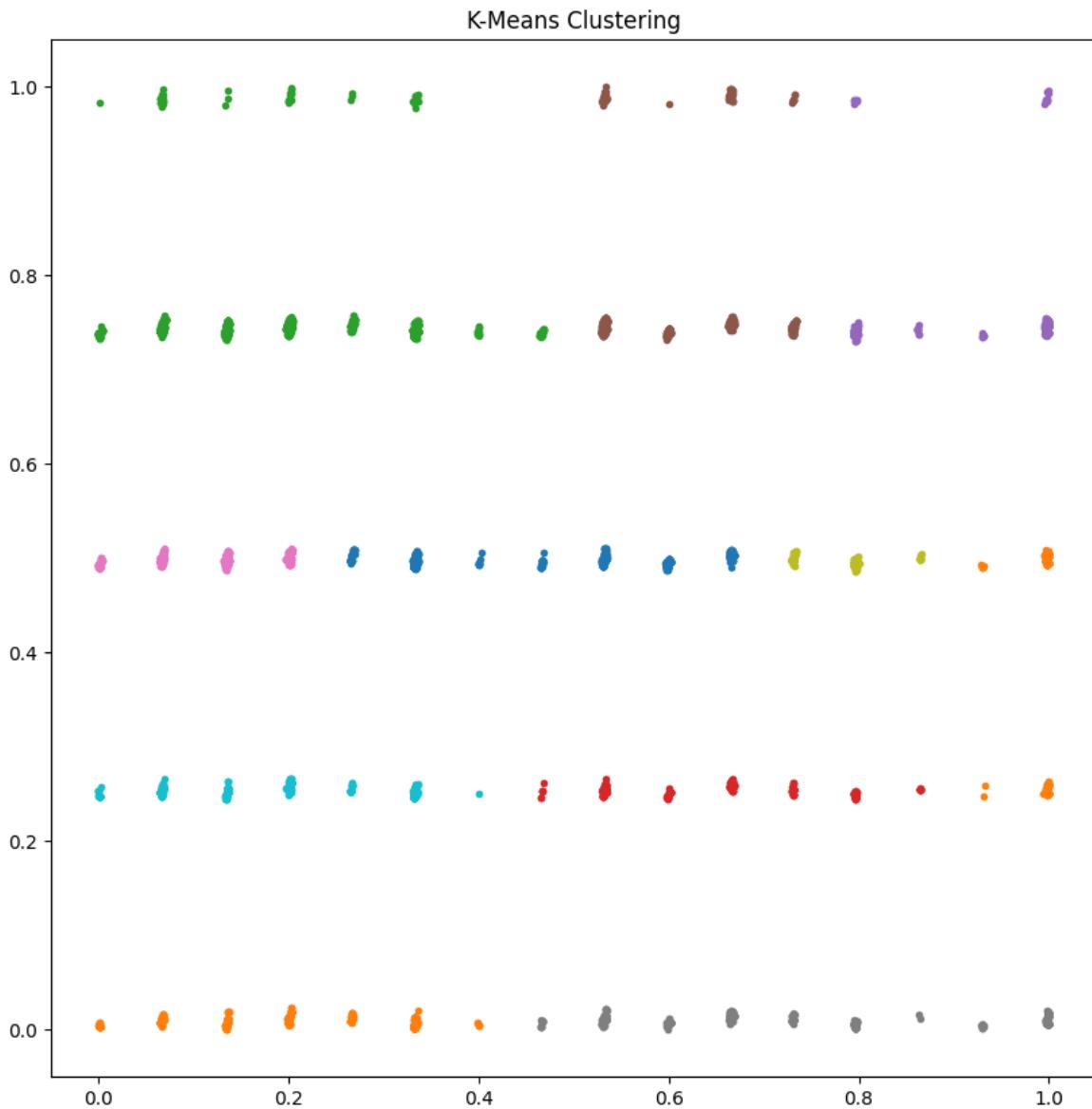
Osnova algoritma K-Sredina je koncept klasterovanja baziranog na centroidima. Algoritam ima za cilj da podeli skup podataka na k različitih klastera, pri čemu svaka tačka pripada klasteru sa najbližim centroidom. Centroidi predstavljaju centralnu ili srednju tačku svakog klastera.

K-means algoritam minimizuje sumu kvadrata unutar klastera (WCSS²) i teži da formira kompaktne i dobro odvojene klastere. Važno je napomenuti da početni nasumični izbor centroida može uticati na konačni rezultat klasterovanja i algoritam može konvergirati ka različitim rešenjima za različite inicijalizacije. Kako bi se ovo ublažilo, algoritam se često pokreće više puta sa različitim inicijalizacijama, a izabere se rešenje sa najmanjom sumom kvadrata unutar klastera. U nastavku ćemo za izbor najboljih parametara za K-Sredina koristiti dve mere, siluetu (*Silhouette*) i inerciju (*Inertia*), a odnos metrika i broja klastera možemo pogledati na slikama ispod.



² WCSS - Within-Cluster Sum of Squares

Kako u našem skupu podataka, kao što smo pre videli, imamo oko 100 različitih klasa, logično je da silueta raste i da je najveća vrednost za 13 klastera. Ipak, da bismo imali kompromis između brzine izvršavanja ali da pritom za broj klastera uzmemmo broj koji ima smisla, pogledaćemo meru inercije. Koristeći metod "lakta" možemo videti da se posle 13 klastera mera inercije skoro uopšte ne menja, a da je upravo za taj broj klastera i vrednost siluete na našoj slici (desno) maksimalna. Uzimajući ovo u obzir možemo pogledati kako je algoritam K-Sredina klasterovao naše podatke u 13 klastera.



U realnosti mi znamo da bi se svaki od ovih malih potklastera mogao klasterovati u poseban, što bi zapravo bilo i tačno s obzirom na to da imamo oko 100 klasa u našem skupu. Uprkos tome, vidimo da je algoritam relativno smisleno odvojio klastere.

Hijerarhijsko klasterovanje

Hijerarhijsko klasterovanje počinje sa svakom tačkom podataka kao posebnim klasterom i iterativno spaja klastera na osnovu njihove sličnosti. Algoritam postepeno gradi hijerarhiju klastera sve dok se ne dostigne željeni broj klastera ili određeni kriterijum.

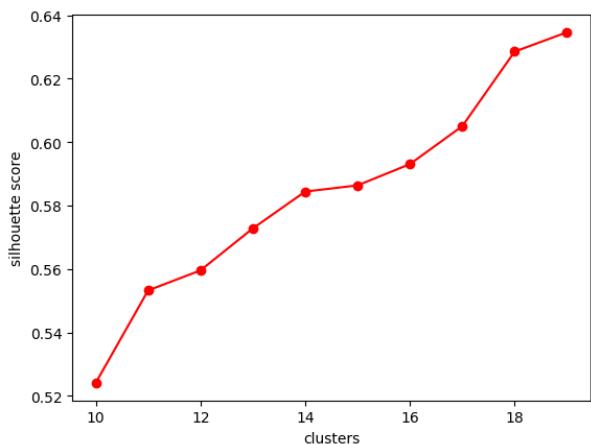
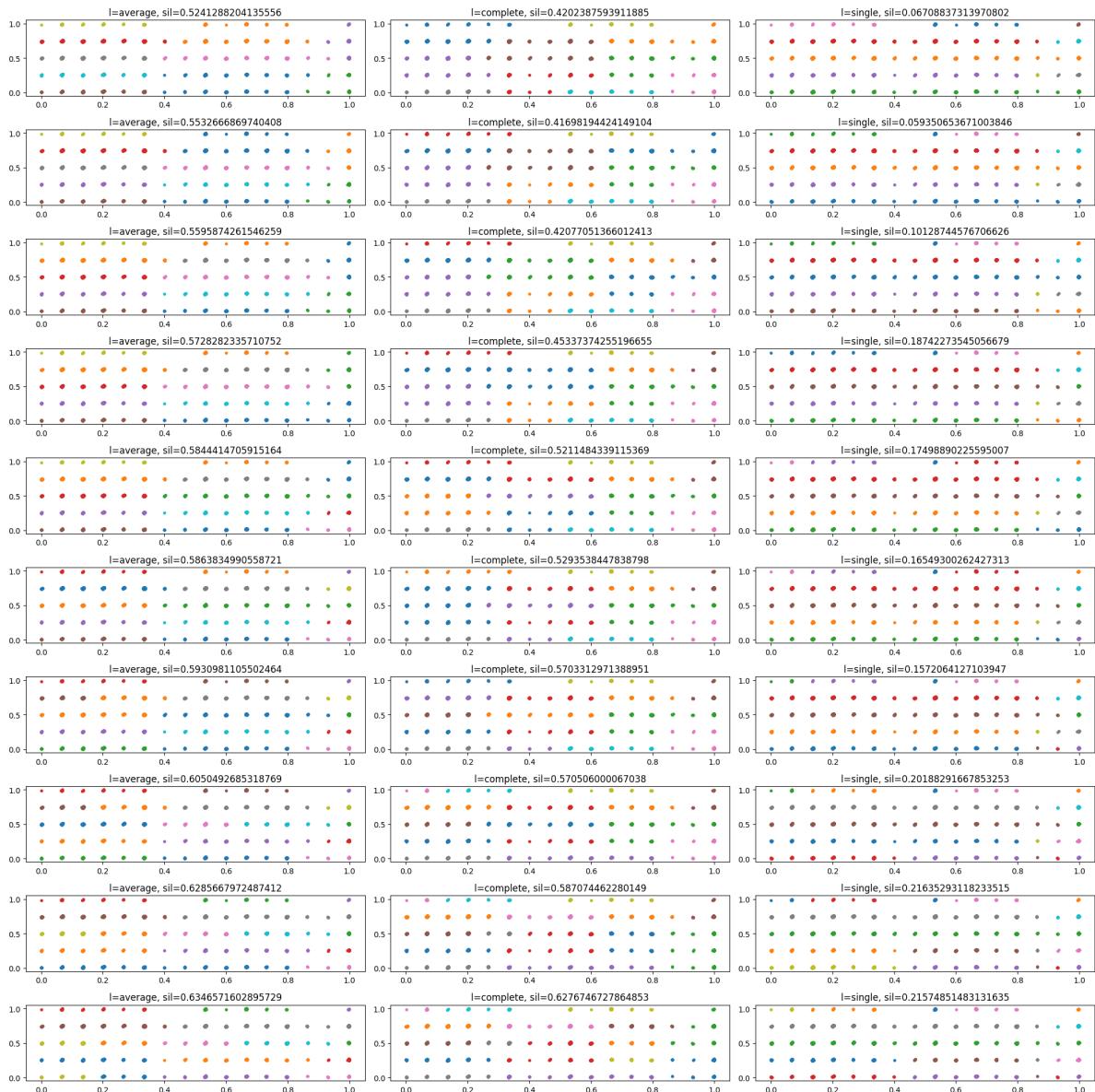
Hijerarhijsko klasterovanje je pristup odozdo nagore, gde se klasteri postepeno spajaju kako bi formirali veće klastere. Ne zahteva od korisnika da unapred odredi broj klastera, jer se hijerarhija može preseći na različitim nivoima kako bi se dobili različiti brojevi klastera.

Izbor mere sličnosti ili različitosti, kao i kriterijum povezivanja koji se koristi za određivanje udaljenosti između klastera (na primer, jednostruko povezivanje, potpuno povezivanje, prosečno povezivanje), mogu uticati na rezultate klasterovanja.

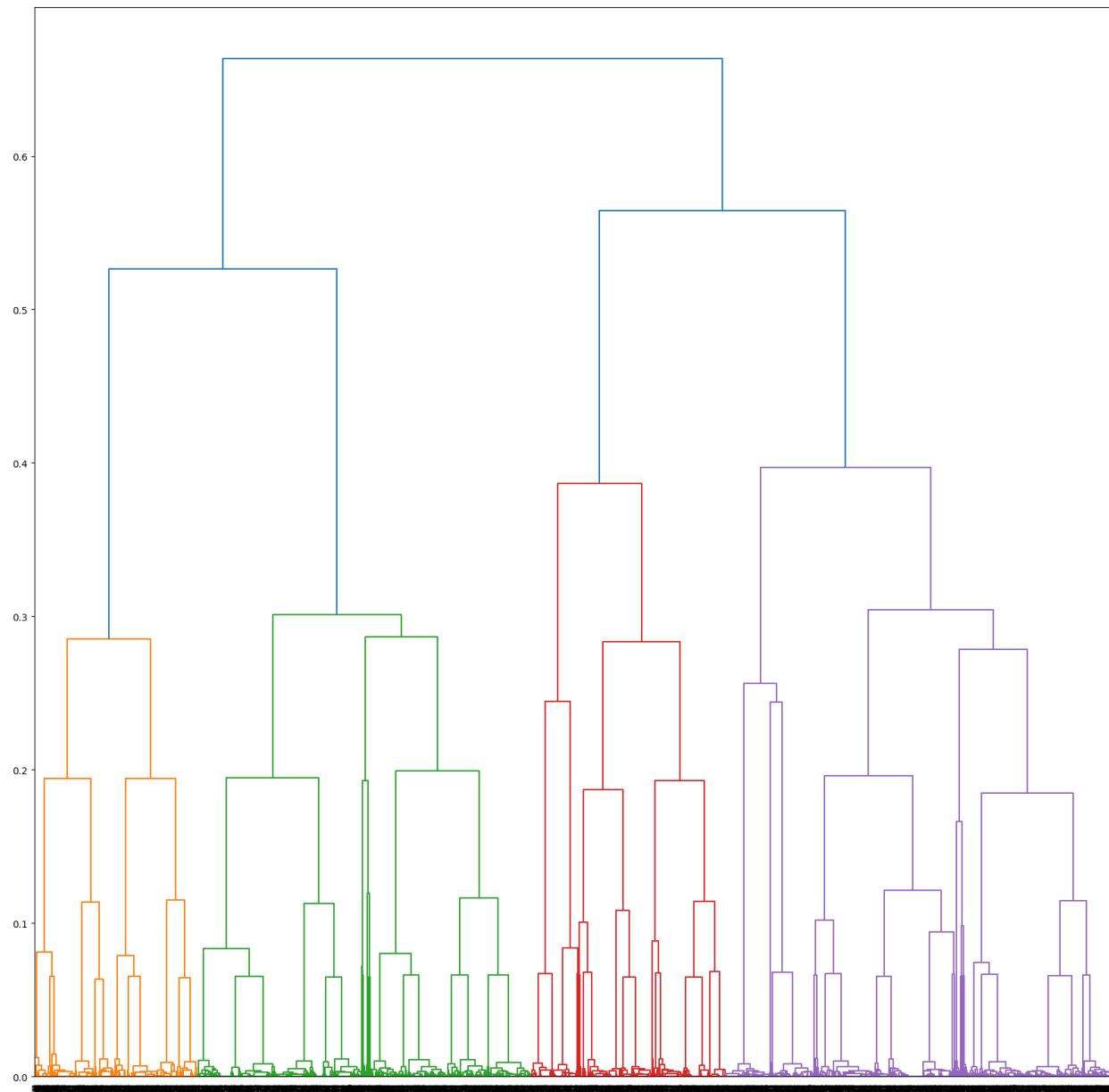
Kao i u prethodnom modelu, nismo za broj klastera stavljali velike brojeve zbog brzine izvršavanja, ali smo isprobali nekoliko u kombinaciji sa tri kriterijuma povezanosti.

U nastavku ćemo prikazati kako model hijerarhijskog klasterovanja grupiše podatke iz našeg skupa u zavisnosti od parametara koje smo mu dali, i da li možemo da primetimo koji je najbolji.

Nakon toga, da bismo našli najbolji model, iskoristićemo meru siluete, ali ne i inercije jer je to specifično za model *K-Sredina*.



Kao i kod *K-Sredina* očekivano je da silueta raste kako rastu klasteri, pa smo za dalji rad i vizualizaciju hijerarhijskog predstavljanja preko dendrograma uzeli broj klastera da bude 19. U nastavku možemo videti nacrtan dendrogram hijerarhijskog klasterovanja za najbolje parametre i za 19 klastera.

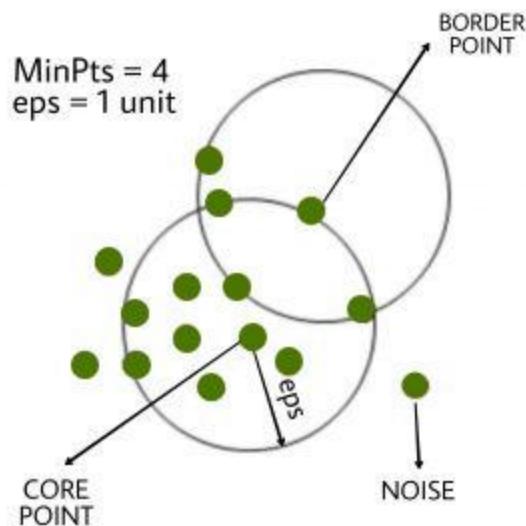


Opet smo svesni činjenice da zapravo postoji mnogo više klastera nego što smo nacrtali, ali smo opet uvideli jedan način da klasterujemo instance a da to ima smisla (pogledati sliku u poslednjem redu i prve dve kolone).

DBSCAN

DBSCAN je algoritam za klasterovanje zasnovan na gustini koji grupiše zajedno podatke koji su bliski jedni drugima u gustoj oblasti, razdvajajući ih od područja niže gustine. Posebno je koristan za otkrivanje klastera proizvoljnog oblika i rukovanje sa šumovitim podacima.

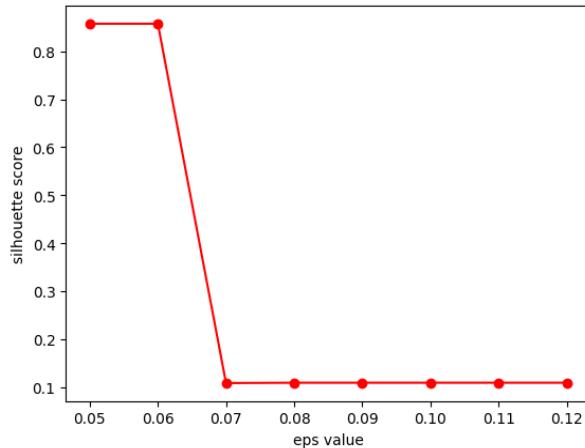
Prednosti DBSCAN-a uključuju sposobnost rukovanja klasterima različitih oblika i veličina, otpornost na šum i sposobnost automatskog otkrivanja broja klastera. Međutim, zahteva pažljiv izbor parametara i može imati poteškoća sa podacima visoke dimenzionalnosti ili klasterima sa promenljivom gustinom.



Pošto je naš skup takav da postoje klase koje imaju veoma malo instanci, DBSCAN je zanimljiv pristup klasterovanja ovog skupa jer u zavisnosti od parametara koje prosledimo, algoritam može da posmatra određene instance kao šum jer ih ima jako malo u okruženju (neretko samo i jedna). Naravno, da bismo ovo postigli, morali smo da se poigramo sa parametrima za *epsilon* vrednost, kao i za minimalni broj instanci u okolini neke tačke da bi se ta tačka proglašila jezgrom.

Pošto su podaci skalirani radi bolje vizualizacije, razmak između njih je manji pa smo za *epsilon* vrednost uzeli raspon od 0.05 do 0.13 sa korakom 0.01. Za vrednosti minimalnog broja instanci u okruženju smo uzeli raspon od 10 do 50 sa korakom 10.

Na slikama ispod možemo videti vizualizaciju klasterovanja DBSCAN modela i poređenje klastera u zavisnosti od parametara koji su korišćeni, kao i poređenje mere siluete u zavisnosti od broja klastera koji su nađeni.



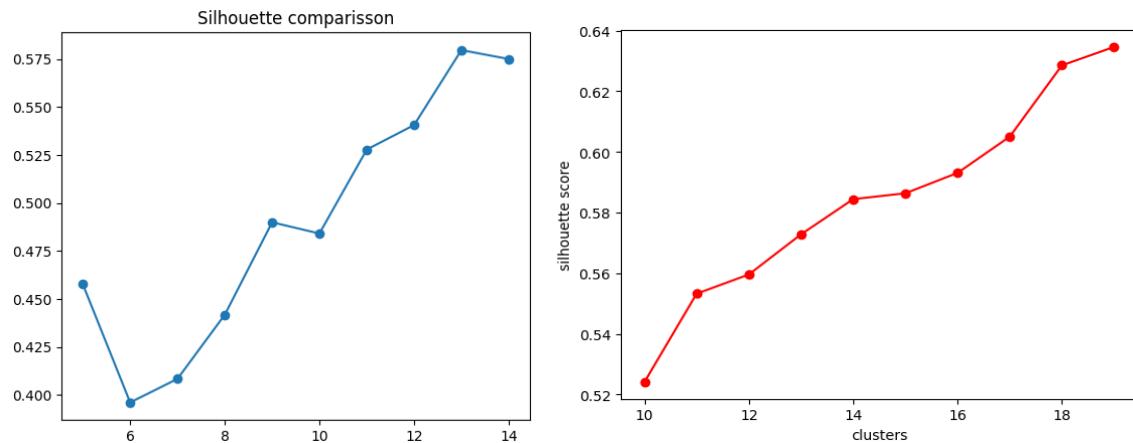
Grafik zavisnosti siluete od epsilon vrednosti nam nagoveštava da je klasterovanje uspešnije i smislenije izvršeno (bar po DBSCAN algoritmu) za male *epsilon* vrednosti (0.05, 0.06). Takođe, model za veće vrednosti parametra *min_samples*, nalazi veći broj šumova. Analizirajmo da li ovo ima smisla za naš skup podataka.

Epsilon vrednost predstavlja razdaljinu ili radijus koja definiše poluprečnik unutar kojeg se traže susedne tačke. Što je epsilon veći, znači da će veći broj manjih klastera pripasti jednom koji će se prikazivati kao jedan veći. Ovo možemo da primetimo na slici iznad gde, na primer, u prvom ili drugom redu, klasteri koji su obojeni kao zasebni, zaista i deluju kao zasebni, dok u poslednjem redu, vidimo da su vizualno mali klasteri pridruženi većim i da se crtaju kao delovi jednog većeg klastera. Zapravo možemo uvideti korelaciju između ovoga, kao i uporediti sa onim što mi sami možemo da vidimo. Instance koje pripadaju klasama koje se retko (ili ređe) javljaju, su okarakterisane kao šumovi, dok je obezbeđeno da one instance kojih ima dovoljno da se smeste u jedan mali klaster ne postanu deo većeg klastera. Time, ustvari, i dobijamo najrealniju sliku klastera nad našim skupom harmonija u horalima.

Poređenje modela klasterovanja

Videli smo nekoliko modela kojima smo pokušali da klasterujemo harmonije zabeležene u Bahovim horalima. Neki od njih su bili uspešni onoliko koliko je to bilo dozvoljeno vremenskim ograničenjima (primer *K-Sredina*), dok su neki imali mogućnost da nam prikažu klastera na način na koji nismo očekivali da podaci budu grupisani (primer *DBSCAN*).

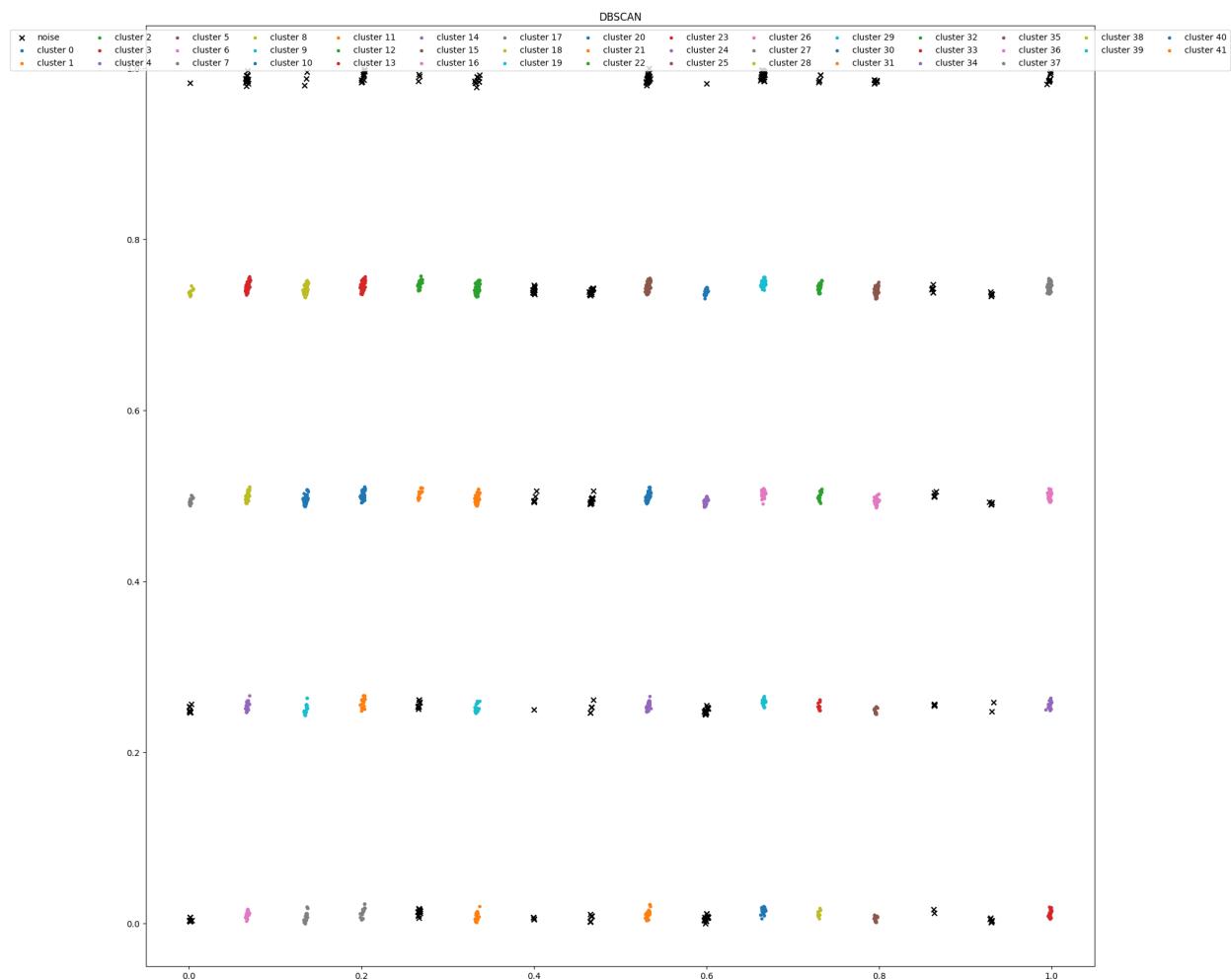
Pošto algoritmi *K-Sredina* i *Hijerarhiskog klasterovanja* oba imaju kao parametar broj klastera koji smo izabrali da nam nalazi u zavisnosti od mere siluete, pa je i logično da ova dva modela uporedimo po tome. Na slikama ispod možemo jednu pored druge videti vrednost siluete za isti broj klastera modela *K-Sredina* (levo) i *Hijerarhiskog klasterovanja* (desno).



Možemo videti sa slike da *Hijherarhijsko klasterovanje* ostvaruje za nijansu bolje vrednosti silueta za iste brojeve klastera. Samo ovo poređenje nam, naravno, nije dovoljno da bismo definitivno favorizovali jedan model u odnosu na drugi.

Što se tiče modela *DBSCAN*, nemamo neki preterano dobar način da ga uporedimo sa prethodnim, jer nemamo broj klastera koji je zadat, budući da DBSCAN algoritam sam pronalazi broj klastera u zavisnosti od parametara koji su mu dati za *epsilon* i *min_samples*. Ono što možemo reći je to da DBSCAN, za dobre *eps* i *min_samples* vrednosti, klasteruje vrednosti približno onako kako bismo i mi to uradili, sa dodatkom da neke celokupne klase gleda kao šumove, što ako malo bolje razmislimo, u realnosti i ima smisla. Klase koje se pojavljuju izuzetno malo puta (ili drastično manji broj puta od ostalih), u realnosti mogu da se posmatraju kao neki šumovi.

Sa ovim na umu, verovatno nije ni pogrešno da kažemo da, uz sve specifičnosti skupa podataka, model DBSCAN vrši najprirodnije kasterovanje istih.



Pravila pridruživanja

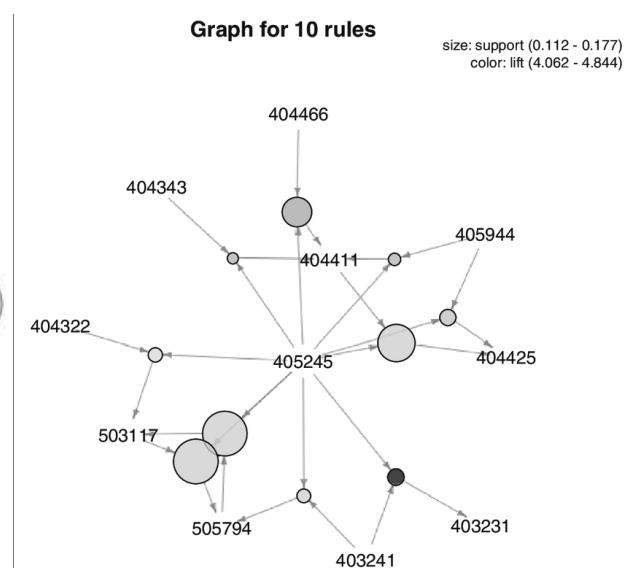
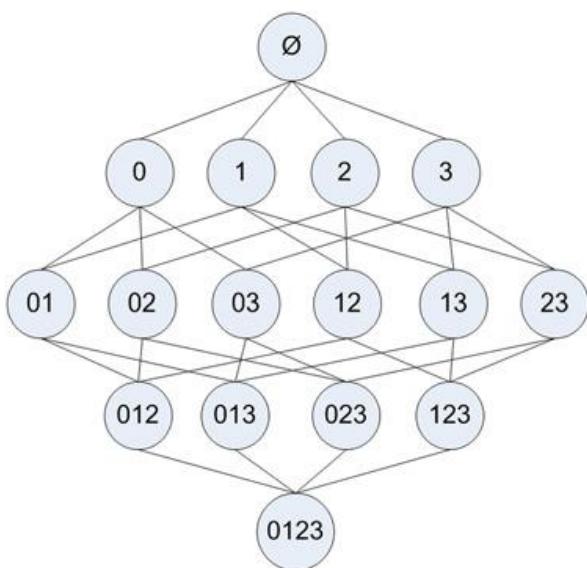
Uvod

Pravila pridruživanja je tehnika istraživanja podataka koja se koristi za otkrivanje zanimljivih veza ili asocijacija među elementima u velikim skupovima podataka. Fokusira se na pronalaženje obrazaca ili pravila koja opisuju zajedničko pojavljivanje ili zavisnost između različitih elemenata u transakcionej bazi podataka ili skupu transakcija (uglavnom u transakcionom modelu).

Jedan od najbitnijih pojmove u tehnici pravila pridruživanja jeste podrška. Podrška elementnog skupa je proporcija transakcija u kojima se elementni skup pojavljuje. Podrška se računa kako bi se identifikovali skupovi, koji se često zajedno pojavljuju u skupu podataka. Prag podrške je unapred definisan od strane korisnika.

Iz čestih elementnih skupova generišu se asocijativna pravila tako što se svaki elementni skup dekomponuje na dva neprazna podskupa, poznata kao "antecedent" (leva strana) i "consequent" (desna strana) pravila. Asocijativna pravila su oblika "ako leva strana, onda desna strana" i opisuju odnose između elemenata.

Generisana pravila se evaluiraju na osnovu različitih metrika kao što su pouzdanost (*confidence*), podizanje (lift) i podrška (*support*).



Preprocesiranje

Kao što smo već napomenuli gore, pravila pridruživanja se uglavnom primenjuju na transakcionim tipovima podataka, kao što su dobro poznati primeri potrošačke korpe. Pošto struktura našeg skupa podataka jeste manje više u transakcionom modelu, a u to se možemo uveriti na slici ispod, nema neke preterane potrebe za preprocesuiranjem našeg skupa za primenu pravila pridruživanja.

	C	C#/Db	D	D#/Eb	E	F	F#/Gb	G	G#/Ab	A	A#/Bb	B	Bass	Meter	Chord label
1	YES	NO	NO	NO	NO	YES	NO	NO	NO	YES	NO	NO	F		3F_M
2	YES	NO	NO	NO	YES	NO	NO	YES	NO	NO	NO	NO	E		5C_M
3	YES	NO	NO	NO	YES	NO	NO	YES	NO	NO	NO	NO	E		2C_M
4	YES	NO	NO	NO	NO	YES	NO	NO	NO	YES	NO	NO	F		3F_M
5	YES	NO	NO	NO	NO	YES	NO	NO	NO	YES	NO	NO	F		2F_M
6	NO	NO	YES	NO	NO	YES	NO	NO	NO	YES	NO	NO	D		4D_m
7	NO	NO	YES	NO	NO	YES	NO	NO	NO	YES	NO	NO	D		2D_m
8	YES	NO	NO	NO	NO	YES	NO	NO	NO	YES	NO	NO	A		3F_M
9	YES	NO	NO	NO	NO	YES	NO	NO	NO	YES	NO	NO	A		2F_M
10	NO	NO	YES	NO	NO	YES	NO	NO	NO	NO	YES	NO	Bb		5BbM
11	NO	NO	YES	NO	NO	YES	NO	NO	NO	NO	YES	NO	Bb		1BbM
12	YES	NO	YES	NO	NO	YES	NO	NO	NO	YES	YES	NO	C		2BbM
13	YES	NO	YES	NO	NO	YES	NO	YES	NO	NO	YES	NO	C		1BbM
14	NO	NO	YES	NO	NO	YES	NO	NO	NO	NO	YES	NO	D		3BbM
15	NO	NO	YES	NO	YES	YES	NO	YES	NO	NO	YES	NO	E		2BbM
16	YES	NO	NO	NO	NO	YES	NO	NO	NO	YES	NO	NO	F		4F_M
17	YES	NO	NO	NO	NO	YES	NO	NO	NO	YES	NO	NO	F		3F_M
18	NO	NO	YES	NO	NO	YES	NO	NO	NO	NO	YES	NO	Bb		5BbM
19	NO	NO	YES	NO	NO	YES	NO	YES	NO	NO	YES	NO	Bb		2BbM
20	NO	NO	YES	NO	NO	YES	NO	NO	NO	YES	YES	NO	Bb		1BbM

Naravno, izbacili smo atribute koji nam nisu od značaja kao što su *Choral ID*, i *Event number*. Ono što jesmo uradili je da smo u SPSS modeleru, u čvoru koji učitava podatke, namestili da su uloge svih atributa i ulaz (*input*) i ciljni atribut (*target*), osim atributa *Chord label*, koji je postavljen samo kako ciljni atribut. Ovako označeni atributi i skup podataka može u neki od algoritama pravila pridruživanja, a mi smo se ovde osvrnuli na dva:

1. Apriori algoritam
2. FP-Growth algoritam

Opis modela

Apriori algoritam

Apriori algoritam radi iterativno i prati "apriori" princip, koji kaže da bilo koji podskup čestog skupa stavki takođe mora biti čest. Algoritam koristi strategiju pretrage po širini (*breadth-first search*) za istraživanje prostora skupova stavki i efikasno pronalaženje čestih skupova stavki.

Apriori princip, takođe poznat kao i princip zatvorenosti nadole, je osnovni koncept asocijativnih pravila. Baziran je na posmatranju da, ako je jedan skup stavki redak (ne zadovoljava prag minimalne podrške), tada će svi njegovi nadskupovi (veći skupovi stavki koji ga sadrže) takođe biti retki.

U čvoru Apriori algoritma podešavamo neke od parametara. Ostavili smo da model koristi predefinisane uloge (*roles*), odnosno ostavili smo modelu da sam odluči šta će da koristi kao *antecedent* i *consequent*.

U odeljku "Model" smo ugasili opciju da koristi particionisane podatke i podesili minimalnu podršku za levu stranu pravila (*antecedent*), minimalnu pouzdanost pravila, i na kraju, maksimalni broj atributa u levoj strani. Sa ovim parametrima smo eksperimentisali, a oni koju su nam dali zanimljive rezultate su sledeći:

1. Minimalna podrška leve strane : 2%
2. Minimalna pouzdanost pravila : 80%
3. Maksimalni broj atributa u levoj strani : 10

Pogledajmo sada neka pravila koje nam je algoritam *Apriori* našao, i analizirajmo njihov smisao u muzici.

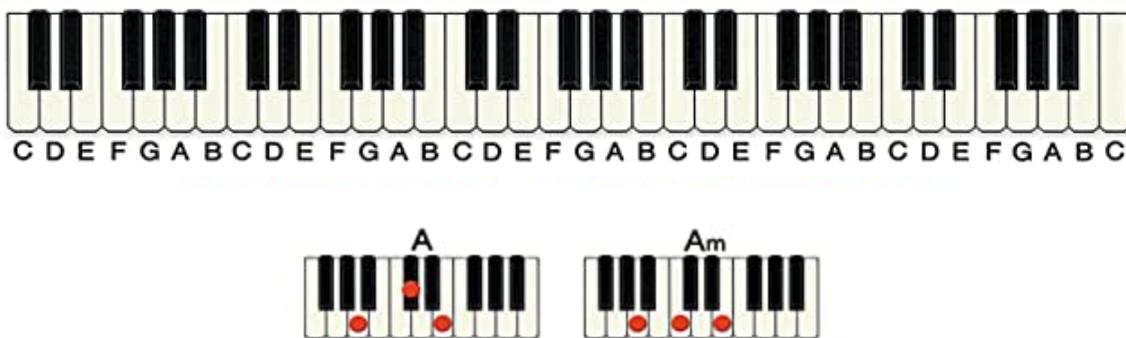
Consequent	Antecedent	Support %	Confidence %	Lift
D	Bass = D	12.162	99.129	2.374
A	Bass = A	12.145	99.419	2.371
G	Bass = G	12.021	99.559	2.633
E	Bass = E	11.968	99.705	2.658
C	Bass = C	10.591	98.5	3.117
F	Bass = F	8.155	97.835	4.316
F#/Gb	Bass = F#	8.049	99.342	3.986
B	Bass = B	7.926	98.441	3.114
G	Bass = G D	7.82	99.323	2.627
D	Bass = D A	7.538	99.063	2.373
C	Bass = C G	7.22	98.533	3.118

Ako pogledamo pravila sortirana po podršci, vidimo da nam algoritam daje povezanosti bas note sa notom koja se poklapa sa njim, a nalazi se u jednoj odsviranoj harmoniji. Ovo zapravo ima smisla jer ako se nota koja predstavlja najniži ton (bas ton) odsvira, velika je verovatnoća da će taj ton u drugoj oktavi da bude odsviran i time se doprinosi stvaranju akorda ili harmonije.

Consequent	Antecedent	Support %	Confidence %	Lift
Chord label = A_m	Bass = A C E A	2.365	85.075	18.68
Chord label = A_m	Bass = A C E	2.383	84.444	18.542
Chord label = BbM	Bass = Bb F D	3.054	94.22	17.108
Chord label = BbM	Bass = Bb A#/Bb F D	3.054	94.22	17.108
Chord label = E_M	G#/Ab Bass = E B	3.16	86.592	16.629
Chord label = E_M	G#/Ab Bass = E B E	3.16	86.592	16.629

Pravila sortirana po podizanju (*lift*) nam više govore o formiraju same harmonije odnosno akorda u odnosu na tonove koji su prisutni. Na primer, prvo pravilo nam govori da ako su tonovi A - C - E prisutni, i da je pritom bas ton A, onda je naziv tog akorda A_m (*A-Mol*).

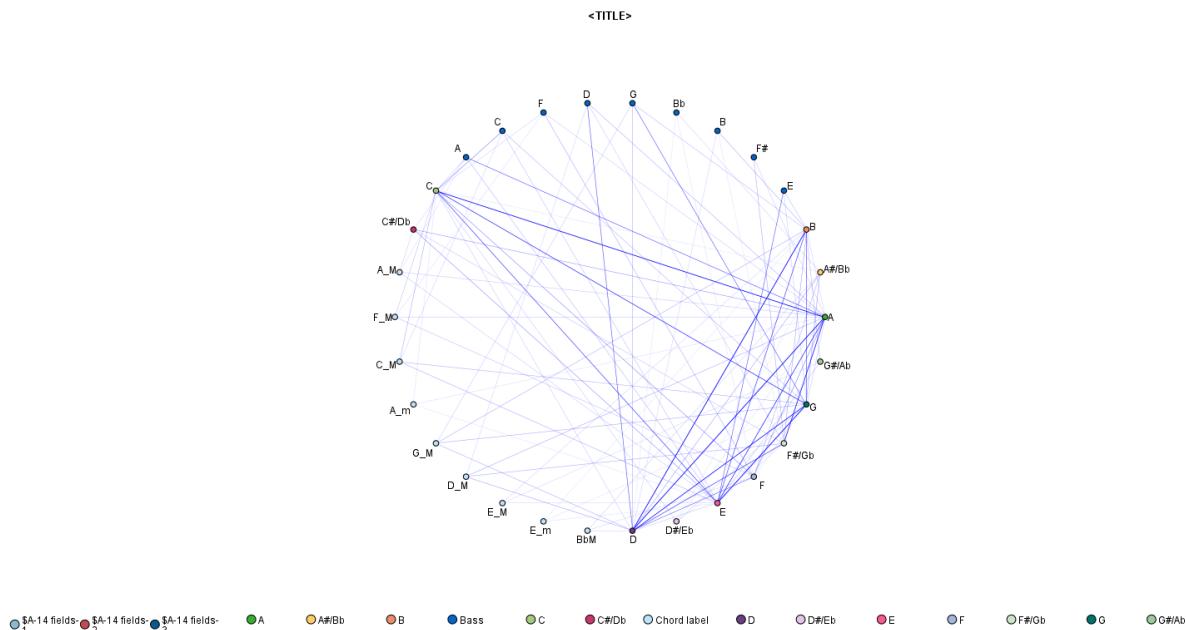
Ako ovo pogledamo na slici na kojoj su obeležene dirke klavira i uporedimo sa akordom A-Mol, možemo se uveriti da ovo zaista jeste tačno.



Istu stvar možemo pogledati i za poslednje pravilo na slici gore koje kaže da, ako su tonovi E - G# - B³, a bas ton E, onda je naziv akorda E_M (E-Dur/E-Major).



Ukupan broj pravila pridruživanja generisanih *Apriori* algoritmom je 106, i pokušaćemo da ih vizualizujemo iscrtavanjem mreže preko "Web" čvora u SPSS-u.



Jače iscrtane linije su bitnija pravila, pa tako možemo za većinu nota videti da je jaka linija povučena do istoimenog bas tona. Recimo, ton A i ton C se često javljaju zajedno, što naravno, ima smisla jer je na klaviru ton C, cela dva stepena viši od tona A. Ovo se naziva terca i ima veliku primjenost u formiranju harmonija ili akorda.

Model sa *Apriori* algoritmom nam je dao neke od bitnijih pravila pridruživanja, za koje smo, analizom i vizualizacijom, videli da imaju smislena objašnjenja u muzici i pravljenju harmonija i akorda o kojima mi upravo i pričamo.

³ Podaci i slike su obeleženi po američkom standardu pa se nota koja prethodi noti C, označava ko B, a kod nas bi to bila nota H.

FP-Growth algoritam

FP-Growth algoritam je efikasan i skalabilan algoritam koji se koristi za nalaženje čestih skupova stavki u procesima istraživanja podataka i asocijativnih pravila. On predstavlja alternativu Apriori algoritmu i posebno je koristan za istraživanje velikih skupova podataka.

Ključna ideja iza FP-Growth algoritma je izgradnja kompaktnih podataka strukture nazvane FP-Stabla, koja predstavlja skup podataka u kompresovanom obliku. Ova struktura stabla omogućava efikasno i brzo nalaženje čestih skupova stavki. Algoritam koristi pristup "podeli i vladaj" (*divide and conquer*) rekurzivno istražujući uslovna FP-Stabla kako bi pronašao sve česte skupove stavki.

Opšti koraci FP-Growth algoritma su:

1. Konstrukcija FP-Stabla:
2. Izgradnja uslovne baze šablonu
3. Rekurzivno rudarenje čestih skupova stavki

U poređenju sa Apriori algoritmom, koji generiše kandidatske skupove stavki i više puta pregleda skup podataka, FP-Growth algoritam ima prednost jer zahteva samo dva prolaska kroz podatke. Posebno je efikasan kada je broj čestih skupova stavki relativno mali u odnosu na veličinu skupa podataka.

Ostavili smo da model koristi predefinisane uloge atributa, maksimalni broj uslova u jednom pravilu je stavljen na 5, a maksimalni broj pravila globalno 1000. Meru smo postavili na podizanje (*lift*). Pogledajmo statistike za pravila i za najčešće atrIBUTE.

Rule Statistics

Measurements	Minimum	Maximum	Mean	Standard Deviation
Condition Support (%)	5.08	41.92	16.77	11.20
Confidence (%)	14.02	100.00	56.55	28.46
Rule Support (%)	5.01	12.07	7.03	1.94
Lift	2.00	11.12	4.05	1.96
Deployability (%)	0.00	36.05	9.74	10.53

Information for Most Frequent Items			
Item name	Records (%)	Conditions (%)	Predictions (%)
A	41.92	24.47	3.19
D	41.75	22.34	3.19
G	37.81	19.15	4.26
E	37.51	18.09	3.19
B	31.62	13.83	4.26
C	31.60	15.96	5.32
F#/Gb	24.92	8.51	3.19
F	22.67	10.64	4.26
A#/Bb	18.02	3.19	2.13
C#/Db	16.84	3.19	1.06
Bass = D	12.16	3.19	3.19
Bass = A	12.14	3.19	3.19
Bass = G	12.02	8.51	5.32
Bass = E	11.97	3.19	3.19
G#/Ab	11.63	1.06	1.06
Bass = C	10.59	3.19	3.19
Chord label = D_M	8.88	0.00	7.45
Chord label = G_M	8.63	0.00	9.57
Chord label = C_M	8.61	0.00	7.45
Bass = F	8.16	4.26	3.19
Bass = F#	8.05	2.13	2.13
Bass = B	7.93	1.06	1.06
Chord label = F_M	6.87	0.00	7.45
Chord label = A_M	6.21	0.00	6.38
Chord label = BbM	5.51	0.00	1.06
Bass = Bb	5.10	1.06	1.06

Primetimo da se u pravilima leve strane (*conditions*), imena akorda ne javljaju nikada, što je nekako i logično s obzirom na to da pravila pridruživanja uglavnom idu od tonova, i njima pridružuje akord koji im odgovara. Shodno tome, vidimo da su akordi najviše zastupljeni u desnoj stvari pravila (*Predictions*).

Pogledajmo sada neka pravila generisana FP-Growth algoritmom sortirana po različitim merama redom, pouzdanost, podizanje i podrška.

Most Interesting Rules by Confidence

Rank	Rule ID	Condition	Prediction	Other Evaluation Statistics				
				Sorted By Confidence(%)	Condition Support (%)	Rule Support (%)	Lift	Deployability (%)
1	40	A Bass = F#	F#/Gb	100.00	5.49	5.49	4.01	0.00
2	59	G Bass = E	E	100.00	5.68	5.68	2.67	0.00
3	60	B Bass = E	E	100.00	6.65	6.65	2.67	0.00
4	76	F#/Gb Bass = D	D	100.00	5.68	5.68	2.40	0.00
5	61	Bass = E	E	99.71	11.97	11.93	2.66	0.04
6	77	C Bass = A	A	99.68	5.44	5.42	2.38	0.02
7	17	Bass = Bb	A#/Bb	99.65	5.10	5.08	5.53	0.02
8	65	Bass = G	G	99.56	12.02	11.97	2.63	0.05
9	68	B Bass = G	G	99.47	6.71	6.67	2.63	0.04
10	81	Bass = A	A	99.42	12.14	12.07	2.37	0.07
11	41	Bass = F#	F#/Gb	99.34	8.05	8.00	3.99	0.05
12	69	D Bass = G	G	99.32	7.82	7.77	2.63	0.05

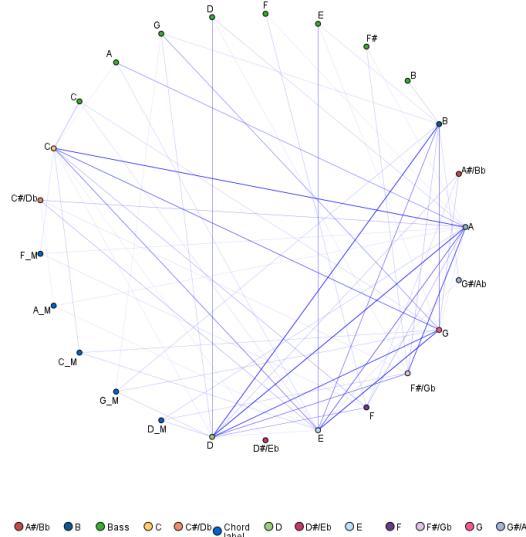
Most Interesting Rules by Lift

Rank	Rule ID	Condition	Prediction	Other Evaluation Statistics				
				Sorted By Lift	Condition Support (%)	Confidence (%)	Rule Support (%)	Deployability (%)
1	1	A C F	Chord label = F_M	11.12	7.10	76.37	5.42	1.68
2	2	G E C	Chord label = C_M	9.14	8.37	78.69	6.58	1.78
3	3	A C#/Db	Chord label = A_M	9.06	9.53	56.30	5.37	4.17
4	4	A D F#/Gb	Chord label = D_M	8.67	9.29	77.00	7.15	2.14
5	5	D G B	Chord label = G_M	8.65	8.70	74.65	6.50	2.21
6	6	C F	Chord label = F_M	8.45	10.22	58.03	5.93	4.29
7	7	E C#/Db	Chord label = A_M	8.13	9.92	50.53	5.01	4.91
8	8	A F	Chord label = F_M	8.11	10.36	55.71	5.77	4.59
9	9	D F#/Gb	Chord label = D_M	6.51	13.26	57.79	7.66	5.60
10	10	E C	Chord label = C_M	6.49	13.26	55.93	7.41	5.84
11	11	A F	Bass = F	6.31	10.36	51.45	5.33	5.03
12	12	G B	Chord label = G_M	6.16	13.54	53.19	7.20	6.34

Most Interesting Rules by Rule Support

Rank	Rule ID	Condition	Prediction	Other Evaluation Statistics				
				Sorted By Rule Support(%)	Condition Support (%)	Confidence (%)	Lift	Deployability (%)
1	81	Bass = A	A	12.07	12.14	99.42	2.37	0.07
2	82	A	Bass = A	12.07	41.92	28.80	2.37	29.85
3	78	D	Bass = D	12.06	41.75	28.88	2.37	29.69
4	79	Bass = D	D	12.06	12.16	99.13	2.37	0.11
5	65	Bass = G	G	11.97	12.02	99.56	2.63	0.05
6	66	G	Bass = G	11.97	37.81	31.65	2.63	25.84
7	61	Bass = E	E	11.93	11.97	99.71	2.66	0.04
8	62	E	Bass = E	11.93	37.51	31.81	2.66	25.58
9	50	Bass = C	C	10.43	10.59	98.50	3.12	0.16
10	51	C	Bass = C	10.43	31.60	33.02	3.12	21.17
11	87	A D	F#Gb	9.29	16.54	56.14	2.25	7.26
12	85	D	Chord label = D_M	8.49	41.75	20.34	2.29	33.26

Sa slike se vidi da su pravila prudruživanja koja su generisana, iako sa drugačijim ocenama, uglavnom ista ili vrlo slična kao pravila generisana *Apriori* algoritmom. Pravila sa najvećim podizanjem su ona koja nam povezuju odsvirane tonove sa nazivom akorda (harmonije), dok nam pravila sa najvećom podrškom govore da, ako je prisutan određeni bas ton, da će postojati isti taj ton u samom akordu koji će biti odsviran.



Na slici iznad možemo videti vizualnu reprezentaciju pravila u vidu mreže povezanosti.

Kao i kod *Apriori* algoritma, vidimo da je mreža prilično slična, sa tim što je ova mreža malo ređa. Ovo naravno nije nikakvo merilo efikasnosti, iako FP-Growth jeste efikasniji od *Apriori* algoritma, već smo samo hteli da pokažemo ona pravila koja su najjača radi lakšeg vizualnog poređenja sa *Apriori* algoritmom.

Poređenje modela pravila pridruživanja

Iako nemamo baš neku realnu meru pomoću koje bismo uporedili modele koje smo obrađivali, možemo reći da smo videli da oba algoritma generišu smislena pravila, i da vrlo slično rade.

Ako bismo uporedili brzinu izvršavanja, tu je jasno da se FP-Growth model izvršava brže nego Apriori, što je bilo i rečeno kada smo govorili o karakteristikama FP-Growth algoritma. Ono što bismo takođe mogli uzeti kao kriterijum poređenja je to da je Apriori algoritam izgenerisao pravila koja imaju veću meru podizanja (*lift*). S obzirom na to da smo pronašli ista pravila u oba modela, ovo poređenje nam i dalje nije dalo neki bitan rezultat.

Sve u svemu, oba algoritma su dobar izbor za posmatranje odnosa između atributa, i međusobno pridruživanje skupova istih.

Zaključak

Ovaj projekat istraživanja podataka fokusiran je na analizu skupa koji sadrži harmonije iz horala koje je komponovao J.S. Bah. Projekat je obuhvatio različite tehnike, uključujući modele klasifikacije, modele klasterovanja i modele asocijativnih pravila.

Za zadatak klasifikacije, korišćena su tri modela: slučajna šuma, naivni Bajes (dve varijante) i stablo odlučivanja. Među njima, model slučajne šume se pokazao kao najbolji izbor. Demonstrirao je veću tačnost i pouzdanost u klasifikaciji harmonija, čime je dokazao svoju efikasnost na ovom specifičnom skupu podataka.

Prelazeći na modele klasterovanja, u projektu su korišćeni K-sredina, hijerarhijsko klasterovanje i DBSCAN. Među njima, DBSCAN se pokazao kao najpogodniji izbor zbog svoje sposobnosti da efikasno obradi šumove. Uspešno je identifikovao i grupisao harmonije, istovremeno efikasno otkrivajući i uzimajući u obzir prisustvo klasa sa veoma malim brojem instanci u skupu podataka.

Za fazu istraživanja podataka metodom asocijativnih pravila korišćena su dva modela: Apriori i FP-Growth. Oba modela su se istakla u otkrivanju značajnih asocijativnih pravila u skupu podataka. Međutim, FP-Growth model je bio bolji od Apriori modela u smislu brzine i efikasnosti, što ga čini preferiranim izborom za veće skupove podataka.

U celini, projekat je pokazao primenljivost i efektivnost tehnika istraživanja podataka u izvlačenju vrednih uvida iz skupa podataka sa harmonijama J.S. Baha. Model slučajne šume se pokazao kao najtačniji klasifikator, DBSCAN se istakao u identifikaciji izuzetaka, a FP-Growth model je demonstrirao brzinu u otkrivanju asocijativnih pravila. Ovi rezultati mogu doprineti dubljem razumevanju Bahovih kompozicija i potencijalno pomoći muzikolozima, kompozitorima i zaljubljenicima u daljem istraživanju i analizi njegovih dela, kao i mogućnostima veštačke inteligencije da, uz znanje o harmonijama i samoj strukturi Bahovih horala, samostalno generiše muzičke kompozicije u Bahovom stilu što je to moguće preciznije i lepše.