Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Mušič

Razlaga klasifikatorjev na podlagi podkonceptov

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: prof. dr. Marko Robnik Šikonja

Ljubljana, 2023

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Kandidat: Nejc Mušič
Naslov: Razlaga klasifikacijskih problemov na podlagi podkonceptov
Vrsta naloge: Diplomska naloga na univerzitetnem programu prve stopnje Računalništvo in informatika
Mentor: prof. dr. Marko Robnik Šikonja

Opis:
Besedilo teme diplomskega dela študent prepiše iz študijskega informacijskega sistema, kamor ga je vnesel mentor. V nekaj stavkih bo opisal, kaj pričakuje od kandidatovega diplomskega dela. Kaj so cilji, kakšne metode naj uporabi, morda bo zapisal tudi ključno literaturo.

Title: Explanation of Classifiers Based on Subconcepts

Description:

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
MDEC	multidiversified ensemble clustering	Večdimenzionalno multidiversificirano združevanje v gruče
XAI	explainable artificial intelligence	razložljiva umetna inteligenca
SHAP	SHapley Additive exPlanations	Shapleyjeve aditivne razlage
DBSCAN	Density-Based Spatial Clustering of Applications with Noise	enako ime
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise	enako ime
XGBoost	Extreme Gradient Boosting	enako ime
DBCV	Density-Based Clustering Validation	enako ime
t-SNE	t-distributed Stochastic Neighbor Embedding	enako ime
PCA	Principal component analysis	enako ime
UMAP	Uniform Manifold Approximation and Projection	enako ime
MNIST	Modified National Institute of Standards and Technology	enako ime

Povzetek

Naslov: Razlaga klasifikatorjev na podlagi podkonceptov

Avtor: Nejc Mušič

Pri nekaterih klasifikacijskih problemih je poleg natačnosti in zanesljivosti pomembna tudi razlaga odločitev (npr. v medicini). Ker imajo ponavadi kompleksni modeli, kot so na primer nevronske boljše, boljše rezultate jih velikokrat raje vzamemo kot npr. linearno regresijo, ki ima lastnost, da je dobro razložljiva. Diplomska naloga se ukvarja s tem, da bi takšne kompleksne modele lahko razložili. Na podlagi gručenja podatkov razložimo klasifikator s pomočjo odločitvenih pravil, medoidov gruč in s SHAP vrednostmi.

Ključne besede: razložljiva umetna inteligenca, MDEC, DBSCAN, HDBSCAN, KMEANS, SHAP, odločitvena pravila, medoid/prototip.

Abstract

Title: Explanation of Classifiers based on Subconcepts

Author: Nejc Mušič

In some classification problems, in addition to accuracy and reliability, the explanation of decisions is also important (e.g., in medicine). Complex models, such as neural networks, often outperform simpler models like linear regression, which are inherently interpretable. My thesis addresses the challenge of explaining such complex models. Based on data clustering, we explain the classifier using decision rules, cluster medoids, and SHAP values.

Keywords: XAI, MDEC, DBSCAN, HDBSCAN, KMEANS, SHAP, decision rules, medoid/prototype.

Poglavje 1

Uvod

V zadnjem času smo bili priča dobrim dosežkom umetne inteligence na področju strojnega učenja. Kompleksni modeli pri klasifikacijskih problemih omogočajo izjemno natančnost pri napovedovanju rezultatov. Vendar se s tem pojavlja izziv nerazložljivosti "black box" modelov, pri katerih je težko razumeti, kako in zakaj so sprejeli določene odločitve. Za primer lahko vzamemo globoke nevronske mreže ali kompleksne ansamble (XGBoost), ki imajo na tisoče ali celo milijone parametrov. To pomeni, da so odločitveni procesi izjemno zapleteni, kar otežuje razumevanje, kako določen vhod vpliva na izhod. Z diplomsko nalogo želimo prispevati k razvoju razlage kompleksnih modelov ter omogočiti boljše razumevanje njihovega delovanja, kar bo pripomoglo k dodatni izboljšavi modelov. Prav tako je razumevanje odločitev modelov ključno za zaupanje v njihove rezultate ter za širšo sprejetost uporabe umetne inteligence v različnih kritičnih domenah (npr. medicina, varnost računalniških omrežij, letalska in avtomobilska industrija). Postopek razlage je naslednji. Klasifikator je naučen na učni množici, na kateri poženemo algoritme za gručenje. Za avtomatski izbor parametrov uporabimo hevristiko silhuete in DBCV indeks. Pri izboru najboljšega gručenja si bomo pomagali z vizualizacijami podatkov v dvodimezionalnem prostoru (za to bomo uporabili algoritem T-SNE pri visokodimenzionalnih podatkih). Predpogoj razlage je ločenost podatkov na gruče, ki so medsebojno dobro ločljive

in prisotnost posameznega razreda v dveh ali več različnih gručah. Ko bomo imeli označene gruče, bomo posamezno gručo razložili z odločitvenimi pravili po konceptu ena gruča proti vsem ostalim. Nato bomo preverili prisotnost razredov v gruči. V primeru, da je gruča homogena (primeri pripadajo enemu razreda), bomo posamezno gručo predstavili z medoidom. V primeru, da je gruča heterogena bomo znotraj gruče uporabili odločitvena pravila, ki bodo ločila primere različnih razredov. Dodatno bomo izračunali SHAP vrednosti za vsako gručo in jih predstavlili z grafom. SHAP vrednosti nam bodo razložile pomembnost atributov za naš klasifikator, kar nam dodatno pomaga pri interpretaciji odločitvenih pravil in medoidov. Sledi izbira naključnega primera iz testne množice, ki ga bomo z algoritmom za gručenje vmestili v eno izmed gruč in s tem razložili z zgornjimi metodami. Dodatno bomo izračunali SHAP vrednosti le za novi primer. V zadnjih letih so bili poskuseni številni pristopi za razlago kompleksnih modelov. Pristop s CBR (Case-Based Reasoning) rešitvijo je zanimil, saj najde pomembne značilke na podlagi lokalnih informacij in nato zbere primer, ki je bil pomemben za izgradnjo modela kot primer razlage [19]. Poskusi so bili tudi na področju odločitvenih dreves in induciranih pravilih, s katerimi dobimo vpogled v odločitvene meje v podatkih in poenostavljeno logiko [19]. SHAP vrednosti veljajo za dobro metodo razlage (pomembnost atributov), zato so bili tudi uporabljeni na raznih področjih kot je na primer NLP (Natural Language Processing) [7]. Članek [7] dobro kategorizira probleme v zvezi z razlogo klasifikatorjev. Prav tako razdeli razlage glede na to, katero vrsto "black box"modela poskuša razložiti in poda širši pregled področja. Dodatna sorodna dela so našteta v podpoglavju 4.6 (na koncu pri primerjavi z obstoječimi razlagami). V diplomski nalogi bomo predstavili naslednja poglavja: za uvodom sledi predstavitev uporabljenih tehnologij in izbor parametrov za algoritme, nato opis pristopa s podskupinami, kjer bo predstavljen preprost ilustrativni primer razlage, zatem poglavje Evalvacija, kjer bodo predstavljene podatkovne množice, potem primerjava algoritmov gručenja na umetnih podatkovnih množicah in realni podatkovni množici MNIST, sledijo ocene smiselnosti prototipov (me-

doidov), pravil in poglavje o številu gruč, poglavje se konča z razlago realne podatkovne množice KDD99, kjer bodo podane še druge obstoječe razlage. Diplomska naloga se konča z zaključkom, kjer bomo na kratko povzeli dobljene rezultate opisanih metod. Izpostavili bomo dobre in slabe (omejitve pristopa) lastnosti ter podali mnenje o nadaljnih raziskavah na tem področju.

Poglavje 2

Predstavitev tehnologij

2.0.1 Algoritmi za gručenje

MDEC

Algoritem "Multidiversified Ensemble Clustering"je zasnovan gručenje visokodimenzionalnih podatkov. Združuje več rešitev gručenja za izboljšanje splošne uspešnosti gručenja. Algoritem MDEC ustvari veliko število raznolikih metrik z naključno skalirano eksponentno funkcijo podobnosti in jih poveže z naključnimi podprostori. S tem dobimo pare metrika-podprostor. Na osnovi matrik podobnosti, pridobljenih iz metrika-podprostor parov, nato sestavi ansambel raznolikih združenj v gruče. Zatem uporabi kriterij zasnovan na entropiji, za oceno raznolikosti združevanj glede na gruče, na osnovi katerega uporabi tri specifične algoritme ansambelskega združevanja (čonsensus functions"). Več informacij o algoritmu je na voljo v članku [9]. Pri algoritmu smo opredelili le parameter za število skupin.

KMEANS

K-means algoritem je iterativni algoritem, ki poskuša razdeliti nabor podatkov v K predhodno določenih neprekrivajočih se podskupin (gruč), pri čemer vsaka podatkovna točka pripada le eni skupini. Algoritem iterativno dodeljuje primere najbližjemu centroidu in zatem premika centroide. Več

informacij je na voljo v članku [8]. Pri algoritmu smo opredelili le parameter za število skupin.

DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) je algoritem za gručenje, ki temelji na osnovi gostote podatkovnih točk v prostoru. Algoritem je sposoben zaznati gruče v podatkih, ki imajo različne oblike ter hkrati zaznati osamelce (šum). Več informacij o algoritmu in njegovih izpeljankah je na voljo v članku [10]. Pri uporabi algoritma je ključna izbira dveh parametrov - epsilon in min samples. Članek [17] predlaga vrednost min samples = 2 * dim, kjer je dim dimenzija podatkovne množice. Za vrednost parametra epsilon je predlagana tehnika, ki izračuna povprečno razdaljo med vsako točko in njenimi k najbližjimi sosedi, kjer je k enak vrednosti min samples. Povprečne k-razdalje nato prikažemo v naraščajočem vrstnem redu na grafu. Optimalna vrednost za epsilon se nahaja pri točki največje ukrivljenosti (tj. kjer ima graf največji nagib). Največji nagib na grafu smo našli s pomočjo algoritma Kneed, ki je predstavljen v članku [11]. Pri obeh parametrih smo poskusili tudi okoliške vrednosti, ki se nahajajo okoli vrednosti parametrov min samples in epsilon.

HDBSCAN

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) izvaja algoritem DBSCAN pri različnih vrednostih epsilon in združi rezultate, da najde gručenje, ki zagotavlja najboljšo stabilnost pri različnih vrednostih epsilon. To omogoča HDBSCAN-u, da najde gruče z različnimi gostotami (za razliko od DBSCAN-a) in je bolj odporen na izbiro parametrov [15]. Pri HDBSCAN algoritmu smo določili parametra min cluster size = (ix1*5) in min samples = (ix2+2)*5, kjer $ix1 \in [2, 10]$ in $ix2 \in [0, 8]$ in $ix_1, ix_2 \in \mathbb{N}$. Podoben postopek z drugimi vrednostmi je opisan v članku [4].

2.0.2 Hevristike za rezultate gručenja

Koeficient silhuete

Koeficient silhuete je metoda za ocenjevanje kakovosti gručenja, ki se osredotoča na merjenje, kako so točke v isti gruči podobne med seboj v primerjavi z drugimi gručami. Njegova glavna naloga je ponuditi kvantitativno merilo za oceno, kako dobro so točke znotraj iste gruče povezane in kako dobro so razmejene od drugih gruč. Za oceno smo uporabili hevristiko povprečne sihuete, ki se ne osredotoča na oceno kakovosti posameznih točk, kot to počne koeficient silhuete za vsako točko, temveč daje celostno sliko o rezultatu gručenja. Več informacij o izračunu v članku [20].

DBCV indeks

DBCV (Density-Based Clustering Validation) indeks je mera kakovosti gručenja, ki se uporablja za ocenjevanje učinkovitosti algoritmov za gručenje. Mera kot je koeficient silhuete je primerna za ocenjevanje gruč s sferičnimi oblikami, medtem ko algoritmi, ki temeljijo na gostoti podatkovnih točk velikokrat zaznajo nepravilne oblike, kjer se bolje odreže DBCV indeks. Izračun DBCV indeksa tudi temelji na gostoti gruč. DBCV indeks je podbrobno predstavljen v članku [16].

2.0.3 Klasifikatorji

XGBoost

XGBoost (Extreme Gradient Boosting) je zmogljiva in priljubljena metoda strojnega učenja. Gre za algoritem, ki kombinira moč odločitvenih dreves in tehnike gradientnega dviga za doseg visoke točnosti. XGBoost je kompleksen model in je dobro opisan v članku [18].

Logistična regresija

Logistična regresija je metoda, ki se pogosto uporablja v strojnem učenju za reševanje klasifikacijskih problemov. Kljub svojemu imenu se ne ukvarja z "regresijo", temveč se ukvarja z napovedovanjem verjetnosti, da bo določen vhodni primer spadal v en izmed dveh ali več razredov. Več o logistični regresiji je v članku [3].

2.0.4 Metode razlage

Odločitvena pravila

Kljub temu da so odločitvena pravila metoda strojnega učenja, smo jih uporabili za razlago gruč, saj so lahko razložljiva z obliko "IF conditions THEN response". Metoda, ki smo jo uporabili, iz ansambla odločitvenih dreves izvleče pravila z največjima vrednostima "Precision"in "Recall". Najboljša pravila smo nato uporabili za opis ene gruče proti ostalim. Uporabna so tudi za ločitev primerov, ki pripadajo različnim razredom v eni gruči (medoid težko pojasne tako gručo). Več podrobnosti o odločitvenih pravilih najdemo v tem članku [5].

Medoid

Za razlago posamezne gruče smo uporabili medoid, ki je izračunan na atributih primerov iz gruče. Gre za tipičen primer, ki enostavno opiše gručo. Pri uporabi razlage z medoidom moramo biti pazljivi na sestavo gruče. Gruča, ki je sestavljena iz dveh ali več razredov, ne bo dobro razložena z medoidom (tu si pomagamo z odločitvenimi pravili znotraj gruče).

SHAP

SHAP (SHapley Additive exPlanations) je metoda, ki se uporablja za razumevanje in razlaganje pomembnosti posameznih značilk v modelih strojnega učenja. Njeno ime izhaja iz koncepta Shapleyevih vrednosti v teoriji iger, kjer

se meri prispevek vsakega igralca k skupni vrednosti. SHAP vrednosti bomo za vsako gručo posebaj predstavili v obliki grafa. Z metodo SHAP bomo za izbran klasifikator izvedli pomembnost atributov, kar nam bo nadaljno pomagalo pri interpretaciji pravil in medoidov in s tem pri razumevanju posamezne gruče. Več o SHAP metodi najdemo v članku [13].

2.0.5 Zmanjšanje dimenzionalnosti

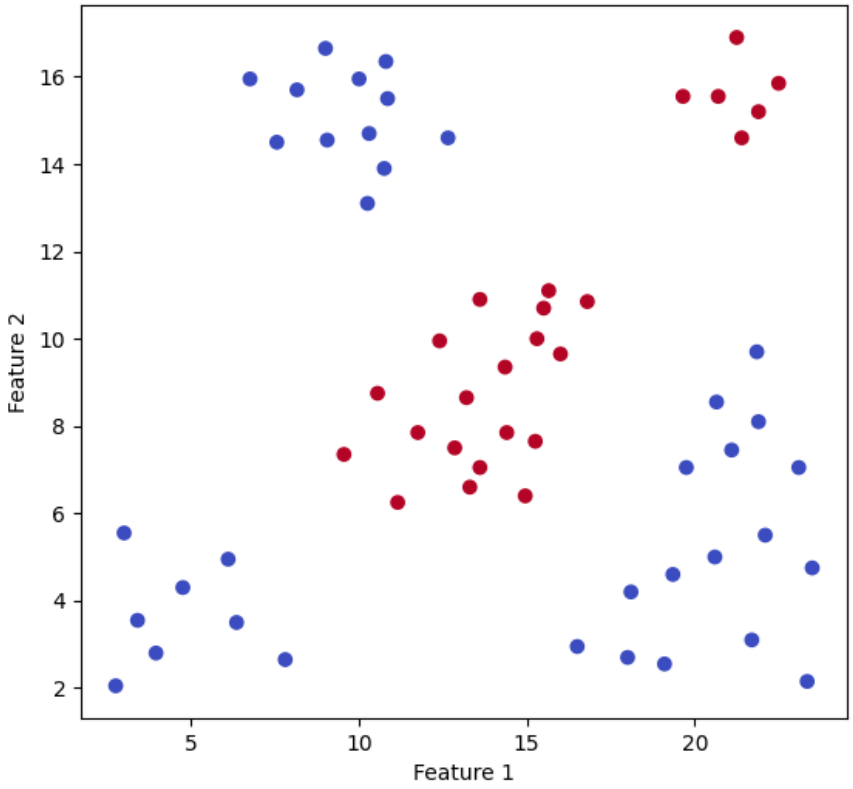
Za zmanjšanje dimenzionalnosti podatkov smo v članku uporabili T-SNE (T-distributed Stochastic Neighbor Embedding), ki se pogosto uporablja za vizualizacijo visokodimenzionalnih podatkov v nižje dimenzionalnem prostoru. Nesistematično sta bili preizkušeni tehniki UMAP (Uniform Manifold Approximation and Projection) in PCA (Principal component analysis), a je t-SNE proizvedel bolj primerne oblike gruč, zato smo uporabili le T-SNE v nadaljevanju. T-SNE deluje tako, da v visokodimenzionalnem prostoru izračuna verjetnostne porazdelitve, ki predstavljajo podobnost med primeri. Nato ustvari podobno verjetnostno porazdelitev v nižje dimenzionalnem prostoru in poskuša minimizirati razliko med tema dvema porazdelitvama. T-SNE smo uporabili tudi za predprocesiranje visokodimenzionalnih podatkov, s katerim smo dobili boljši rezultat pri gručenju (KMEANS, DBSCAN, HDBSCAN). Več informacij o T-SNE metodi je v članku [23].

Poglavje 3

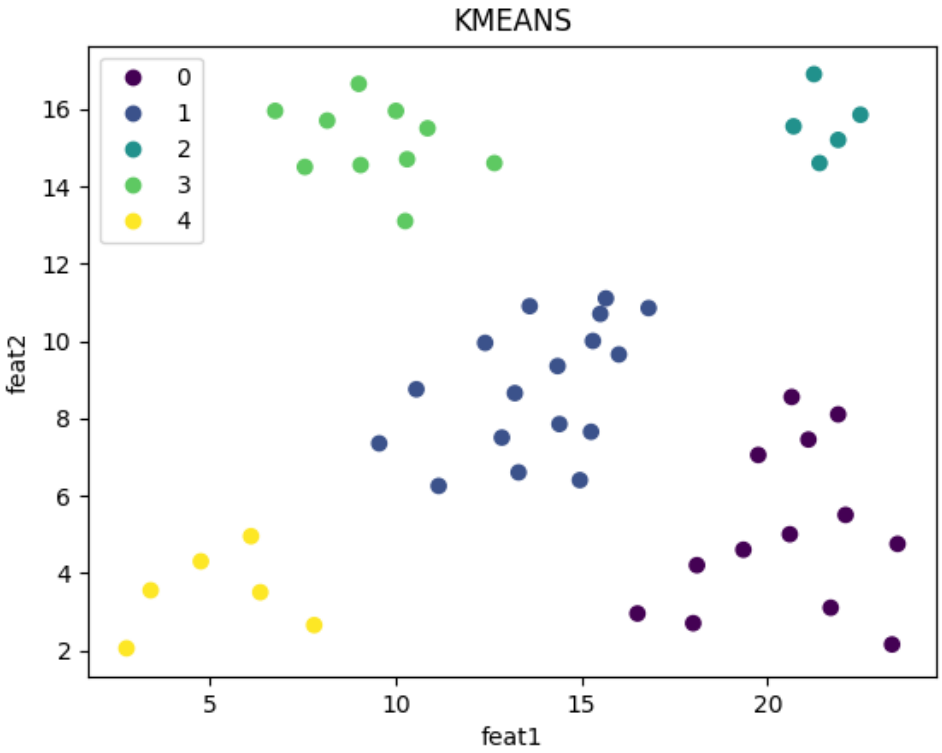
Opis pristopa s podskupinami

V tem poglavju predstavimo pristop razlage klasifikatorja s podskupinami. Primer bo preprost za lažje razumevanje. Najprej spoznamo umetno množico podatkov, ki je prikazana na sliki 3.1. Vidimo 5 gruč in vsaka od gruč v celoti vsebuje primere enega od dveh razredov, kar je razvidno z modro in rdečo barvo. Prvo so bili podatki razdeljeni na učno in testno množico. Klasifikator, naučen na učni množici, ki ga uporabimo v tem primeru je XGBoost, ima klasifikacijsko točnost 0.9 in F1-score 0.9 na testnih podatkih. Sledi gručenje podatkov, s katerim bomo pridobili podskupine podatkov. V sklopu diplomske naloge so bili testirani 4 algoritmi gručenja, ki imajo drugačne lastnosti. Njihova izbira je odvisna od posameznih podatkov. Za ta primer je bil uporabljen algoritem gručenja k-means. Algoritem kot obvezni parameter zahteva število skupin, katerim bo dodelil podatke. Za avtomatizacijo izbora tega parametra sem uporabil mero silhuete, ki nam pove kako dobro so bili podatki gručeni. Algoritem k-means sem pognal čez različno število skupin in največja vrednost silhuete (0,601) je prišla pri 5 skupinah. Rezultat gručenja je viden na sliki 3.2.

Na tem mestu je potrebno omeniti, da sem za algoritme gručenja, ki temeljijo na gostoti podatkov uporabil mero dbcv, ki je analogna meri silhuete, a je namenjena za oceno gručenja prav teh algoritmov. V 2 dimenzijah je človeškemu očesu jasno, kje so meje gruč, a v višjih dimenzijah temu ni tako.



Slika 3.1: Ilustracija umetne množice podatkov, ki vsebuje več konceptov. Vidimo podatke 3 razredov predstavljenih z barvami (siva, modra, rdeča) in 5 gruč. Imamo 2 značilki, ki se razprostirata v dvodimenzionalnem prostoru.



Slika 3.2: Rezultat gručenja z algoritmom k-means. Največja vrednost hevristike silhuete je bila pri 5 skupinah.

Vizualizacije ne povejo celotne zgodbe. To je eden izmed razlogov uporabe odločitvenih pravil, ki nam povejo, kako se izbrana gruča loči od preostalih. S tem pridobimo občutek za meje posamezne gruče, ki nam bodo kasneje pri razlagi primera pomagale pri umestitvi primera v gručo. Vidi se tudi katere značilke so bolj pomembne za umestitev v gručo, saj bodo tiste, ki najbolje ustrezajo skupini, med pogoji odločitvenih pravil. Pravila pridobljena na našem primeru so vidna v tabeli 3.1.

Tabela 3.1: Odločitvena pravila za meje posameznih gruč

Gruča	**Odločitvena pravila**	**Precision**	**Recall**
0	feat1 > 16.07 in feat2 <= 8.95	1.0	1.0
1	feat1 <= 19.35 in feat2 <= 12.10 in feat2 > 5.88	1.0	1.0
2	feat1 > 20.68 in feat2 > 11.35	1.0	1.0
3	feat1 <= 16.68 in feat2 > 12.80	1.0	1.0
4	feat1 <= 7.97 in feat2 <= 9.72	1.0	1.0

Tabela 3.2: Medoidi posameznih gruč

Gruča	**Feat1**	**Feat2**	**Razred**
0	20.6	5	0
1	14.35	9.35	1
2	21.9	15.2	1
3	9.05	14.55	0
4	4.75	4.3	0

Gruče bomo razložili z medoidom, ki predstavlja tipičen primer gruče. Prav tako si bomo pogledali SHAP vrednosti celotne gruče in novega primera. V tabeli 3.3 vidimo nov primer. S pomočjo teh metod bomo nato razložili klasifikator na tem primeru.

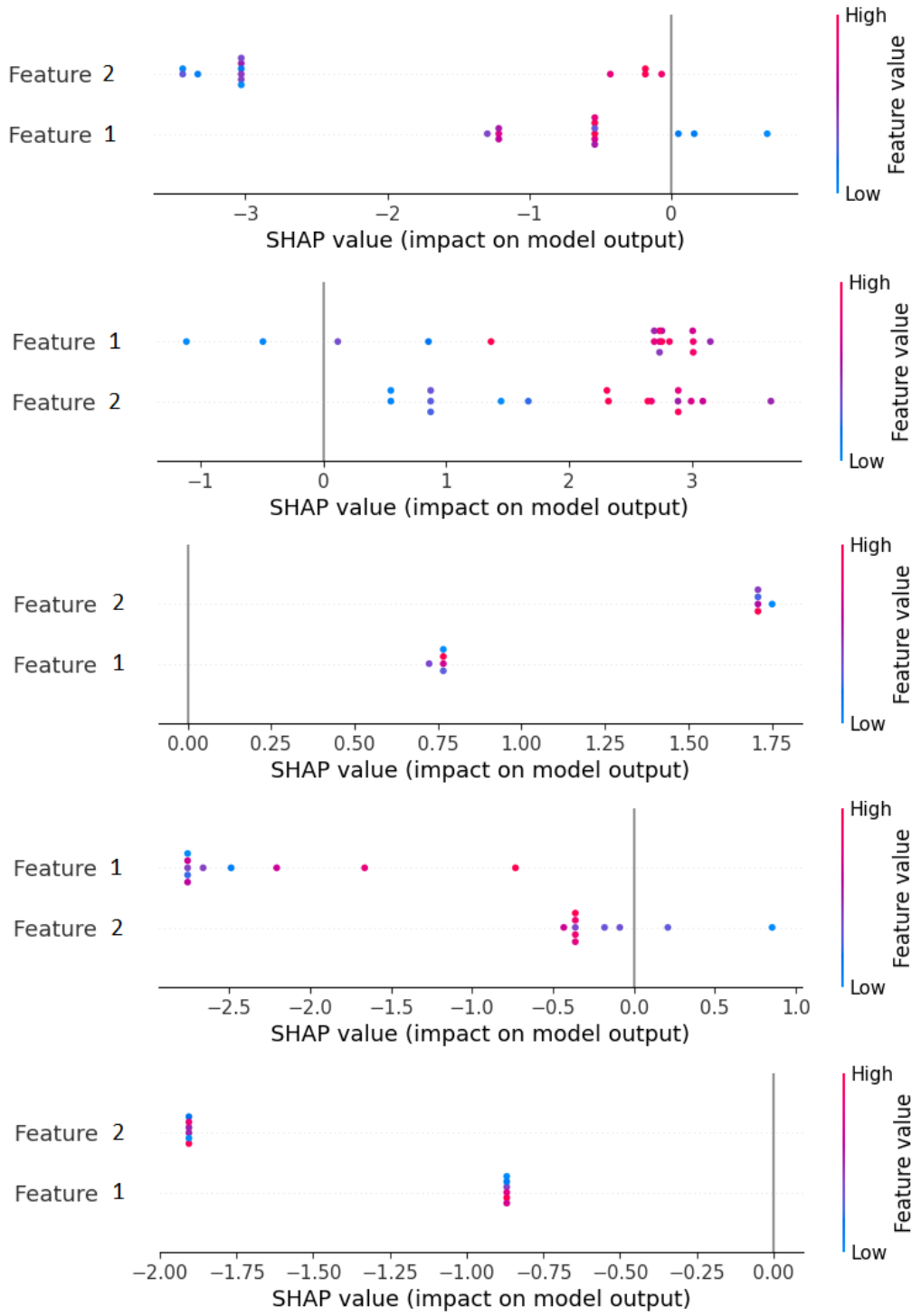
Vzamimo neviden. Klasifikator ga je uvrstil v razred 1. Primer lahko po pravilih za mejo gruče razvrstimo v gručo številka 1, seveda ima tudi knjižnica

Tabela 3.3: Neviden primer

Feat1	**Feat2**	**Predikcija klasifikatorja**
13	9	1

za gručenje opcijo predikcije novega primera, ki prav tako trenutni primer razvrsti v gručo 1. Prvo nas zanimajo verjetnosti razredov v gruči 1. Vidimo, da je gruča v celoti homogena in vsebuje le razred 1. V takem primeru nas zanima medoid, saj dobro opiše gručo. V tabeli 3.2 imamo podane medoide za gruče. Vidimo, da medoid spada pod razred 1, kar podpre odločitev klasifikatorja. Pri razlagi nam velikokrat pomaga podatek o pomembnosti značilk, ki ga bomo pridobili s SHAP vrednostmi. Na sliki 3.3 vidimo SHAP vrednosti za posamezne gruče. Nov primer ima SHAP vrednosti 3.008 in 3.091, kar pomeni, da sta obe značilki pomembni za klasifikacijo primera v razred 1.

Podatkovna množica "Homogene gruče"je sestavljena iz samih homogenih gruč. Če je kakšna gruča sestavljena iz več razredov moramo uporabiti odločitvena pravila za ločitev razredov znotraj gruče.



Slika 3.3: Na grafu so prikazane SHAP vrednosti za vse gruče 0-4 od zgoraj navzdol. Vidimo, da sta za gručo 1 pomembni obe značilki in da so SHAP vrednosti bolj razpršene kot pri drugih gručah. Ta gruča je sredinska in je bolj kompleksna za klasifikator kot druge.

Poglavje 4

Evalvacija

4.1 Podatkovne množice

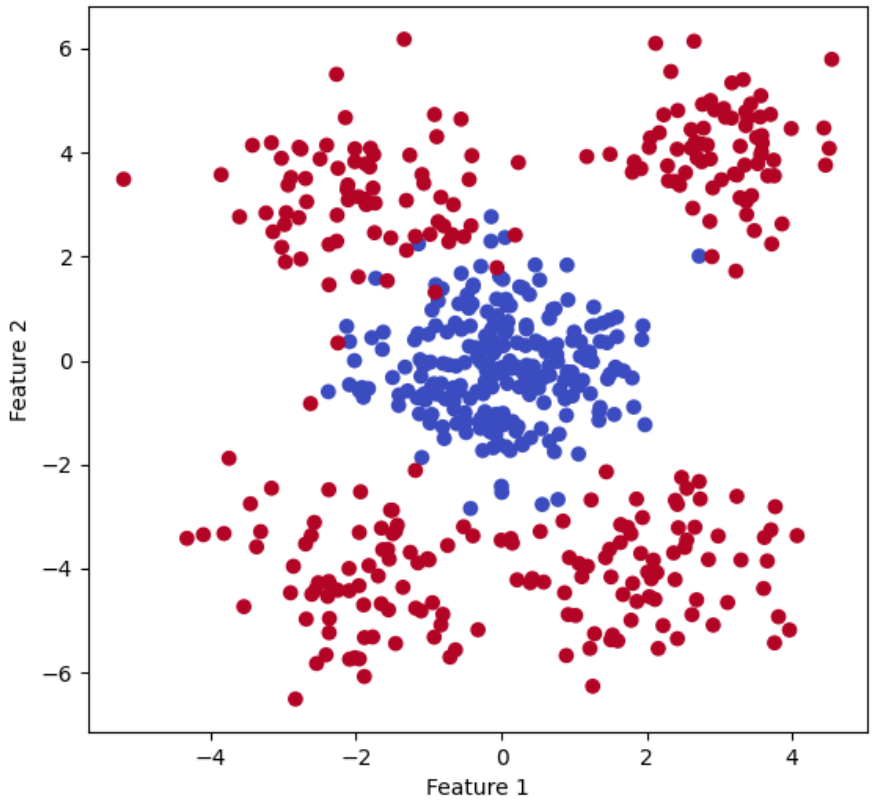
4.1.1 Umetne podatkovne množice

Umetna podatkovna množica "Krožne gruče 5-2"

Prva umetna podatkovna množica (slika 4.1) je dvodimenzionalna in ima 550 primerov. Podatkovna množica ima 2 razreda, opisana sta z dvema atributoma. Zanima nas, kako dobro algoritmi najdejo pet gruč. Gruče so sferične oblike in se na mejah rahlo prekrivajo. Primeri v posameznih gručah so porazdeljeni normalno v 2 dimenzijah z različnimi standardnimi odkloni. V modri gruči je 250 primerov, v vsaki od rdečih pa 75. Podatkovna množica je bila namenjena primerjavi algoritmov in preizkusu hevristike silhuete in DBCV indeksa.

Umetna podatkovna množica "Trakovi 4-3"

Druga umetna podatkovna množica (slika 4.2) je dvodimenzionalna in ima 545 primerov, porazdeljenih v 3 razrede in opisanih z dvema atributoma. Primeri so porazdeljeni v štiri zavite in podolgovate gruče, pozicionirane druga poleg druge. Primeri v posameznih gručah so zelo gosti do meje gruč, med in okoli gruč najdemo primere z nizko gostoto, ki predstavljajo šum.



Slika 4.1: Dvodimenzionalna umetna podatkovna množica "Krožne gručeš 5 gručami in 2 razredoma. Razred 0 je označen z modro barvo, razred 1 z rdečo.

Podatkovna množica je bila namenjena primerjavi algoritmov in preizkusu hevristike silhuete in DBCV indeksa.

Umetna podatkovna množica "Testna"

Tretja umetna podatkovna množica "Testna"(slika 4.3) je dvodimenzionalna in ima 61 primerov. Podatkovna množica ima 2 razreda in 5 gruč. Gruče so homogene (vsebujejo primere le enega razreda) in vsak razred je razčlenjen na več gruč, kot je razvidno na sliki 4.3. Podatkovna množica je bila namenjena opisu postopka razlage klasifikatorja.

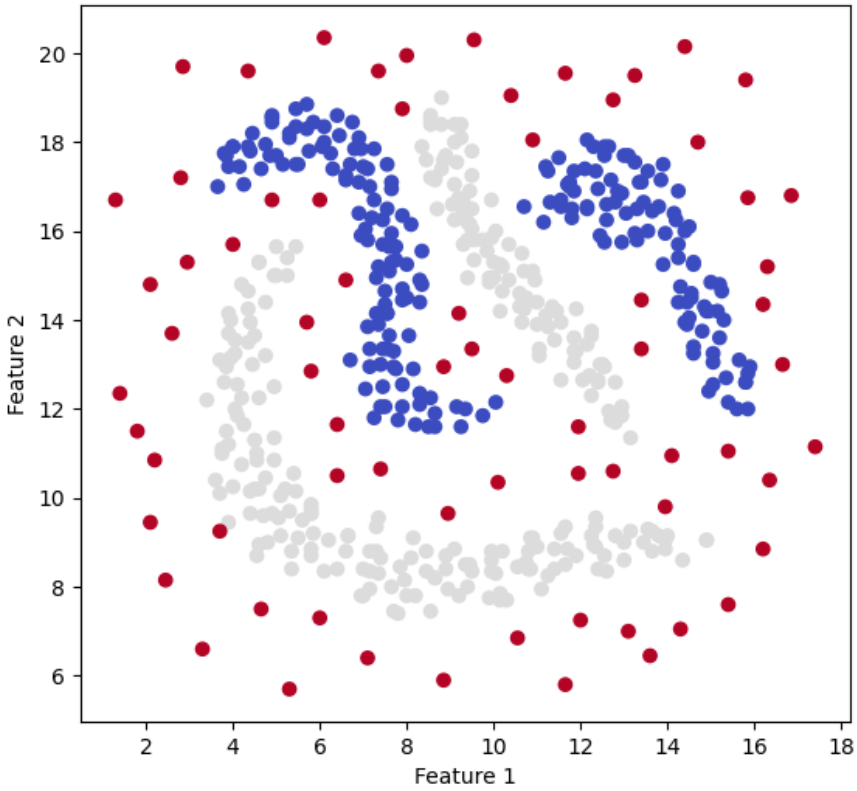
4.1.2 Realne podatkovne množice

Podatkovna množica MNIST

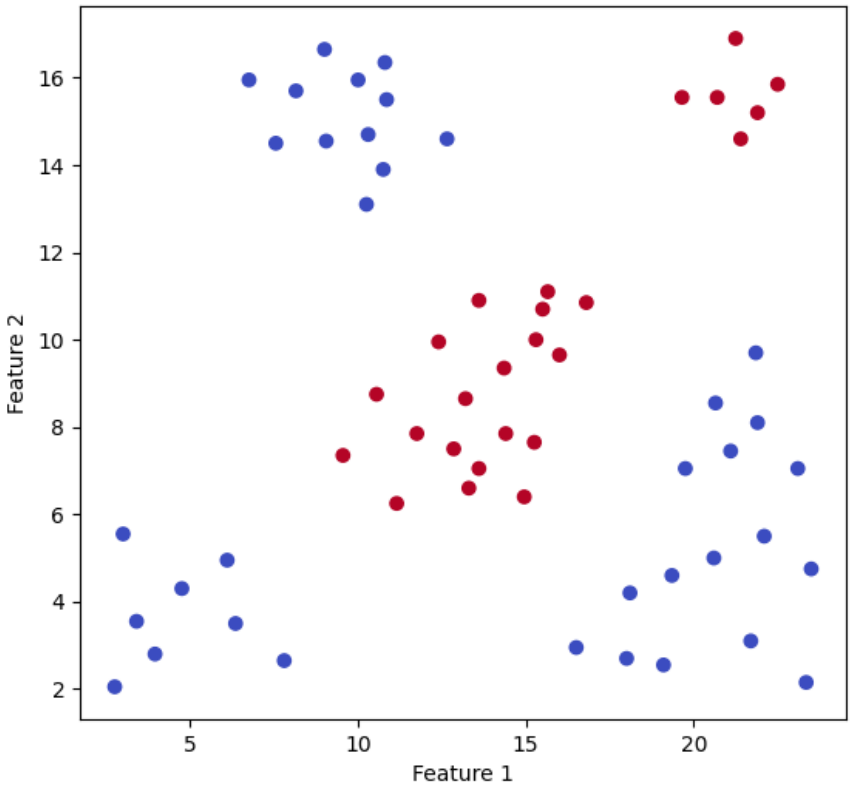
Podatkovna množica MNIST (Modified National Institute of Standards and Technology) je široko uporabljena na področju strojnega učenja in računalniškega vida. Vsebuje zbirko ročno napisanih številk (slik), ki se pogosto uporabljajo za učenje različnih algoritmov za klasifikacijo (vsebuje 10 razredov: številke 0-9). Vsaka slika je sivinska in ima resolucijo 28x28 = 784. Podatkovna množica ima 70000 slik, ki se ponavadi delijo na učno množico (60000) in testno množico (10000). Podatkovno množico smo uporabili za testiranje algoritmov gručenja. Zaradi časovne kompleksnosti algoritmov smo vzeli le prvih 10000 primerov iz učne množice in vsako sliko, ki je 2D matrika razvili v vektor. Slike vidimo na sliki 4.4 [2].

Podatkovna množica KDD99

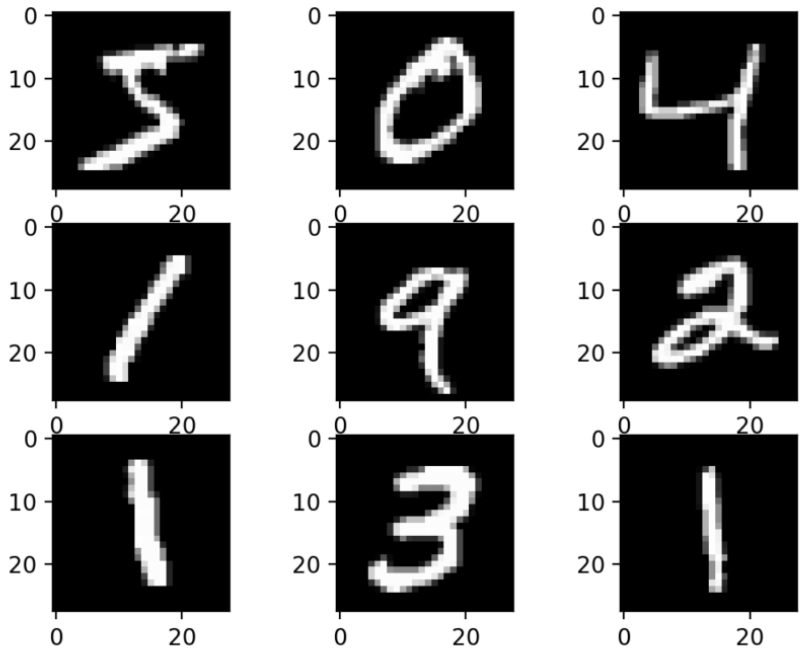
Podatkovna množica KDD99 je široko poznana in pogosto uporabljena na področju varnosti in odkrivanja anomalij v računalniških omrežjih. Podatkovna množica je uporabljena za razvoj in ocenjevanje sistemov, ki odkrivajo vdore v računalniška omrežja. Podatkovna množica vsebuje različne vrste omrežnih prometov, vključno z običajnimi omrežnimi aktivnostmi in potencialnimi vdori. Vsebuje 23 različnih vrst napadov (ciljna spremenljivka). V



Slika 4.2: Dvodimenzionalna umetna podatkovna množica "Trakovi 4-3". Ima 4 gruče in 3 razrede. Razreda 0 in 1 (mofra in siva barva) predstavljata sestavo gruč, medtem ko razred 2 (rdeča barva) predstavlja šum.



Slika 4.3: Dvodimenzionalna umetna podatkovna množica s 5 gručami in 2 razredoma. Razred 0 je označen z modro, razred 1 z rdečo.



Slika 4.4: Slike števk iz podatkovne množice MNIST, ki jih uporabljamo za strojno učenje in testiranje modelov.

našem primeru smo jih razdelili na škodljive in normalne povezave. Ima 41 atributov, kjer so 3 kategorični (protocol, service in flag), ostali pa zvezni. Podatkovno množico smo (poleg ciljne spremenljivke) predprocesirali po naslednjih korakih: odstranili smo primere, ki se ponavljajo, zatem smo uporabili tehniko "One Hot Encoding", kjer smo za vsako vrednost kategorične spremenljivke dodali nov atribut (zatem imamo 118 atributov) in vzeli naključnih 5000 primerov zaradi časovne zahtevnosti algoritmov. V tabeli 4.1 vidimo imena vseh atributov (z enakimi kraticami so atributi poimenovani tudi v poglavju 4.6). Predprocesiranje in dodatne informacije so opisane v članku [22].

4.2 Primerjava algoritmov za gručenje

Za ocenjevanje algoritmov gručenja bomo uporabili dve hevristiki: koeficient silhuete in DBCV. Indeks silhuete boljši za ocenjevanje algoritmov, ki delujejo na osnovi razdalj, DBCV indeks pa je boljši za algoritme, ki delujejo na osnovi gostote in del podatkov označijo kot šum [4]. DBSCAN in HDBSCAN ne razdelita vseh primerov v gruče, tako da zapišemo tudi delež označenih primerov, ostali so označeni kot osamelci oziroma šum. V nadaljevanju si bomo ogledali 2 umetni podatkovni množici in realno podatkovno množico MNIST, na katerih bomo lahko primerjali algoritme. V nadaljevanju bodo predstavljeni le rezultati, ki so dobili najboljšo hevristiko.

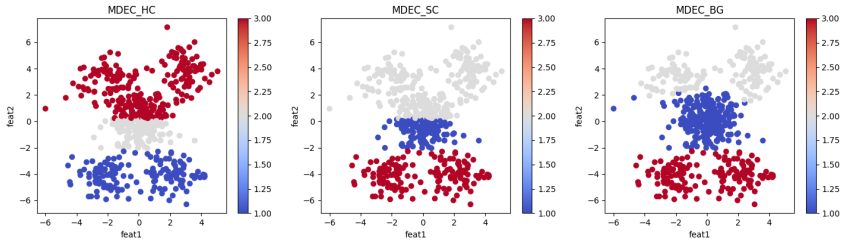
4.2.1 "Krožne gruče"

Algoritem MDEC

Algoritem MDEC vrne 3 rezultate gručenja. Najboljša vrednost silhuete (tabela 4.2) je bila pri parametru za število gruč k = 3, kar ni najboljši rezultat, saj je razvidnih 5 skupin. Vredno je omeniti, da MDEC vsebuje naključne elemente, tako da se vrednosti in ocene lahko vsakič rahlo spremenijo, a so bile v večini zelo blizu. Vzet je bil najboljši rezultat izmed izvedenih po-

Tabela 4.1: Imena in kratice atributov podatkovne množice KDD99

Kratica	Ime	Kratica	Ime	Kratica	Ime
F0	duration	F40	protocol_type_b'udp'	F80	service_b'ntp_u'
F1	src_bytes	F41	service_b'IRC'	F81	service_b'other'
F2	dst_bytes	F42	service_b'X11'	F82	service_b'pm_dump'
F3	land	F43	service_b'Z39_50'	F83	service_b'pop_2'
F4	wrong_fragment	F44	service_b'auth'	F84	service_b'pop_3'
F5	urgent	F45	service_b'bgp'	F85	service_b'printer'
F6	hot	F46	service_b'courier'	F86	service_b'private'
F7	num_failed_logins	F47	service_b'csnet_ns'	F87	service_b'red_i'
F8	logged_in	F48	service_b'ctf'	F88	service_b'remote_job'
F9	num_compromised	F49	service_b'daytime'	F89	service_b'rje'
F10	root_shell	F50	service_b'discard'	F90	service_b'shell'
F11	su_attempted	F51	service_b'domain'	F91	service_b'smtp'
F12	num_root	F52	service_b'domain_u'	F92	service_b'sql_net'
F13	num_file_creations	F53	service_b'echo'	F93	service_b'ssh'
F14	num_shells	F54	service_b'eco_i'	F94	service_b'sunrpc'
F15	num_access_files	F55	service_b'ecr_i'	F95	service_b'supdup'
F16	num_outbound_cmds	F56	service_b'efs'	F96	service_b'systat'
F17	is_host_login	F57	service_b'exec'	F97	service_b'telnet'
F18	is_guest_login	F58	service_b'finger'	F98	service_b'tftp_u'
F19	count	F59	service_b'ftp'	F99	service_b'tim_i'
F20	srv_count	F60	service_b'ftp_data'	F100	service_b'time'
F21	serror_rate	F61	service_b'gopher'	F101	service_b'urh_i'
F22	srv_serror_rate	F62	service_b'hostnames'	F102	service_b'urp_i'
F23	rerror_rate	F63	service_b'http'	F103	service_b'uucp'
F24	srv_rerror_rate	F64	service_b'http_443'	F104	service_b'uucp_path'
F25	same_srv_rate	F65	service_b'imap4'	F105	service_b'vmnet'
F26	diff_srv_rate	F66	service_b'iso_tsap'	F106	service_b'whois'
F27	srv_diff_host_rate	F67	service_b'klogin'	F107	flag_b'OTH'
F28	dst_host_count	F68	service_b'kshell'	F108	flag_b'REJ'
F29	dst_host_srv_count	F69	service_b'ldap'	F109	flag_b'RSTO'
F30	dst_host_same_srv_rate	F70	service_b'link'	F110	flag_b'RSTOS0'
F31	dst_host_diff_srv_rate	F71	service_b'login'	F111	flag_b'RSTR'
F32	dst_host_same_src_port_rate	F72	service_b'mtp'	F112	flag_b'S0'
F33	dst_host_srv_diff_host_rate	F73	service_b'name'	F113	flag_b'S1'
F34	dst_host_serror_rate	F74	service_b'netbios_dgm'	F114	flag_b'S2'
F35	dst_host_srv_serror_rate	F75	service_b'netbios_ns'	F115	flag_b'S3'
F36	dst_host_rerror_rate	F76	service_b'netbios_ssn'	F116	flag_b'SF'
F37	dst_host_srv_rerror_rate	F77	service_b'netstat'	F117	flag_b'SH'
F38	protocol_type_b'icmp'	F78	service_b'nnsp'	N/A	N/A
F39	protocol_type_b'tcp'	F79	service_b'nntp'	N/A	N/A



Slika 4.5: Tri rezultati MDEC algoritma pri številu gruč k = 3. Zanima nas najbolj desni rezultat z najboljšo hevristiko.

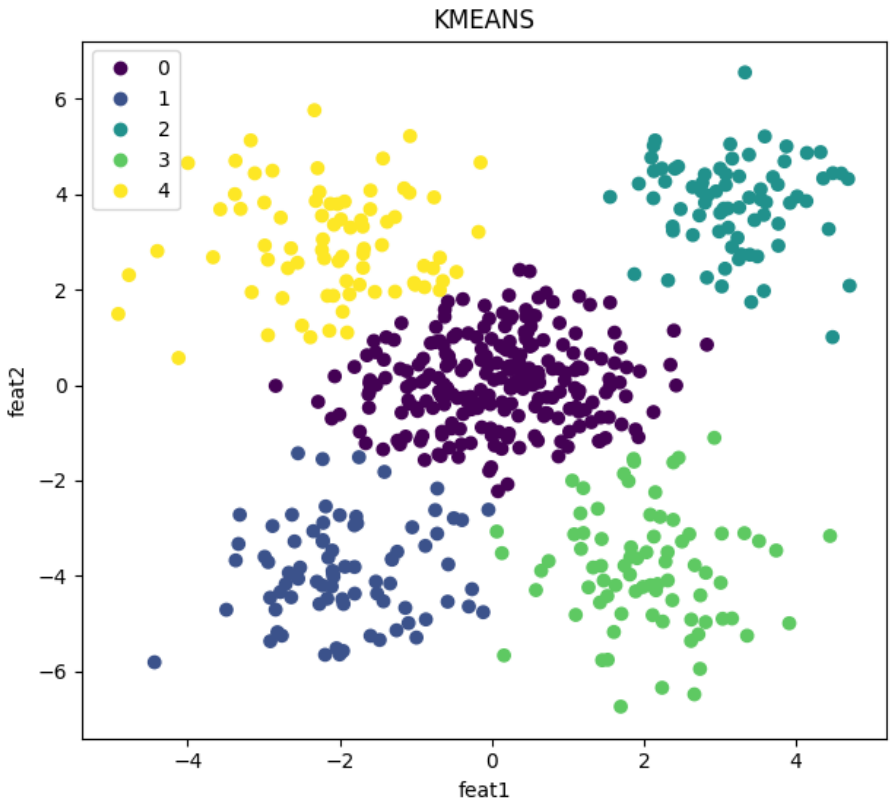
skusov. Na sliki 4.5 je bil to rezultat MDEC BG. Algoritem MDEC je pri dvodimenzionalnih podatkih ustvaril ravne pasove, kar ni bila dobra rešitev gručenja. Pomembno je omeniti, da je algoritem MDEC namenjen uporabi v visoko dimenzionalnih prostorih, kjer se odreže bolje.

Tabela 4.2: Tabela ocen in deleža označenih primerov

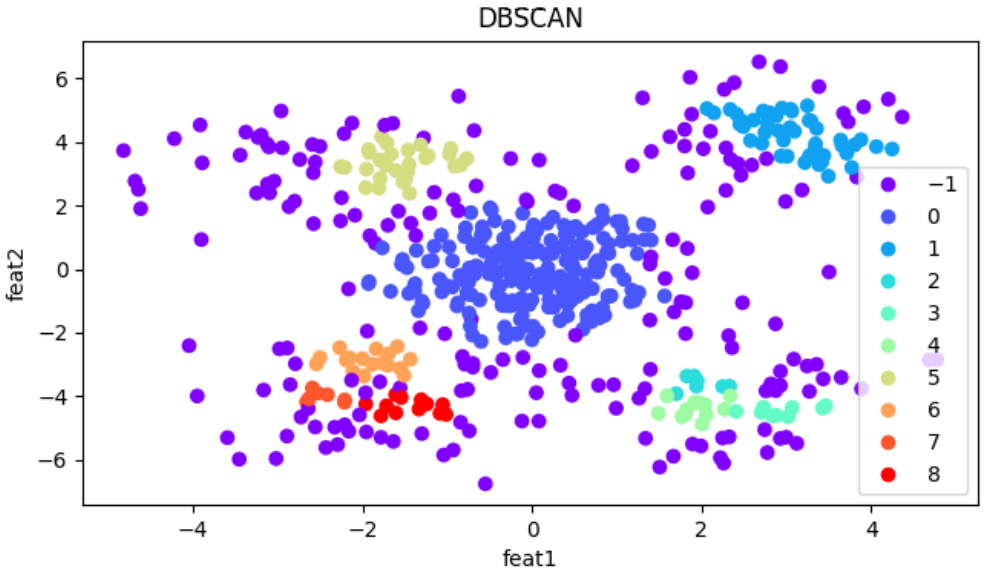
Algoritem	Silhueta	DBCV	%
MDEC	0.40	N/A	100
KMEANS	0.52	N/A	100
DBSCAN	N/A	0.34	65
HDBSCAN	N/A	0.38	69

Algoritem K-MEANS

Algoritem k-means je vrnil zelo dober rezultat, saj so oblike gruč sferične in dovolj ločene. Najboljša vrednost silhuete (tabela 4.2) je bila pri parametru za število gruč k = 5, kar je dejansko število skupin. Silhueta k-means je v tem primeru boljša od silhuete MDEC algoritma, kar nam potruje boljše grupiranje k-means algoritma. Na sliki 4.6 je razviden rezultat gručenja.



Slika 4.6: Rezultat K-MEANS algoritma pri številu gruč k = 5.



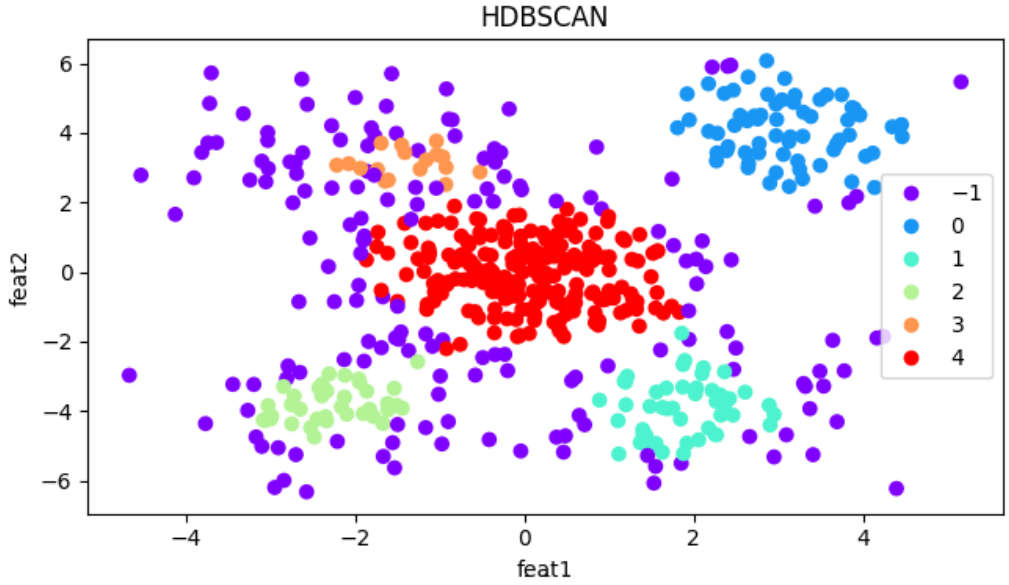
Slika 4.7: Rezultat DBSCAN algoritma.

Algoritem DBSCAN

Algoritem DBSCAN nima parametra za nastavitev števila gruč, ima pa dva druga parametra - min samples in epsilon. Izbira parametrov je opisana pod predstavitvijo tehnologij, tukaj bo predstavljen le najboljši rezultat. Algoritem je zelo veliko primerov označil kot osamelce in jih ni dodelil v nobeno gručo. Oznako je dobilo 65 % primerov za najboljši rezultat. Omeniti je potrebno, da DBCV ocene ni pametno primerjati z silhueto, saj delujeta na drugačnih principih. DBSCAN je našel 9 gruč kar pomeni, da je našel 4 dodatne gruče, ki niso samoumevne iz podatkov. Rezultati gručenja za DBSCAN so na sliki 4.7 in v tabeli 4.2.

Algoritem HDBSCAN

Algoritem HDBSCAN samodejno določi epsilon. Namesto njega smo izbrali parametra min cluster size in min samples. Oznako je dobilo 69 % primerov za najboljši rezultat. HDBSCAN je našel 5 gruč kar pomeni, da je našel pravilno število gruč kot k-means. Vidimo, da je DBCV indeks boljši kot pri



Slika 4.8: Rezultat HDBSCAN algoritma.

DBSCAN algoritmu kar dodatno potrjuje boljše gručenje. Rezultati gručenja za HDBSCAN so na sliki 4.8 in v tabeli 4.2.

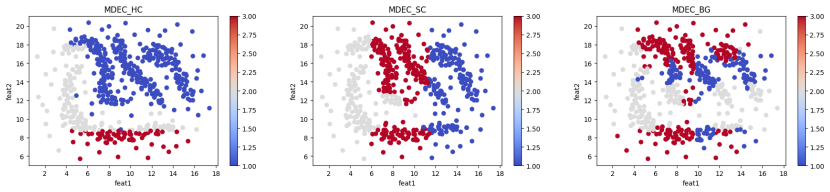
4.2.2 "Trakovi 4-3"

Algoritem MDEC

Ponovno vidimo, da je algoritem MDEC prostor razdelil na pasove. Rezultat je na sliki 4.9. Najboljša rezultat (tabela 4.3) je dosegel MDEC_HC pri parametru za število gruč k = 3.

Tabela 4.3: Tabela ocen in deleža označenih primerov

Algoritem	Silhueta	DBCV	%
MDEC	0.25	N/A	100
KMEANS	0.44	N/A	100
DBSCAN	N/A	0.0034	87
HDBSCAN	N/A	0.095	81



Slika 4.9: Tri rezultati MDEC algoritma pri številu gruč k = 3. Najbolj levi je dobil najboljšo oceno.

Algoritem K-MEANS

Algoritem k-means je vrnil najboljši rezultat pri številu gruč k = 10. Rezultat gručenja je viden v tabeli 4.3 in na sliki 4.10. Vidimo, da se vsak centroid vmesti v prostor in zavzame približno sferično obliko.

Algoritem DBSCAN

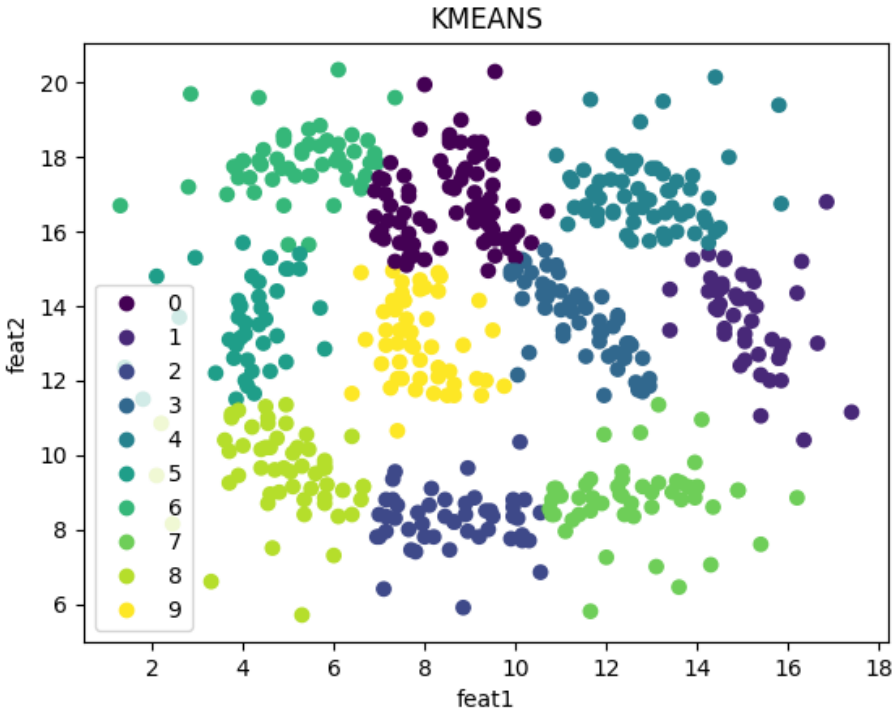
Algoritem DBSCAN je uporaben pri gručenju naravnih podolgovatih oblik, kar je razvidno tudi iz rezultata gručenja na sliki 4.11 in v tabeli 4.3. DBSCAN je našel 4 gruče kar pomeni, da je našel vse gruče in hkrati izločil šum iz podatkov.

Algoritem HDBSCAN

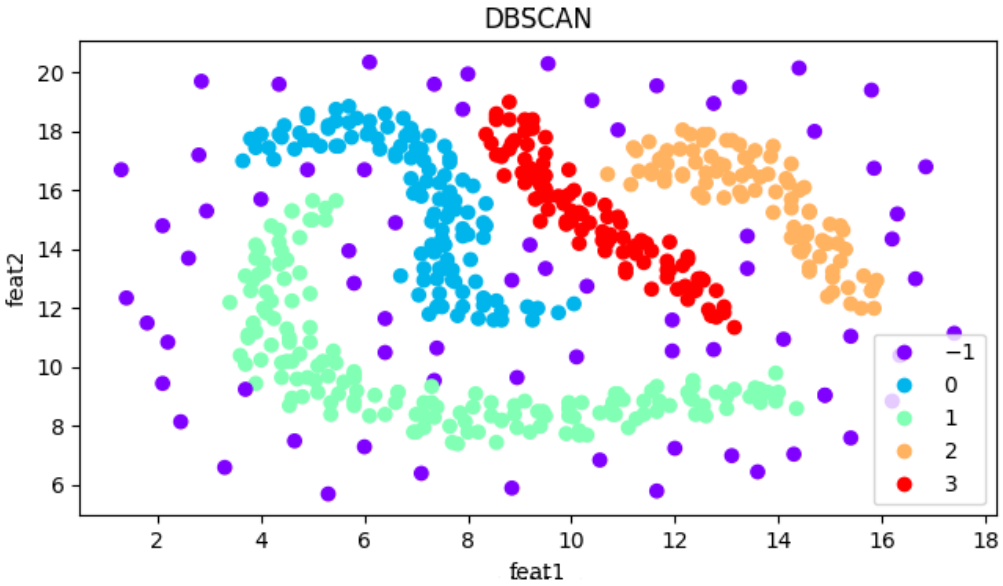
Algoritem HDBSCAN je hierarhična različica DBSCAN algoritma, a vrne slabši rezultat kot DBSCAN kot je razvidno iz slike 4.12 in tabele 4.3. HDBSCAN je našel 3 gruče kar pomeni, da je dve ločeni gruči združil v eno. Najboljša DBCV vrednost je 0.095 kar je presenetljivo boljši DBCV indeks kot od DBSCAN algoritma glede na to da vidimo slabše gručenje s strani HDBSCAN algoritma. Šum je dobro ločen od skupin.

4.2.3 MNIST

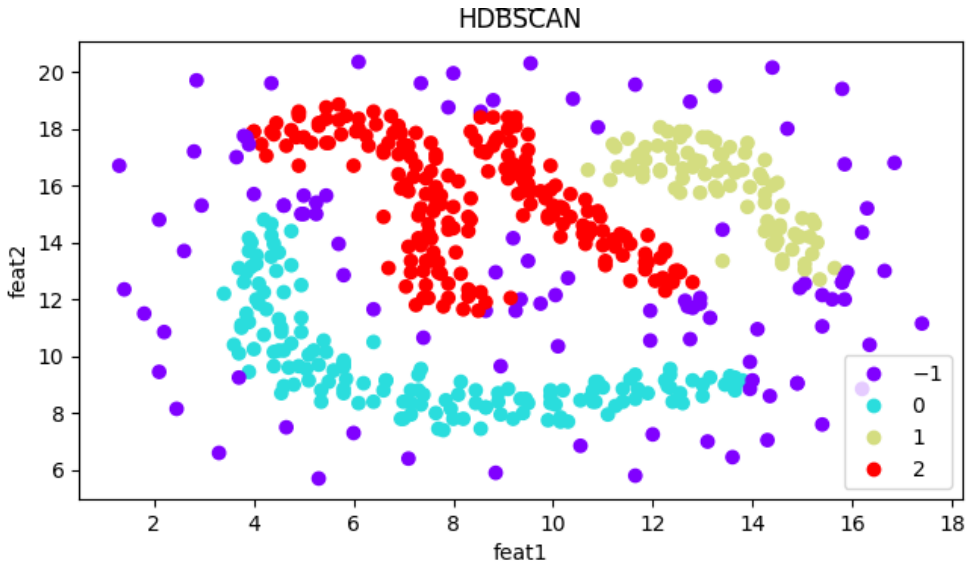
Podatkovna množica MNIST ima 784 atributov, zato smo za prikaz v 2 dimenzijah uporabili vložitev T-SNE. Zaradi časovne zahtevnosti algoritmov



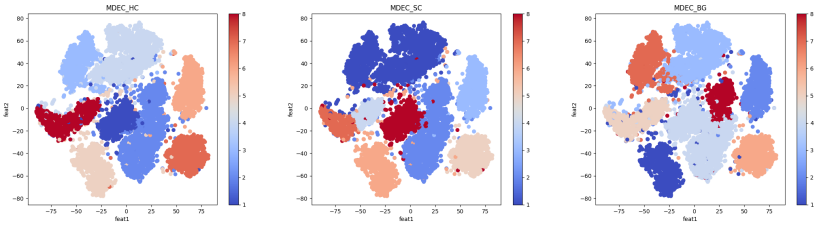
Slika 4.10: Rezultat K-MEANS algoritma pri številu gruč k = 10.



Slika 4.11: Rezultat DBSCAN algoritma. Na sliki se vidi označbe vseh 4 skupin in izločitev šuma iz podatkov.



Slika 4.12: Rezultat HDBSCAN algoritma. Na sliki se vidi označbe 3 skupin in izločitev šuma iz podatkov.



Slika 4.13: Rezultat MDEC algoritma. Algoritem MDEC_HC (najbolj leva slika) je dosegel najboljšo hevristiko silhuete pri številu gruč k = 8. Našel je 2 gruči manj, najdene gruče so smiselne.

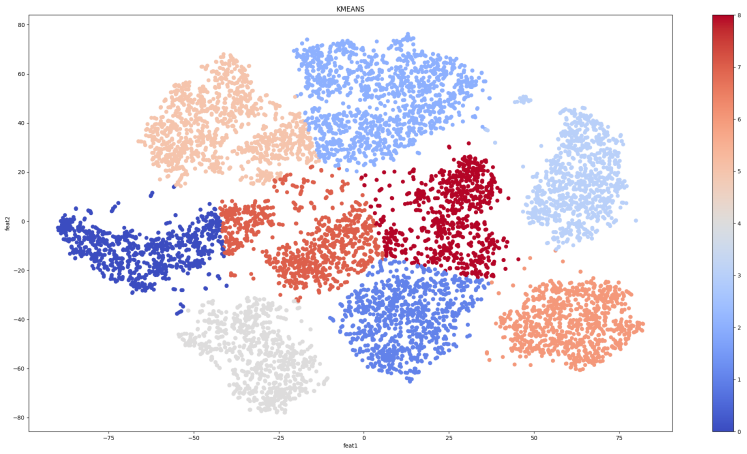
smo uporabili le prvih 10000 primerov. Prav tako smo pri KMEANS, DBSCAN in HDBSCAN algoritmih za gručenje vhodne podatke najprej pretvorili v 2 dimenziji. Izjema je algoritem MDEC, ki je bolje deloval v originalnem, visokodimenzionalnem prostoru. V tabeli 4.4 vidimo rezultate.

Tabela 4.4: Rezultati algoritmov za gručenje

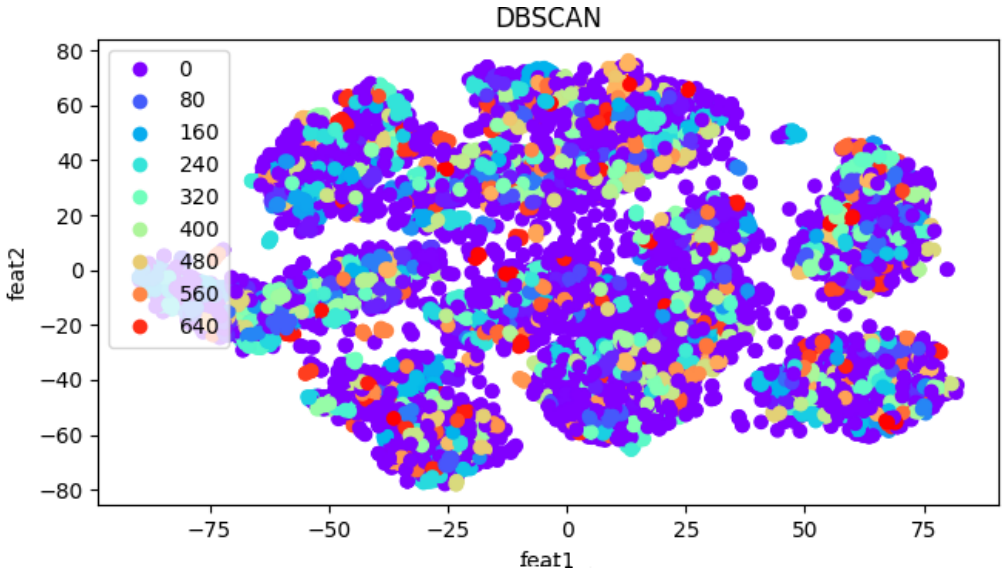
Algoritem	Silhueta	DBCV	%	Vizualizacija
MDEC	0.0471	N/A	100	Slika 4.13
KMEANS	0.4738	N/A	100	Slika 4.14
DBSCAN	N/A	0.282	63	Slika 4.15
HDBSCAN	N/A	-0.162	95	Slika 4.16

4.3 Razlaga realne podatkovne množice in primerjava z obstoječimi razlagami

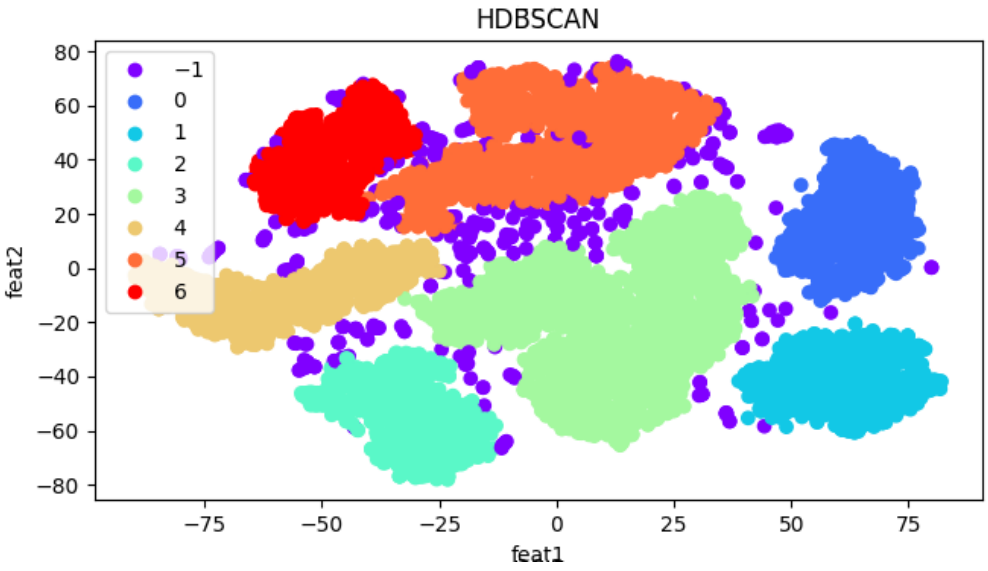
V tem poglavju smo uporabili podatkovno množico KDD99, saj ima strukturo, ki jo zahteva naša metoda. Večina preostalih podatkovnih množic nima primerov enega razreda razčlenjenega v več gruč ali pa ima medsebojno slabo ločene gruče. Predprocesiranje podatkovne množice in imena atributov so na voljo v poglavju o realnih podatkovnih množicah (4.1.2). Pri gručenju smo si



Slika 4.14: Rezultat KMEANS algoritma. Vidimo, da je algoritem dosegel najboljšo hevristiko silhuete pri številu gruč k = 9. Vse gruče niso smiselne, kar se vidi pri temno modri gruči.



Slika 4.15: Rezultat DBSCAN algoritma. Algoritem ni deloval po pričakovanjih, kar se vidi po nerazumnlivih oznakah. Ločene gruče so neobstoječe. Naša izbira parametrov se v tem primeru ne obrestuje, saj je algoritem našel 640 gruč.

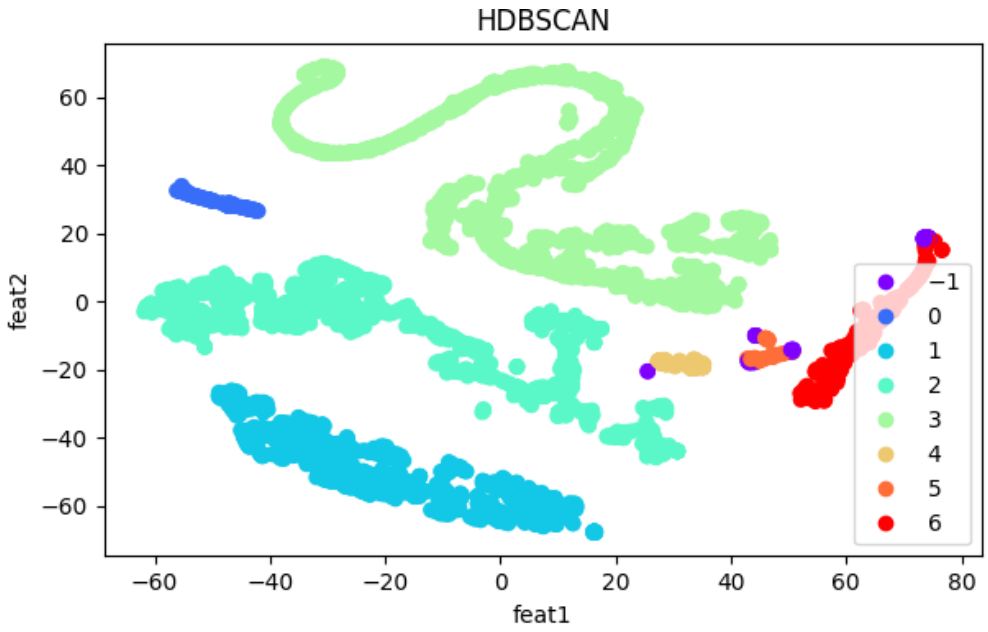


Slika 4.16: Rezultat HDBSCAN algoritma. Algoritem je našel smiselne gruče (7), a ne vseh. Na sliki se vidi, da bi morali biti oranžna in zelena gruča razdeljeni na dodatne podgruče.

pomagali z tehniko T-SNE, saj je pri algoritmih na osnovi gostote (v našem primeru na sliki 4.17 je to HDBSCAN) pomagala pri primerih, ki so bili označeni za šum zaradi visoke dimenzionalnosti. S tem smo pridobili 7 gruč, ki so na prvi pogled smiselno opredeljene. V tabeli 4.5 vidimo verjetnosti razredov za vsako gručo. Gruči 1 in 2 vsebujeta škodljive povezave, ostale normalne. Opazimo, da sta gruči 2 in 5 bolj mešani kot preostale gruče, kar bomo kasneje omenili pri razlagi z medoidi, kjer si bomo pomagali še z odločitvenimi pravili znotraj gruče. Za klasifikacijo povezav smo uporabili logistično regresijo, ki je imela na testnih podatkih klasifikacijsko točnost 0.9587 in F1 score 0.9587. V tabeli 4.6 imamo podana odločitvena pravila, ki med seboj ločijo gruče. Vidimo, da imamo za vsako gručo več pravil, saj so podatki visokodimenzionalni in se jih lahko dobro loči po različnih atributih. Pri interpretaciji pravil je pomembno pogledati meri Precision in Recall, saj so nekatera pravila manj natančna ali pa ne zajamejo vseh primerov. Ko imamo občutek kako so skupine medsebojno ločene, vsako lahko opišemo z medoidom, ki je tipičen primer gruče. Medoide vidimo v tabeli

4.7. Zaradi velikega števila atributov smo obdržali v tabeli le tiste, kateri se razlikujejo vsaj pri enem izmed medoidov. Poleg medoidov nas zanima tudi kako so ločeni razredi znotaj gruče 2 in 5, kar lahko vidimo v tabeli 4.8. Zanima nas tudi, kateri atributi so pomembni za naš klasifikator (logistično regresijo), kar vidimo na grafih za SHAP vrednosti na slikah 4.18, 4.19, 4.20. Vidimo, da je veliko atributov nepomembnih. Pomembnost atributov nato upoštevamo pri odločitvenih pravilih in medoidih, kjer nam pravila in medoidi sestavljeni iz pomembnih atributov povejo več o našem klasifikatorju in odločitvi. Sedaj vzamemo naključen primer iz testne množice. Zaradi visoke dimenzionalnosti niso podane vrednosti atributov. Primer je logistična regresija opredelila v razred 1, algoritem za gručenje pa ga je razvrstil v gručo 1. Pogledamo tabelo verjetnosti razredov in vidimo, da je gruča 1 v celoti homogena s 100% primerov, ki pripadajo razredu 1 (škodljiva povezava). V tabeli z odločitvenimi pravili 4.5 vidimo 9 pravil, ki z visoko natančnostjo opišejo gručo z majhno podmnožico atributov. Skupina je homogena, zato jo dobro opiše medoid M1 v tabeli 4.6. V primeru, da bi bil nov primer iz gruče 2 ali 5 (nečisti gruči), bi si ogledali še pravila znotraj gruče, ki ločijo razrede. SHAP vrednosti na grafu za gručo 1 prikazujejo, da so v večini pomembni atributi F19, F29, F20, F28. Nov primer je dobil SHAP vrednosti F19: 13.9, F29: 2.4, F20: 0.34, F28: -0.23, vrednosti ostalih atributov so zanemarljivo majhne.

Na podatkovni množici KDD99 že obstajajo poskusi razlage, kot v članku [14], kjer je bila podatkovna množica razložena z odločitvenim drevesom. Atributi so bili glede na pomembnost ocenjeni z mero entropije. Prav tako so za odkrivanje znanja iz podatkov uporabili odločitveno drevo v članku [6]. Članek [24] različne modele za strojno učenje razloži s SHAP vrednostmi, ki jih potrdi na podlagi KDE (kernel density estimation) grafov. V članku [1] je predstavljena izbira podmnožice najbolj pomembnih atributov s hibridnim pristopom. Z algoritmom KMEANS so na podlagi gruč v članku [21] iskali povezavo med tipom napada in uporabljenim protokolom. Podatkovno



Slika 4.17: Rezultat T-SNE + HDBSCAN algoritma na učni množici. Algoritem je našel 7 gruč. DBSV indeks je 0.5617.

množico so opisali tudi z odločitvenimi pravili v članku [12].

Tabela 4.5: Verjetnosti razredov v gručah

Razred	**G0**	**G1**	**G2**	**G3**	**G4**	**G5**	**G6**
0	0.966	0	0.344	0.993	0.787	0.641	0.972
1	0.334	1	0.656	0.007	0.213	0.359	0.028

4.4 Število gruč

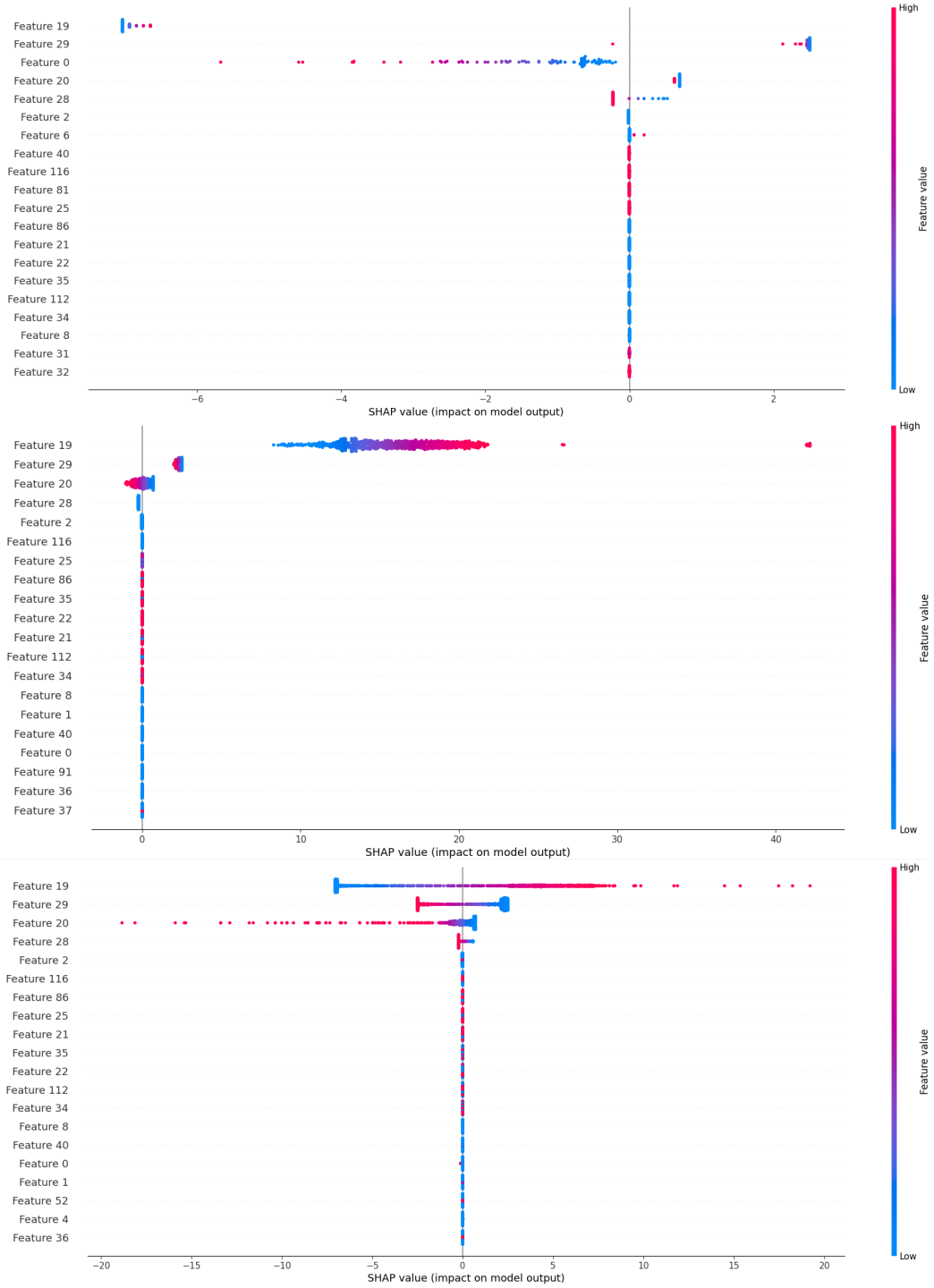
Število gruč je pri algoritmih kot so k-means in MDEC igralo pomembno vlogo prav tako kot pri razlagi. Pri umetnih podatkih smo vedeli število gruč, tako da je bila smiselno najboljša ocena gručenja prav takrat, ko smo za parameter števila gruč uporabili resnico. Težje je ta parameter najti v realnih podatkih, ko ne veš koliko gruč se skriva v njih. Takrat je bilo smiselno

Tabela 4.6: Odločitvena pravila za meje posameznih gruč

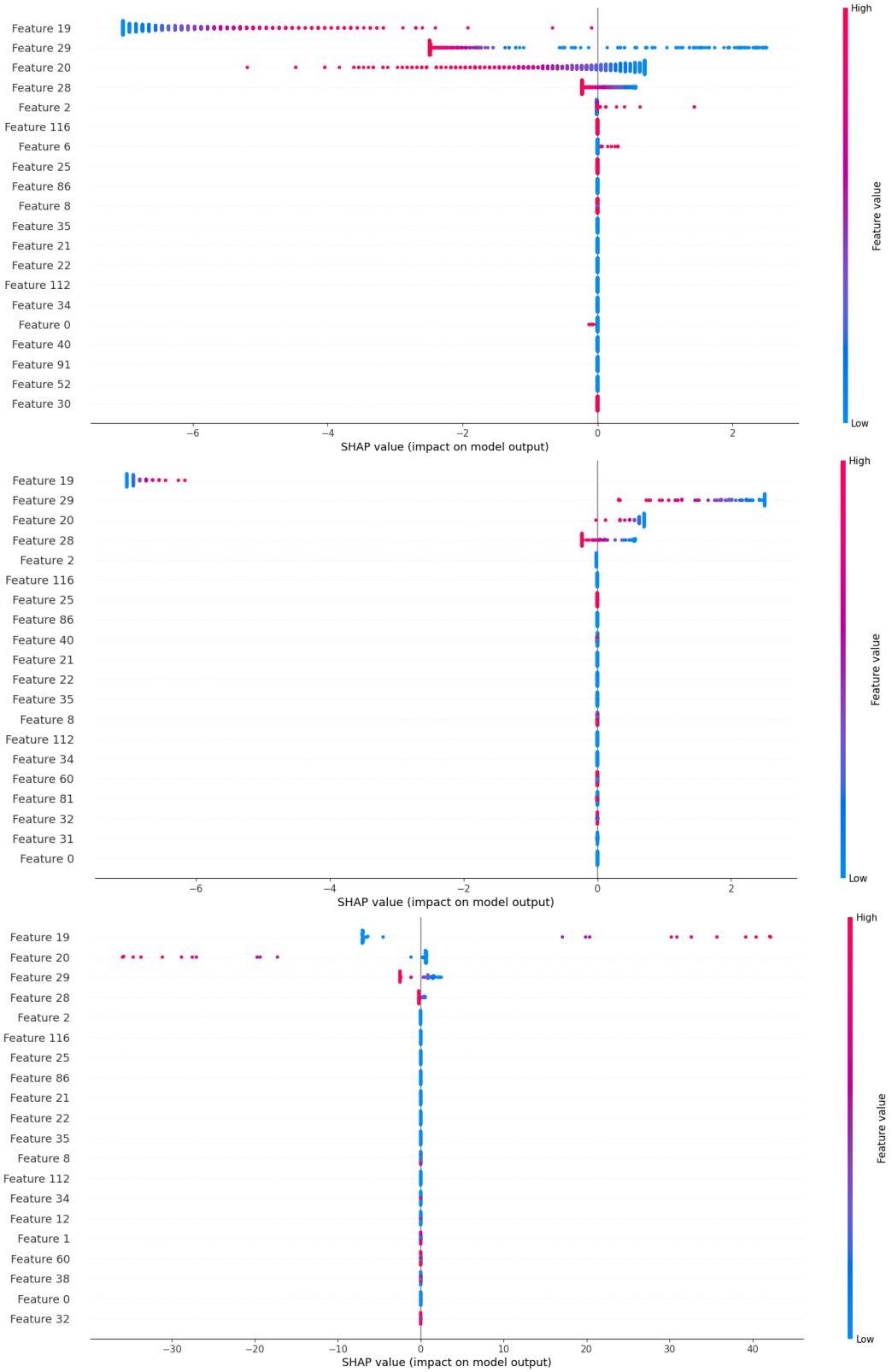
Gruča	**Odločitvena pravila**	**Precision**	**Recall**
0	F0 > 640.5 in F2 > 52.5	1.0	1.0
0	F0 > 629.5 in F30 <= 0.7755	1.0	1.0
0	F0 > 724.5 in F31 > 0.01	1.0	1.0
0	F0 > 640.5 in F1 <= 2567830.0	1.0	1.0
0	F0 > 640.5 in F28 > 11.5	1.0	1.0
0	F0 > 640.5 in F33 <= 0.02	1.0	1.0
0	F0 > 640.5 in F60 <= 0.5	1.0	1.0
1	F116 <= 0.5 in F19 > 162.0 in F32 <= 0.005	1.0	1.0
1	F19 > 162.0 in F25 <= 0.265 in F28 > 185.0	1.0	1.0
1	F19 > 162.0 in F30 <= 0.115 in F32 <= 0.005	1.0	0.9991
1	F19 > 162.0 in F28 > 185.0 in F30 <= 0.115	1.0	0.9987
1	F19 > 162.0 in F20 <= 49.0 in F32 <= 0.005	1.0	0.9985
1	F19 > 162.0 in F28 > 185.0 in F29 <= 48.5	1.0	0.9985
1	F19 > 162.0 in F32 <= 0.005 in F39 > 0.5	1.0	0.9985
1	F1 <= 14.0 in F19 > 162.0 in F32 <= 0.005	1.0	0.9985
1	F116 <= 0.5 in F19 > 162.0 in F28 > 185.0	1.0	0.9985
2	F1 <= 137.0 in F19 <= 162.0 in F2 <= 1269.5	1.0	0.991
3	F1 <= 27526.0 in F63 > 0.5 in F8 > 0.5	0.999	0.984
3	F1 <= 27540.5 in F1 > 71.0 in F63 > 0.5	1.0	0.981
3	F1 <= 27540.5 in F2 > 36.5 in F63 > 0.5	1.0	0.981
3	F23 <= 0.135 in F63 > 0.5 in F9 <= 0.5	0.998	0.982
3	F1 > 70.5 in F63 > 0.5 in F9 <= 0.5	0.999	0.979
3	F63 > 0.5 in F8 > 0.5 in F9 <= 0.5	0.999	0.978
4	F1 <= 475.0 in F1 > 143.5 in F102 <= 0.5 in F60 > 0.5	1.0	0.66
5	F1 <= 1482.5 in F1 > 475.0 in F20 <= 279.0 in F60 > 0.5	1.0	0.77
6	F1 > 598.0 in F91 > 0.5	1.0	0.812
6	F1 > 644.5 in F12 <= 4.5 in F32 <= 0.315 in F33 <= 0.11 in F6 <= 2.5 in F8 > 0.5 in F97 <= 0.5	1.0	0.812

Tabela 4.7: Medoidi z atributi, ki se razlikujejo.

Atribut	**M0**	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**
F0	3058	0	0	0	0	0	5
F1	147	0	0	286	201	854	1536
F2	105	0	0	1768	0	0	328
F8	0	0	0	1	1	0	1
F19	1	240	109	24	6	2	1
F20	1	1	9	35	6	2	1
F21	0	1	1	0	0	0	0
F22	0	1	1	0	0	0	0
F25	1	0.04	0.08	1	1	1	1
F26	0	0.06	0.06	0	0	0	0
F27	0	0	0	0.11	0	0	0
F28	255	255	255	143	170	106	80
F29	1	1	15	255	28	100	154
F30	0	0.04	0.06	1	0.09	0.47	0.98
F31	0.6	0.07	0.07	0	0.02	0.04	0.03
F32	0.93	0	0	0.01	0.09	0.47	0.01
F33	0	0	0	0.01	0.07	0.02	0.01
F34	0	1	1	0	0	0	0
F35	0	1	1	0	0	0	0
F39	0	1	1	1	1	1	1
F40	1	0	0	0	0	0	0
F60	0	0	0	0	1	1	0
F63	0	0	0	1	0	0	0
F81	1	0	0	0	0	0	0
F86	0	1	1	0	0	0	0
F91	0	0	0	0	0	0	1
F112	0	1	1	0	0	0	0
F116	1	1	0	1	1	1	1
Class	0	1	1	0	0	0	0



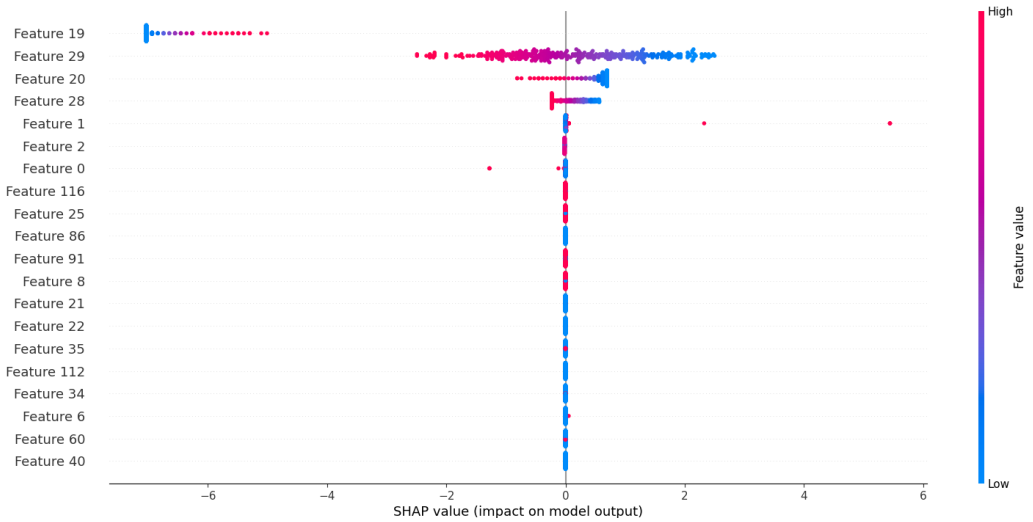
Slika 4.18: Grafi s SHAP vrednostmi za gruče 0-2



Slika 4.19: Grafi s SHAP vrednostmi za gruče 3-5

Tabela 4.8: Pravila za primere različnih razredov znotraj gruče. V oklepajih so zapisane vrednosti Precision in Recall.

Gruča	**Pravila za razred 0**	**Pravila za razred 1**
2	F111 <= 0.5 in F25 > 0.49 in F29 > 1.5 in F31 <= 0.845 in F33 <= 0.44 in F35 <= 0.03 in F4 <= 1.5 in F54 <= 0.5 (1,0.96)	F1 <= 74.5 in F25 <= 0.49 (0.9992,0.93)
5	F30 <= 0.665 (1,1)	F30 > 0.665 (1,1)



Slika 4.20: Graf s SHAP vrednostmi za gručo 6

preučiti podatke v nižjedimenzionalnih prostorih ter iterirati čez različne vrednosti parametra gruč in iskati najboljšo oceno gručenja. Prav tako se preveliko število gruč pozna pri razlagi, kjer se nestrnjenost gruč kaže v slabo razložljivih prototipih in pravilih, ki bi morali spadati pod isto razlago. Premajhno število gruč pa nam preveč posploši razlago in poda razlage, ki bi morale biti ločene na več množic pravil in prototipov (lahko dobimo nečiste skupine). V obeh primerih je lahko razlaga neverodostojna.

4.5 Smiselnost prototipov

Prototipi kot so medoidi v našem primeru so smiselni takrat, ko je gruča homogena in jo lahko predstaviš z reprezentativnim primerom, ki velja za element gruče, ki je najbolj avtentičen. Pomemben je tudi razpored primerov z različnimi razredi v gruči, saj lahko medoid predstavlja manjšino gruče v primeru, da so primeri z manjšinjskim razredom razporejeni v središču sferične gruče. Prav tako nam medoid ne pove katera značilka je najpomembnejša. Če vzamemo podolgovato gručo v dvodimenzionalnem prostoru v obliki linije, nam je pri medoidu bolj pomembna le ena dimenzija, kar ne bo razvidno samo iz prototipa. Seveda si pri takšni težavi lahko pomagamo s SHAP vrednostmi. V visokih dimenzijah so medoidi zelo neintuitivni in nepregledni, kar se ponovno rešuje s SHAP vrednostmi za pomembnost atributov.

4.6 Smiselnost pravil

Pri odločitvenih pravilih se lahko zgodi, da so gruče dobro ločene v eni dimenziji in slabše v neki drugi. Tako je smiselno gledati samo pogoj v pravilu, ki boljše loči skupino. Prav tako se v visokih dimenzijah zgodi, da veliko različnih podmnožic pravil dobro loči skupino. Tu si lahko pomagamo s SHAP vrednostmi za gručo ter za posamezen primer, ki nam povejo katere značilke so pomembne za dani klasifikator in to upoštevamo v pogojih pravil.

Vredno je omeniti, da pravila delijo prostor s hiperpravokotniki, kar pomeni, da ne zajamejo kompleksnih oblik v celoti, kar tudi poenostavi razlago. Smiselnost se da oceniti tudi z vrednostmi Precision in Recall, ki jih model vrne zraven pravil.

Poglavje 5

Zaključek

V diplomski nalogi smo predstavili metodo razlage klasifikatorja na podlagi podkonceptov (gruče primerov istega razreda) z uporabo algoritmov za gručenje, odločitvenih pravil, medoidov in grafov s SHAP vrednostmi. Poleg tega smo izvedli eksperimente s štirimi algoritmi za gručenje na 2 umetnih in 1 realni (MNIST) podatkovni množici. Za izbor parametrov in oceno kvalitete gručenja smo uporabili hevristiko silhuete in DBCV indeks. Pri visokodimenzionalnih podatkih je bila uporabljena tudi tehnika za zmanjšanje dimenzij T-SNE. Pri primerjava algoritmov za gručenje smo ugotovili, da se algoritem MDEC slabo odreže v nizkodimenzionalnih prostorih (2D), medtem ko se je dobro odrezal v visokodimenzionalnem prostoru. Algoritem KMEANS se je odrezal dobro, kjer so bile oblike gruč sferične, tudi če so si bile gruče blizu. V visokodimenzionalnem prostoru se je bolje odrezal v kombinaciji s T-SNE, a vseeno ne tako dobro kot HDBSCAN. Algoritma DBSCAN in HDBSCAN sta brez T-SNE veliko primerov klasificirala kot šum. Algoritem DBSCAN je v kombinaciji s T-SNE slabo gručil, medtem ko je na umetni podatkovni množici "Trakovi 4-3"gručil zelo dobro. Algoritem HDBSCAN se je izkazal za bolj stabilnega (od DBSCAN) v kombinaciji s T-SNE, zato je bil uporabljen tudi pri razlagi realne podatkovne množice KDD99. Postopek razlage se je v večini primerov realnih podatkovnih baz izkazal za neuporabnega, saj imajo redke podatkovne množice lastnost, kjer so primeri razreda razdeljeni

v 2 ali več gruči (premajhna razčlenjenost primerov istega razreda na gruče). Predstavljena metoda razlage razdeli podatkovno množico na manjše dele, kar naredi razlago bolj razumljivo in bolj podrobno kot če bi metode uporabili na celotni podatkovni množici. Pri razlagi z medoidi so se pojavile težave v visokodimenzionalnem prostoru, saj je medoid z veliko atributi dokaj neintuitiven in težko razumljiv. Na tem mestu smo si pomagali s SHAP vrednostmi. Ogledali smo si le atribute pomembne za klasifikator in obdržali smo le atribute v katerih so se medoidih različnih gruč razlikovali. Omenimo tudi odločitvena pravila, ki so v nekaterih primerih vsebovala atribute nepomembne za klasifikator, a so nam vseeno podala dodatno informacijo o gruči. Dodatno težavo predstavljajo gruče sestavljene iz 2 ali več razredov, saj medoid ne predstavi celotne slike. Znova smo uporabili odločitvena pravila, tokrat znotraj gruče, da so nam pomagala ločiti primere enega razreda od primerov preostalih razredov. Raziskave se lahko nadaljujejo v smeri dodatnih razlag za posamezno gručo. V to je vključena vizualizacija dodatnih primerov (ne samo medoida) iz različnih delov gruče in dodatna raziskava v smeri nečistih gruč, predvsem kako naj bo gruča predstavljena. Dodatno bi se lahko atributi opisali z statističnimi metodami, saj o samem atributu ne izveš veliko v naši razlagi. Uporabna raziskava bi bila tudi o primernosti podatkovnih množic za to metodo (avtomatsko testiranje ali je metoda aplikativna za podatkovno množico).

Literatura

[1] Nelcileno Araújo in sod. "Identifying important characteristics in the KDD99 intrusion detection dataset by feature selection using a hybrid approach". V: *2010 17th International Conference on Telecommunications*. IEEE. 2010, str. 552–558.

[2] Jason Brownlee. "How to Develop a CNN for MNIST Handwritten Digit Classification". V: *Machine Learning Mastery* (2019). URL: `https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/`.

[3] Alfred DeMaris. "A tutorial in logistic regression". V: *Journal of Marriage and the Family* (1995), str. 956–968.

[4] Charles Frenzel. "How To Tune HDBSCAN". V: *Towards Data Science* (2021). URL: `https://towardsdatascience.com/tuning-with-hdbscan-149865ac2970`.

[5] Jerome H Friedman in Bogdan E Popescu. "Predictive learning via rule ensembles". V: (2008).

[6] Jim Georges in Anne H Milley. "Kdd'99 competition: Knowledge discovery contest". V: *ACM SIGKDD Explorations Newsletter* 1.2 (2000), str. 79–84.

[7] Riccardo Guidotti in sod. "A survey of methods for explaining black box models". V: *ACM computing surveys (CSUR)* 51.5 (2018), str. 1–42.

[8] John A Hartigan in Manchek A Wong. "Algorithm AS 136: A k-means clustering algorithm". V: *Journal of the royal statistical society. series c (applied statistics)* 28.1 (1979), str. 100–108.

[9] Dong Huang in sod. "Toward multidiversified ensemble clustering of high-dimensional data: From subspaces to metrics and beyond". V: *IEEE Transactions on Cybernetics* 52.11 (2021), str. 12231–12244.

[10] Kamran Khan in sod. "DBSCAN: Past, present and future". V: *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*. IEEE. 2014, str. 232–238.

[11] Daniel Kleine. "Detecting knee- / elbow points in a graph of a function". V: *Towards Data Science* (2021). URL: `https://towardsdatascience.com/detecting-knee-elbow-points-in-a-graph-d13fc517a63c`.

[12] Shih-Wei Lin in sod. "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection". V: *Applied Soft Computing* 12.10 (2012), str. 3285–3290.

[13] Scott M Lundberg in Su-In Lee. "A unified approach to interpreting model predictions". V: *Advances in neural information processing systems* 30 (2017).

[14] Basim Mahbooba in sod. "Explainable artificial intelligence (XAI) to enhance trust management in intrusion detection systems using decision tree model". V: *Complexity* 2021 (2021), str. 1–11.

[15] Leland McInnes, John Healy in Steve Astels. "hdbscan: Hierarchical density based clustering." V: *J. Open Source Softw.* 2.11 (2017), str. 205.

[16] Davoud Moulavi in sod. "Density-based clustering validation". V: *Proceedings of the 2014 SIAM international conference on data mining*. SIAM. 2014, str. 839–847.

[17] Tara Mullin. "DBSCAN Parameter Estimation Using Python". V: *Medium* (2020). URL: https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd.

[18] Ashutosh Nayak. "XGBoost: An Intuitive Explanation". V: *Towards Data Science* (2019). URL: https://towardsdatascience.com/xgboost-an-intuitive-explanation-88eb32a48eff.

[19] Conor Nugent in Pádraig Cunningham. "A case-based explanation system for black-box systems". V: *Artificial Intelligence Review* 24 (2005), str. 163–178.

[20] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". V: *Journal of Computational and Applied Mathematics* 20 (1987), str. 53–65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. URL: https://www.sciencedirect.com/science/article/pii/0377042787901257.

[21] Mohammad Khubeb Siddiqui in Shams Naahid. "Analysis of KDD CUP 99 dataset using clustering based data mining". V: *International Journal of Database Theory and Application* 6.5 (2013), str. 23–34.

[22] Saurabh Singh. "Building an Intrusion Detection System using KDD Cup'99 Dataset". V: *Medium* (2020). URL: https://medium.com/analytics-vidhya/building-an-intrusion-detection-model-using-kdd-cup99-dataset-fb4cba4189ed.

[23] Laurens Van der Maaten in Geoffrey Hinton. "Visualizing data using t-SNE." V: *Journal of machine learning research* 9.11 (2008).

[24] Remah Younisse, Ashraf Ahmad in Qasem Abu Al-Haija. "Explaining intrusion detection-based convolutional neural networks using shapley additive explanations (shap)". V: *Big Data and Cognitive Computing* 6.4 (2022), str. 126.