Prediktivno modeliranje kvarova

Pilošta, Bruno

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Bruno Pilošta

PREDIKTIVNO MODELIRANJE KVAROVA

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Bruno Pilošta

Matični broj: 0016129843

Studij: Informacijski sustavi

PREDIKTIVNO MODELIRANJE KVAROVA

ZAVRŠNI RAD

Mentorica:

Doc. dr. sc. Dijana Oreški

Varaždin, kolovoz 2020.

Bruno Pilošta

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Ovaj rad bavi se proučavanjem prediktivnog modeliranja kvarova na diskovnim jedinicama. Praktični dio rada odnosi se na predviđanje kvara na diskovnoj jedinici temeljem stanja diska dan prije pojave kvara. U uvodnom djelu dan je opis teme rada, razlozi te motivacije za odabir ove teme. U nastavku se opisuju prethodna istraživanja, navode se i opisuju metode strojnog učenja koje su važne za izradu modela kao i opis S.M.A.R.T. sustava. S.M.A.R.T. sustav izvještava korisnika računala o trenutnom stanju diskova pomoću velikog broja pokazatelja kao što su broj grešaka, čitanja ili zapisivanja. Pomoću preuzetih skupova podataka o stanju diskova kasnije se izrađuju modeli predviđanja potencijalnih kvarova. Algoritmi izrade modela su *Random Forest,* metoda potpornih vektora, stabla odlučivanja i algoritam k najbližih susjeda. Izrađeni modeli se analiziraju pomoću odabranih metrika te se na kraju uspoređuju dobiveni rezultati i donosi zaključak ovog rada.

Ključne riječi: prediktivno modeliranje, rudarenje podataka, strojno učenje, čvrsti disk, predviđanje kvara, S.M.A.R.T.

Sadržaj

1. Uvod

Tema ovog rada jest proučavanje metoda prediktivnog modeliranja kvara diskovnih jedinica, ali i izrada prediktivnih modela koji se temelje na algoritmima nadziranog strojnog učenja. Danas postoji veliki broj metoda, odnosno, algoritama kojima se pokušavaju izraditi prediktivni modeli, no za ovaj rad odabrao sam četiri izrazito popularna, ali i snažna algoritma. Osim proučavanja metoda strojnog učenja vrlo važna komponenta prediktivnog modeliranja jest i rudarenje podacima kako bi otkrili koji su podaci zapravo reprezentativni za područje u kojem želimo izraditi prediktivne modele.

Pojavom SSD diskova čvrsti diskovi su i dalje jedan od neizostavnih dijelova svakog računala zbog velikog kapaciteta uz nisku cijenu, no oni su također i puno više podložni kvarovima zbog čega je rana identifikacija kvara nužna. Kako sam se i osobno susreo sa nekoliko ispada diskovnih jedinica, ali i netočnih upozorenja o kvaru na diskovima, odlučio sam se istražiti postoji li bolji način otkrivanja budućeg kvara na diskovima. Današnji standardi otkrivanja kvarova temelje se na usporedbi S.M.A.R.T. stanja diskova sa tvornički zadanim pragovima koje su proizvođači odredili prilikom proizvodnje diskova. Na realnom primjeru pokazat će se kako takav pristup često ne prikazuje pravo „zdravlje“ diskova, ali ipak daje dobro upozorenje o mogućim novonastalim problemima. Sve navedeno govori o važnosti predviđanja kvarova na, jednoj od kvarovima najpodložnijih komponenata svakog računala, disku. Za razliku od prethodnih istraživanja koja se većinom bave predviđanjem kvara na diskovnim jedinicama nekoliko dana ili tjedana prije, odlučio sam se istražiti mogućnost predviđanja kvara na disku temeljem S.M.A.R.T. stanja diska dan prije njegove pojave.

Prije pisanja ovog rada nisam imao velikog znanja o samom prediktivnom modeliranju i strojnom učenju, no uz određena znanja iz područja statistike i matematike koje sam stekao na fakultetu odlučio sam se za istraživanje ovog, osobno, vrlo zanimljivog područja. Smatram da sam u proteklih nekoliko mjeseci stekao mnogo iskustva kroz istraživanje ovog izrazito traženog, zanimljivog, ali i zahtjevnog područja koje želim i dalje nastaviti proučavati.

2. Metode i tehnike rada

Većina dostupnih istraživanja temelji se na promatranju stanja diskova kroz određeno vremensko razdoblje te predviđanje kvara nekoliko dana/tjedana unaprijed. U ovom istraživanju pokušat ću predvidjeti kvar čvrstog diska dan prije njegove pojave pomoću algoritama i metoda strojnog učenja. Algoritme koje ću za izradu prediktivnih modela koristiti prilikom ovog istraživanja su *Random Forest* algoritam, metoda potpornih vektora, stablo odlučivanja te k najbližih susjeda.

Za izradu prediktivnih modela korišteno je nekoliko alata. Najvažniji alat korišten je Jupyter Notebook[1] uz programski jezik Python verzije 3.7.3. Za učitavanje i uređivanje skupova podataka korištene su Python biblioteke „pandas“[2], vizualizacijske biblioteke „Seaborn“[3] i „Matplotlib“[4] te biblioteka za strojno učenje „scikit-learn“[5]. Algoritmi strojnog učenja izvode se na računalu sa osnovnim specifikacijama:

- Intel četverojezgreni i5-6500 3.2GHz procesor
- 8 GB radne memorije
- ADATA SSD SP580 128GB

[1] https://jupyter.org/
[2] https://pandas.pydata.org/
[3] https://seaborn.pydata.org/
[4] https://matplotlib.org/
[5] https://scikit-learn.org/stable/

3. Prethodna istraživanja

Tijekom godina razni znanstvenici pokušavaju izraditi prediktivni model za ranu identifikaciju ispada diskovnih jedinica. Danas je dostupan veliki broj istraživanja koja uključuju izradu prediktivnih modela, a u ovom radu će se opisati nekoliko njih.

U istraživanju pod nazivom „Hard Drive Failure Prediction Using Classification and Regression Trees“ [1] predloženi su prediktivni modeli temeljen na klasifikacijskim i regresijskim stablima. Navedeni modeli predviđaju potencijalni ispad diskovne jedinice jedan tjedan unaprijed uz FDR (Failure Detection Rate) od 95.49% i FAR (Failure Alarm Rate) od 0.09%. Osim klasifikacijskog modela koji predviđa ispad diskovne jedinice unutar tjedan dana predložen je i model koji određuje „zdravlje“ čvrstog diska temeljen na regresijskim stablima. Za razliku od binarnog klasifikatora (disk ispravan ili ne) navedeni model pridodaje disku stvarnu vrijednost koja određuje njegovo trenutno stanje ispravnosti.

U istraživanju naziva „Bayesian Approaches to Failure Prediction for Disk Drives“ [2] pristupa se problemu identifikacije kvara 48 sati prije nego se on dogodi na dva načina. Prvi način je pomoću mješavine „Naive Bayes“ klasifikatora i EM (expectation-maximization) algoritma. Drugi način je obični „Naive Bayes“ klasifikator. Kako bi se pronašao disk kod kojeg je moguća neispravnost u skoroj budućnosti pokušava se pronaći anomalija. Zdravi diskovi imaju vrlo slična stanja, to jest, SMART podatke dok potencijalno neispravni diskovi imaju određene vrijednosti SMART atributa koji odstupaju od normale. Na taj način se pomoću navedenih metoda pokušavaju pronaći anomalije, odnosno predvidjeti ispadi diskovnih jedinica. Metoda NBEM (Naive Bayes EM) daje bolje rezultate od standarda industrije iz 2000. godine kod koje je uz FPR (false positive rate) 0.002 TPR (true positive rate) između 0.04 i 0.11. Njihova metoda detekcije kvara uz FPR 0.002 daje TPR od 0.3.

Izrazito dobre rezultate pokazalo je i istraživanje „Random-forest-based failure prediction for hard disk drives“ [3] . Ovdje se za predviđanje kvara koristi metoda strojnog učenja „Random-Forest“. Za eksperiment korištena su tri skupa SMART podataka. Rezultati na prvom skupu koji se sastoji od 22,962 ispravna i 433 neispravna diska bili su FDR 97.67% uz FAR od 0.017%, drugom skupu (2892 ispravna, 1344 neispravna) FDR 100% uz FAR 1.764% i trećem skupu (35,521 ispravna, 1041) FDR 94.89% uz FAR 0.44%.

U istraživanju „Hard drive failure prediction using non-parametric statistical methods“ [4] uspoređuje se nekoliko metoda za predviđanje kvara čvrstog diska (Support Vector Machines, Wilcoxon-Mann-Whitney rank-sum test, klasteriziranje). Najbolje rezultate pokazalo je korištenje „Wilcoxon-Mann-Whitney rank-sum“ testa sa FDR-om od 33.2% i FAR-om od 0.5%.

Skup podataka nad kojim se izvršilo istraživanje sastoji se od 369 diskova od kojih je 178 ispravnih i 191 neispravna.

4. Opis metoda

4.1. Tipovi strojnog učenja

Strojno učenje može se klasificirati na nekoliko načina, no najpoznatija klasifikacija je na temelju ljudske uključenosti u strojno učenje. Tipovi strojnog učenja prema navedenoj klasifikaciji mogu biti nadzirano (supervised) učenje, nenadzirano (unsupervised), polunadzirano (semisupervised) i „Reinforcement“ učenje [5, str. 7].

Kod nadziranog ili „supervised“ učenja algoritmu je potrebno dati označene podatke (training data) na temelju kojih algoritam uči i na kraju može klasificirati ili odrediti vrijednost novim podacima (test data). Metode nadziranog učenja su klasifikacija i regresija [5, str. 8]. Primjer regresije može biti da želimo, na temelju S.M.A.R.T. vrijednosti atributa (prediktora), odrediti koliko će još sati od trenutka promatranja neki disk raditi do njegovog kvara. Kod klasifikacije želimo, pomoću prediktora, odrediti vrijednost kategorijske varijable. Algoritam u ovom slučaju pregledava dani skup podataka koji se sastoji od prediktora i već klasificiranih varijabli i „uči“ koje kombinacije prediktora utječu na ciljnu (kategorijsku) varijablu (eng. label). Na temelju naučenih kombinacija algoritam može klasificirati još neklasificirane zapise u testnom skupu podataka [6, str. 301]. Pomoću algoritama klasifikacije u nastavku rada će se pokušati istražiti je li moguće, uz dane S.M.A.R.T. atribute (prediktore), odrediti kvar na disku dan prije pojave istog.

4.2. Algoritmi klasifikacije

Danas postoji veliki broj algoritama klasifikacije. Određeni algoritmi mogu osim za klasifikaciju poslužiti i za regresiju. Algoritmi koji će se koristiti u ovom radu su:

- Algoritam k-najbližih susjeda
- Stabla odlučivanja
- *Random-Forest* algoritam
- *SVM* (Support Vector Machines) algoritam

4.3. Algoritam K-najbližih susjeda

K-najbližih susjeda (eng. K-nearest neighbors, kraće KNN) je algoritam nadziranog strojnog učenja te je jedan od algoritama koji se može koristiti kod regresije i klasifikacije. Također, KNN algoritam je jedan od najkorištenijih algoritama za klasifikaciju zbog svoje jednostavnosti i brze izračunljivosti kod manjeg skupa podataka [7].

Algoritam KNN može se opisati na sljedeći način [7]:

1. Imamo zadani skup podataka
2. Odabiremo neki broj K kao broj susjeda
3. Za svaki zapis u skupu podataka radi se sljedeće:
    a. Računa se udaljenost između trenutnog testnog zapisa i svakog od trening zapisa
    b. Sortiraju se vrijednosti prema udaljenost od najmanje do najveće
    c. Gleda se prvih K redaka te se uzima vrijednost ciljne kategorijske varijable koja se najviše puta pojavila (mod) ako se radi klasifikaciji ili srednja vrijednost kontinuiranih varijabli spomenutih K redaka ako se radi o regresiji.

4.3.1. Funkcija udaljenost

Kao što je spomenuto u prethodnom poglavlju kod KNN algoritma potrebno je izračunati udaljenost između podataka. Udaljenost nam zapravo govori kakva je sličnost između podataka na temelju koje algoritam klasificira podatke u kategorije. Udaljenost se računa na dva načina.

Prvi način je za kontinuirane varijable, a koristi se „Euklidska udaljenost“, to jest, udaljenost između dvije točke u Euklidskom prostoru [6, str. 305]. Formula za Euklidsku udaljenost glasi:

$$D(x, y) = \sqrt{\sum (x_i - y_i)^2}$$

pri čemu su $x = x_1, x_2, \dots, x_m$ i $y = y_1, y_2, \dots, y_m$ m vrijednosti atributa za dva zapisa [6, str. 305]. Primjer: Postoje dva diska sa jednim S.M.A.R.T. atributom „*Raw Read Error Rate*“. Prvi disk ima vrijednost za navedeni atribut $x_1 = 55$, a drugi disk ima vrijednost $y_1 = 22$. Udaljenost u Euklidskom prostoru računa se:

$$D(x, y) = \sqrt{(x_1 - y_1)^2} = \sqrt{(55 - 22)^2} = \sqrt{1.089} = 33$$

Na taj način možemo izračunati i za ostale zapise, te na kraju u obzir uzimamo samo K najkraćih udaljenosti. U primjeru je pokazan slučaj u jednodimenzionalnom Euklidskom prostoru (jedan atribut/prediktor), no prema formuli za Euklidsku udaljenost moguće je računati udaljenost za višedimenzionalne prostore.

Kod računanja udaljenosti moramo u obzir uzimati i atribute koji poprimaju vrlo velike vrijednosti u odnosu na ostale atribute zbog čega izračunata udaljenost može biti vrlo velika bez obzira na važnost navedenog atributa kod klasifikacije. Zbog toga se preporučuje u fazi čišćenja podataka, osobito kod KNN algoritma, raditi normalizaciju vrijednosti atributa [6, str. 305].

Drugi slučaj računanja udaljenosti uključuje kategorijske varijable. U ovom slučaju koristimo funkciju [6, str. 306]:

$$D(x_i, y_i) = \begin{cases} 0 & ako\ je\ x_i = y_i \\ 1 & u\ ostalim\ slučajevima \end{cases}$$

Primjer: Postoje tri diska, dva sa kvarom i jedan bez. Kategorijska varijabla u ovom slučaju je kvar, odnosno, kvar postoji ili ne. Vrijednost „*kvar*“ za prvi disk možemo zapisati kao $x_1 = 1$, drugi disk s kvarom $y_1 = 1$, a treći disk bez kvara kao $z_1 = 0$. Vrijednosti koje disk može poprimiti za varijablu kvar samo su 0 i 1. Prema navedenoj formuli ako promatramo prvi i drugi disk imamo $x_1 = y_1$ što znači da je udaljenost 0, a u slučaju promatranja prvog diska i trećeg udaljenost je 1 zbog $x_1 \neq z_1$.

Ako imamo skup podataka koji se sastoji od kontinuiranih i kategorijskih varijabli možemo iskoristiti min-max normalizaciju nad kontinuiranim varijablama. Min-max normalizacija normalizira podatke na skalu 0 – 1 [6, str. 31].

4.3.2. Odabir vrijednosti za K

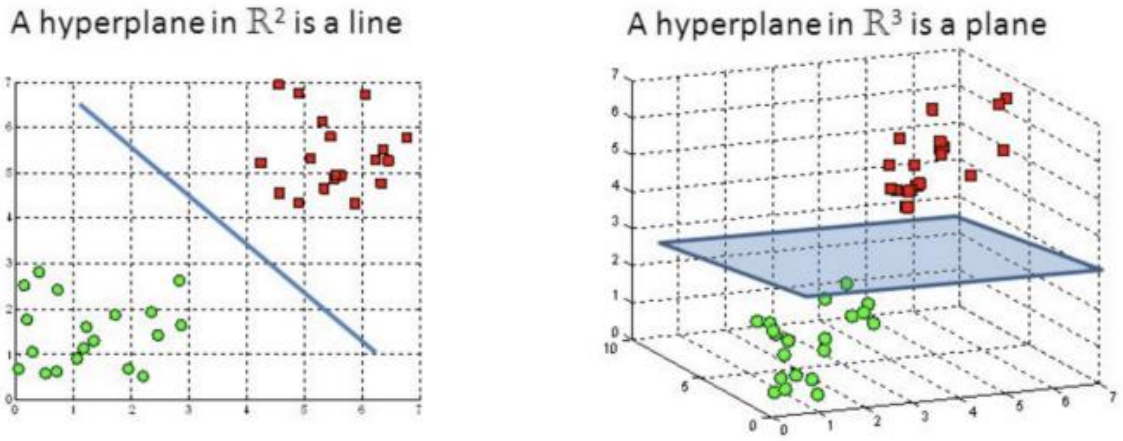
Odabir dovoljno velike vrijednosti za broj susjeda koji će se uzimati u obzir ima značajan utjecaj na performanse modela. Odabir male vrijednosti K, primjerice 1, imat će najčešće nepovoljan utjecaj na performanse modela.

Primjer: Imamo četiri diska i želimo klasificirati peti disk. Disk 1, 2 i 3 imaju kvar dok disk 4 nema. Izračunali smo udaljenosti između petog i ostalih četiri diskova. Udaljenost između diska 5 i 4 je najmanja i iznosi 44 mjerne jedinice, a između petog i ostalih je 45 mjernih jedinica. U slučaju da uzmemo za K = 1, gledamo samo jednog, najbližeg susjeda te će se promatrani disk klasificirati kao njegov najbliži susjed, disk 4, bez kvara. U slučaju da uzimamo K = 3, gledamo tri najbliža susjeda, disk 4 čija je udaljenost 44 i primjerice, disk 1 i 2 sa udaljenosti 45. U ovom slučaju sa K = 3 imamo dva sa kvarom i jedan bez kvara te će disk 5 biti klasificiran kao pokvaren.

Dobra je praksa isprobati izraditi model sa različitim vrijednostima za K te na kraju odabrati model sa najmanjim klasifikacijskim greškama [7].

4.4. Support Vector Machines

Support Vector Machines (metoda potpornih vektora) također je algoritam nadziranog strojnog učenja koji se koristi kod problema klasifikacije, ali i regresije. Algoritam radi na način da se pokušava pronaći linija (u dvodimenzionalnom prostoru) ili hiper-ravnina (u višedimenzionalnim prostorima) koja dijeli podatkovne točke (zapise u dataset-u) u dvije skupine te na taj način klasificira svaku od podatkovnih točaka [8]. SVM je često korišten za kompleksne skupove podataka male i srednje veličine kao i za linearnu i nelinearnu klasifikaciju i regresiju [5, str. 145].



Slika 1 Hiper-ravnine u 2D i 3D prostoru [8]

Na slici 1 možemo vidjeti na koji način izgleda klasifikacija u dvodimenzionalnom realnom prostoru pomoću linije i trodimenzionalnom prostoru pomoću 2D ravnine. Primjer: U ovom slučaju zelene točke mogu označavati zdrave diskove, a crvene točke pokvarene diskove. 2D prostor zapravo označava da promatramo samo dva atributa (npr. smart_1, smart_2), a 3D prostor 3 atributa. Nakon završetka treniranja SVM modela novi zapis možemo klasificirati na način da gledamo pripada li lijevo od linije u 2D prostoru ili desno. Ako se zapis klasificirao lijevo od linije (kao zelena točka na slici) model bi odredio da je navedeni disk ispravan, odnosno neispravan ako bi se klasificirao desno od linije. Na slični način se izvodi klasifikacija u 3D prostoru, no u ovom slučaju se gleda pozicija ispod ili iznad ravnine. Također, za višedimenzionalne prostore klasifikacija se izvodi ovisno o poziciji od hiper-ravnine.

4.4.1. Potporni vektori

Potporni vektori zapravo označavaju podatkovne točke koje određuju poziciju i orijentaciju hiper-ravnine. Svakim dodavanjem novog potpornog vektora ili brisanjem mijenjamo poziciju hiper-ravnine [8].

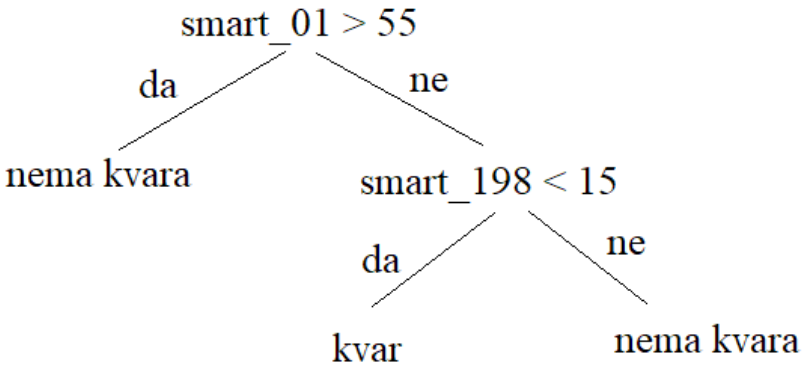
Osim toga, potporni vektori označavaju gdje se nalaze margine hiper-ravnine. Margine hiper-ravnine su udaljenosti ravnine od navedenih potpornih vektora. Ako su margine veće model će biti robusniji, dok sa manjim marginama postoji veća šansa za krivu klasifikaciju [9].

4.4.2. Kernel funkcija

U nekim slučajevima nije moguće linearno odvojiti klase (pomoću linije, ravnine) pa je potrebno upotrijebiti neku od kernel funkcija. Kernel funkcija se koristi za preslikavanje trenutnog prostora u prostor više dimenzije. U prostoru više dimenzije pokušava se pronaći linearna razdvojivost te kad se pronađe ponovno se radi preslikavanje u originalni prostor [10]. Primjerice: u 2D prostoru problem je linearno nerazdvojiv (ne može se pronaći linija razdvajanja) te kernel funkcijom radimo preslikavanje u 3D prostor gdje pronalazimo razdvojivost (ravninu). Nakon toga ponovno se radi preslikavanje u 2D prostor. Najčešće korištene kernel funkcije su polinomijalna funkcija i RBF [10].

4.5. Stabla odlučivanja

Kao i već opisani algoritmi stablo odlučivanja je algoritam nadziranog strojnog učenja te može poslužiti za probleme klasifikacije i regresije. Stablo odlučivanja sastoji se od čvorova, grana i listova [11]. Način rada jednostavnog stabla odlučivanja opisan je na sljedećoj slici.



Slika 2 Jednostavno stablo odlučivanja

Svako stablo odlučivanja kreće od korijenskog čvora. U ovom primjeru ispituje se je li vrijednost atributa „*smart_01*“ nekog diska veća od 55. Ako jest prema izrađenom stablu odlučivanja model će odrediti da nema kvara. U slučaju da je manje ili jednako od 55 kreće se na desni čvor te se ispituje je li vrijednost atributa „*smart_198*“ manja od 15 te ukoliko jest model određuje da postoji kvar, u suprotnom slučaju odredit će se da ne postoji kvar.

Kao što je prikazano na primjeru, testiranje se izvodi izrazito jednostavno po principu „ako ne - onda“. Što se tiče izgradnje stabla postoji nekoliko algoritama koji se koriste, a najpoznatiji i najkorišteniji su CART (Classification and regression trees) algoritam i C4.5 algoritam [6, str. 319].

Skup alata Scikit-Learn koji se koristi u ovom radu za izradu prediktivnog modela koristi CART algoritam za izgradnju stabla. Algoritam dijeli skup podataka za treniranje na dva dijela na temelju promatranog atributa *k* i praga t_k. Traži se par (k, t_k) koji daje najpovoljnije grananje. Za određivanje najbolje podjele najčešće se koristi se *Gini indeks* [5, str. 171, 172].

Stabla odlučivanja imaju brojne prednosti pred ostalim algoritmima strojnog učenja ali također i nedostatke [12].

Prednosti [12]:

- Lagano ih je interpretirati i analizirati
- Mogu se vizualizirati te na taj način pratiti tijek donošenja odluka
- Nije potrebno raditi veliku količinu preprocesuiranja skupa podataka poput normalizacije

Nedostaci [12]**:**

- Često se izrađeni model ne generalizira na nove testne podatke (overfitting)
- Lagano se mogu izraditi stabla koja su pristrana određenoj klasi kod nebalansiranih skupova podataka

4.6. *Random Forest* algoritam

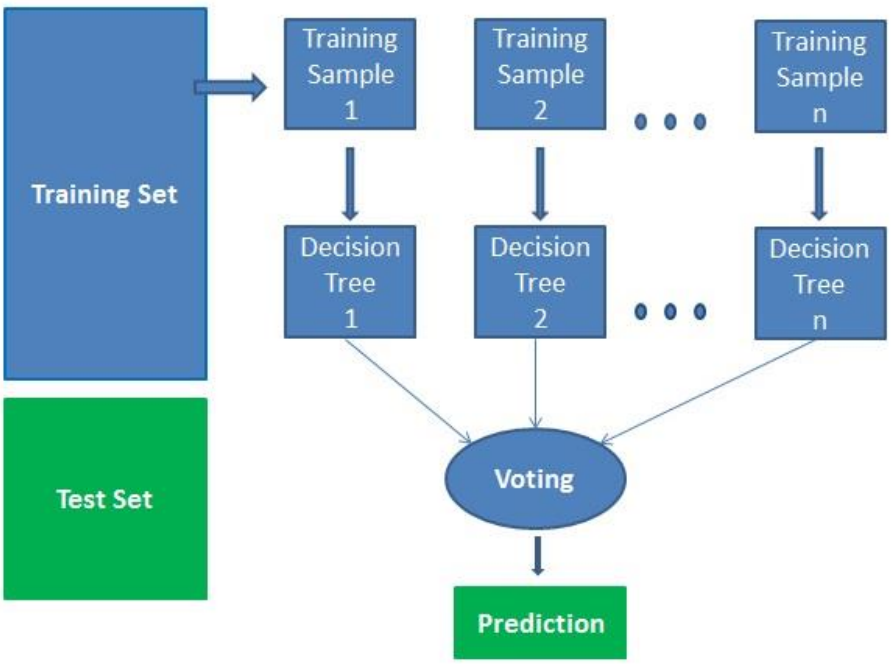
Random Forest algoritam razlikuje se od ranije spomenutih algoritama nadziranog strojnog učenja. *Random Forest* (hrv. „slučajna šuma“) pripada grupi *ensemble* metoda za izradu prediktivnih modela, a temeljena je na stablima odlučivanja. Riječ *ensemble* označava grupu elemenata na koju se gleda samo kao cjelinu. Zbog toga *ensamble* algoritmi kombiniraju više algoritama (iste ili različite vrste) za izradu modela [13]. Ranije spomenuti algoritmi/metode izvršavaju svoju zadaću samostalno, primjerice: koristi se jedno stablo odlučivanja za izradu modela. Kod ove metode koristi se više stabala odlučivanja nad skupom podataka kako bi donosile odluke [14]. Također, *Random Fores*t algoritam može se koristiti za klasifikacijske i regresijske probleme.

Ovaj je algoritam trenutno jedan od najpreciznijih zbog grupiranja većeg broja stabala odlučivanja. Osim toga često je korišten i kod analize i odabira atributa koji imaju najveći utjecaj na klasifikaciju [14].

4.6.1. Način rada algoritma

Rad algoritma može se jednostavno opisati u četiri koraka [14]:

1. Slučajno se odabire *n* uzoraka iz skupa podataka
2. Prema svakom uzorku izrađuje se zasebno stablo odlučivanja (izrađuje se „šuma“ stabala odlučivanja)
3. Izvodi se glasanje/klasifikacija za svaki element iz testnog skupa podataka pomoću svih stabala u „šumi“
4. Odabire se rezultat koji je dobio najviše „glasova“



Slika 3 Rad Random Forest algoritma [14]

Princip rada može se prikazati prema sljedećem primjeru: Imamo skup podataka sa 30 diskova, ispravnih i neispravnih te 1 disk koji želimo testirati. Za broj uzoraka odabiremo 3 pa skup podataka dijelimo na 3 dijela od kojih se svaki sastoji od 10 diskova. Nakon toga izrađuje se stablo odlučivanja za svaki od uzoraka te dobivamo 3 stabla odnosno „šumu“. U sljedećem koraku disk koji želimo testirati prolazi kroz svako od tri izrađena stabla odlučivanja. Dva stabla su disk, primjerice, klasificirala kao neispravan dok ga je jedno stablo klasificiralo kao ispravan. Zadnji korak je odabir klasifikacije za koje su stabla dala najviše glasova, u ovom primjeru to je disk neispravan.

5. S.M.A.R.T sustav

S.M.A.R.T. (Self-Monitoring, Analysis and Reporting Technology) je sustav koji se nalazi u čvrstim diskovima čija je zadaća izvještavanje o stanju diska [15]. Prva pojava S.M.A.R.T. sustava bila je 1992. godine kada je IBM u svojim diskovima uključio takozvani PFA sustav (Predictive Failure Analysis) koja je mjerila razne atribute stanja diska te je slala korisniku poruku upozorenja ukoliko je vrijednost određenog atributa premašila zadani prag. PFA tehnologija je ubrzo dobila naziv S.M.A.R.T. te je postala standard industrije za predviđanje kvara na diskovima [16].

S.M.A.R.T. prikuplja razne podatke o stanju diskova iz kojih je moguće predvidjeti budući kvar na disku. Postoji veliki brojevi S.M.A.R.T. atributa koji opisuju stanje diskova, neki od njih nalaze se na svim diskovima, dok neke S.M.A.R.T. sustave proizvođači proširuju svojim dodatnim atributima [17].

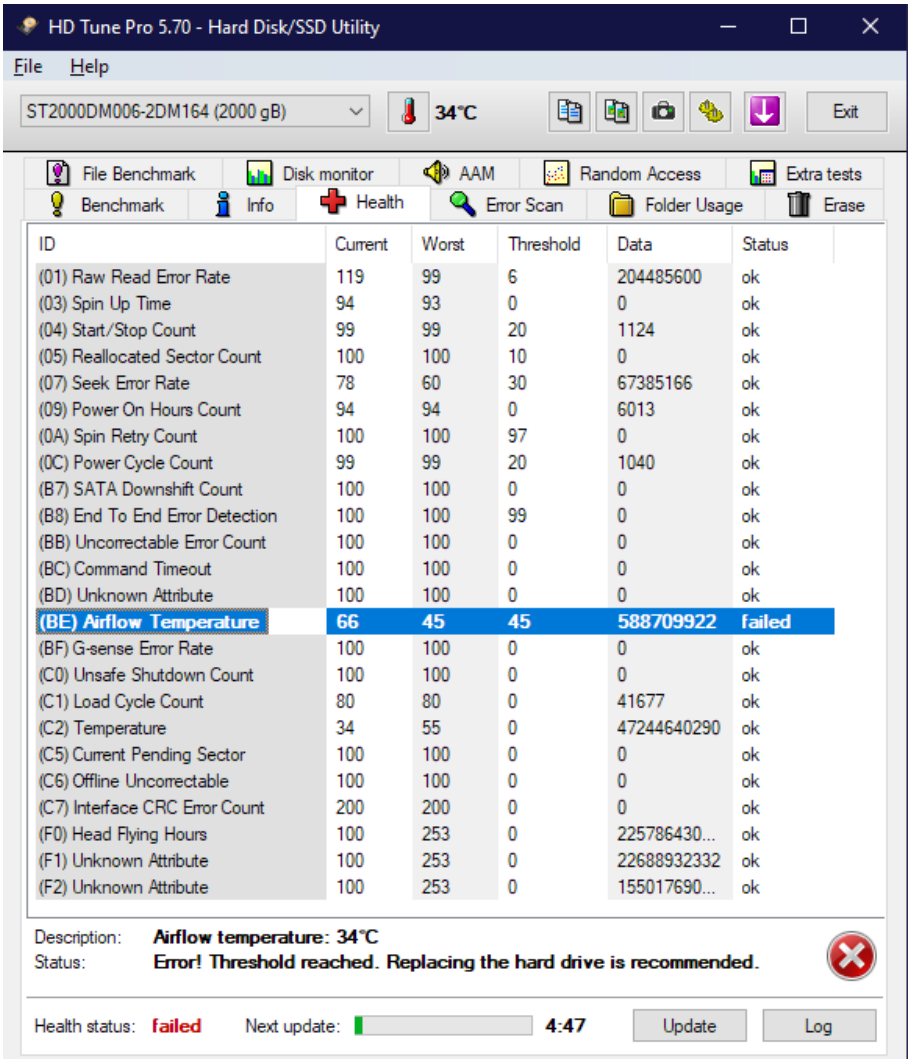
ID #	Attribute name	Meaning of attribute
1	**Raw Read Error Rate**	Count of non-corrected read errors. More errors (i.e. lower attribute value) means worse condition of disk surface.
2	**Throughput Performance**	Overall (general) throughput performance of HDD
3	**Spin Up Time**	Average time of spindle spin up time (from stopped to fully operational)
4	**Start/Stop Count**	Count of spindle start/stop cycles. Raw value probably shows total number of on/off HDD.
5	**Reallocated Sectors Count**	Count of reallocated sectors. When the HDD finds a read/write error, it will mark this sector as "reallocated" and transfer data to the special reserved area. That's why on a modern HDD you can't see "BAD blocks" while testing the surface - all bad blocks are hidden in reallocated sectors. The more sectors reallocated (i.e.lower attribute value), the worse the condition of disk surface.
7	**Seek Error Rate**	Count of seek errors. When your HDD reads data, it positions heads in the needed place. If there is a failure in the mechanical positioning system, a seek error arises. More seek errors (i.e. lower attribute value) - indicates worse condition of a disk surface and disk mechanical subsystem.
8	**Seek Time Performance**	Performance of seek operations. Shows how fast seek operations are going. So, if this attribute is lowering, it's a symptom of bugs in the HDD mechanical subsystem.
9	**Power-On Hours**	Count of hours in power on state. Raw value of this attribute shows total count of hours (or minutes, or seconds - it depends on disk manufacturer) in power on state.

Slika 4 Popis S.M.A.R.T. atributa koji se nalaze na svim diskovima [17]

Na slici 4 mogu se vidjeti S.M.A.R.T. atributi čije vrijednosti prikupljaju svi diskovi. Neki od njih na prvi pogled mogli bi utjecati na potencijalnu pojavu kvara na disku poput „*Raw Read Error Rate*“ koji broji greške kod čitanja sa diska, „*Seek Error Rate*“, „*Realocated Sectors Count*“ ili „*Seek Time Performance*“.

5.1. Pristup S.M.A.R.T. podacima

Svaki korisnik računala može pristupiti S.M.A.R.T. podacima svojeg diska, a i preporučljivo je s vremena na vrijeme pregledati stanje diska, osobito ako korisnik ima pohranjene važne podatke. Danas postoji veliki broj besplatnih aplikacija koje čitaju stanje diska i prikazuju korisniku. Na sljedećoj slici prikazani su S.M.A.R.T. podaci jednog čvrstog diska preko programa HD Tune Pro 5.70[6].



Slika 5 Stanje čvrstog diska

[6] https://www.hdtune.com/

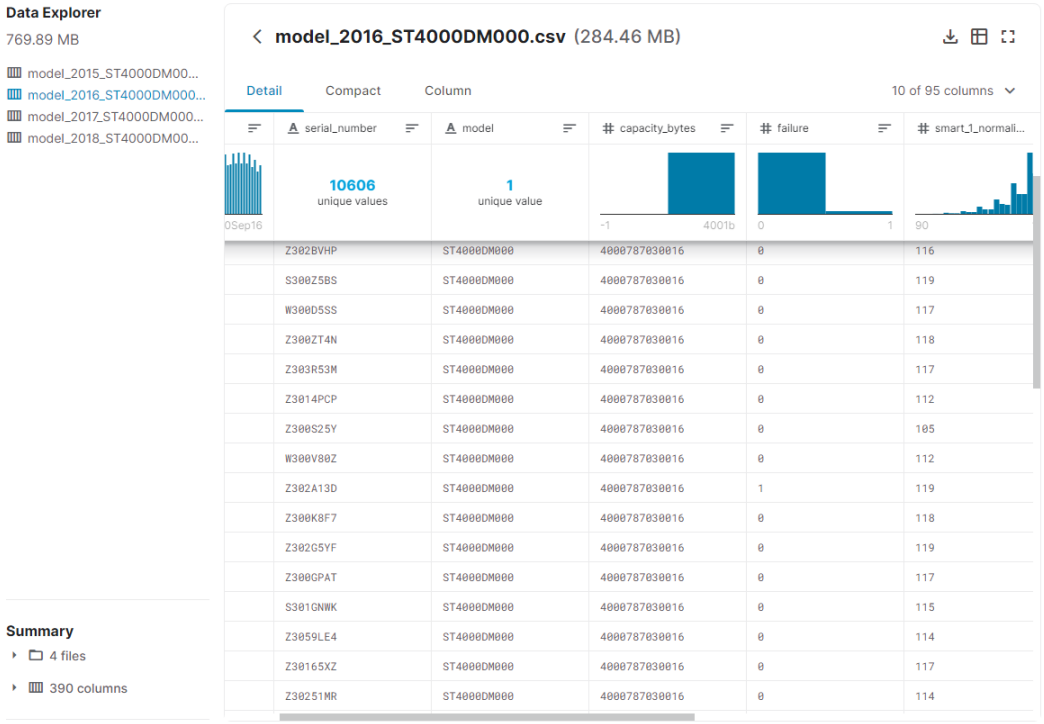
Kao što je navedeno, svaki proizvođač diskova određuje koje S.M.A.R.T. podatke želi da diskovi prikupljaju. Svaki atribut koji se prikuplja ima ID, trenutnu vrijednost (Current), najnižu prikupljenu vrijednost (Worst), prag (Threshold), vrijednost koju je senzor prikupio (Data) i Status. ID označava o kojem se atributu radi, trenutna vrijednost je zapravo normalizirana vrijednost koju je senzor prikupio (Data), prag označava minimalnu dopuštenu vrijednost koju je proizvođač postavio, a Status govori o statusu atributa. U slučaju ovog diska za atribut „*Airflow Temperature*“[7] normalizirana vrijednost je u jednom trenutku došla do praga od 45 te sustav preporučuje zamjenu diska. Naravno, to ne mora značiti da će se u skoroj budućnosti dogoditi potpuni kvar diska, ali je jedan od pokazatelja degradacije diska te je potrebno detaljnije pratiti ponašanje ostalih S.M.A.R.T. atributa. Navedena vrijednost atributa došla je do praga godinu dana prije pisanja ovog rada, a disk i dalje radi normalno zbog čega bi bilo dobro imati točniji i precizniji sustav za predviđanje kvarova na disku koji će se u ovom radu pokušati pronaći i istražiti.

[7] https://kb.acronis.com/content/9125#:~:text=Description,allows%20setting%20the%20minimum%20threshold.

6. Skup podataka

Skup podataka koji se koristi u ovom radu preuzet je sa web stranice Kaggle[8]. Originalni skup podataka preuzet je sam sa stranice backblaze.com te je procesuiran na sljedeće načine:

- U skup podataka uzeti su samo modeli čvrstog diska Seagate ST4000DM000 zbog povećeg broja kvarova ovog modela nad ostalim modelima čvrstih diskova
- Originalno je skup podataka podijeljen na kvartale godine, a kasnije spojen prema godinama, od 2015. – 2018.
- Zbog izrazito velikog broja zdravih diskova u odnosu na pokvarene uzeto je samo 10,000 zdravih diskova iz svake godine i svi pokvareni
- Za pokvarene diskove svakodnevno su, kroz maksimalno 120 dana, uzeta S.M.A.R.T. stanja diskova sve do pojave kvara
- Za zdrave diskove nasumično je dnevno kroz godinu uzeto 120 S.M.A.R.T. stanja



Slika 6 Skup podataka za 2016. godinu

[8] https://www.kaggle.com/awant08/hard-drive-failure-prediction-st4000dm000

Skup podataka za svaku od godina sastoji se od 95 stupaca, a to su datum uzimanja „snapshota“, serijski broj diska, model, kapacitet u bajtovima, prisustvo kvara (failure) i 90 S.M.A.R.T. atributa (pola raw podaci, pola normalizirani). Atribut „*failure*“ u ovom skupu podataka ima vrijednost „*1*“ za sve „*snapshote*“ (retke u datasetu, zapise) koji se odnose na disk kod kojeg se u budućnosti javlja ispad. Svaki disk možemo raspoznati prema njegovom serijskom broju. Primjerice: za disk sa serijskim brojem, na primjer „*Z304JL7B*“, kod kojeg se ne javlja greška u promatranom razdoblju vrijednost atributa „*failure*“ biti će za sve retke „*0*“, dok za disk sa serijskim brojem „*Z3051QQ3*“ kod kojeg se javlja greška vrijednost atributa „*failure*“ za sve retke biti će „*1*“. Detaljnije značenje važnih S.M.A.R.T. atributa za predviđanje kvara bit će opisana u nastavku rada kao i dodatna procesuiranja skupa podataka

7. Čišćenje i priprema skupa podataka

Prvi korak pripreme skupa podataka jest učitavanje potrebnih biblioteka i skupova podataka.

```
In [4]: #učitavanje .csv datoteka i prikaz prvih 5 elemenata
        hard1 = pd.read_csv('../zavr/model_2015_ST4000DM000.csv')
        hard2 = pd.read_csv('../zavr/model_2016_ST4000DM000.csv')
        hard3 = pd.read_csv('../zavr/model_2017_ST4000DM000.csv')
        #spajanje datasetova za 2015 i 2016
        hard_drive_data = pd.concat([hard1, hard2])
        test_data = hard3
        hard_drive_data.head()
```

Slika 7 Učitavanje skupova podataka

Ovisno o algoritmu koji će se koristiti za izradu modela, učitavaju se skupovi podataka. Algoritam „Random-Forest“ pokazao je slične rezultate s većim skupom podataka za treniranje, to jest, skupovi podataka 2015. i 2016. zajedno spojeni u jedan skup podataka, ali i sa korištenjem samo skupa podataka iz 2016. godine. Skup podataka iz 2017. godine koristi se kod svih modela kao testni skup podataka. Razlog zbog čega je uzet taj skup podataka jest bolja mogućnost uspoređivanja rezultata testiranja između različitih modela jer se na taj način testiranja modela odvijaju nad istim skupom podataka.

	date	serial_number	model	capacity_bytes	failure	smart_1_normalized	smart_1_raw	smart_2_normalized	smart_2_raw	smart_3_normalized	...
0	2015-01-01	Z300YN6R	ST4000DM000	4000787030016	0	117.0	134283496.0	NaN	NaN	97.0	...
1	2015-01-01	W300T09N	ST4000DM000	4000787030016	0	120.0	235812736.0	NaN	NaN	91.0	...
2	2015-01-01	Z3025923	ST4000DM000	4000787030016	1	117.0	159417104.0	NaN	NaN	99.0	...
3	2015-01-01	Z300GPJ7	ST4000DM000	4000787030016	0	110.0	25333728.0	NaN	NaN	92.0	...
4	2015-01-01	Z300VALY	ST4000DM000	4000787030016	0	117.0	139107936.0	NaN	NaN	96.0	...

5 rows × 95 columns

Slika 8 Prikaz podataka pomoću pandas biblioteke

Nakon učitavanja podataka može se primijetiti da skup podataka sadrži istu vrijednost za atribute *model* (ST4000DM000), *capacity_bytes* (4TB), ali i određene S.M.A.R.T. atribute. Oni će se kasnije izbaciti iz skupa podataka.

Jedna od najvažnijih stvari kod čišćenja skupa podataka je rješavanje problema vrijednosti koje nedostaju.

```
In [10]: hard_drive_data.isna().sum()
         #iz tablice možemo primijetiti da postoje atrubuti s NaN vrijednostima

Out[10]: date                        0
         serial_number               0
         model                       0
         capacity_bytes              0
         failure                     0
         smart_1_normalized          3
         smart_1_raw                 3
         smart_2_normalized    2320064
         smart_2_raw           2320064
         smart_3_normalized          3
         smart_3_raw                 3
         smart_4_normalized          3
         smart_4_raw                 3
         smart_5_normalized          3
         smart_5_raw                 3
         smart_7_normalized          3
         smart_7_raw                 3
         smart_8_normalized    2320064
         smart_8_raw           2320064
```

Slika 9 Broj podataka koji nedostaju prema atributima

Također se može primijetiti da određeni atributi nemaju zapisanu vrijednosti ni za jedan redak te je takve atribute potrebno izbaciti.

```
In [11]: #dropna(how='all') briše sve atribute koji imaju samo NaN vrijednosti
         hard_drive_data = hard_drive_data.dropna(axis=1, how='all')
         test_data = test_data.dropna(axis=1, how='all')
         hard_drive_data.head()
         #nakon brisanja atributa sa samo NaN vrijednosti preostalo je 53 atributa,
```

Slika 10 Koraci brisanja stupaca sa NaN vrijednostima

Nakon brisanja navedenih atributa preostalo je još odrediti što će se učiniti sa onim zapisima koji imaju vrijednosti koje nedostaju. Njih je samo tri.

```
In [14]: hard_drive_data[hard_drive_data.isna().any(axis=1)]
         #može se primijetiti da postoji 3 redaka sa samo NaN vrijednostima SMART podataka, očito se dogodila greška kod zapisivanja vrij
         #također vidljivo je da je zapisani kapacitet za dva diska -0.0 GB
         #potrebno je obrisati navedene retke
```

Out[14]:

	date	serial_number	model	capacity_bytes	failure	smart_1_normalized	smart_1_raw	smart_3_normalized	smart_3_raw	smart_4_normalized
401231	2015-06-19	S3010M5P	ST4000DM000	3726.0	0	NaN	NaN	NaN	NaN	NaN
16394	2016-01-31	Z300K1L6	ST4000DM000	-0.0	0	NaN	NaN	NaN	NaN	NaN
16433	2016-01-31	W300CSYX	ST4000DM000	-0.0	0	NaN	NaN	NaN	NaN	NaN

3 rows × 53 columns

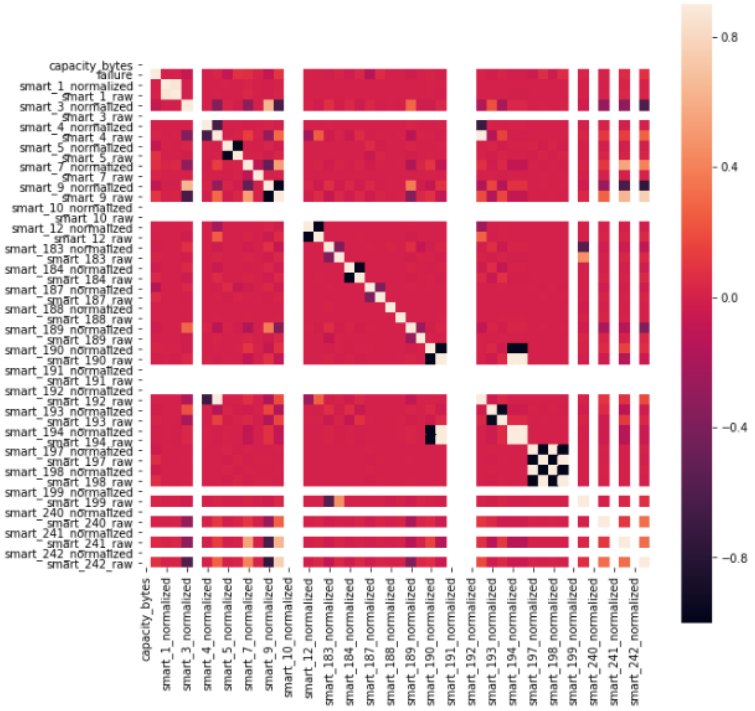
Slika 11 Koraci brisanja redaka samo sa NaN vrijednostima

Dva diska, osim vrijednosti S.M.A.R.T. atributa, imaju krivo zapisani kapacitet. Može se zaključiti da se vjerojatno dogodila greška kod prikupljanja podataka za navedena dva diska, ali i za treći disk pa je potrebno ta tri retka izbaciti iz skupa podataka. Nakon izvršenih radnji skupovi podataka nemaju više podatke koji nedostaju.

Sljedeće će se analizirati matrica korelacija.

```
In [16]: korelacijska_matrica = hard_drive_data.corr()
         plt.subplots(figsize=(10,10))
         sns.heatmap(korelacijska_matrica, vmax=0.9, square=True)
         #u korelacijskoj matrici mogu se primijetiti praznine

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x2382f080048>
```



Slika 12 Matrica korelacija

U matrici možemo primijetiti praznine kod određenih atributa pa je za detaljniju analizu potrebno pogledati kakve vrijednosti imaju navedeni atributi. Praznine zapravo označavaju „*NaN*“ vrijednosti za korelaciju s ostalim atributima. Atributi s vrijednošću „*NaN*“ za korelacije detaljnije će se analizirati.

In [18]:
```
non_correlated_features = hard_drive_data[['capacity_bytes', 'smart_3_raw', 'smart_10_normalized', 'smart_10_raw', 'smart_191_no
non_correlated_features.describe()

#standardna devijacija za navedene atribute je 0 što znači da su vrijednosti za sve retke tih atributa jednake(vrijednosti ne va
#te je atribute potrebno izbaciti pošto oni ne prikazuju korelaciju među pojavljivanja i nepojavljivanja kvara
```

Out[18]:

	capacity_bytes	smart_3_raw	smart_10_normalized	smart_10_raw	smart_191_normalized	smart_191_raw	smart_192_normalized	smart_199_normalized
count	2320061.0	2320061.0	2320061.0	2320061.0	2320061.0	2320061.0	2320061.0	2320061.0
mean	3726.0	0.0	100.0	0.0	100.0	0.0	100.0	200.0
std	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
min	3726.0	0.0	100.0	0.0	100.0	0.0	100.0	200.0
25%	3726.0	0.0	100.0	0.0	100.0	0.0	100.0	200.0
50%	3726.0	0.0	100.0	0.0	100.0	0.0	100.0	200.0
75%	3726.0	0.0	100.0	0.0	100.0	0.0	100.0	200.0
max	3726.0	0.0	100.0	0.0	100.0	0.0	100.0	200.0

Slika 13 Statistički opis atributa

Sada se može vidjeti razlog „*NaN*“ vrijednosti za te atribute. Standardna devijacija je jednaka 0 za sve atribute što znači da vrijednosti ne variraju, odnosno, za svaki zapis u skupu podataka vrijednost za navedene atribute su jednake. Zbog toga ti atributi ni na koji način ne mogu pridonijeti predviđanju kvara pa ih se može izbaciti iz skupa podataka.

Kao što je i prije navedeno skupovi podataka koji se koriste sadrže određeni broj „snapshotova“ S.M.A.R.T. stanja diskova. Svaki disk raspoznaje se prema njegovom serijskom broju zbog toga možemo vidjeti koliko ima zapisa S.M.A.R.T. stanja za svaki od diskova. Svaki disk ima veći broj prikupljenih stanja, zdravi diskovi 120 zapisa stanja po godini, a diskovi s kvarom onoliko zapisa koliko se uspjelo prikupiti do pojave kvara. Ono što se u ovom istraživanju pokušava istražiti jest mogućnost predviđanja pojave kvara jedan dan prije na temelju S.M.A.R.T. stanja diska zadnjeg dana njegovog životnog vijeka. Kako bi se dobila samo stanja pokvarenih diskova na zadnji dan, skup podataka sortira se prema datumu, nakon toga grupira se prema serijskom broju te se uzima posljednja vrijednost iz svake grupe. Na taj način dobivamo posljednji „snapshot“ S.M.A.R.T. stanja svih diskova. Za one diskove koji nisu pokvareni na taj način uzima se zadnje prikupljeno stanje. Također je isprobano uzimanje prvih „snapshot-a“ stanja zdravih diskova (kada su diskovi noviji zbog moguće degradacije diskova kroz vremenski period od godine dana), no nisu uočeni bolji rezultati.

```
In [25]: #za svaki disk uzet je određeni broj snapshotova S.M.A.R.T. stanja
         #za pokvarene diskove snapshotovi su se uzimali sve do zadnjeg dana do kad je disk radio
         #a za zdrave kroz godinu 120 snapshotova
         #skup podataka se sortira po datumu, grupira po serijskom broju i uzima posljednji element grupe sa tail(1)
         hard_drive_tail = hard_drive_data.sort_values('date').groupby('serial_number').tail(1)
         hdd_test = test_data.sort_values('date').groupby('serial_number').tail(1)

In [35]: hard_drive_tail['serial_number'].value_counts()
         #sada za svaki disk imamo samo jedan snapshot
         #možemo krenuti s izradom modela

Out[35]: Z30250ZY    1
         Z30149EM    1
         Z3025KLK    1
         Z305KVJ3    1
         S300Z6GA    1
         W300SNV4    1
         W300JG6A    1
         Z304GXHV    1
         Z3025WBY    1
         Z302FZBS    1
         S300Z53S    1
         Z3026Z72    1
         Z301A7JM    1
         Z300H1GZ    1
```

Slika 14 Filtriranje prema serijskom broju

Ovisno o modelima kasnije su odrađena dodatna procesuiranja skupa podataka, no to će se detaljnije analizirati kod analize rezultata.

8. Opis metrika i analiza rezultata

8.1. Metrike

Za potrebe analize rezultata potrebne su određene metrike. U ovom radu koriste se točnost, klasifikacijska greška, matrica konfuzija, recall (sensitivity, hrv. osjetljivost), preciznost, F1 mjera, TNR (true negative rate), FPR (false positive rate) i ROC AUC (Receiver Operating Characteristics krivulja – Area Under The Curve – prostor ispod krivulje). Za potrebe brzog izračuna svih navedenih metrika za svaki model izrađena je funkcija koja za parametar prima listu modela, podatke za testiranje i stvarne klasifikacije.

```
#funkcija za računanje metrika
def metrike(modeli, testData, y_test):
    for model in modeli:
        print("Model: ", type(model).__name__)
        y_pred = model.predict(testData)
        accuracy = metrics.accuracy_score(y_test, y_pred)
        print("Točnost: ", accuracy)
        klasifikacijska_greska = 1 - accuracy
        print("Klasifikacijska greška: ",klasifikacijska_greska)

        print("Confusion matrica")
        CM = metrics.confusion_matrix(y_test, y_pred) #confusion matrica
        TN = CM[0, 0]
        TP = CM[1, 1]
        FP = CM[0, 1]
        FN = CM[1, 0]
        print("    P0          P1")
        print(f"S0  {TN}        {FP}")
        print(f"S1  {FN}        {TP}")

        recall = TP / float(FN + TP)
        print("Recall: ",recall)
        precision = TP / float(TP + FP)
        print("Preciznost: ",precision)
        f1 = metrics.f1_score(y_test, y_pred)
        print("F1 score: ", f1)

        TNR = TN / (TN + FP)
        print("TNR: ",TNR)

        FPR = FP / float(TN + FP)
        print("FPR: ",FPR)

        ROC_AUC = metrics.roc_auc_score(y_test, y_pred)
        print('ROC AUC:', metrics.roc_auc_score(y_test, y_pred))
        print("-----------------------------------------------")
```

Slika 15 Funkcija za izračun metrika

Točnost (eng. accuracy) označuje postotak točnih klasifikacija te je jedna od najkorištenijih metrika, no kod izrade modela u ovom radu ne može dati dobru sliku rada modela zbog nebalansiranih skupova podataka. Klasifikacijska greška je suprotnost točnosti, odnosno označava postotak neispravnih klasifikacija. Najvažnija metrika analize rezultata kod klasifikacijskih problema jest matrica konfuzije (confusion matrix). Sastoji se od četiri polja koja

označavaju točan broj ispravnih, odnosno, neispravnih klasifikacija. Najvažnije metrike koje se računaju iz matrice konfuzije su osjetljivost (eng. Recall) i preciznost. Osjetljivost govori koliko je klasifikator osjetljiv na otkrivanje pozitivnih vrijednosti klasifikacije (osjetljiv na otkrivanje kvara) dok preciznost govori koliko je klasifikator precizan kada predviđa pozitivne vrijednosti klasifikacije (precizan u otkrivanju kvara). Ukratko, osjetljivost u obzir uzima stvarne pozitivne (TP) i krive negativne (FN) vrijednosti klasifikacije, a preciznost stvarne pozitivne (TP) i krive pozitivne vrijednosti (FP). TNR (True Negative Rate) označava stopu stvarnih negativnih vrijednosti klasifikacije, a FPR (False Positive Rate) stopu krivih pozitivnih vrijednosti klasifikacije. Bolje mjere ukupnih performansi modela od točnosti u ovom slučaju su F1 mjera i ROC AUC. F1 mjera računa se iz preciznosti i osjetljivosti, dok ROC AUC označava prostor ispod ROC (Reciever Operating Characteristics) krivulje. ROC se crta pomoću TPR-a i FPR-a gdje je TPR (osjetljivost ,recall, sensitivity) na y-osi, a FPR na x-osi [19].

8.2. Rezultati *Random Forest*

Broj diskova s kvarovima u ovom skupu podataka je 971, dok je broj ispravnih 16948.

```
In [27]: hard_drive_tail['failure'].value_counts()

Out[27]: 0    16948
         1      971
         Name: failure, dtype: int64
```

Slika 16 Odnos broja ispravnih i neispravnih diskova

Može se zaključiti da je skup podataka izrazio nebalansiran. Postoji mogućnost balansiranja skupa podataka na način da se smanji broj neispravnih diskova koji će ući u treniranje ili kopiranje podataka o neispravnim diskovima do broja ispravnih diskova, no balansiranje ipak nije pokazalo bolje rezultate što će se prikazati u nastavku.

Za početak potrebno je odrediti koji će se atributi koristiti za treniranje modela.

```
In [43]: X = hard_drive_tail.drop(['failure','model',  'date', 'serial_number'], axis=1)
         Y = hard_drive_tail['failure']
         x_testData = hdd_test.drop(['failure','model',  'date', 'serial_number'], axis=1)
         y_testData = hdd_test['failure']
         X.shape, x_testData.shape

Out[43]: ((17919, 38), (3885, 38))
```

Slika 17 Određivanje atributa za treniranje i testiranje

Izbacuje se *model*, pošto je isti za sve diskove, *date* jer nam ne govori ništa o pojavi kvara te *serial_number*, pošto se sada jedinstven za svaki zapis. Preostaje 38 S.M.A.R.T. atributa za

treniranje. Kod ovog algoritma koriste se svi dostupni atributi jer je, kao što je navedeno u opisu algoritama, *Random Forest* algoritam izrazito dobar u detekciji važnih prediktora pa će se odabir važnosti prepustiti samom algoritmu. Također, na kraju će se analizirati najvažniji prediktori te će se samo oni koristiti kod ostalih algoritama koji ne mogu raditi sa višedimenzionalnim skupovima podataka. Isti atributi koriste se u testnom skupu za 2017. godinu. Nakon toga skup se dijeli na treniranje i testiranje, 70% skupa podataka su podaci za treniranje i 30% su podaci za testiranje.

Posljednji korak prije treniranja je odabir algoritama i njegovih hiperparametara. Modele koji će se izraditi mogu se vidjeti na sljedećoj slici.

```
In [55]: from sklearn.ensemble import RandomForestClassifier
         model1 = RandomForestClassifier(n_estimators=10,max_depth=None, n_jobs=-1)
         model2 = RandomForestClassifier(n_estimators=50, max_depth=None, n_jobs=-1)
         model3 = RandomForestClassifier(n_estimators=100,max_depth=None, n_jobs=-1)
         model4 = RandomForestClassifier(n_estimators=200,max_depth=None, n_jobs=-1)
         model5 = RandomForestClassifier(n_estimators=500,max_depth=None, n_jobs=-1)
         model6 = RandomForestClassifier(n_estimators=1000,max_depth=None, n_jobs=-1)
```

Slika 18 Odabir hiperparametara algoritma

Parametar *n_estimators* predstavlja broj stabala u „šumi“, *max_depth* označava dubinu do koje se stablo može ići, a *n_jobs* omogućava podjelu poslova, odnosno mogućnost korištenja svih jezgri procesora u vrijeme treniranja modela kako bi samo treniranje bilo daleko brže. Izrađuje se 6 modela, svaki sa različitim brojem stabala, maksimalnom dubinom postavljenom na „None“ i brojem poslova na „-1“ kako bi se koristile svi dostupni procesori/jezgre sustava (4 jezgre). Mijenjanjem maksimalne dubine može se spriječiti takozvani *overfitting*, no u ovom slučaju stavljanjem maksimalne dubine ispod 5 padaju performanse modela zbog čega se koristi zadana vrijednost „None“.

Nakon određivanja hiperparametara može se krenuti s treniranjem modela i izradom metrika.

```
In [57]: print("Test-original podaci")
         metrike(modeli,x_test, y_test)
         print("Test-2017")
         metrike(modeli,x_testData, y_testData)

Test-original podaci
Model:  RandomForestClassifier
Točnost:  0.96875
Klasifikacijska greška:  0.03125
Confusion matrica
    P0          P1
S0  5050        31
S1  137         158
Recall:  0.535593220338983
Preciznost:  0.8359788359788359
F1 score:  0.6528925619834711
TNR:  0.9938988388112576
FPR:  0.006101161188742373
ROC AUC: 0.7647460295751203
------------------------------------------------
```

Slika 19 Izračun metrika

Rezultati pokazuju da je nad ovim skupom podataka najbolje koristiti 200 stabala.

Metrike za testni skup od 30% originalnog skupa podataka za 2015.+2016. godinu su:

```
Model: RandomForestClassifier
Točnost:  0.9702380952380952
Klasifikacijska greška:  0.029761904761904767
Confusion matrica
P0          P1
S0  5046        35
S1  125        170
Recall:  0.576271186440678
Preciznost:  0.8292682926829268
F1 score:  0.6799999999999999
TNR:  0.9931115922062587
FPR:  0.0068884077937413895
ROC AUC: 0.7846913893234684
```

Za testni skup podataka za 2017. godinu rezultati su sljedeći:

```
Model: RandomForestClassifier
Točnost:  0.9583011583011583
Klasifikacijska greška:  0.04169884169884175
Confusion matrica
P0          P1
S0  3579        75
S1  87         144
Recall:  0.6233766233766234
Preciznost:  0.6575342465753424
F1 score:  0.64
TNR:  0.9794745484400657
FPR:  0.020525451559934318
ROC AUC: 0.8014255859083446
```

Sada se može vidjeti da točnost u ovom slučaju ne govori puno zbog nebalansiranosti skupa podataka. Ono što je puno važnije je analizirati osjetljivost (recall) i preciznost. Osjetljivost je za skup podataka diskova iz 2017. 62.3%, a preciznost 65.8%. Ostali modeli testiranjem nad testnim skupom 2017. godina su dali ukratko sljedeće rezultate:

Tablica 1 Random Forest Trening-2015.+2016., Test-2017.

Broj stabala/ Mjera	Recall	Preciznost	F1 mjera	ROC AUC
10	0.6017	0.7473	0.6667	0.79443
50	0.6061	0.6604	0.6320	0.793178
100	0.6104	0.6651	0.6366	0.795479
200	0.6234	0.6575	0.6400	0.80142
500	0.6147	0.6544	0.6339	0.79709
1000	0.6277	0.6473	0.6373	0.80304

Kao što se može vidjeti iz tablice nema velikih razlika među modelima, no bilo bi dobro odabrati najbolji od prikazanih. Ono što može primijetiti iz tablice jest što je veća osjetljivost to je manja preciznosti. Zbog toga je potrebno odrediti kompromis, odnosno želimo li veću osjetljivost ili preciznost. Model sa 200 stabala ima sljedeću matricu konfuzije:

```
Confusion matrica
    P0       P1
S0  3579     75
S1  87       144
```

Značenja pojedinih elemenata mogu se vidjeti na sljedećoj matrici:

```
Confusion matrica
    P0       P1
S0  TN       FP
S1  FN       TP
```

Ako želimo povećati osjetljivost (recall) onda će se smanjiti broj krivih negativnih klasifikacija (FN), a ako želimo povećati preciznost smanjit ćemo broj krivih pozitivnih klasifikacija (FP). To se može iščitati iz formula za osjetljivost i preciznost:

$$osjetljivost = \frac{TP}{FN\ +\ TP}$$

$$preciznost = \frac{TP}{TP + FP}$$

U slučaju predviđanja kvara diska važnije je da se smanji broj krivih negativnih klasifikacija (predviđanja da ne postoji kvar kada zapravo postoji).

Izrada modela je također napravljena korištenjem samo skupa podataka iz 2016. godine za treniranje, a rezultati su sljedeći:

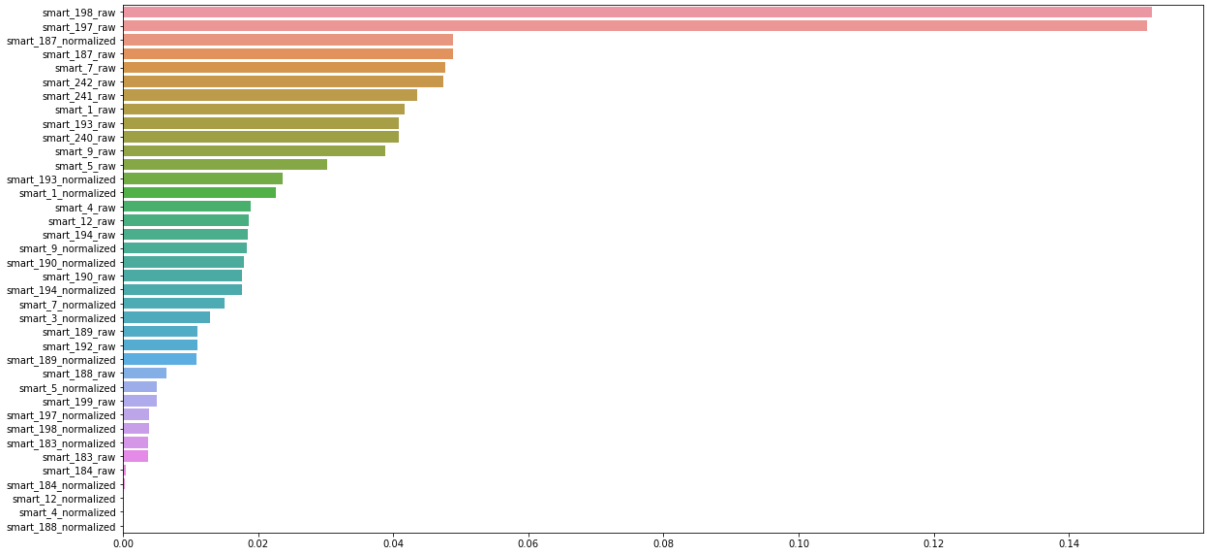
Tablica 2 Random Forest Trening-2016. Test-2017.

Broj stabala/Mjera	Recall	Preciznost	F1 score	ROC AUC
10	0.5844	0.625	0.6040	0.7811
50	0.6103	0.705	0.6542	0.7971
100	0.6190	0.7186	0.6651	0.8019
200	0.6233	0.6990	0.6590	0.8032
500	0.6277	0.7073	0.6651	0.8056
1000	0.6277	0.6971	0.6605	0.8052

Rezultati za Trening-2016. Test-2017.- 500 stabala:

```
Model:  RandomForestClassifier
Točnost:  0.9624195624195624
Klasifikacijska greška:  0.03758043758043761
Confusion matrica
    P0           P1
S0  3594        60
S1  86         145
Recall:  0.6277056277056277
Preciznost:  0.7073170731707317
F1 score:  0.6651376146788991
TNR:  0.9835796387520526
FPR:  0.016420361247947456
ROC AUC: 0.8056426332288401
```

Kao što je ranije navedeno „Random Forest“ algoritam izrazito dobro određuje važnost prediktora. Najvažniji atributi mogu se vidjeti na sljedećoj slici:



Slika 20 Popis najvažnijih prediktora

Pet najvažnijih atributa su: *smart_198_raw*, *smart_197_raw*, *smart_187_raw*, *smart_7_raw*, *smart_242_raw*. Atribut *smart_187_normalized* neće se uzimati u obzir pošto on prikazuje normalizirane vrijednosti *smart_187_raw*, a normalizacija će se odrađivati po potrebi ovisno o algoritmu koji se koristi. Bilo bi dobro analizirati što zapravo označuju navedeni atributi. Značenja pojedinih atributa mogu se pronaći na stranici backblaze.com[9] i detaljnija u kb.acronis.com bazi znanja[10].

- *smart_197_raw* - Current_Pending_Sector (Raw Value)
- *smart_198_raw* - Offline_Uncorrectable (Raw Value)

[9] https://www.backblaze.com/blog-smart-stats-2014-8.html#S242R
[10] https://kb.acronis.com/

- *smart_187_raw* - Reported_Uncorrect (Raw Value)
- *smart_7_raw* - Seek_Error_Rate (Raw Value)
- *smart_242_raw - Total_LBAs_Read (Raw Value)*

smart_197_raw i *smart_198_raw* odnosi se na neispravnost sektora diska. *smart_187_raw* predstavlja greške koje ECC (error-correcting code) nije uspio rješiti. *smart_7_raw* broji greške kod pretraživanja dok *smart_242_raw* predstavlja ukupan broj čitanja logičkih blokova. koje su se dogodile sa diskom, dok atribut *smart_242_raw* predstavlja broj čitanja logičkih blokova.

8.2.1. Testiranja s balansiranim skupom podataka

Određeno je i nekoliko testiranja s balansiranim skupom podataka. Prvo balansiranje izvodi se na način da se uveća broj pokvarenih diskova tako da se kopiraju slučajno odabrani zapisi o pokvarenim diskovima. Drugo balansiranje izvodi se tako da se smanji broj nepokvarenih diskova. Rezultati su sljedeći za algoritam *Random Forest* sa 500 stabala:

Tablica 3 Random Forest-balansirani Trening-2016. Test-2017.

Balansiranje/Mjera	Točnost	Klasifikacijska greška	Recall	Preciznost	TNR	FPR
Nepokvareni_downsampled	0.7732	0.2267	0.7922	0.1801	0.772	0.227
Pokvareni_upsampled	0.9637	0.0363	0.6233	0.7272	0.9852	0.015

Balansiranje na način da se smanji broj nepokvarenih diskova dovodi do poboljšanja u osjetljivosti, no dovodi do velikog pada u preciznosti, točnosti i stopi stvarno negativnih vrijednosti (TNR) što je zapravo i očekivano jer se model trenira s daleko manje nepokvarenih diskova zbog čega model ne može dobro detektirati ispravne diskove Ako se poveća broj pokvarenih diskova vidimo vrlo slične rezultate kao i kada ne koristimo balansiranje jer zapravo kopiramo već postojeće zapise među kojima snažan „Random Forest“ algoritam može pronaći prediktivne važnosti i na manjem skupu podataka poput originalnog.

Rezultati za balansirani skup – Pokvareni_upsampled

```
Model:  RandomForestClassifier
Točnost:  0.9637065637065637
Klasifikacijska greška:  0.03629343629343629
Confusion matrica
    P0          P1
S0  3600        54
S1  87        144
Recall:  0.6233766233766234
Preciznost:  0.7272727272727273
F1 score:  0.6713286713286714
TNR:  0.9852216748768473
FPR:  0.014778325123152709
ROC AUC: 0.8042991491267354
```

8.3. Rezultati stablo odlučivanja

Drugi algoritam koji se koristio za izradu prediktivnih modela je stablo odlučivanja. Skup podataka koji se koristi za treniranje isti je kao i kod algoritma „Random Forest“, S.M.A.R.T. stanja diskova za 2015.+2016. godinu. Za testiranje korišten je skup podataka za 2017. godinu. Algoritam je dao najbolje rezultate kada su za treniranje i testiranje uzeti ranije spomenutih **pet najvažnijih atributa** otkrivenih pomoću „Random Forest“ algoritma. Izrađena su dva modela, za prvi model korištene su zadane postavke, a za drugi model dodan je hiperparametar „*max_depth = 3*“. Sa vrijednošću 3 za „*max_depth*“ dobivaju se najbolji rezultati.

```
In [42]: from sklearn.tree import DecisionTreeClassifier
         model1 = DecisionTreeClassifier()
         model2 = DecisionTreeClassifier(max_depth=3)
```

Slika 21 Hiperparametri-stabla odlučivanja

Trening 2015+2016. Test 2017. - *max_depth = None*

```
Model:  DecisionTreeClassifier
Točnost:  0.9284427284427285
Klasifikacijska greška:  0.07155727155727154
Confusion matrica
    P0          P1
S0  3466        188
S1  90        141
Recall:  0.6103896103896104
Preciznost:  0.42857142857142855
F1 score:  0.5035714285714286
TNR:  0.9485495347564313
FPR:  0.051450465243568694
ROC AUC: 0.7794695725730209
```

Trening 2015+2016. Test 2017. - *max_depth = 3*

```
Model:  DecisionTreeClassifier
Točnost:  0.9608751608751609
Klasifikacijska greška:  0.039124839124839106
Confusion matrica
    P0          P1
S0  3587        67
S1  85        146
Recall:  0.6320346320346321
Preciznost:  0.6854460093896714
F1 score:  0.6576576576576577
TNR:  0.981663929939792
FPR:  0.01833607006020799
ROC AUC: 0.8068492809872121
```

Algoritam stabla odlučivanja daje vrlo slične rezultate kao i *Random Forest* algoritam.

8.4. Rezultati SVM algoritam

Prvi algoritam koji se ne temelji na stablima odlučivanja je SVM. Ovaj algoritam pokazao je bolje rezultate sa manjim trening skupom podataka koji uključuje samo 2016. godinu. Također najbolji rezultati su kada se koriste prije spomenutih pet prediktora *smart_198_raw*, *smart_197_raw*, *smart_187_raw*, *smart_7_raw*, *smart_242_raw*. Atributi *smart_7_raw*, *smart_242_raw* sadrže vrijednosti koje jako variraju i poprimaju vrlo velike vrijednosti u odnosu na ostala tri atributa zbog čega je potrebno odraditi standardizaciju. Najbolji rezultati se dobivaju ako se standardiziraju vrijednosti samo navedena dva atributa.

	smart_197_raw	smart_198_raw	smart_187_raw	smart_242_raw	smart_7_raw
count	10606.000000	10606.000000	10606.000000	1.060600e+04	1.060600e+04
mean	4.338676	4.338676	0.279559	7.791894e+10	4.139376e+10
std	174.665371	174.665371	9.111081	9.091986e+10	3.042763e+12
min	0.000000	0.000000	0.000000	5.824718e+08	1.249410e+05
25%	0.000000	0.000000	0.000000	6.482316e+09	2.605322e+08
50%	0.000000	0.000000	0.000000	1.693797e+10	3.839847e+08
75%	0.000000	0.000000	0.000000	1.466635e+11	6.234051e+08
max	13864.000000	13864.000000	844.000000	4.248264e+11	2.814715e+14

Slika 22 Statistike pet najvažnijih atributa

```
In [29]: from sklearn.preprocessing import StandardScaler
         X[['smart_242_raw','smart_7_raw']] = StandardScaler().fit_transform(X[['smart_242_raw','smart_7_raw']])
         x_testData[['smart_242_raw','smart_7_raw']] = StandardScaler().fit_transform(x_testData[['smart_242_raw','smart_7_raw']])
```

Slika 23 Standardiziranje

Nakon toga može se prijeći na izradu modela. Koristeći ovaj algoritam izradit će se 7 modela sa različitim hiperparametrima. Vrijednosti parametara mogu se vidjeti na sljedećoj slici.

```
In [32]: from sklearn.svm import SVC
         model1 = SVC(kernel="rbf", gamma=5, C=0.1)
         model2 = SVC(kernel="rbf", gamma=5, C=1)
         model3 = SVC(kernel="rbf", gamma=5, C=1000)
         model4 = SVC(kernel="rbf", gamma=0.1, C=0.1)
         model5 = SVC(kernel="rbf", gamma=0.1, C=1)
         model6 = SVC(kernel="rbf", gamma=0.1, C=1000)
         model7 = SVC(kernel="rbf", gamma=10, C=0.1)
```

Slika 24 Hiperparametri - SVM

Kao kernel funkciju koristi se Gaussian RBF te se isprobavaju modeli sa različitim vrijednostima „*gamma*“ i „*C*“ parametra. Navedeni parametri utječu na izgled hiperravnine razdvajanja [5, str. 152].

Što se tiče rezultata, točnost se ponovno kreće oko 95-97%, no to nam i u ovom slučaju ne govori puno. Rezultate osjetljivosti, preciznosti, F1 mjera i ROC AUC možemo vidjeti prema sljedećoj tablici:

Tablica 4 SVM - Trening-2016. Test-2017

Paremetri/Mjera	Recall	Preciznost	F1 score	ROC AUC
Gamma=5,C=0.1	0.6537	0.6622	0.6580	0.8163
Gamma=5,C=1	0.6363	0.735	0.6821	0.8109
Gamma=5,C=1000	0.6363	0.7424	0.6853	0.8112
Gamma=0.1,C=0.1	0.6233	0.7579	0.6841	0.8054
Gamma=0.1,C=1	0.6190	0.7566	0.6810	0.8032
Gamma=0.1,C=1000	0.6017	0.7354	0.6619	0.7940
Gamma=10,C=0.1	0.6580	0.5937	0.6242	0.8147

Moglo bi se reći da se najoptimalniji rezultati dobivaju sa parametrima „*Gamma=5*“, „*C=1000*“ ako gledamo F1 mjeru, ROC AUC i preciznosti, no s druge strane s parametrima „*Gamma=5*“, „*C=0.1*“ najbolja je osjetljivost uz dovoljno dobru preciznost.

Trening 2016., Test 2017. - *Gamma=5, C=0.1*

```
Model:  SVC
Točnost:  0.9595881595881596
Klasifikacijska greška:  0.04041184041184043
Confusion matrica
    P0          P1
S0  3577        77
S1  80        151
Recall:  0.6536796536796536
Preciznost:  0.6622807017543859
F1 score:  0.6579520697167756
TNR:  0.9789272030651341
FPR:  0.0210727969348659
ROC AUC: 0.8163034283723939
```

Trening 2016., Test 2017. - *Gamma=5, C=1000*

```
Model:  SVC
Točnost:  0.9652509652509652
Klasifikacijska greška:  0.03474903474903479
Confusion matrica
    P0          P1
S0  3603        51
S1  84        147
Recall:  0.6363636363636364
Preciznost:  0.7424242424242424
F1 score:  0.6853146853146853
TNR:  0.9860426929392446
FPR:  0.013957307060755337
ROC AUC: 0.8112031646514404
```

8.5. Rezultati KNN algoritam

Posljednji algoritam koji će se analizirati jest algoritam k najbližih susjeda. Skup podataka za treniranje opet su diskovi promatrani 2016. godine, a za testiranje diskovi iz 2017. Što se tiče prediktora koji se koriste to su u ovom slučaju tri najvažnija atributa pronađena pomoću „Random Forest“ algoritma. KNN algoritam radi dobro sa manjim, ali i niskodimenzionalnim skupom podataka te su i u ovom slučaju vidljivi bolji rezultati sa samo tri prediktora, a ne pet ili više.

```
In [28]: X = hard_drive_tail[['smart_197_raw','smart_198_raw','smart_187_raw']]
         Y = hard_drive_tail['failure']

         x_testData = hdd_test[['smart_197_raw','smart_198_raw','smart_187_raw']]
         y_testData = hdd_test['failure']
         X.shape, x_testData.shape

Out[28]: ((10606, 3), (3885, 3))
```

Slika 25 Odabir atributa za treniranje i testiranje – KNN

Osim već prije opisanog procesuiranja skupa podataka potrebno je i standardizirati/normalizirati podatke. Standardiziranje, nasuprot min-max normalizaciji, pokazalo je bolje rezultate. Izradit će se pet modela, svaki sa različitim hiperparametrom brojem susjeda.

```
In [31]: from sklearn.neighbors import KNeighborsClassifier
         model1 = KNeighborsClassifier(n_neighbors=1)
         model2 = KNeighborsClassifier(n_neighbors=3)
         model3 = KNeighborsClassifier(n_neighbors=5)
         model4 = KNeighborsClassifier(n_neighbors=7)
         model5 = KNeighborsClassifier(n_neighbors=9)
```

Slika 26 Hiperparametri - KNN

Tablica 5 KNN - Trening-2016. Test-2017

Broj susjeda/ Mjera	Recall	Preciznost	F1 score	ROC AUC
1	0.5887	0.8047	0.68	0.7899
3	0.6190	0.7487	0.6777	0.8030
5	0.6320	0.6985	0.6636	0.8074
7	0.6320	0.6985	0.6636	0.8074
9	0.6320	0.6986	0.6636	0.8074

Može se primijetiti da se rezultati poboljšavaju, što se tiče osjetljivosti, do K = 5. Nakon toga više se rezultati ne mijenjaju. Možemo zaključiti da najoptimalnije rezultate daje model sa K=5.

Trening 2016., Test 2017. „*n_neighbors=5*“

```
Model:  KNeighborsClassifier
Točnost:  0.9619047619047619
Klasifikacijska greška:  0.03809523809523807
Confusion matrica
    P0          P1
S0  3591        63
S1  85        146
Recall:  0.6320346320346321
Preciznost:  0.6985645933014354
F1 score:  0.6636363636363636
TNR:  0.9827586206896551
FPR:  0.017241379310344827
ROC AUC: 0.8073966263621436
```

9. Zaključak

Kako bi se moglo pristupiti izradi prediktivnih modela, prije svega potrebno je bilo istražiti postojeća istraživanja te sam došao do zaključka da želim problemu predviđanja kvara pristupiti na malo drugačiji način od većine već odrađenih istraživanja tako da pokušam predvidjeti kvar dan prije njegove pojave pomoću S.M.A.R.T. stanja diska. Također, potrebno je bilo i odabrati alate koji će se koristiti prilikom izrade modela. Kako sam se već susretao sa programskim jezikom Python na fakultetu, ali i programiranjem u slobodno vrijeme, odlučio sam iskoristiti stečeno znanje i istražiti mogućnosti ovog programskog jezika u svrhu izrade prediktivnih modela. Sljedeći korak bio je istraživanje metoda i načina izrade prediktivnih modela. Tu sam naišao na veliki broj mogućnosti te sam se odlučio za četiri algoritma: Random Forest algoritam, stabla odlučivanja, metodu potpornih vektora i algoritam k-najbližih susjeda. *Random Forest* algoritam, za razliku od ostalih, poslužio je u dvije svrhe. Zbog svoje poznate prediktivne snage može jako dobro uočiti važne prediktore te je zbog toga, osim za izradu modela, poslužio za pronalazak važnih prediktora koji su kasnije korišteni kod izrade modela pomoću drugih algoritama koji ne mogu raditi sa visokodimenzionalnim skupovima podataka. Prije izrade modela potrebno je bilo i proučiti skupove podataka na temelju kojih će se izrađivati modeli. To su podaci o S.M.A.R.T. stanjima diskova koje je bilo potrebno dodatno procesuirati. Osim toga opisane su i metrike kojima su se određivale performanse modela.

Na kraju dobro bi bilo izdvojiti najbolje modele i usporediti ih. Svi modeli testirani su nad skupom podataka S.M.A.R.T. stanja diskova za 2017. godinu. Najbolji model koji je izrađen pomoću *Random Forest* algoritma koristi skup podataka S.M.A.R.T. stanja diskova iz 2016. godine kao trening skup podataka te kao hiperparametar 500 stabala. Najbolji model izrađen pomoću stabla odlučivanja kao hiperparametar ima određenu maksimalnu dubinu jednaku tri. Kao najbolji model izrađen pomoću metode potpornih vektora ima određene hiperparametre *Gamma=5, C=0.1.* Najbolji hiperparametar kod KNN algoritma pokazao se broj susjeda K=5.

Detaljne metrike najboljih modela izrađenih pomoću svakog od algoritama možemo vidjeti na sljedećoj tablici:

Tablica 6 Metrike najboljih algoritama

Algoritam/ Mjera	Točnost	Klas. greška	Osjetljiv.	Preciz.	F1	TNR	FPR	ROC AUC
Random Forest	0.9624	0.0376	0.6277	0.7073	0.6651	0.9836	0.0164	0.8056
Stablo odluč.	0.9608	0.0391	0.6320	0.6854	0.6576	0.9816	0.0183	0.8069
SVM	0.9595	0.0404	0.6536	0.6622	0.6580	0.9789	0.0211	0.8163
KNN	0.9619	0.0381	0.6320	0.6985	0.6636	0.9827	0.0172	0.8074

Ono što se na prvu može primijetiti da svi algoritmi daju vrlo slične rezultate. Ako se gledaju ukupne performanse prema ROC AUC najbolji model bio bi izrađen pomoću SVM algoritma. Također, ovaj model ima najbolju osjetljivost koja nam je u ovom istraživanju jedna od metrika koju je potrebno dovesti do čim više razine pošto želimo da imamo čim manje krivih negativnih klasifikacija s tim da imamo dovoljno dobru preciznost. Model izrađen pomoću Random Forest algortima pak ima najbolju preciznost kao i najniži FPR, odnosno najmanje krivo određenih pozitivnih klasifikacija. Pronalaženje najvažnijih prediktora pomoću *Random Forest* algoritma omogućili jednako dobre rezultate za modele izrađene i pomoću ostalih algoritama.

Zaključak ovog istraživanja bio bi da je moguće predvidjeti kvar na disku prema stanju diska na zadnji dan njegovog životnog ciklusa, no ne sa savršenom sigurnošću. Kako bi se sa većom sigurnošću mogao predvidjeti kvar čini se da je potrebno dulje praćenje stanja diska, odnosno, dnevne promjene vrijednosti određenih S.M.A.R.T. atributa te bi se potencijalno na taj način mogao predvidjeti kvar na disku. Osobno smatram da bi daljnja istraživanja trebala imati u cilju pronaći način da se smanji broj krivo određenih negativnih klasifikacija uz dovoljno mali broj krivo određenih pozitivnih klasifikacija.

Popis literature

[1] J. Li i. sur. „Hard Drive Failure Prediction Using Classification and Regression Trees. Proceedings of the International Conference on Dependable Systems and Networks„ 2014. [Na internetu]. Dostupno: ResearchGate, https://www.researchgate.net/publication/286602543_Hard_Drive_Failure_Prediction_Using_Classification_and_Regression_Trees. [Pristupano: 05.08.2020.]

[2] G. Hamerly, C. Elkan, „Bayesian Approaches to Failure Prediction for Disk Drives“. 2003. [Na internetu]. Dostupno: ResearchGate, https://www.researchgate.net/publication/2893283_Bayesian_Approaches_to_Failure_Prediction_for_Disk_Drives, [Pristupano: 05.08.2020.]

[3] J.Shen, J.Wan, S.Lim, L.Yu „Random-forest-based failure prediction for hard disk drives“. 2018., [Na internetu]. Dostupno: ResearchGate, https://www.researchgate.net/publication/328807441_Random-forest-based_failure_prediction_for_hard_disk_drives. [Pristupljeno: 05.08.2020.]

[4] J. Murray, G. Hughes, K.Kreutz-Delgado „Hard drive failure prediction using non-parametric statistical methods“. 2003. [Na internetu]. Dostupno: ResearchGate, https://www.researchgate.net/publication/228972414_Hard_drive_failure_prediction_using_non-parametric_statistical_methods [Pristupljeno: 06.08.2020.]

[5] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, CA, USA: O'Reilly Media, 2017.

[6] D. Larose, C. Larose, *Dana Mining and Predictive Analytics*, USA: John Wiley & Sons, 2015.

[7] O. Harrison „Machine Learning Basics with the K-Nearest Neighbors Algorithm“. 2018. [Na internetu]. Dostupno: towardsdatascience, https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 [Pristupljeno: 09.08.2020.]

[8] R. Gandhi „Support Vector Machine — Introduction to Machine Learning Algorithms“. 2018. [Na internetu]. Dostupno: towardsdatascience, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 [Pristupljeno: 10.08.2020.]

[9] S. Ray „Understanding Support Vector Machine(SVM) algorithm from examples (along with code)“. 2018. [Na internetu]. Dostupno: Analytics Vidhya, https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/ [Pristupljeno: 10.08.2020.]

[10] S. Madhavan, M. Sturdevant „Learn classification algorithms using Python and scikit-learn“. 2019. [Na internetu]. Dostupno: IBM Developer, https://developer.ibm.com/technologies/data-science/tutorials/learn-classification-algorithms-using-python-and-scikit-learn/ [Pristupljeno: 10.08.2020.]

[11] A. Chakure „Decision Tree Classification“. 2019. [Na internetu]. Dostupno: towardsdatascience, https://towardsdatascience.com/decision-tree-classification-de64fc4d5aac [Pristupljeno: 11.08.2020.]

[12] scikit-learn developers (bez dat.) „Decision Trees“. [Na internetu]. Dostupno: scikit-learn, https://scikit-learn.org/stable/modules/tree.html [Pristupljeno: 11.08.2020.]

[13] A. Chakure „Random Forest Classification“. 2019. [Na internetu]. Dostupno: towardsdatascience, https://towardsdatascience.com/random-forest-classification-and-its-implementation-d5d840dbead0 [Pristupljeno: 11.08.2020.]

[14] A. Navlani, „Understanding Random Forests Classifiers in Python“. 2019. [Na internetu]. Dostupno: DataCamp, https://www.datacamp.com/community/tutorials/random-forests-classifier-python [Pristupljeno: 11.08.2020.]

[15] A. Klein, „What SMART Stats Tell Us About Hard Drives“. 2016. [Na internetu]. Dostupno: BACKBLAZE, https://www.backblaze.com/blog/what-smart-stats-indicate-hard-drive-failures/ [Pristupljeno: 08.08.2020.]

[16] „Hard Disk SMART Drives“. [Na internetu]. Dostupno: pctechguide.com, https://www.pctechguide.com/hard-disks/hard-disk-smart-drives [Pristupljeno: 08.08.2020.]

[17] „S.M.A.R.T. attribute meaning“. [Na internetu]. Dostupno: siguardian.com, https://web.archive.org/web/20110226194620/http:/www.siguardian.com/products/siguardian/on_line_help/s_m_a_r_t_attribute_meaning.html [Pristupljeno: 08.08.2020.]

[18] „S.M.A.R.T. attribute meaning“. [Na internetu]. Dostupno: siguardian.com, https://web.archive.org/web/20110226194620/http:/www.siguardian.com/products/siguardian/on_line_help/s_m_a_r_t_attribute_meaning.html [Pristupljeno: 08.08.2020.]

[19] „Evaluating a Classification Model“. [Na internetu]. Dostupno: ritchieng.com, https://www.ritchieng.com/machine-learning-evaluate-classification-model/ [Pristupljeno: .14.08.2020.]

Popis slika

Popis tablica

Prilozi

[1] Korišten skup podataka, Kaggle, https://www.kaggle.com/awant08/hard-drive-failure-prediction-st4000dm000

[2] Jupyter Notebook-ovi sa izrađenim modelima, https://drive.google.com/file/d/1NZ56Pstkd0qjJRkzYSLacuk9Sh0lVG5x/view?usp=sharing