SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2388

ALGORITMI STROJNOG UČENJA ZA PREDVIĐANJE GLEDANOSTI TELEVIZIJSKIH KANALA

Marko Josipović

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2388

ALGORITMI STROJNOG UČENJA ZA PREDVIĐANJE GLEDANOSTI TELEVIZIJSKIH KANALA

Marko Josipović

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

DIPLOMSKI ZADATAK br. 2388

Pristupnik:	**Marko Josipović (0036496833)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	izv. prof. dr. sc. Alan Jović

Zadatak: **Algoritmi strojnog učenja za predviđanje gledanosti televizijskih kanala**

Opis zadatka:

Predviđanje gledanosti je vrlo težak problem koji je osjetljiv na kvalitetu podataka na temelju kojih se predviđa. Poboljšanje točnosti predviđanja može donijeti velike uštede u marketinškoj industriji i u industriji emitiranja televizijskih (TV) kanala i drugih sadržaja. U ovom radu potrebno je opisati teorijsku podlogu regresijskih algoritama strojnog učenja za predviđanje poput linearne regresije, stroja s potpornim vektorima i algoritma slučajne šume. Osim algoritama potrebno je i ostvariti implementaciju web aplikacije s pametnim sustavom za predviđanje gledanosti koja će se razviti u suradnji s tvrtkom Odašiljači i veze d.o.o. Razmatrat će se različite tehnologije za implementaciju takve web aplikacije i praktično usporediti točnosti predviđanja navedenih algoritama korištenjem stvarnih podataka o gledanosti TV kanala.

Rok za predaju rada: 28. lipnja 2021.

Hvala roditeljima, Karli i mentoru na pomoći i podršci tijekom studija.

SADRŽAJ

1. Uvod

Strojno učenje je grana umjetne inteligencije čiji je razvoj rezultirao primjenom algoritama strojnog učenja za rješavanje raznih problema. Ovi algoritmi se snalaze u problemima koje klasični algoritmi ne mogu riješiti poput računalnog vida, raspoznavanja govora, višeagentskih sustava i pronalaženja znanja u velikoj količini podataka. Predviđanje gledanosti televizijskih kanala također spada u probleme koji se ne mogu riješiti klasičnim algoritmima.

Cilj sustava za predviđanje je što točnije predviđanje koje se može koristiti za pocjenu rizika davanja kredita, prilagođavanje poslužitelja opterećenju ili pak za optimizaciju rasporeda emitiranja sadržaja televizijskih programa.

U ovom radu opisana je teorijska podloga regresijskih algoritama nadziranog strojnog učenja za predviđanje: linearne regresije, stroja s potpornim vektorima i algoritma slučajne šume. Nakon algoritama opisana je implementacija web aplikacije s pametnim sustavom za predviđanje gledanosti televizijskih kanala razvijene u suradnji s tvrtkom Odašiljači i veze d.o.o. Aplikacija koristi stvarne podatke o gledanosti i korisnicima omogućava predviđanje gledanosti na temelju učitanog rasporeda emitiranja. Razmatraju se različite tehnologije koje se mogu koristiti za implementaciju takve web aplikacije i na kraju se uspoređuju točnosti predviđanja opisanih algoritama korištenjem stvarnih podataka o gledanosti.

2. Algoritmi strojnog učenja za predviđanje gledanosti televizijskih kanala

Algoritmi strojnog učenja mogu se opisati pomoću tri komponente: modela, funkcije pogreške i optimizacijskog postupka. Model je skup hipoteza, tj. funkcija koje primjerima pridaju oznake. Sve hipoteze u modelu su parametrizirane parametrima o kojima ovisi kako će hipoteza preslikavati iz skupa primjera na skup oznaka. Zadaća algoritama strojnog učenja je učenje modela što je zapravo pronalaženje najbolje hipoteze, odnosno parametara koji definiraju najbolju hipotezu. Kriterij po kojem uspoređujemo hipoteze je funkcija pogreške a način na koji dolazimo do najbolje naziva se optimizacijskim postupkom.

Veliki problem strojnog učenja predstavljaju podnaučenost i prenaučenost modela, ovdje u priču ulaze hiperparametri algoritma strojnog učenja kojima se određuje složenost modela. Prije nego što algoritam strojnog učenja može pronaći optimalne parametre hipoteze, razvijatelj sustava za predviđanje mora obaviti pronalazak optimalnih hiperparametara. Ovi problemi i postupak optimizacije hiperparametara opisani su u zadnjem poglavlju.

Postoje tri pristupa u strojnom učenju koji se razlikuju u načinu na koji algoritam strojnog učenja uči model. Nadzirano učenje koristi skup podataka kojem su već pridijeljene oznake, nakon učenja modela označenim skupom podataka sposoban je zaključivati koje oznake treba pridijeliti primjerima koji nisu bili u označenom skupu. Nenadzirano učenje se obavlja pomoću skupa podataka koji nije označen što znači da model pri učenju pronalazi pravilnosti u podatcima, primjer je grupiranje podataka po sličnosti. Zadnji pristup je podržano učenje u kojem programski agent na temelju ulaznih podražaja odlučuje o akcijama koje će poduzeti. Akcije se na neki način nagrađuju ili kažnjavaju i tako agent uči koje akcije je dobro poduzeti u određenim situacijama.

Nadzirano učenje je prikladno za rješavanje problema predviđanja gledanosti. Za

učenje modela koji će predviđati gledanost potreban je označeni skup podataka koji se sastoji od primjera i pripadajućih oznaka. Primjer je vektor značajki koje opisuju jedan podatak, npr. kod predviđanja rizika davanja kredita, značajke mogu biti mjesečni prihodi tražitelja kredita, mjesečni rashodi i veličina uštedevine. Oznake mogu biti numeričke ili kategoričke, u prvom slučaju je onda riječ o regresiji a u drugom o klasifikaciji. Neki algoritmi strojnog učenja mogu se koristiti i za regresiju i za klasifikaciju, uz nužne preinake. Na primjeru predviđanja rizika davanja kredita oznaka može biti kategorička ako se kao rezultat očekuje odluka odobriti kredit ili ne ili pak neka numerička mjera koja opisuje stopu rizika odobravanja kredita. Budući da se gledanost televizijskih kanala iskazuje brojem, fokus ovog rada je na regresijskim algoritmima strojnog učenja.

Označeni skup podataka zapisujemo kao $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, gdje $\mathbf{x}$ označava vektor značajki jednog primjera a y oznaku odgovarajućeg primjera. Broj primjera u skupu je N, a pomoću varijable i indeksiramo sve parove vektora značajki i njihovih oznaka. Konstrukcija značajki iz dostupnih neobrađenih podatka koje će pridonijeti točnosti predviđanja je zahtjevan posao i o tome će više biti rečeno u idućem poglavlju.

2.1. Linearna regresija

Prvi algoritam je linearna regresija, to je relativno jednostavan algoritam koji pretpostavlja da je predviđena varijabla linearno zavisna o ulaznom vektoru značajki $\mathbf{x}$. Model ovog algoritma je definiran kao:

$$h(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_1 + \cdots + w_n x_n = w_0 + \sum_{j=1}^{n} w_j x_j \tag{2.1}$$

Hipoteze modela su parametrizirane vektorom težina $\mathbf{w}$, a broj članova u izrazu ovisi o broju značajki ulaznog vektora. Može se primijetiti da model s dvije značajke opisuje ravnine u trodimenzionalnom prostoru, a za veći broj značajki hipoteze su zapravo hiperravnine.

Druga komponenta algoritma je funkcija pogreške koja se izražava kao očekivana greška hipoteze na označenom skupu primjera. Greška hipoteze na jednom primjeru definirana je funkcijom gubitka:

$$L(y^{(i)}, h(\mathbf{x}^{(i)})) = (y^{(i)} - h(\mathbf{x}^{(i)}))^2 \tag{2.2}$$

Koristi se kvadrat razlike između stvarne vrijednosti i vrijednosti koju hipoteza predvidi za taj primjer i naziva se kvadratnim gubitkom. Kvadrat je tu zbog toga što razlika može biti pozitivna i negativna što znači da bi se inače greške suprotnih predznaka

poništavale. Funkcija pogreške je tada srednja vrijednost gubitaka svih primjera:

$$E(h|\mathcal{D}) = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - h(\mathbf{x}^{(i)}))^2 \tag{2.3}$$

Dijeli se s 2, a ne N zbog jednostavnijeg izračuna i optimizacijskog postupka.

Optimizacijski postupak ovog algoritma naziva se metoda najmanjih kvadrata zbog toga što je funkcija pogreške definirana pomoću kvadratnog gubitka, a rezultat postupka su težine koje definiraju hipotezu s najmanjom greškom. Postupak koristi matrični račun, funkcija pogreške tada se definira kao:

$$E(\mathbf{w}|\mathcal{D}) = \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathrm{T}}(\mathbf{X}\mathbf{w} - \mathbf{y}) \tag{2.4}$$

Matrica $\mathbf{X}$ je matrica dizajna čiji su retci primjeri iz označenog skupa podataka kojima je dodana jedinica na početak. Raspisivanjem ovog izraza i derivacijom po vektoru $\mathbf{w}$ dođemo do gradijenta funkcije pogreške. Ova funkcija je konveksna što znači da je njezina stacionarna točka ujedno i minimum. Izjednačavanjem gradijenta s nulom dobije se izraz za vektor težina:

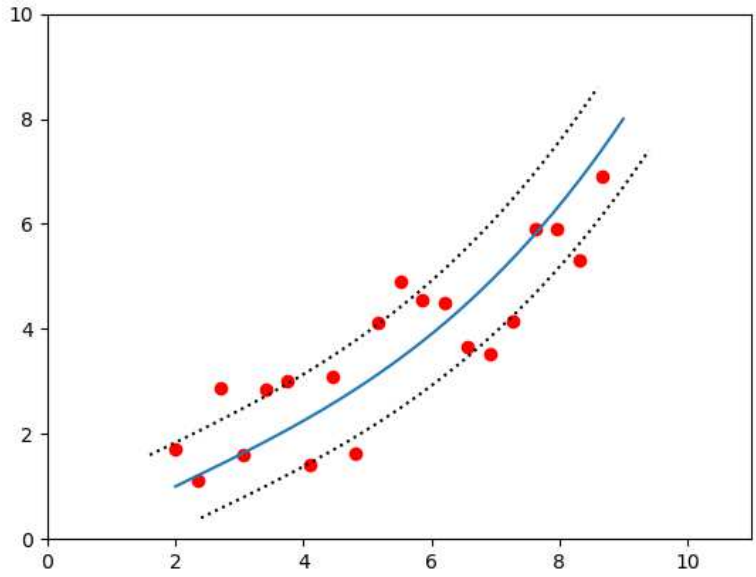
$$\mathbf{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} = \mathbf{X}^{+}\mathbf{y} \tag{2.5}$$

Matrica $\mathbf{X}^{+}$ naziva se Moore-Penroseov inverz koji ima svaka matrica. Ovaj postupak nije egzaktan nego minimizira kvadratno odstupanje, ali uvijek ima rješenje u zatvorenoj formi.

Algoritam linearne regresije često se koristi u statistici i vrlo je važan u strojnom učenju, ali zbog manje složenosti točnost predviđanja težih problema bit će manja.

2.2. Stroj s potpornim vektorima

Algoritam stroja s potpornim vektorima (engl. *support vector machine, SVM*) bio je razvijen za optičko prepoznavanje znakova ali je kasnije prilagođen za klasifikaciju i regresiju [1]. Ideja algoritma je pomoću odabranih primjera iz označenog skupa pronaći hiperravninu koja ima najmanje odstupanje od funkcije koju se želi naučiti. Podatci na temelju kojih se pronalazi najbolja hiperravnina nalaze se na granicama, ili blizu granica određenih hiperravninom i odstupanjem ε, ti primjeri nazivaju se potpornim vektorima. Na slici 2.1 crvene točke predstavljaju podatke iz označenog skupa, plava krivulja je hiperravnina, a iscrtkane krivulje su granice koje su pomaknute od hiperravnine za $\pm\varepsilon$.



Slika 2.1: SVM za regresiju

Model SVM algoritma je jednostavan:

$$h(\mathbf{x}; \mathbf{w}, w_0) = \sum_{j=1}^{n} w_j x_j + w_0 = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 \tag{2.6}$$

ali je zato optimizacijski postupak dosta složeniji.

Najprije se definiraju vrijednosti koje model poprima za primjere na granicama tako da primjeri na granici na pozitivnoj strani hiperravnine (na slici, iznad plavog pravca) poprimaju $h(\mathbf{x}) = 1$, a na drugoj granici $h(\mathbf{x}) = -1$. Tada je funkcija koju se minimizira oblika:

$$\frac{1}{2} \|\mathbf{w}\|^2 \tag{2.7}$$

uz ograničenja:

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0 - y^{(i)} \leq \varepsilon$$

$$y^{(i)} - \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} - w_0 \leq \varepsilon$$

što znači da želimo što manju udaljenost granica od hipperravnine, ali da se obuhvate svi podatci u označenom skupu. Dijeljenje s 2 je tu opet zbog matematičke jednostavnosti.

Neće uvijek biti moguće obuhvatiti sve primjere unutar granica, zbog toga se uvodi meka margina koja dopušta izlazak primjera izvan granica i pridjeljuje im određenu grešku. To se postiže uvođenjem dviju rezervnih varijabli ξ i ξ^* koje za svaki primjer

opisuju udaljenost primjera od pojedinih granica. Postoji više funkcija gubitaka koje se mogu koristiti, primjeri simetričnih funkcija gubitaka su linearna, kvadratna i gubitak Hubera. U nastavku je korištena linearna funkcija gubitka koja primjerima unutar granica ne pridaje nikakvu grešku dok greška primjera izvan granica linearno ovisi o udaljenosti od granice. Nakon uvođenja meke margine optimizacijski problem je minimizacija funkcije:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{2.8}$$

s ograničenjima:

$$y^{(i)} - \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} - w_0 \leq \varepsilon + \xi_i$$

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0 - y^{(i)} \leq \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0$$

Koeficijent *C* je hiperparametar i on određuje važnost komponenti problema, veća vrijednost koeficijenta daje veću važnost minimizaciji greške.

Funkcija koju se optimira je konveksna i kvadratna, a ograničenja su linearne funkcije. Ovakav optimizacijski problem naziva se kvadratni program. Neke od metoda za rješavanje kvadratnog programa su metode kazne, unutarnje točke, konjugiranog gradijenta i Lagrangeova dualnost. Ovdje je opisana metoda Lagrangeove dualnosti koja započinje definiranjem Lagrangeove funkcije optimizacijskog problema:

$$\begin{aligned} L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\lambda}, \boldsymbol{\lambda}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) - \sum_{i=1}^{N}(\lambda_i\xi_i + \lambda_i^*\xi_i^*) \\ - \sum_{i=1}^{N}\alpha_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0 - y^{(i)} + \varepsilon + \xi_i) \\ - \sum_{i=1}^{N}\alpha_i^*(y^{(i)} - \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} - w_0 + \varepsilon + \xi_i^*) \end{aligned} \tag{2.9}$$

Ovime smo dobili funkciju u koju su ugrađena ograničenja, jedini uvjet je da Lagrangeovi multiplikatori λ_i, λ_i^*, α_i i α_i^* poprimaju nenegativne vrijednosti. Iduće se definira dualna Lagrangeova funkcija, parametri funkcije više neće biti primarne varijable $\mathbf{w}$, w_0 i rezervne varijable nego će to biti Lagrangeovi multiplikatori, tj. dualne varijable. Do dualne Lagrangeove funkcije dođemo tako da u Lagrangeovu funkciju uvrstimo izraze dobivene izjednačavanjem parcijalnih derivacija Lagrangeove funkcije po primarnim varijablama s nulom. Rezultat je funkcija koja minimizira primarnu Lagrangeovu

funkciju po primarnim varijablama:

$$\begin{aligned}\widetilde{L}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(\mathbf{x}^{(i)})^{\mathrm{T}}\mathbf{x}^{(j)} \\ - \varepsilon\sum_{i=1}^{N}(\alpha_{i^*} + \alpha_i) + \sum_{i=1}^{N} y^{(i)}(\alpha_i - \alpha_i^*)\end{aligned} \tag{2.10}$$

i maksimizacija dualne funkcije daje minimum primarne funkcije. Ograničenja su sada:

$$\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0$$

$$\alpha_i, \alpha_i^* \in [0, C]$$

također moraju vrijediti i Karush-Kuhn-Tuckerovi uvjeti:

$$\alpha_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0 - y^{(i)} + \varepsilon + \xi_i) = 0$$

$$\alpha_i^*(y^{(i)} - \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} - w_0 + \varepsilon + \xi_i^*) = 0$$

$$(C - \alpha_i)\xi_i = 0$$

$$(C - \alpha_i^*)\xi_i^* = 0$$

Primjeri kod kojih je jedan od Lagrangeovih multiplikatora α i α^* različit od nule su potporni vektori i oni pridonose predviđanju. Definiranjem dualnog problema omogućena je uporaba algoritma slijedne minimalne optimizacije. Ovaj algoritam dijeli problem na manje probleme i rješava ih analitički, u ovom radu neće se ulaziti u detalje algoritma.

Na kraju ovog podpoglavlja razmotrit će se jezgreni trik. Nekada je podatke potrebno preslikati u prostor značajki, npr. ako podatci koje klasificiramo nisu linearno odvojivi, prije učenja ih se preslikava u prostor više dimenzije gdje će biti linearno odvojivi [2]. Dualna forma modela SVM algoritma je:

$$h(\mathbf{x}) = w_0 + \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\mathbf{x}^{\mathrm{T}}\mathbf{x}^{(i)}$$

do koje se dođe uvrštavanjem izraza za vektor težina iz parcijalne derivacije Lagrangeove funkcije izjednačene s 0. U dualnoj formulaciji modela i optimizacijskog problema (jednadžba 2.10) pojavljuje se skalarni produkt primjera. Umjesto da se računa skalarni produkt preslikanih vektora, ubacit će se jezgrena funkcija $\kappa(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{x}')$ koja ujedno preslikava primjere funkcijom ϕ i računa sličnost između dva primjera. Preslikavanje se obavlja implicitno i tako se smanjuje računalna složenost, jednostvno

je definirati jezgrenu funkciju i povećava se složenost modela koja se ne može postići preslikavanjem. Više o jezgrenim funkcijama i optimizaciji hiperparametara algoritma bit će rečeno u zadnjem poglavlju.

2.3. Slučajna šuma

Algoritam slučajne šume je zapravo algoritam ansambla stabala odluke. Stablo odluke ima takav naziv zbog toga što algoritam gradi graf koji se sastoji od čvorova koji predstavljaju odluke iz kojih se granaju dvije ili više grana koje su rezultati odluke. Početni čvor je korijen, a krajnji čvorovi su listovi koji ne sadrže odluke nego rezultantne oznake. Konačna predviđena oznaka šume se dobije uprosječivanjem ako je riječ o regresiji ili brojanjem predviđenih oznaka pojedinih stabala kod klasifikacije.

Stabla se mogu izgraditi na mnogo načina korištenjem različitih algoritama i metrika pomoću kojih se odabiru značajke na temelju kojih se donosi odluka o grananju tj. podjeli podataka. Primjeri metrika za odabir najpovoljnije značajke za klasifikaciju su entropija, informacijska dobit i nečistoća Gini, a kod regresije često se koriste srednja apsolutna pogreška i srednja kvadratna pogreška. Neki od najpoznatijih algoritama za izradu stabla su ID3 [3], C4.5 [4] i CART (Classification and Regression Trees) [5].

Entropijom se mjeri nasumičnost, definirana je kao $E = -\sum_{i=1}^{K} p_i \log_2 p_i$ gdje su p_i postotci udjela skupova primjera dobivenih podjelom. Cilj je minimizirati entropiju, najbolja podjela po određenoj značajki je ona koja ima entropiju jednaku nuli i tada svi primjeri pripadaju istoj grupi. Informacijska dobit mjeri smanjenje entropije koje bi se dogodilo podjelom na temelju određene značajke, a nečistoća Gini je mjera kojom se izražava koliko često bi se nasumično odabrani primjer krivo klasificirao ako se klasificira nasumičnim odabirom oznake na temelju raspodjele oznaka u podatcima. Kada je u pitanju regresija, potrebni su kriteriji za odabir značajki koji su prikladni za kontinuirane oznake. Srednja apsolutna pogreška računa apsolutne razlike između oznaka skupa i stvarne oznake, a srednja kvadratna pogreška računa kvadrat iste razlike što je zapravo varijanca.

Sva tri spomenuta algoritma za izradu stabala odluke imaju isti postupak:

```
1. Počni s korijenskim čvorom.
2. Napravi podjelu koristeći značajku koja na temelju
    kriterija rezultira najboljom podjelom.
3. Ako je postignut uvjet za zaustavljanje, izađi.
    Inače ponovi drugi korak za svaki čvor dijete.
```

Algoritam ID3 je jednostavan algoritam za izradu stabla odluke i prikladniji je za klasifikaciju zbog toga što izračun mogućih podjela na temelju kontinuiranih značajki može biti zahtjevan. Postupak je takav da se odabire značajka koja nakon podjele daje najmanju entropiju ili najveću informacijsku dobit, to ne mora rezultirati najoptimalnijim stablom zbog čega se algoritam smatra pohlepnim. Postupak se dalje rekurzivno nastavlja nad podijeljenim podskupovima razmatrajući druge značajke dok se ne obavi podjela na temelju svih značajki ili dok svi primjeri podskupa ne pripadaju jednoj klasi ili ako je podskup prazan.

Nasljednik ovog algoritma je C4.5 koji olakšava korištenje kontinuiranih značajki, omogućava učenje na podatcima za učenje s vrijednostima značajki koje nedostaju i uvodi podrezivanje stabla čime se smanjuje prenaučenost. Podrezivanje se obavlja tako da se potpuno izgrađeno stablo pretvori u skup pravila koja za svaki list određuju put od korijena do lista, zatim se ta pravila podrezuju ako to poboljšava točnost. Rezultantni skup pravila se sortira prema procijenjenoj točnosti i može ih se koristiti za predviđanje oznake.

Algoritam CART dijeli skupove na svakom čvoru u samo dva podskupa i omogućuje višestruko dijeljenje skupova na temelju iste značajke. Podrezivanje potpuno izgrađenih stabala obavlja se korištenjem podrezivanja minimalnog troška i kompleksnosti koji je parametriziran parametrom kompleksnosti, njega se optimizira unakrsnom provjerom. Rezultat algoritma je više stabala a konačno, optimalno stablo se odabire korištenjem skupa podataka za testiranje ili unakrsnom provjerom [6].

Problemi algoritma stabla odluke su prenaučenost i velika varijanca koja se očituje tako što male promjene u podatcima za učenje rezultiraju posve drugačijim stablom koje će se izgraditi. Prenaučenost se može spriječiti podrezivanjem i ograničavanjem maksimalne dubine stabla, a za smanjenje varijance koristi se ideja o ansamblu procjenitelja. Postoji mnogo metoda ansambala, ovdje ću istaknuti algoritam uzdizanja AdaBoost (Adaptive Boosting) [7] te *bagging* (*bootstrap aggregating*) metode u koje spada algoritam slučajne šume.

Kod metoda uzdizanja slabiji procjenitelji se uče jedan za drugim s ciljem popravljanja greški prethodnog procjenitelja, tj. smanjenja pristranosti. Rezultantni procjenitelj dobije se ponderiranom kombinacijom slabih procjenitelja na temelju njihove točnosti. Algoritam AdaBoost to postiže učenjem slabih procjenitelja s modificiranim verzijama podataka za učenje kojima se pridaje vektor težina odnosno važnosti. Nakon svakog učenja, podatcima kojima je oznaka krivo predviđena pridaje se veća važnost pri učenju idućeg procjenitelja, a točno predviđenima se smanjuje važnost. Tako se fokus kasnijih procjenitelja preusmjerava na primjere koje je teško naučiti.

Bagging metode su metode uprosječivanja kod kojih se više procjenitelja uči zasebno koristeći različite skupove za učenje i potom se njihovi izlazi spajaju kao rezultat predviđanja. Smanjenje varijance se postiže uvođenjem nasumičnosti tako da se podskupovi za učenje pojedinih procjenitelja generiraju nasumičnim uzorkovanjem s ponavljanjem početnog skupa za učenje. Metode uzdizanja se koriste sa slabim procjeniteljima, a *bagging* radi bolje s jakim procjeniteljima zbog toga što smanjuje prenaučenost. Primjer jakog procjenitelja je potpuno izgrađeno stablo odluke.

Algoritam slučajne šume je *bagging* algoritam koji kao osnovne procijenitelje koristi stabla odluke. Osim nasumičnih podskupova kojima se uče stabla, ovaj algoritam kod svake podjele značajku na temelju koje se obavlja podjela odabire iz nasumičnog podskupa značajki. Broj značajki koji se uzima u obzir može se optimirati ovisno o problemu koji se rješava. Uvođenjem još jedne razine nasumičnosti dolazi se do algoritma ExtraTrees (*extremely randomized trees*) [8]. Taj algoritam dodatno nasumično određuje vrijednosti značajki koje se razmatraju kao kriterij za podjelu skupa na čvoru stabla, tako se još više smanjuje varijanca procjenitelja.

3. Implementacija web aplikacije s pametnim sustavom za predviđanje gledanosti

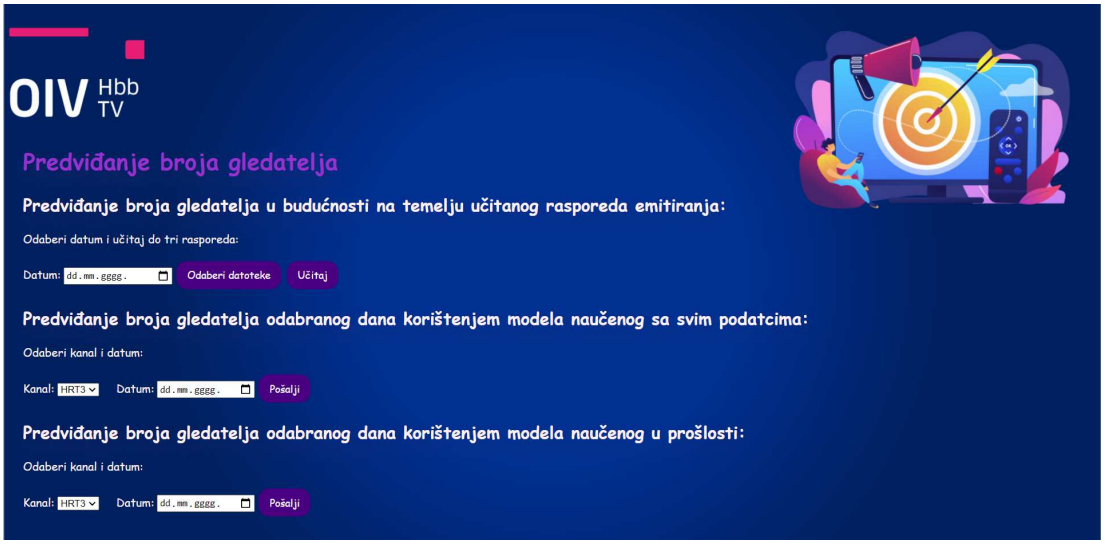
Kako bi krajnji korisnici mogli koristiti sustav za predviđanje gledanosti, odlučeno je razviti i implementirati web aplikaciju koja pruža lak pristup i intuitivno sučelje za korištenje pametnog sutava za predviđanje gledanosti. Web aplikacija je implementirana u sklopu diplomskog projekta kao rješenje studentskog zadatka tvrtke Odašiljači i veze d.o.o., a tijekom izrade ovog rada su napravljena poboljšanja.

Za rad aplikacije potrebni su podatci o emitiranom sadržaju i pripadajuće gledanosti televizijskih kanala tijekom emitiranja tih sadržaja. Predviđaju se gledanosti dva kanala Hrvatske radiotelevizije (HRT), javne televizije Republike Hrvatske. Kanali čiji su podatci dostupni su HRT3 i HRT4 na kojima se ne emitira najvažniji program. Na HRT3 se emitiraju zabavni, kulturni i u doba pandemije obrazovni sadržaj za niže razrede osnovne škole, a HRT4 je informativnog karaktera. Podatci o emitiranom sadržaju dohvaćaju se iz rasporeda emitiranja s poslužitelja HRT-a, a podatci o gledanosti pohranjeni su u sustavu Elasticsearch kojem se pristupa pomoću Pythonovog klijenta. Za spremanje podataka korištena je baza podataka SQLite [9], modul sqlite3 koji je ugrađen u Python pruža potpuno sučelje za rad s bazama SQLite.

3.1. Web aplikacija

Funkcionalnosti koje aplikacija pruža su predviđanje gledanosti u budućnosti za učitani raspored emitiranja, učenje modela i predviđanje gledanosti u prošlosti te dohvaćanje predviđanja koje se obavlja svakog dana u kojem se predviđa gledanost idućeg dana. Početna stranica web aplikacije može se vidjeti na slici 3.1.

Web aplikacija je implementirana programskim jezikom Python i radnim okvirom Flask [10]. Aplikaciju pomoću modula mod_wsgi korisnicima predočava poslužitelj



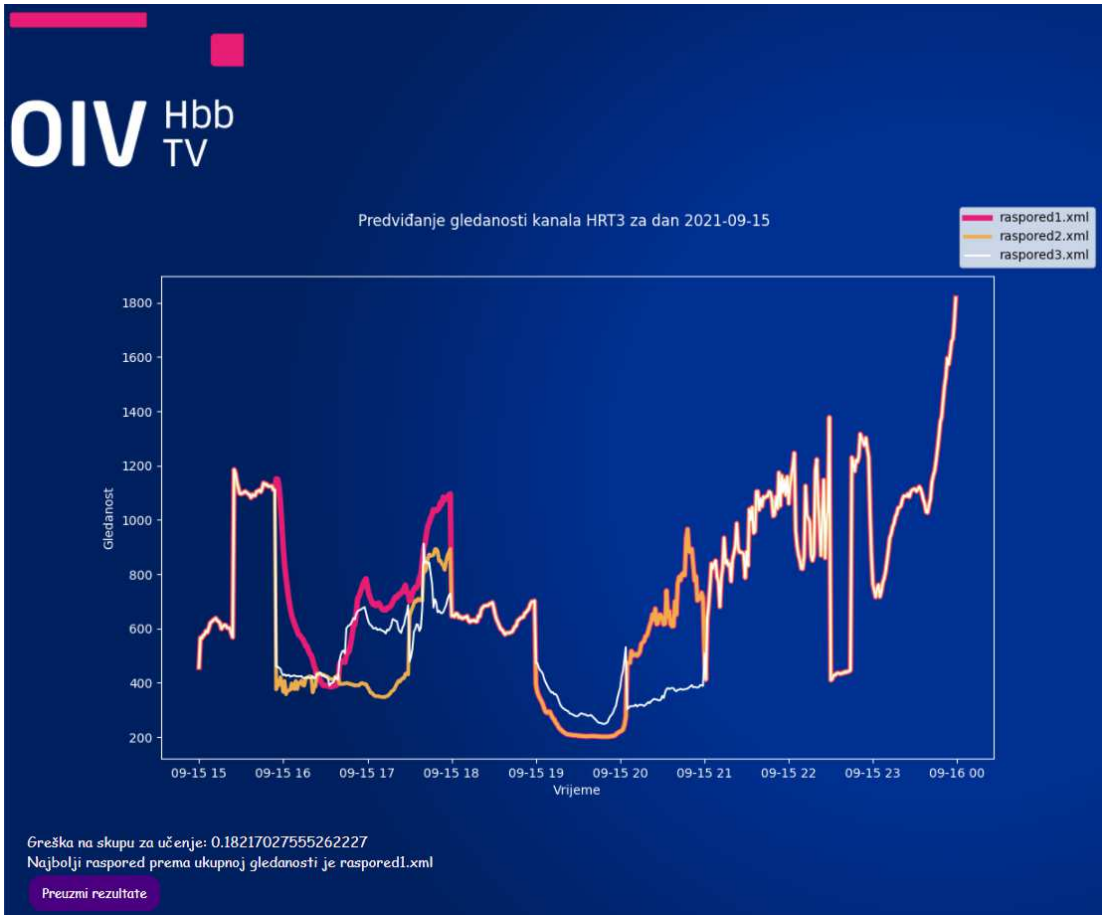
Slika 3.1: Početna stranica web aplikacije

Apache. Budući da učenje modela i predviđanje može potrajati, takvi zadatci se prosljeđuju drugom procesu kako bi aplikacija mogla nastaviti posluživati druge zahtjeve korisnika. Konkretno ti procesi su radnici radnog okvira Celery [11], raspodijeljenog reda zadataka. Pomoću njega moguće je rasporediti zadatke na više dretvi ili na više računala. Za prosljeđivanje zadataka do radnika potreban je posrednik poruka, u implementaciji je korišten Redis [12]. Redis je sustav za pohranu struktura podataka u radnoj memoriji i može se koristiti kao raspodijeljena baza podataka, priručna memorija ili posrednik poruka. Kada korisnik pomoću web aplikacije pokrene zadatak, poslužitelj Redis ga stavi u red čekanja. Taj red povremeno provjeravaju Celeryjevi radnici i kada jedan od njih bude spreman odraditi zadatak preuzet će ga i izvršiti.

Za implementaciju korisničkog sučelja korišteni su HTML, CSS, JavaScript i AJAX (Asynchronous JavaScript and XML). Kada korisnik pošalje zahtjev za predviđanje ili učenje modela preusmjerit će ga se na stranicu s rezultatima koja isprva prikazuje animaciju učitavanja i moli korisnika za strpljenje. Za to vrijeme zadatak se šalje u red i čeka obradu, a JavaScriptov kod koji se izvršava na klijentskom računalu svake sekunde pomoću AJAX-ovog poziva provjerava je li zadatak završen. Kada radnik izvrši zadatak, klijentov preglednik će ažurirati stranicu i zamijeniti animaciju čekanja s rezultatima izvođenja zadatka. Tako se omogućava promjena sadržaja web stranice bez ponovnog učitavanja cijele stranice.

Prva funkcionalnost omogućava učitavanje do tri rasporeda i obavlja predviđanje gledanosti učitanih rasporeda. Prikaz rezultata prve funkcionalnosti može se vidjeti na slici 3.2. Na grafu se prikazuju predviđene gledanosti svih učitanih rasporeda za

svaku minutu odabranog dana. Ovako se rasporedi mogu vizualno usporediti i može se vidjeti kako različiti sadržaji ili vrijeme emitiranja jednog sadržaja utječe na predviđenu gledanost. Ispod grafa ispiše se greška modela ostvarena predviđanjem skupa za učenje, greška je definirana jednadžbom 4.3. Uz grešku ispiše se i naziv datoteke rasporeda koji postiže najveću ukupnu predviđenu gledanost tijekom dana te je omogućeno dohvaćanje rezultata predviđanja najboljeg rasporeda u csv formatu klikom na poveznicu.



Slika 3.2: Stranica s rezultatima

Rezultati druge dvije funkcionalnosti aplikacije su grafovi koji prikazuju predviđanje gledanosti odabranog dana u prošlosti i stvarnu gledanost tog dana. Razlika je u tome što druga funkcionalnost na korisnikov zahtjev uči model na temelju svih dostupnih podataka osim odabranog dana, a zadnja funkcionalnost omogućava samo dohvaćanje predviđanja koje se obavlja samo za protekli dan pomoću modela naučenog sa podatcima koji su bili dostupni do tog dana.

3.2. Model strojnog učenja

Podatci na temelju kojih se želi obavljati predviđanje mogu biti strukturirani na različite načine. Preduvjet za dobro predviđanje je konstrukcija značajki koje dobro opisuju fenomen koji se predviđa. Značajke konstruirane iz početnih podataka trebale bi biti informativne i pridonijeti predviđanju, redundantne značajke ne pridonose predviđanju, npr. značajka mjerenja temperature u Celzijima i druga značajka u Fahrenheitima.

Raspored emitiranja formatiran je u XML-u (Extensible Markup Language) i sastoji se od planiranog rasporeda emitiranja za iduća dva tjedna, ali moguće su promjene stoga se svakog dana preuzima raspored idućeg dana koji će biti najtočniji. U rasporedu svaki zapis sadrži detalje o jednom sadržaju: vrijeme početka i kraja emitiranja u satima i minutama, naslov, i kategoriju sadržaja. Raspored emitiranja određen je za svaku minutu u danu, stoga su i gledanosti zbrojene za svaku minutu i model predviđa gledanosti u minutama. U tablici 3.1 prikazan je jedan primjer sadržaja emitiranog na kanalu HRT3 naslova "Hrvati koji su mijenjali svijet (1): Nikola Tesla (R)". Najveći utjecaj na gledanost sadržaja koji se emitira ima vrijeme emitiranja. Vrijeme emitiranja definirano je s dvije značajke, potrebne su obje značajke kako bi se opisala ciklička priroda 24 satnog dana. To se postiže sinusnom i kosinusnom transformacijom definiranom jednadžbom:

$$sin\_time = \sin(2 * \pi * \text{sekunda u danu})$$

gdje sekunda u danu označava broj sekundi proteklih od ponoći do minute u kojoj se emitira sadržaj. Druga značajka, cos_time, definirana je pomoću kosinusne funkcije umjesto sinusne. Iz početka i kraja emitiranja još se definiraju značajke trajanja emitiranja i dana u tjednu. Sadržaj koji se ponovno emitira u naslovu ima oznaku "(R)" pomoću koje se određuje značajka repriziranja. Posljednja značajka je značajka kategorije sadržaja koji se emitira, moguće kategorije ovise o kanalu, a primjeri kategorija su vijesti, umjetnost i kultura, glazba, obrazovanje i znanost, igrani program i religija. Primjeri su indeksirani po vremenskim oznakama minuta a oznaka primjera je u stupcu *viewership* i označava broj jedinstvenih gledatelja emitiranog sadržaja u toj minuti.

Tablica 3.1: Konstruirane značajke

timestamp	sin_time	cos_time	rerun	duration	category	viewership
2021-05-31 07:56	0.87461	-0.48480	True	4	ZNANOST	315

Sadržaji u rasporedu mogu imati elemente u kojima je opis sadržaja, popis glumica i glumaca, redatelja, pisaca ili urednika, originalan naslov i godina izlaska sadržaja, ali

ti podatci nisu odabrani za konstrukciju značajki zbog malog broja zapisa, nekonzistentnosti ili se nisu činili korisnima za predviđanje.

Model je implementiran kao zaseban razred i može ga se lako mijenjati. Konkretni algoritam koji je odabran i postupak odabira opisan je u idućem poglavlju.

3.3. Alternativne implementacije

Ovdje su opisane alternativne tehnologije i radni okviri programskog jezika Python za razvoj web aplikacije koje sadrže i podsustav sa strojnim učenjem.

Postoji mnogo besplatnih radnih okvira za razvoj web aplikacija, za izradu ove aplikacije odabran je Flask zbog toga što omogućava brz razvoj jednostavnijih aplikacija i zbog prethodnog iskustva rada s tim mikrookvirom. Okvir Django [13] je još jedan vrlo popularan web radni okvir za Python, on omogućava lak razvoj i implementaciju čak i složenijih aplikacija. Django ima ugrađenu potporu za forme, autentifikaciju i autorizaciju korisnika i objektno-relacijsko mapiranje za lakši rad s bazama podataka. Zbog ugrađene potpore smanjena je sloboda izbora komponenti koje korisnik okvira može koristiti i zahtjeva se upoznavanje s pojedinostima prije početka rada. Za Flask su razvijena razna proširenja koja se mogu koristiti za implementaciju navedenih značajki. Osim ova dva radna okvira spomenut ću web2py, CubicWeb i Pyramid.

Alternative redu zadataka Celery su RQ (Redis Queue), Huey i Dramatiq. Poput web okvira, razvijatelj sustava mora na temelju zahtjeva i ograničenja odabrati implementaciju reda koja mu odgovara. Navedeni redovi se razlikuju u potpori za prioritet zadatka, automatsko ponavljanje neuspješno izvedenog zadatka, odgađanje izvođenja i podržanih posrednika poruka. Tablica usporedbe može se vidjeti na web stranici Dramatiq reda zadataka [14].

Celery podržava rad sa sljedećim posrednicima poruka: Redis, RabbitMQ i Amazon SQS. Posrednici se razlikuju u brzini kojom mogu slati poruke i perzistentnosti dostavljanja. Redis je prikladan za brzo slanje malih poruka koje su kratkog životnog vijeka, a perzistentnosti nema zbog toga što je to praktički baza podataka koja se čuva u radnoj memoriji. RabbitMQ je prvotno razvijen kao posrednik poruka i podržava manji broj poruka u jedinici vremena ali je pouzdaniji za isporuku, može koristiti SSL za zaštitu sigurnosti podataka i podržava i veće poruke. Amazon SQS (Simple Queue Service) je usluga koji nudi tvrtka Amazon, a mogu se spomenuti i Apache Kafka te carrot.

4. Usporedba točnosti predviđanja algoritama

U ovom poglavlju opisan je postupak usporedbe točnosti predviđanja ranije opisanih algoritama. Usporedba je odrađena pomoću virtualiziranog računala s 4 procesorske jezgre i 8 GB radne memorije te operacijskim sustavom CentOS. Korištene su implementacije algoritama strojnog učenja iz Pythonovog modula scikit-learn [15] ili sklearn. To je knjižnica otvorenog koda koja je fokusirana na strojno učenje i sastoji se od implementacija većeg broja algoritama strojnog učenja i alata za pripremu podataka, učenje i evaluaciju modela. Sve implementacije algoritama u ovom modulu sadrže funkcije *fit* i *predict*. Funkcija *fit* kao argumente prima skup primjera i skup pripadajućih oznaka i odrađuje učenje modela, a funkcijom *predict* predviđaju se oznake neviđenih primjera.

4.1. Implementacije algoritama

Redom, prvi algoritam objašnjen u drugom poglavlju je algoritam linearne regresije. U modulu sklearn algoritam je implementiran kao razred:

```
class sklearn.linear_model.LinearRegression()
```

Kako je ranije objašnjeno, ovaj algoritam pronalazi težine koje minimiziraju kvadratno odstupanje. Nakon učenja, procijenjeni vektor težina modela $\mathbf{w}$ može se dohvatiti pomoću atributa *_coef*.

Drugi algoritam je SVM za regresiju:

```
class sklearn.svm.SVR(kernel='rbf', gamma='scale', C=1.0,
    epsilon=0.1)
```

Konstruktor objekta prima određene argumente, ovdje su navedene zadane vrijednosti pojedinih istaknutih argumenata. Argument *C* odgovara koeficijentu važnosti minimizacije greške, a *epsilon* udaljenosti margina od hiperravnine unutar kojih primjeri ne

pridonose izračunu greške, ova dva argumenta već su spomenuta ranije. Argumenti *kernel* i *gamma* odnose se na jezgrene funkcije, vrijednost "rbf" označava radijalnu baznu funkciju čija je definicija:

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}) = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2) \tag{4.1}$$

Ova jezgrena funkcija poprima vrijednosti između 1 i 0, a vrijednost je veća što su primjeri sličniji jedan drugome tj. što su bliži u ulaznom prostoru. Koeficijent γ određuje brzinu smanjivanja vrijednosti funkcije kada udaljenost između primjera pada. Osim radijalne funkcije implementirane su i linearna, polinomijalna i sigmoidalna jezgrena funkcija, ali moguće je definirati i vlastitu funkciju.

Algoritam ExtraTrees definiran je razredom:

```
class sklearn.ensemble.ExtraTreesRegressor(n_estimators=100,
    criterion='mse', max_depth=None, min_samples_split=2)
```

Implementacija koristi algoritam CART koji je ranije spomenut. Argumentom *n_estimators* postavlja se broj stabala u šumi. Što se tiče kriterija za izračun kvalitete podjele argumentom *criterion* može se odabrati između srednje apsolutne pogreške i srednje kvadratne pogreške. Pomoću *max_depth* može se spriječiti prenaučenost tako da se ograniči maksimalna dubina stabala. Posljednjim argumentom određuje se minimalni broj primjera potrebnih kako bi se razmotrilo podjelu tog skupa, broj 2 označava potpunu izgradnju stabla dok svi završni čvorovi nemaju samo po jedan primjer.

4.2. Optimizacija hiperparametara

Neki algoritmi imaju slobodne parametre, odnosno hiperparametre, kojima se određuje složenost modela. Optimizacijom se pronalaze hiperparametri za koje će rezultirajući model najbolje predviđati određeni skup podataka. To se postiže maksimizacijom odabrane metrike točnosti predviđanja, ali uz točnost predviđanja cilj je doći do modela koji dobro generalizira odnosno koji nije prenaučen.

Za procjenu pogreške modela u ovom radu koristi se unakrsna validacija. U postupku k-struke unakrsne validacije skup za učenje dijeli se na k preklopa te se učenje obavlja k puta tako da se svaki put odabere drugi preklop kao testni podskup, a model se uči na ostalim preklopima. Rezultantna točnost modela je prosjek izračunatih točnosti predviđanja testnih podskupa i ona pokazuje koliko dobro model generalizira.

Neke od najpopularnijih metoda za optimizaciju hiperparametara su slučajno pretraživanje, pretraživanje po rešetci (engl. *grid search*), Bayesova optimizacija, optimizacija na temelju gradijenta i evolucijska optimizacija. Ovdje je objašnjena samo metoda pretraživanja po rešetci zbog toga što je ona korištena pri odabiru najboljih parametara. To je metoda iscrpnog pretraživanja u kojoj se iz zadanog skupa hiperparametara, tzv. rešetke, pronalaze parametri koji rezultiraju najboljim modelom. Metoda uči i validira model za svaku kombinaciju parametara iz rešetke. Budući da je ovo iscrpna metoda, optimizacija može za veći broj parametara i njihovih mogućih vrijednosti biti vremenski zahtjevna. Iz tog razloga vrijednosti parametara koje se razmatraju moraju biti dobro odabrane i tu često iskustvo može pomoći.

U modulu sklearn implementirane su metode slučajnog pretraživanja i metoda pretraživanja po rešetci. Razred u kojem je implementirana metoda pretraživanja po rešetci je:

```
class sklearn.model_selection.GridSearchCV(estimator,
    param_grid, scoring='r2', cv=5)
```

Konstruktor prima procjenitelja čije će hiperparametre optimirati, rešetku parametara u obliku Pythonovog rječnika, metriku na temelju koje se uspoređuje naučene modele i vrstu unakrsne validacije. Korištena je metrika r2 koja je definirana kao:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.2}$$

gdje je $\bar{y}$ srednja vrijednost oznaka, u brojniku je suma kvadrata reziduala a u nazivniku suma kvadrata razlika oznaka od njihove srednje vrijednosti. Ova metrika se naziva koeficijent odlučnosti i izražava koliko dobro će model predvidjeti neviđene primjere.

Neki algoritmi daju dobre rezultate za većinu primjena i bez optimizacije hiperparametara, jedan od njih je ExtraTrees, dok algoritam SVR gotovo nikad neće dobro predviđati sa zadanim hiperparametrima. Hiperparametri koji su optimirani u ovom radu su parametri *gamma* i *C* algoritma SVR te *max_depth* i *min_samples_split* algoritma ExtraTreesRegressor. Stvar na koju je potrebno obratiti pažnju je međuovisnost parametara *gamma* i *C* zbog čega ih je potrebno zajedno optimizirati.

Optimizacija je obavljena tijekom izrade ovog rada, primjer optimiranja parametara algoritma SVR može se vidjeti u nastavku:

```
svm_gsc = GridSearchCV(
    estimator=SVR(),
    param_grid={
```

```
        'C': [0.1, 0.5, 1, 5, 10, 20, 50, 100, 1000],
        'gamma': [0.01, 0.05, 0.1, 0.5, 1, 5, 10, 100]
    },
    scoring='r2',
    cv=5
)
grid_result = svm_gsc.fit(X_train, y_train)
model = SVR(**grid_result.best_params_)
model.fit(X_test, y_test)
```

U isječku koda najprije se inicijalizira objekt pretrage po rešetci koji će obavljati optimizaciju, konstruktoru se predaje rječnik s dva ključa koji odgovaraju nazivima parametara procjenitelja i listama vrijednosti tih parametara. U ovom primjeru lista vrijednosti parametra *C* ima 9 vrijednosti a lista parametra *gamma* sadrži 8 vrijednosti što znači da će postupak optimizacije isprobati 72 kombinacije parametara. Optimizacija se obavlja pozivanjem funkcije *fit* nad objektom pretrage po rešetci, njezini argumenti su podatci za učenje i njihove pripadajuće oznake. Nakon učenja najbolji parametri pohranjeni su u atributu *best_params_* i pomoću njega može se inicijalizirati optimalni model. Vrijednost 5 parametra *cv* odgovara peterostrukoj unakrsnoj validaciji.

Optimalne vrijednosti hiperparametara za dostupne podatke mogu se vidjeti u tablici 4.1.

Tablica 4.1: Optimalni hiperparametri

	C		gamma	
SVR	HRT3	HRT4	HRT3	HRT4
	1000	1000	100	100
	max_depth		min_samples_split	
ExtraTrees	HRT3	HRT4	HRT3	HRT4
	20	15	2	2

Učenje modela algoritmom SVR koristeći veći broj primjera traje jako dugo zbog toga što vremenski kvadratno ovisi o broju primjera. Dobivene vrijednosti optimalnih hiperparametara *C* i *gamma* su više od očekivanih što znači da je model vrlo složen i da je utjecaj pojedinih vektora na greške drugih vrlo mali. Zbog vremenske zahtjevnosti nije obavljena detaljna optimizacija i visoke vrijednosti optimalnih hiperparametara ukazuju na to da je moguće doći i do boljih vrijednosti hiperparametara algoritma SVR. Algoritam ExtraTrees davat će bolje rezultate s povećanjem broja estimatora

kojeg se postavlja argumentom *n_estimators*. To poboljšanje monotono raste i stagnira nakon određenog broja estimatora, ali zbog ograničene veličine radne memorije broj estimatora je ostavljen na zadanoj vrijednosti 100.

4.3. Usporedba i rezultati

Dostupni su podatci od 13.11.2020. do datuma provedbe usporedbe, 13.6.2021. odnosno do dana prije što je skoro 7 mjeseci podataka. Međutim, nedostaju podatci od par sati tijekom tog razdoblja kada se zbog greške nisu dohvatili i spremili podatci o gledanosti. Osim toga dogodio se ispad sustava od kojeg se dohvaćaju podatci o gledanosti pa se ne uzimaju u obzir podatci od dva dana (4.2. i 5.2.). Odbačeni su podatci 5 dana oko Nove godine zbog toga što je gledanost višestruko veća i odudara od ostalih podataka. Također se primjeri sa gledanosti manjom od 5 ne uzimaju u obzir zbog toga što to ukazuje na grešku u radu sustava za prikupljanje podataka. Na kraju ostaje više od 294000 primjera za svaki od dva televizijska kanala.

U postupku usporedbe algoritama najprije se početni skup podataka nasumično razdvoji na skup za učenje i ispitni skup. Za ispitivanje se obično odvaja 30 do 40% dostupnih podataka, u ovom slučaju odvojena je jedna trećina. Algoritmi se uspoređuju na temelju točnosti predviđanja ispitnog skupa nakon učenja pomoću skupa za učenje. Metrika za točnost predviđanja koja se koristi u ovom radu je srednja apsolutna pogreška:

$$MAPE(y, \hat{y}) = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{4.3}$$

Ova metrika je intuitivna za krajnjeg korisnika i može se koristiti zato što se ne uzimaju u obzir primjeri kojima je gledanost jednaka 0 pa ne može doći do dijeljenja s nulom.

Ostvarene greške predviđanja ispitnog skupa mogu se vidjeti u tablici 4.2.

Tablica 4.2: Greške predviđanja

	HRT3	HRT4
Linearna regresija	0.840	0.417
SVR	0.138	0.157
ExtraTrees	0.170	0.177

Iz rezultata se može vidjeti da model dobiven učenjem algoritmom linearne regresije nije dovoljno složen da bi dobro opisao vezu između značajki i oznaka. Učenje

algoritmom SVR rezultira modelom koji predviđa bolje od modela naučenog algoritmom ExtraTrees, ali ta točnost dolazi uz puno duže učenje modela i predviđanje. Kako druga dva algoritma daju zadovoljavajuće rezultate za svrhu optimizacije rasporeda emitiranja, a učenje i predviđanje modelom dobivenom algoritmom SVR traje višestruko duže, za korištenje u aplikaciji odabran je algoritam ExtraTrees.

5. Zaključak

Razvoj sustava sa strojnim učenjem je težak posao koji zahtjeva individualan pristup problemu kojeg se rješava, od pripreme neobrađenih podataka, odabira algoritma do korisničkog sučelja. Obavljena su poboljšanja web aplikacije i eksperimentalnom usporedbom je odabran algoritam koji ima zadovoljavajuću sposobnost predviđanja i odgovara zahtjevima aplikacije. Usporedba je pokazala da od odabranih algoritama stroj potpornih vektora ima najbolje rezultate predviđanja gledanosti na temelju podataka koji su bili dostupni za izradu ovog rada. Web aplikacija omogućava učenje modela i predviđanje na zahtjev pa vremenska složenost igra ulogu u odabiru algoritma. Iz tog razloga odabrani algoritam je algoritam slučajne šume, odnosno ExtraTrees koji ima nešto manju točnost, ali je puno brži.

Tijekom izrade ovog rada i web aplikacije dobio sam priliku razviti sustav koji koristi strojno učenje od početne ideje do produkcije i tako iskoristiti i produbiti znanje koje sam stekao na studiju. Nadam se da ću imati priliku raditi na sličnim sustavima u budućoj profesionalnoj karijeri.

LITERATURA

[1] Alex J. Smola i Bernhard Schölkopf. A tutorial on support vector regression. 2004. URL https://doi.org/10.1023/B:STCO.0000035301.49549.88.

[2] Jan Šnajder i Bojana Dalbelo Bašić. Strojno učenje, 2014. URL https://www.fer.unizg.hr/_download/repository/StrojnoUcenje.pdf.

[3] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 2008. URL https://doi.org/10.1007/BF00116251.

[4] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1558602380.

[5] L. Breiman, J. Friedman, C.J. Stone, i R.A. Olshen. *Classification and Regression Trees*. Taylor & Francis, 1984. ISBN 9780412048418. URL https://books.google.hr/books?id=JwQx-WOmSyQC.

[6] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Hiroshi Yang, Qiang Motoda, Angus McLachlan, Geoffrey J.and Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, i Dan Steinberg. Top 10 algorithms in data mining. 2008. URL https://doi.org/10.1007/s10115-007-0114-2.

[7] Yoav Freund i Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. ISSN 0022-0000. doi: https://doi.org/10.1006/jcss.1997.1504. URL https://www.sciencedirect.com/science/article/pii/S002200009791504X.

[8] Pierre Geurts, Damien Ernst, i Louis Wehenkel. Extremely randomized trees.

Machine Learning, 63:3–42, 2006. URL https://doi.org/10.1007/s10994-006-6226-1.

[9] Richard D Hipp. SQLite, 2020. URL https://www.sqlite.org/index.html.

[10] Miguel Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[11] Ask Solem. Celery, 2009. URL https://docs.celeryproject.org/en/stable/index.html.

[12] Salvatore Sanfilippo. Redis. URL https://redis.io/.

[13] Django Software Foundation. Django. URL https://djangoproject.com.

[14] Bogdan Popa. Motivation, 2017. URL https://dramatiq.io/motivation.html. Pristupljeno: 2.6.2021.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, i E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Algoritmi strojnog učenja za predviđanje gledanosti televizijskih kanala

Sažetak

Ovaj rad opisuje teorijsku podlogu algoritama strojnog učenja koji su razmatrani za predviđanje gledanosti televizijskih kanala. Eksperimentalna usporedba odabranih algoritama pokazala je da algoritam stroja potpornih vektora za regresiju (SVR) daje najbolje rezultate predviđanja dostupnih podataka o stvarnim gledanostima. Algoritam linearne regresije ne daje zadovoljavajuće rezultate, a algoritam slučajne šume (Extra-Trees) ima malo lošije rezultate od stroja potpornih vektora. Opisana je i implementacija web aplikacije koja korisnicima omogućava predviđanje gledanosti na temelju učitanog rasporeda emitiranja. Aplikacija se može koristiti kao alat za optimizaciju rasporeda. Zbog velike vremenske složenosti algoritma stroja potpornih vektora, za korištenje u aplikaciji odabran je algoritam slučajne šume.

Ključne riječi: strojno učenje, regresija, stroj potpornih vektora, slučajna šuma, web aplikacija.

Machine Learning Algorithms for Forecasting Television Channels Viewership

Abstract

This paper describes the theoretical basis of machine learning algorithms that have been considered for forecasting the viewership of television channels. An experimental comparison of chosen algorithms showed that the support vector regression algorithm (SVR) gives the best results on the available real world viewership data. The linear regression algorithm does not give satisfactory results, and the extremely randomized trees algorithm (ExtraTrees) has slightly worse results than the support vector machine algorithm. The implementation of a web application that allows users to forecast viewership based on the uploaded broadcast schedule is also described. The application can be used as a schedule optimization tool. Due to the high time complexity of the support vector machine algorithm, the randomized trees algorithm was chosen for use in the application.

Keywords: machine learning, regression, support vector machine, random forest, web application.