Predviđanje cijena kuća

Lucija Puškarić, Maja Piskač, Ivona Raguž, Mia Tadić

Sažetak—Promatramo različite metode (*Slučajne šume*, *XGBoost*, *Elastic-Net*, neuronska mreža) u predviđanju cijena kuća te prikazujemo rezultate predviđanja na testnom skupu.

Index Terms—strojno učenje, predviđanje cijena kuća, XGBoost, Random Forests, neuronska mreža, Elastic-NET

1 Uvod

Cijena nekretnina važan su odraz ekonomije. Rasponi cijena kuća od velikog su interesa kako za kupce, tako i za prodavače. Cilj ovog projekta je stvoriti model koji je u stanju precizno procijeniti cijenu kuće u Amesu, Iowi, obzirom na njene značajke, tj. aspekte. Projektni zadatak sastavni je dio Kaggle natjecanja od 2016. godine (https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview).

U nastavku navodimo modele korištene pri predviđanju cijena kuća, uspoređujemo dobivene rezultate primjenom istih metoda, no također pristupamo natjecanju kao tim te analiziramo naše rangiranje na natjecateljskoj ljestvici.

1.1 Motivacija

Kupci se obično pri kupnji kuće vode nekim osnovnim stavkama kao što su broj soba, izgled i stanje kuće. No, zanimljivo je promotriti predviđanje cijene kuća čiji su nam brojni aspekti i pokazatelji njihovih kvaliteta poznati, pa čak i oni za koje na prvu mislimo da uopće nemaju utjecaja na ukupnu cijenu.

1.2 Ciljevi

Cilj ovog projekta je stvoriti modele koji su u stanju što preciznije predvidjeti cijene kuća u Amesu, Iowi, obzirom na njene značajke, tj. aspekte.

1.3 Opis podataka

Skup podataka koje koristimo pri rješavanju problema poznat je pod nazivom *The Ames Housing dataset*. Podaci su sastavljeni su od strane Deana De Cocka u svrhe *data science* obrazovanja. Skup podataka je nadogradnja na standardni *Boston housing dataset*, s većim brojem primjera i značajki. Podaci o cijenama kuća odgovaraju razdoblju od 2006. do 2010. godine.

Za razvoj modela koristimo podatke iz skupa za treniranje (*engl. training dataset*) koji se sastoji od 1460 primjera opisanih s 80 značajki (atributa), od kojih je jedna značajka cilja - ona čija nas vrijednost zanima, tj. ona koju nastojimo predvidjeti. Preciznost modela utvrđujemo njihovim testiranjem na skupu za testiranje (*engl. test dataset*) s 1459 primjera.

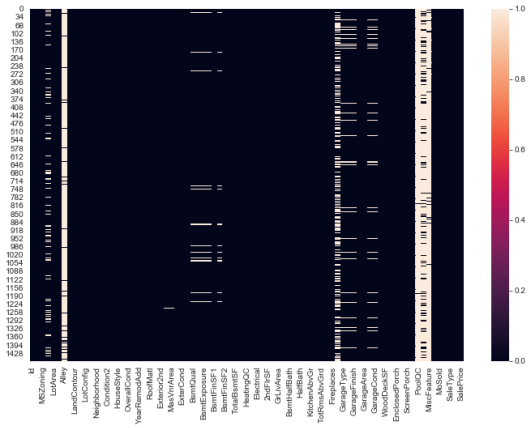
2 Podaci i transformacije podataka

U ovom odjeljku detaljnije opisujemo podatke i njihove značajke te vršimo različite transformacije nad podacima.

Atributa ima 79 uz dodatno ciljni atribut *SalePrice*. Popis svih atributa (*engl. attribute*, *feature*) i njihovih značenja moguće je pronaći na sljedećoj poveznici: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data?select=data_description.txt. Samo neki od atributa su, naprimjer: broj kupaonica, veličina garaže u kvadratnim metrima, veličina bazena, tip fasade, tip trijema...

2.1 Nedostajući podaci (*engl. missing values*)

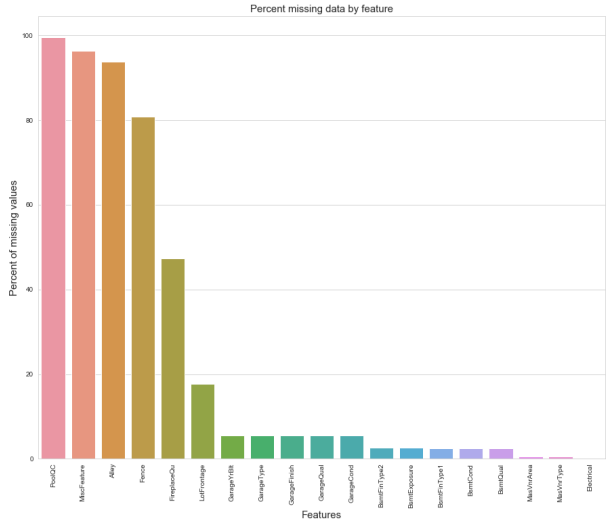
Trening skup podataka je imao čak 6,965 nedostajućih vrijednosti, a testni skup 7000. Prikazujemo kako izgleda heatmap gdje crni podaci predstavljaju "normalne" podatke, a svijetlim crticama su označeni *missing values*:



Slika 1. *Heatmap* nedostajućih vrijednosti. Crnom bojom su označeni "normalni" podaci, a svijetlom bojom nedostajući.

Na slici vidimo postotak nedostajućih vrijednosti za pojedine varijable:

Razlog tako velikom broju nedostajućih podataka je dogovorena notacija za unos podataka. Naime, uz trening i testni skup podataka, dobili smo i datoteku s opisom svake značajke — koje je njezino točno značenje te koje vrijednosti može poprimati. Za većinu kategoričkih značajki, definirano je da ako kuća ne posjeduje dio koji ta značajka opisuje, na primjer kuća ne posjeduje bazen, za njezinu *PoolQC* značajku (koja predstavlja kvalitetu bazena) neće biti unesena nikakva vrijednost. Najveći broj nedostajućih vrijednosti dolazi upravo iz te situacije. Svaku nedostajuću vrijednost usporedili smo s njezinom "susjednom" značajkom, na



Slika 2. Postotak nedostajućih vrijednosti unutar određenih značajki.

primjer za *PoolQC* smo gledali je li onda vrijednost u *PoolArea* (veličina bazena) jednaka 0, jer bi to onda zaista značilo da vrijednost ne nedostaje, dapače, znamo da je to informacija "nema bazena". Za nekoliko stvarnih pronađenih nedostajućih vrijednosti, koristili smo većinom prosječnu vrijednost sličnih kuća (sličnih s obzirom na neke značajne atribute).

2.2 Ispravak tipova značajki

Nekoliko značajki nije bilo ispravnog tipa. To nije predstavljao veliki problem, ali smo htjeli urediti podatke što je moguće više. Tako smo cijene i varijable koje predstavljaju površine u kvadratnim mjerama pretvorili iz tipa *int* u tip *float*, godine iz tipa *float* u *int*.

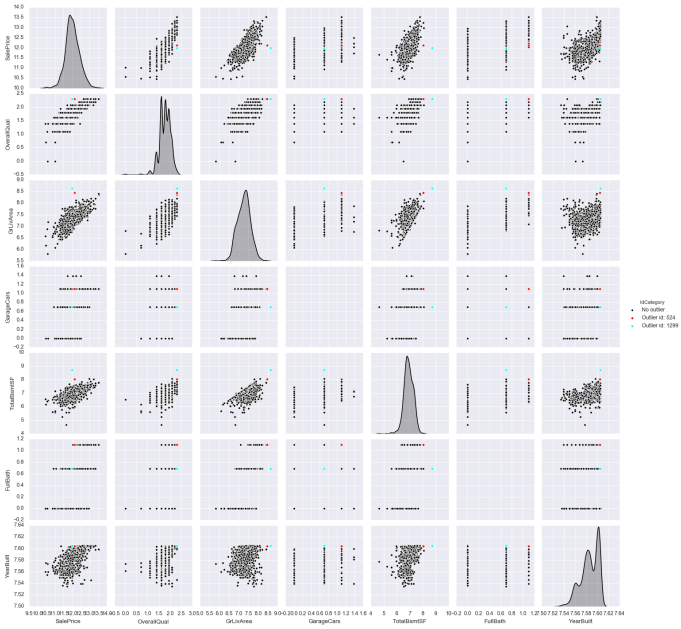
2.3 Netipične vrijednosti (*engl. outliers*)

Outliere smo detaljno proučili univarijantnom i bivarijantnom analizom kako ne bismo olako odbacili vrijedne podatke. Univarijantnu analizu smo primijenili provjeravajući pretpostavke o standardizaciji. Također smo ispitiviali *Power-Law* distribuciju s obzirom da je ona česta pojava u sličnim situacijama s cijenama, ali naš skup ipak ne zadovoljava tu distribuciju. Bivarijantnom analizom smo izvukli sve kandidate za outliere te kasnije *scatterplot* matricom usporedili opravdavaju li neke druge značajke stršeće vrijednosti outliera (slika 3 — zbog male slike je manja vidljivost, ali želimo ukazati na istaknutost outliera drugim bojama.).

Sveli smo cijeli proces na pronalazak dva outliera koji se vide i nakon tranformacije logaritmom (koja inače ublažava outliere, opisana je u jednom od kasnijih potpoglavlja). *scatterplot matrix* nakon transformacije vidi se na slici 4.

Ta dva outliera smo odlučili ukloniti kako nebi stvarali probleme modelima te pošto ih je samo mali broj od dva outliera. Iako mnogi modeli znaju raditi s outlierima, naš skup podataka je mali te ne želimo moguće kompromitirane podatke da poremete efikasnost modela koji se ionako može istrenirati samo na relativno malom skupu podataka.

Slika 3. *Scatterplot matrix* nekoliko najkoreliranijih atributa s atributom *SalePrice*. Bojama su naglašeni *outlieri*.



Slika 4. *Scatterplot matrix* nekoliko najkoreliranijih atributa s atributom *SalePrice*, nakon transformacije logaritmom. Bojama su naglašeni *outlieri*.

2.4 Grupiranje značajki i stvaranje novih

U daljnoj obradi podataka posegnule smo za grupiranjem nekih značajki i kreiranjem novih. Grupiranjem nekih značajki i stvaranjem novih postigle smo cjelokupniju informaciju o pojedinom aspektu cijene kuće što je postignuto stvaranjem suma, prosjeka, *boolean* varijabli. Naime, tim radnjama postigle smo pojednostavljenje modela na način da pojedini atribut uslijed grupiranja predstavlja generalizaciju više atributa čime postižemo kompaktniju informaciju o

podacima.
Konkretno, promotrimo li atribute, uočit ćemo *'TotalBsmtSF'* (površina podruma) te *'GrLivArea'* (površina nadzemnog dijela kuće). Prirodno se nameće uvođenje novog atributa *'TotalSF'* (ukupna površina) koji je jednak zbroju vrijednosti prethodna dva atributa - on je sada nositelj značajnije informacije o ukupnoj površini kuće (dvije informacije su zamijenute jednom). Na sličan način generaliziramo informaciju o ukupnom broju kupaonica (novi atribut *'TotalBath'*) te ukupnoj kvaliteti kuće (novi atribut *'OverallTotal'*). Također, eliminiramo atribute koje smo obuhvatili u gornjem procesu.

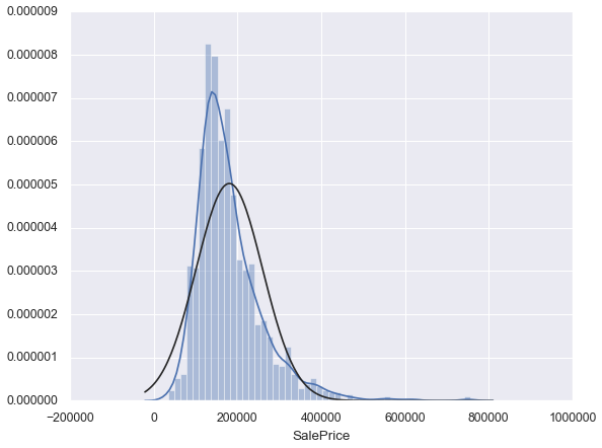
2.5 Rješavanje kategoričkih varijabli

Neki atributi u datasetu zadani su da imaju numeričke vrijednosti, no te vrijednosti ustvari označavaju kategoriju. To su atributi MSSubClass (identificira vrstu stana koji su uključeni u prodaju i poprima vrijednosti od 20 do 190 koje enkodiraju tip stana), YrSold (godine prodaja nekretnina koje su u rasponu od 2016. do 2010. godine, kako je 2008. godine nastupila kriza koja je značajno utječe na cijenu nekretnina, dobro je odvojiti svaku godinu posebno), te MoSold (mjesec u kojem je prodana nekretnina). Transformacija u kategoričke varijble je napravljena tako da su se vrijednosti pretvorile u tip string.

Kako algoritmi strojnog učenja uglavnom rade s numeričkim vrijednostima svih atributa, potrebno je sve kategoričke vrijednosti transformirati u numeričke. Za to su korištena dva tipa transformacije. Prvo su LabelEncoderom transformirane varijable u čijim vrijednostima postoji uređenost, na primjer BsmtCond, odnosno stanje podruma, koje poprima vrijednosti Excellent, Good, ..., Poor. Zatim su pomoću metode get_dummies transformirane sve ostale kategoričke varijable.

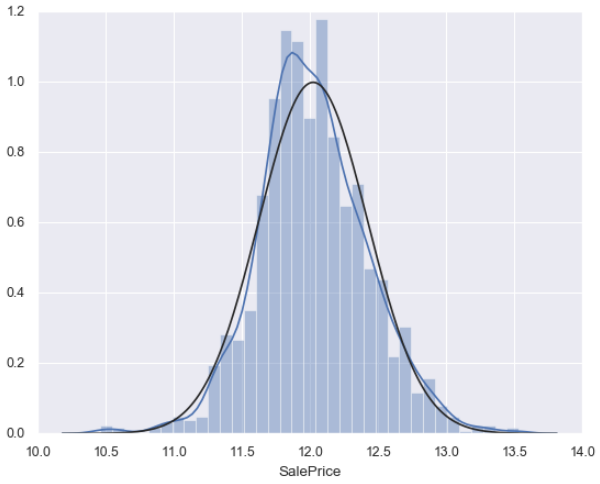
2.6 Normalizacija, asimetrija *(engl. skewness)*

Puno značajki u našem skupu podataka ne prati normalnu distribuciju, te su pozitivno asimetrične (*engl. right-skewed*). Pogledajmo na primjer našu ciljnu varijablu *SalePrice* na slici 5.



Slika 5. Histogram *SalePrice*-a.

Na sve pozitivno asimetrične takve značajke smo primijenili prirodni logaritam te tako pomjerili medijan u "središte" (slika 6).



Slika 6. Histogram *SalePrice*-a nakon transformacije logaritmom.

Transformacija logaritmom je većini značajki sredila i normaliziranost, no nije svima. Proučavanjem raznih izvora na internetu, zaključili smo da je uz logaritam ipak prigodno i primijeniti normalizaciju kako bi podaci sigurno bili normalizirani te smo to i učinili. Odabrane podatke smo dakle normalizirali kako naši modeli ne bi imali problema s velikim i različitim rasponima vrijednosti u različitim značajkama. Samoj cijeni kuća smo pristupali različito ovisno o modelu, kod nekih smo ju normalizirali, a kod nekih samo skalirali da bude u rasponu $< 0, 1]$ tako što smo cijene podijelili s najvećom cijenom kuće.

2.7 Odabir značajki *(engl. feature engineering)*

Radi pojednostavljivanja, boljeg objašnjavanja naših modela te izbjegavanja *overfitting*-a, odlučili smo obratiti pozornost na odabir značajki. Nakon prijašnjih koraka u transformaciji podataka, broj značajki je znatno porastao, od inicijalnog broja od 79 atributa, dobili smo 254 atributa.Glavni razlog je taj što se u skupu podataka nalazi velik broj kategoričkih varijabli koje su transformirane pomoću metode get_dummies. Problemu odabira najbitnijih značajki odlučili smo pristupiti tako da uzmemo u obzir tri kategorije metoda za *feature selection*.

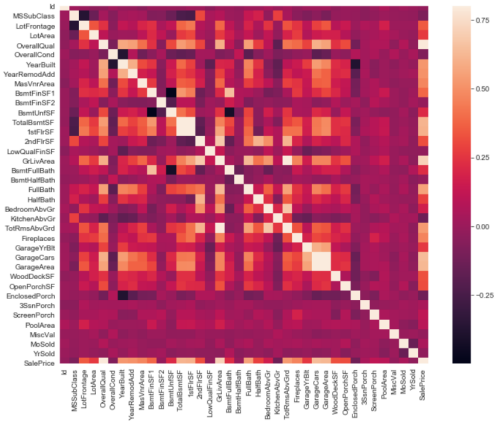
1) **Filter metode**
   U ovoj kategoriji koristili smo metodu iz *Scikit-learn*-a, *SelectKbest* i to na temelju metrike *mutual_info_regression*.
   mutual_info_regression je implementacija metrike *mutual information*. Intuitivno, ona mjeri međuovisnost između dvije varijable. Ako su te varijable međusobno neovisne tada ne postoji informacija o jednoj varijabli koja se može dobiti iz poznavanja druge varijable pa iz tog slijedi da je njihova međusobna informacija jednaka 0. Ako pak postoji deterministička veza između dvije varijable ($f(x) = y$) onda je njihova međusobna informacija jednaka

1. Dakle, $0 <$ *mutual information* < 1. Detaljna i formalna definicija je u [5].
Prednost te metrike u odnosu na F-Test je da funkcionira dobro i kad nemamo linearnu vezu između nezavisne i ciljne varijable.

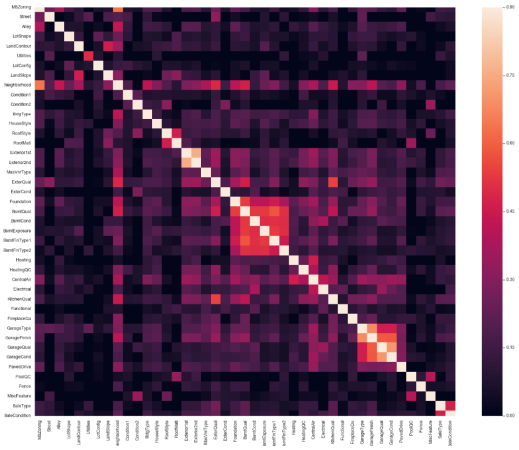
2) **Wrapper metode**
Primjer wrapper metode koji je bio korišten u ovom projektu je *Recursive feature elimination*. To je iterativna metoda koja prvo evaluira važnost značajki u skupu sa svim značajkama, a zatim u svakoj iteraciji eliminira najlošiju značajku i tako sve dok se ne dođe do željenog broja značajki.

3) **Embedded metode**
Budući da mnogi modeli imaju svoje vlastite metode za pronalazak važnih značajki, pokušali smo koristiti i takav pristup kod modela kod kojih je to moguće.

U eksploratornoj analizi utvrdili smo da postoje parovi značajki koji su međusobno vrlo korelirani (kod kategoričkih varijabli se koristio Cramerov V koeficijent). Postojala je sumnja da zbog visoke koreliranosti, značajke u parovima daju gotovu istu informaciju o ciljnoj varijabli te da bi onda po jednu značajku iz tih parova valjalo izbaciti. Promatrali smo kako utječe i taj aspekt na naše modele.



Slika 7. Heatmap korelacija numeričkih varijabli



Slika 8. Heatmap povezanosti kategoričkih varijabli

U opisu pojedinih modela je preciznije navedeno koje su metode za odabir značajki bile korisne i koje su zbog toga bile primjenjivane.

3 MODELI

3.1 Elastic-Net

Postupci koji su bili korišteni u svrhu pripreme podataka su rješavanje nedostajućih podataka i *outlier*-a, ispravak tipova značajki, grupiranje značajki i stvaranje novih, rješavanje kategoričkih varijabli te logaritmiranje asimetričnih podataka.

Cilj regularizacije linearnih regresija je izbjegavanje *overfitting*-a tj. bolja predikcija na test skupovima. Elastic-Net objedinjuje dva tipa regulariziranih linearnih regresija, Lasso i Ridge regresiju tj. u optimizacijskom problemu kombinira l_1-normu i l_2-normu.

$$\min_w \frac{1}{2n_{samples}} \|Xw - y\|_2^2 + \alpha\rho \|w\|_1 + \frac{\alpha(1-\rho)}{2} \alpha\rho \|w\|_2^2$$

U gore prikazanom optimizacijskom problemu, postoje dva bitna parametra. α je parametar koji određuje jačinu regularizacije, a ρ je udio l_1-norme u regularizaciji. Za odabir tih parametara korišten je *GridSearchCV*. Metoda pretražuje mrežu parametara i utvrđuje koja kombinacija parametara dovodi do najtočnijeg modela i pri tom koristi cross validaciju.
Dani su mogući parametri za $\alpha(0.00001, 0.0001, 0.001, 0.1)$ te mogući parametri za $\rho(0.0, .2, .4, .6, .8, 1.0)$. Algoritam je odabrao parametre: $\alpha = 0.0001$ i $\rho = 1.0$.
Budući da je udio l_1-norme stopostotan, u nastavku smo zapravo koristili **Lasso regresiju**.

U sklopu odabira bitnih značajki primjenjena je wrapper metoda rekurzivne eliminacije značajki pomoću koje je odabrano njih 110 koje su značajne za ovaj model.

3.2 XGBoost

XGBoost je skraćenica za "Extreme gradient boosting", dakle riječ je o ansamblu temeljenom na stablu odlučivanja i gradient boosting-u. Neke karakteristike XGBoost-a vidljive su na sljedećoj slici.

Slika 9. Karakteristike metode Extreme gradient boosting

U ovom projektu da bismo primijenili XGBoost, prvo smo pripremili podatke tako da smo riješili problem nedostajućih vrijednosti u train i test datasetu, ispravili tipove

značajki za oba dataseta, maknuli outliere iz train dataseta. Zatim smo kreirali neke nove atribute te pojedine grupirali u grupni atribut, riješili problem kategoričkih varijabli te logaritmirali asimetrične podatke. Također maknuli smo po jedan atribut iz para visoko koreliranih varijabli.

Kako performanse XGBoost-a ovise o hiperparametrima, bilo je vrlo važno dobro njih odabrati, jer optimalni skup parametara pomaže postizanju veće točnosti. Postoji nekoliko metoda za automatsko podešavanje parametara, a to su RandomSearch, GridSearchCV i Bayesian optimization. Kako Bayesian optimization uglavnom daje bolje i brže rezultate, odlučili smo se koristiti njega. Bayesian optimization radi tako da:

1) Izgradi surogatni probabilistički model objektne funkcije.
2) Nađe hiperparametre koji imaju najbolje performanse na surogatu.
3) Primijeni te hiperparametre na originalnoj objektnoj funkciji.
4) Ažurira surogatni model inkorporirajući nove parametre.
5) Ponavlja korake 2) - 4) do kad ne postigne maksimalni broj iteracija ili maksimalno vrijeme izvršavanja.

Parametri koje koristimo su:

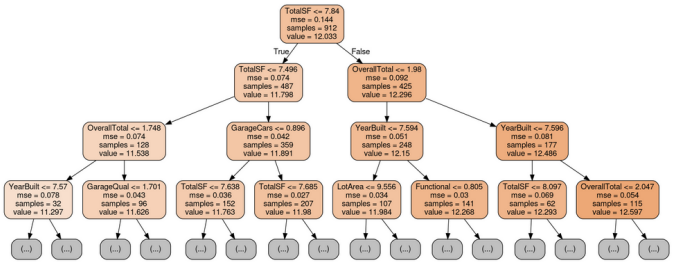
- *eta* - Smanjenje veličine koraka koji se koristi pri ažuriranju kako bi se izbjegao overfitting. Nakon svakog boosting koraka možemo direktno dobiti težine novih atributa, a *eta* smanjuje težine atributa kako bi boosting proces bio konzervativniji.
- *gamma* - Minimalno smanjenje gubitka potrebno da bi se mogla napraviti sljedeća particija na listu stabla. Veća *gamma* daje konzervativniji model.
- *max_depth* - Maksimalna dubina stabla. Povećavanje ove varijable pridonijet će kompliciranijem modelu s kojim je veća mogućnost overfittinga.
- *min_child_weight* - Minimalni zbroj težine primjerka (hessian) potreban djetetu. Ako particija stabla rezultira listom koji ima sumu težina primjerka manji od *min_child_weight*, tada u tom listu neće doći do daljnji particija.
- *subsample* - Podskup skupa za treniranje. Ako se stavi na 0.5, tada će XGBoost slučajno odabrati pola training skupa prije izgradnje stabla, te će to preventirati overfitting. Subsampling se radi jednom u svakoj boosting iteraciji.
- *colsample_bytree* - Podskup stupaca kod izgradnje svakog stabla. Radi se jednom za konstruirano stablo.
- *lambda* - L2 regularizacija težina. Povećavanjem *lambde* dobivamo konzervativniji model.
- *alpha* - L1 regularizacija težina. Povećavanjem *alphe* dobivamo konzervativniji model.
- *eval_metric* - Evaluacijska metrika za validacijski skup. Kako mi imamo regresijski problem, koristit ćemo RMSE.

3.3 Slučajne šume

Algoritam slučajnih šuma zasniva se na modificiranoj verziji *bagging* metode. Algoritam kombinira različite postojeće ideje o slučajnostima iz statistike - spomenutu *bagging* metodu te odabir slučajnog vektora atributa iz cijelog skupa atributa. Obje tehnike pokušavaju dovesti određeni šum u podatke i algoritam, što rezultira preciznijim modelom pri obradi novog skupa podataka.
Algoritam slučajnih šuma gradi veliki broj različitih stabala. Iz skupa za učenje, *bootstrap* uzorkovanjem dobivamo novi uzorak koji je veličinom jednak polaznom skupu za učenje, a potom se nad svakim uzorkom izgradi jedno modificirano stablo. Postupak uzorkovanja i izgradnje stabala ponavljamo onoliko puta koliko će biti stabala u slučajnoj šumi. Svaki takav uzorak se razlikuje, stoga nastaje šum u odnosu na originalni skup podataka. Iz tog razloga, svako izgrađeno stablo se međusobno razlikuje - rezultat je smanjena varijanca modela. Pri uzorkovanju, otprilike jedna trećina originalnih podataka se ne iskoristi - taj skup podataka se naziva *out of bag* skup (OOB), te je pomoću njega moguće provesti unakrsnu validaciju svakog izgrađenog stabla. Takva unakrsna validacija je dobra procjena greške slučajne šume.
Naglasak pri modeliranju bit će smanjenje *overfittinga* te pronalazak tehnika koje će to omogućiti.
Korištene tehnike poboljšanja predikcije:

- uočavanje međusobno koreliranih atributa i izbacivanje nekih od njih
- uočavanje važnosti atributa (*engl. feature importance*) - kako bismo izbjegle pojavu *overtfittinga*, promotrile smo atribute i njihove važnosti, a potom odabrale podskup najznačajnijih (najbitnijih) atributa. Regresor u algoritmu slučajnih šuma posjeduje direktnu informaciju o izračunatoj važnosti atributa koju je vrlo lako dobiti. Utvrđivanjem najbitnijih atributa kreiramo novi *limited features* skup nad kojim treniramo model.
- uključivanje parametra *max_feature* u regresoru algoritma slučajnih šuma - parametar predstavlja broj slučajno odabranih atributa pri izgradnji stabla. Pri svakom dijeljenju (odluci), nasumično se odabire *max_feature* atributa te među njima odabiremo najznačajniji (na temelju odabranog atributa vrši se daljnji *split*). Ovaj parametar može biti proizvoljno odabran te se koristi kao parametar za podešavanje algoritma čije će najbolje vrijednosti ovisiti o problemu. Uvođenjem ovog parametra, primjetno je poboljšanje predikcije našeg modela na testnom skupu podataka.
- postavljanje parametra *oob_score* na *True* - na taj način vršimo validaciju modela slučajnih šuma (prethodno spomenuti OOB skup - primjeri koji nisu korišteni za treniranje modela koriste se pri testiranju tog stabla). Takvim načinom validacije izbjegavamo dijeljenje originalnog trening skupa na novi trening skup i skup za validaciju.

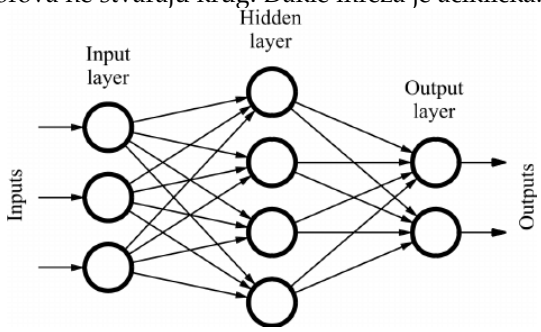
Također, pri samom modeliranju vrlo bitan parametar je *n_estimators* - označava broj ponavljanja postupka uzorkovanja i izgradnje stabala u slučajnoj šumi.



Slika 10. Jedno stablo odluke algoritma slučajnih šuma primjenjen na naš problem

3.4 Neuronska mreža

Izmodelirat ćemo *feedforward* neuronsku mrežu. *Feedforward* neuronska mreža je neuronska mreža gdje veze između čvorova ne stvaraju krug. Dakle mreža je aciklička:



Za izgradnju neuronske mreže koristit ćemo *PyTorch*, paket za strojno učenje u Pythonu.

Neuronska mreža je osjetljiva na nedostajuće podatke, outliere i velike raspone vrijednosti. Stoga je ulazne podatke potrebno pripremiti da bi model imao što manje problema. Za pripremu podataka za neuronsku mrežu koristit ćemo sve postupke iz poglavlja Podaci i transformacije podataka. Najprije ćemo riješiti sve nedostajuće vrijednosti iz trening i testnog skupa podataka, tranformirati određene tipove značajki i ukloniti nekoliko otkrivenih *outliera* kako ne bi ometali model. Provest ćemo grupiranje i kreiranje novih značajki te u takvom novom skupu podataka odabrati najbitnije značajke za treniranje i predikciju cijena. Za ovaj model, testirat ćemo dva različita odabira značajki:

1) izdvojili smo 30 najbitnijih značajki dobivenih filter metodom *SelectKBest*
2) izdvojili smo 100 najbitnijih značajki dobivenih wrapper metodom *Recursive feature elimination*

Ispostavit će se da podjednake rezultate daju oba pristupa, odlučit ćemo se za prvi pristup zbog kraćeg trajanja treninga. U odabranim podacima ćemo zatim rješavati problem asimetričnosti i denormaliziranosti podataka kao što je opisano u gornjim poglavljima.
Trening skup ćemo podijeliti na trening skup i validacijski skup podataka u omjeru 80% – 20%. Trening skup će nam služiti za treniranje modela, a validacijski skup će nam pomoći u biranju najpogodnijih hiperparametara modela. Naime, različitim kombinacijama hiperparametara i optimizacijskih tehnika u modeliranju ćemo istrenirati više različitih modela te ćemo na validacijskom skupu evaluirati svaki od njih i odrediti hiperparametre i tehnike s kojima dobivamo bolje rezultate.

Nakon što smo opisali tzv. *preprocessing* podataka koji će biti input našoj neuronskoj mreži, recimo sada nešto više i o samom modelu neuronske mreže kojeg ćemo izgraditi. Kao što je spomenuto, korišten je poznati PyTorch paket koji je baziran na tenzorima. Kreirat ćemo klasu koja realizira *feedforward* neuronsku mrežu. U njoj ćemo definirati primjenu linearnih transformacija na podacima, a kao aktivacijsku funkciju koristit ćemo ReLU. Ispostavit će se da najbolje evaluacije dobivamo s mrežom koja ima što manje slojeva (*engl. layers*), točnije s jednim slojem, te sa što više neurona u tom sloju. Koristit ćemo još i tri optimizacijske tehnike u kreiranju modela:

1) **Batch normalization** je tehnika koja može poboljšati *learning rate*. To čini minimiziranjem kovarijantnog pomaka koji je suštinski fenomen distribucije ulaznih podataka u svakom *layeru*. Dakle dinamički normaliziramo ulazne podatke prije ulaska u svaki sloj.
2) **Dropout** nasumično deaktivira neke od neurona u svakom koraku tijekom treniranja. Preciznije, dropout tehnika nulira neke elemente ulaznog tenzora s vjerojatnošću p koristeći primjerke Bernoullijeve distribucije. Ova tehnika se uspoređuje sa ansamblima s obzirom da izbacivanjem različitih neurona u svakom koraku, efektivno kombiniramo različite modele. Iako se često ne preporuča korištenje obiju prethodno opisanih tehnika u jednom modelu, u našem testiranju bolji rezultati su bili koristeći obje.
3) **Kaiming inicijalizacija težina** je pogodna jer sprječava izlazne vrijednosti aktivacijskih funkcija da eskaliraju. Tada bi gradijent postao preveliki ili premali da bi se jednostavno vratio u optimalnije stanje pa će mreži trebati duže vremena da dođe do stanja konvergencije, ako uopće dođe do njega. Ovom tehnikom se sprječava i pretjerano prilagođavanje neurona, uspješna je u regularizaciji.

Kao optimizacijsku funkciju u treniranju koristit ćemo Adam optimizacijski algoritam, a za računanje gubitka (*engl. loss*) koristimo kasnije opisanu RMSE metriku.

4 TESTIRANJE I REZULTATI

Trening skup i test skup su sastavni dio Kaggle natjecanja - mogu se pronaći na poveznici: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data. Kako vrijednosti ciljne varijable imamo samo u trening skupu, podijelili smo taj skup na dva dijela, jedan za treniranje, a drugi za validaciju. Veličina validacijskog skupa je 25% trening skupa. Time smo prije primjene modela na danom testnom skupu i učitavanja rezultata na Kaggle, mogle dobiti sliku o tome koliko je model dobar.

Pri evaluaciji naših modela te međusobnom uspoređivanju njihovih preciznosti korištena je RMSE mjera čije je objašnjenje navedeno u nastavku.

4.1 RMSE (Root Mean Square Error)

RMSE je često korištena mjera razlike između vrijednosti predviđenih modelom i stvarnih vrijednosti. To je mjera

točnosti za usporedbu pogrešaka predviđanja različitih modela za određeni skup podataka, a ne između skupova podataka, budući da ovisi o skaliranju.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}}, \quad (1)$$

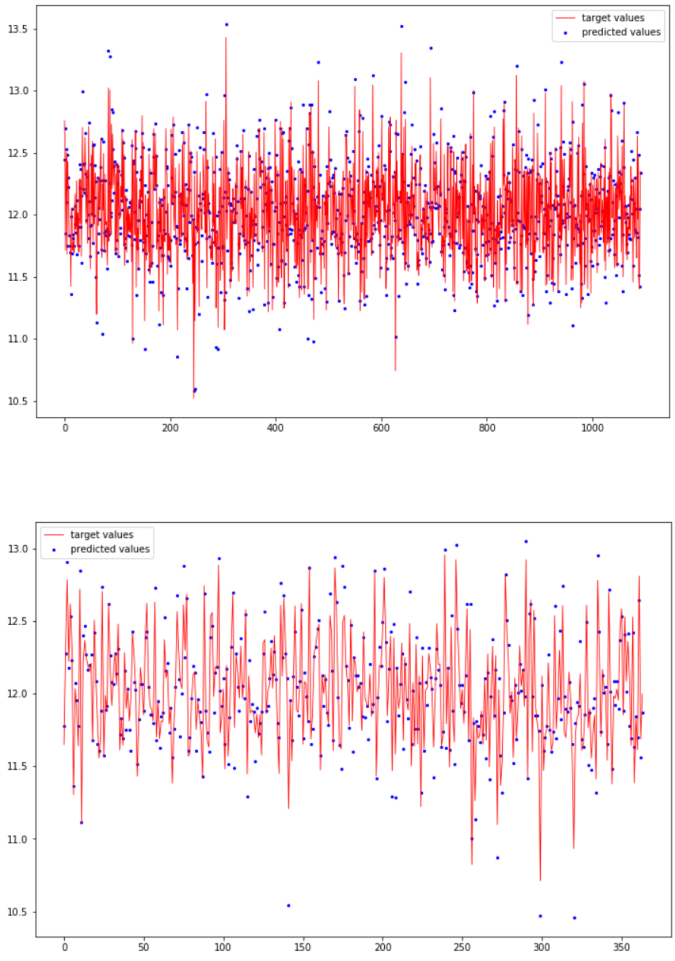
gdje je N broj primjera skupa, y_i vrijednost stvarne vrijednosti ciljne varijable i-tog primjera te $\hat{y}_i$ modelom predviđenja vrijednost ciljne varijable i-tog primjera.

4.2 Usporedba rezultata modela

U nastavku, za svaki pojedini model navodimo ključne parametre i njihove vrijednosti za najbolji mogući rezultat, kao i pripadni RMSE. Također, pokazujemo koji atributi su bili najvažniji u izgradnji modela.

4.2.1 *Elastic-Net*

- **parametri**:
  - $normalize = True$,
  - $alpha = 0.0001$,
  - $l1\_ratio = 1.0$.

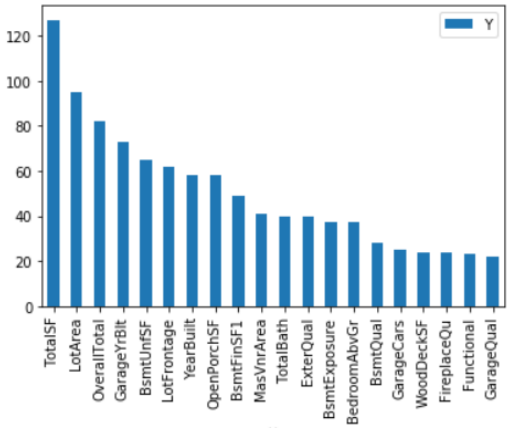


Slika 11. Grafička usporedba originalnih cijena i cijena predviđenih Elastic-Net modelom u trening i validacijskom skupu

4.2.2 *XGBoost*

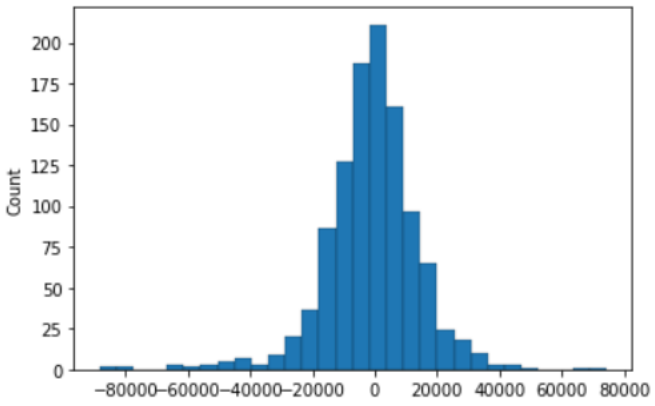
- **parametri**:
  - $lambda = 2.0$,
  - $alpha = 1.0$,
  - $colsample\_bytree = 0.1$,
  - $eta = 0.1$,
  - $gamma = 0.0$,
  - $max\_depth = 4$,
  - $min\_child\_weight = 2$,
  - $subsample = 1.0$.

XGBoost nam daje popis atributa koji su bili najznačajniji kod modeliranja, a koje prikazujemo na sljedećem grafu. Vidimo da su među njima TotalSF (koji uključuje GrLivArea i TotalBsmtSF), LotArea, OverallTotal (u koji je uključen OverallQual) za koje smo i ranije tokom ekploratorne analize uočili da su bitne.

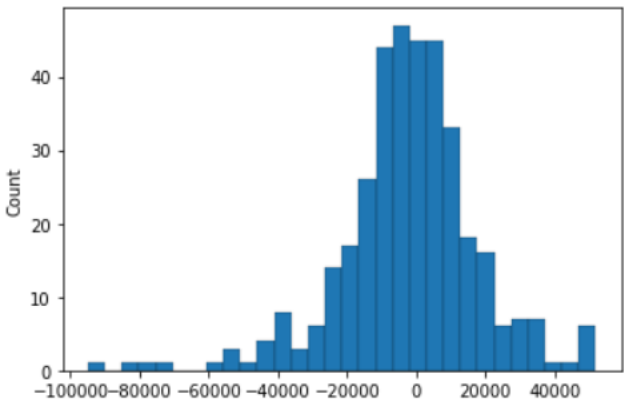


Slika 12. Najvažniji atributi kod modeliranja XGBoost-a

Zatim, prikazujemo graf reziduala originalnih vrijednosti ciljne varijable *SalePrice* i predviđenih vrijednosti te varijable prvo na trening skupu, a zatim na validacijskom skupu.

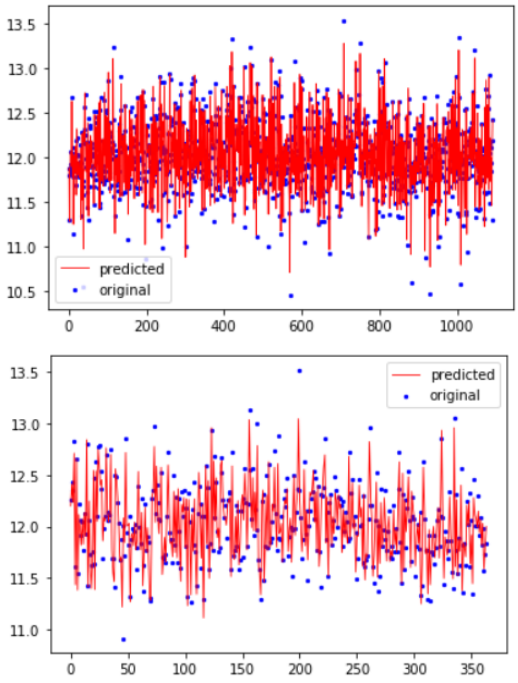


Slika 13. Reziduali originalnih i predviđenih vrijednosti ciljne varijable *SalePrice* na trening skupu podataka



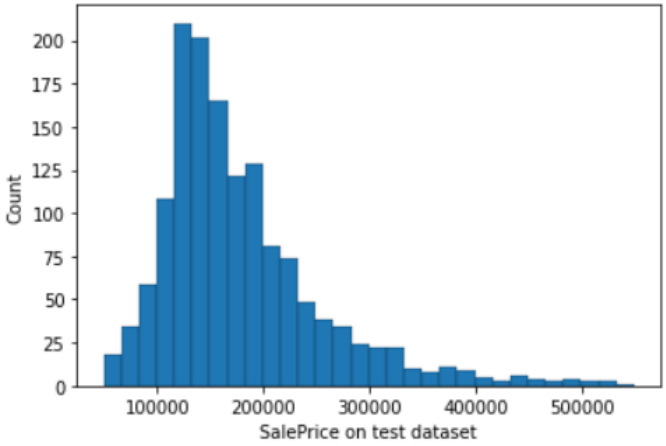
Slika 14. Reziduali originalnih i predviđenih vrijednosti ciljne varijable *SalePrice* na validacijskom skupu podataka

Vidimo da su reziduali normalno distribuirani oko 0, dakle veći broj predviđenih cijena je približno jednak stvarnim cijenama, dok je za mali broj kuća (manje od desetak) razlika između predviđenih i stvarnih cijena oko 100 000 USD.
Nadalje, poklapanje predviđenih i originalnih podataka možemo vidjeti i na sljedećim grafovima gdje koristimo logaritmirane vrijednosti varijable *SalePrice*.



Slika 15. Grafička usporedba originalnih cijena i cijena predviđenih XGBoost modelom u trening i validacijskom skupu

Konačno pogledajmo koja je distribucija predviđenih cijena na danom test datasetu, dakle onom na kojem nemamo unaprijed dane vrijednosti *SalePrice*. Vidimo da je cijena uglavnom između 100 000 i 200 000 USD.

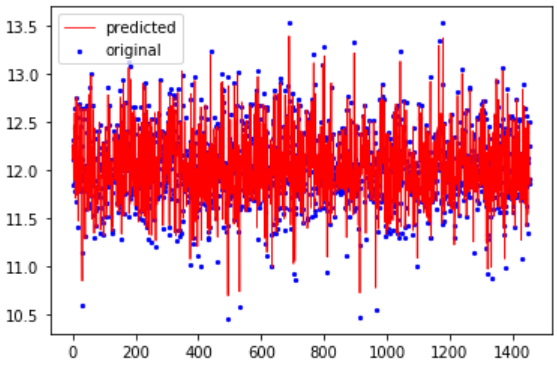


Slika 16. Predviđene cijene kuća iz originalnog test dataseta

Nakon nalaženja najvažnijih atributa, XGBoost model je treniran i na datasetu koji sadrži samo 100 najvažnijih atributa te jednake parametre kao i ranije, no rezultati su bili otprilike jednaki, stoga možemo zaključiti da su parametri dobro određeni te oni sami uspiju dati dobar model s kojim ne dolazi do overfittinga čak i kod većeg broja atributa.

4.2.3 *Slučajne šume*

- **parametri**:
  - $n\_estimators = 400$,
  - $max\_feature = 0.5$,
  - $oob\_score = True$,
  - odabrani broj atributa nad kojima treniramo model: 25 (približno trećina ukupnog broja atributa) najznačajnijih (najveći *feature importance*).



Slika 17. Grafička usporedba originalnih cijena i cijena predviđenih algoritmom slučajnih šuma u trening skupu

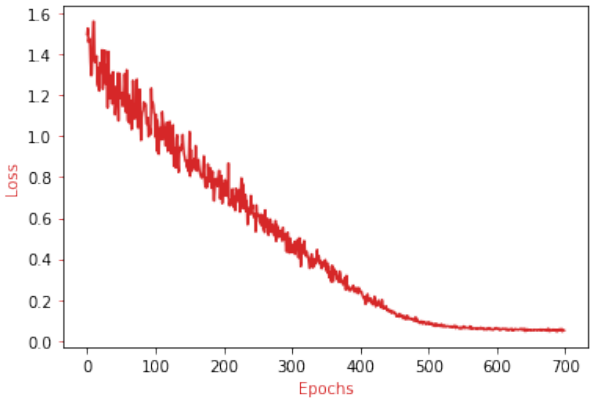
Na trening skupu, RF model postiže *score* od 98.5982%.

4.2.4 *Neuronska mreža*

- **parametri i hiperparametri**:
  - $layers = [30, 500, 1]$,
  - $batch\_size = 500$,

- $learning\_rate = 0.0001$,
- $dropout\_probability = 0.8$,
- $epochs = 700$.

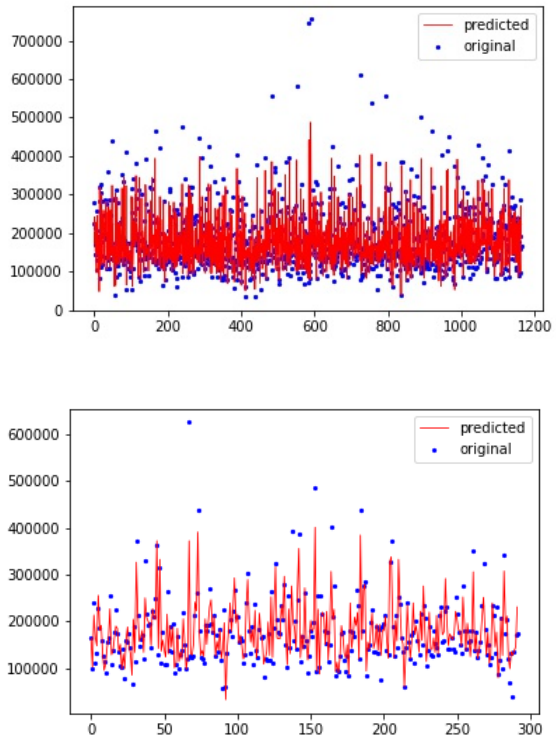
Grafički prikaz konvergencije *loss* funkcije na trening skupu je na slici 18.



Slika 18. Konvergencija *loss*-a neuronske mreže

Skakanje u vrijednostima je rezultat *dropout*-a. Usprkos skakanju, trening loss konvergira.

Prikazujemo i grafičku usporedbu originalnih cijena i cijena predviđenih neuronskom mrežom u trening i validacijskom skupu (19).



Slika 19. Grafička usporedba originalnih cijena i cijena predviđenih neuronskom mrežom u trening i validacijskom skupu

4.3 Korišteni softverski paketi

Pri implementaciji navedenih modela koristile smo sljedeće Python module: numpy, pandas, sklearn, matplotlib, scipy, seaborn.

Korištene biblioteke koje su izvan Python modula: xgboost (moguće je instalirati u anaconda prompt), bayes_opt (instaliran pomoću pip-a), pydotplus (korišten za vizualizaciju stabla u algoritmu slučajnih šuma), PyTorch za neuronsku mrežu (instaliran naredbom `conda`).

4.4 Pozicioniranje na tablici Kaggle rezultata

Učitani podaci (predikcije cijena testnog skupa podataka) se evaluiraju metrikom Root-Mean-Squared-Error (RMSE) između logaritma predviđene cijene i logaritma točne cijene. Uzimanje logaritma znači da će greška u predviđanju skupe kuće i greška u predviđanju jeftine kuće imati jednak utjecaj u mjerenju evaluacije.
Najbolje evaluacije za svaki model su:

- **Elastic-Net**: 0.12509
- **XGBoost**: 0.12224
- **Random Forests**: 0.13598
- **neuronska mreža**: 0.15537

XGBoost modelom postigli smo najbolji rezultat, tj. najmanji RMSE (0.12224) te se pozicionirale u top 21% najboljih rezultata u sklopu natjecanja.

5 Nastavak istraživanja i nove metode

Ideje za nastavak i poboljšanja:

- **Korištenje ansambla modela i kombiniranja predikcija u ansamblu**
  Osim što već koristimo *bagging* i *boosting* u sklopu napravljenih modela, u nastavku bi bilo poželjno isprobati metode kombiniranja predikcija u ansamblu kao što su *Stacking* i *Voting*.
- **Korištenje CatBoost algoritma**
  Budući da u skupu podataka ima puno kategoričkih varijabli, CatBoost bi olakšao posao jer nije potrebno kategoričke varijable pretvarati u numeričke već algoritam to radi automatski. Kao i u XGBoost-u, možemo doznati koje su značajke najbitnije pa bi algoritam bio koristan i zbog odabira značajki.
- **Elegantinije korištenje RFE**
  Uz sam *Recursive feature elimination*, mogla bi se koristiti unakrsna validacija koja bi dala informaciju s koliko značajki model postiže najbolji rezultat.

6 Zaključak

Primjenom i implementiranjem različitih metoda strojnog učenja pri predviđanju cijena kuća uočile smo njihove prednosti, mane, proučile teorijske aspekte metoda te se poigrale s podešavanjem parametara. Sudjelovanje na Kaggle natjecanju potaknulo nas je na dublje razumijevanje implementiranog, a ulazak u top 21% najboljih rezultata je više nego dobar poticaj za nastavak rada.
Implementacija navedenih metoda i rezultati mogu se pronaći ovdje: https://github.com/ivona13/Strojno-ucenje-2020.

LITERATURA

[1] https://xgboost.readthedocs.io/en/latest/parameter.html
[2] https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/
[3] https://medium.com/analytics-vidhya/hyperparameters-optimization-for-lightgbm-catboost-and-xgboost-regressors-using-bayesian-6e7c495947a9/
[4] https://towardsdatascience.com/understanding-random-forest-58381e0602d2
[5] C.E. Shannon *A Mathematical Theory of Communication*, Bell System Technical Journal, 1948.
[6] https://towardsdatascience.com/the-5-feature-selection-algorithms-every-data-scientist-need-to-know-3a6b566efd2
[7] https://towardsdatascience.com/why-how-and-when-to-apply-feature-selection-e9c69adfabf2
[8] https://www.coursera.org/learn/deep-neural-networks-with-pytorch
[9] https://towardsdatascience.com/batch-normalization-and-dropout-in-neural-networks-explained-with-pytorch-47d7a8459bcd
[10] https://morioh.com/p/32c506939ad5