



UNIVERZITET U ZENICI
Politehnički fakultet
Softversko inženjerstvo



Rudarenje podataka

Predikcija cijena nekretnina

Dokumentacija projekta

Studenti:
Belma Delilović, II-113
Ajla Brdarević, II-120

Profesor:
doc. dr. Adnan Dželihodžić
Asistenti:
Faris Hambo, Ahmed Mujić

Zenica, juni 2025.

SADRŽAJ

1. UVOD.....	4
2. SKUP PODATAKA.....	5
2.1. Analiza podataka.....	5
2.1.1. Statistika o podacima.....	6
2.1.2. Provjera nedostajućih vrijednosti i outlier-a.....	6
2.1.3. Analiza numeričkih i kategorijskih varijabli.....	8
2.1.4. Analiza SalePrice i korelacija sa SalePrice.....	8
2.1.5. Unique vrijednosti za kategorijske kolone.....	10
2.1.6. Dodatna analiza ključnih feature-a.....	11
2.1.7. Provjera duplikata i low variance.....	15
2.1.8. Vizualna analiza odnosa varijabli.....	15
2.1.9. Kolone sa NaN vrijednostima.....	16
2.2. Čišćenje podataka.....	17
2.2.1. Uklanjanje ID kolone, kolona sa istim vrijednostima i outlier-a.....	17
2.2.2. Spajanje train i test podataka u jedan skup podataka.....	18
2.2.3. Ispravljanje grešaka u podacima.....	18
2.2.4. Popunjavanje LotFrontage po Neighborhood.....	18
2.2.5. Pretvaranje numeričkih varijabli u kategorijske.....	19
2.2.6. Z-score metoda za uklanjanje outlier-a.....	19
2.2.7. Popunjavanje nedostajućih numeričkih vrijednosti.....	20
2.2.8. Popunjavanje kategorijskih kolona specifičnim vrijednostima.....	21
2.2.9. Transformacija “skewed” kolona sa Box-Cox.....	22
2.2.10. Enkodiranje.....	22
2.3. Provjera čišćenja.....	23
2.3.1. Provjera NaN vrijednosti.....	23
2.3.2. Provjera tipova podataka.....	24
2.3.3. Provjera distribucije ciljne varijable (y).....	24
2.3.4. Provjera korelacija.....	25
2.3.5. Spremanje pripremljenih podataka.....	26
3. ALGORITMI ZA KLASIFIKACIJU.....	27
3.1. Random forest.....	27
3.2. XGboost.....	28
3.3. Ansambl model.....	29
4. IMPLEMENTACIJA MODELA.....	31
4.1. Uvod u implementaciju: biblioteke i učitavanje podataka.....	31
4.1.1. Biblioteke i priprema okruženja.....	31
4.1.2. Učitavanje podataka.....	31
4.1.3. Podjela podataka na trening i validaciju.....	32
4.2. Tehnike i metode za poboljšanje modela.....	32

4.2.1. Definisanje evaluacijskih metrika.....	32
4.2.2. Podešavanje hiperparametara s GridSearchCV.....	33
4.2.3. Korišćenje Pipeline objekta.....	33
4.3. Implementacija i analiza modela.....	33
4.3.1. Random Forest.....	33
4.3.2. Značajnost obilježja u Random Forest.....	34
4.4. XGBoost model.....	34
4.4.1. Značajnost obilježja u XGBoostu.....	35
4.5. Evaluacija modela na validacionom skupu.....	35
4.6. Predikcija na test skupu i spremanje rezultata.....	35
5. ZAKLJUČAK.....	37
6. REFERENCE.....	38

1. UVOD

Predviđanje cijena nekretnina jedan je od najvažnijih zadataka u domenu procjene imovine, urbanističkog planiranja i investicionih odluka. Razvojem mašinskog učenja, modeli poput Random Forest-a i XGBoost-a postali su ključni alati zbog svoje sposobnosti da hvataju nelinearne odnose između velikog broja faktora – kao što su karakteristike kuće, njena lokacija i stanje tržišta.

Industrija nekretnina predstavlja važan stub ekonomije i ima značajan utjecaj na društveni i ekonomski razvoj. Predviđanje cijena kuća predstavlja ključan izazov u okviru tržišta nekretnina, s velikim implikacijama na planiranje, investicije i donošenje strateških odluka. Kupovina kuće uključuje velika finansijska ulaganja, što dodatno pokreće potrošnju u oblastima kao što su građevinarstvo, uređenje prostora, namještaj i slično. Time se direktno i indirektno utiče na razvoj privrednih sektora, zapošljavanje i rast BDP-a (bruto domaćeg proizvoda).

Dosadašnja istraživanja u ovom domenu obuhvatila su različite pristupe. A. Kaushal i A. Shankar [1] opisali su proces predikcije cijena koristeći višestruku linearnu regresiju, analizirajući utjecaj različitih faktora na cijenu i postupno razvijajući model. S. Abdulla, S. Jeberson [2], te ostali [3] sprovedli su komparativnu analizu više algoritama mašinskog učenja kao što su linearna regresija, SVM i Random Forest, fokusirajući se na evaluaciju performansi i odabir relevantnih osobina.

U ovom radu fokusiramo se na implementaciju naprednih regresionih modela, konkretno **Random Forest Regressor** i **XGBoost Regressor**, s ciljem tačne procjene tržišne vrijednosti kuća. Pristup uključuje sve korake profesionalnog ML procesa: od učitavanja i skaliranja unaprijed obrađenih podataka, preko podjele na trening i validacione skupove, do evaluacije modela i poređenja njihovih performansi korištenjem ključnih metrika kao što su **RMSE**, **MAE**, **RMSLE** i **R²**. Modeli su dodatno optimizovani pomoću **GridSearchCV**, a njihova interpretabilnost je poboljšana vizualizacijom značajnosti obilježja, kako bismo razumjeli koji faktori najviše utiču na cijenu kuće. Na kraju, kreiramo i **jednostavni ansambl model** koji kombinuje predikcije oba algoritma, u cilju postizanja još boljih performansi.

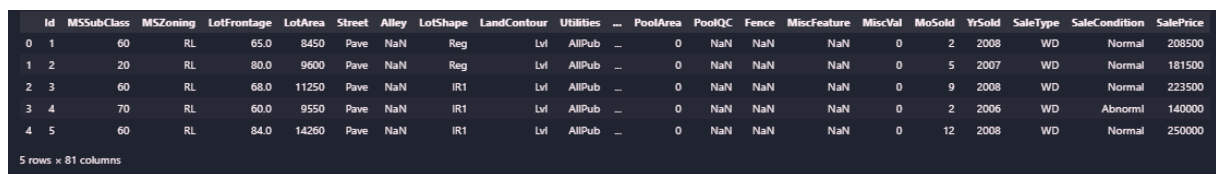
Ovo rješenje ne samo da pruža konkretne vrijednosti za poređenje modela, već demonstrira i važnost izbora algoritma, pravilnog preprocesiranja i analize rezultata u realističnim problemima regresije.

2. SKUP PODATAKA

Za potrebe ovog istraživanja korišten je javno dostupni skup podataka za predikciju cijena nekretnina - House Prices [4]. Na Kaggle-u [4] je u `data_description.txt` dostupan detaljan opis podataka, i nazivi kolona sa opisima. Ovaj skup podataka sadrži informacije o rezidencijalnim nekretninama i koristi se za predviđanje prodajne cijene kuće (`SalePrice`). Dataset se sastoji od 79 feature-a koji opisuju različite aspekte kuće, zemljišta i lokacije, te jedne ciljne varijable (`SalePrice`). Prije analize, čišćenja i vizualizacije podataka, skup podataka je učitán korištenjem biblioteke *pandas*.

2.1. Analiza podataka

Nakon učitavanja podataka (`train.csv` i `test.csv`), pomoću *.shap* dobijen je broj redova i kolona, a *.head()* funkcija prikazuje prvih nekoliko redova da bismo dobili uvid u strukturu i sadržaj podataka.



	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2008	WD	Normal	208500
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	5	2007	WD	Normal	181500
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	9	2008	WD	Normal	223500
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2006	WD	Abnorml	140000
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	12	2008	WD	Normal	250000

5 rows x 21 columns

Slika 1. Prvih pet redova iz skupa podataka

Koristeći metodu *info()*, dobijamo osnovne informacije o DataFrame-u:

- Ukupan broj redova i kolona,
- Imena kolona i njihove tipove podataka (`'object'`, `'int64'`, `'float64'`, itd.),
- Broj nenedostajućih vrijednosti po kolonama.

Skup podataka se sastoji od 1460 redova (primjera) i 81 kolone, uključujući:

- Ciljnu varijablu: `SalePrice` (prodajna cijena kuće)
- 80 ulaznih feature-a, među kojima su:
- 35 numeričkih tipa `int64` (npr. `GrLivArea`, `GarageCars`)
- 3 numerička tipa `float64` (npr. `LotFrontage`, `MasVnrArea`)
- 43 kategorijalna (`object`) (npr. `Neighborhood`, `Exterior1st`)

Neke kolone imaju nedostajuće vrijednosti, što zahtijeva obradu prije modeliranja.:

- `Alley`: samo 91 vrijednost (većinom nedostaje)

- PoolQC, MiscFeature, Fence: također s velikim brojem nedostajućih vrijednosti
- MasVnrType, GarageType, BsmtQual itd. imaju parcijalne nedostatke

2.1.1. Statistika o podacima

Pomoću deskriptivne statistike - *df.describe()* su dobijene statistike za sve numeričke kolone:

- count: broj nenedostajućih vrijednosti,
- mean: aritmetičku sredinu,
- std: standardnu devijaciju,
- min: minimalnu vrijednost,
- 25%, 50%, 75%: kvartile,
- max: maksimalnu vrijednost.

Analiza deskriptivne statistike pokazuje da su podaci uglavnom kompletni, osim za nekoliko kolona s nedostajućim vrijednostima (npr. širina zemljišta). Površine parcela i kuća variraju značajno, što ukazuje na prisustvo nekih veoma velikih nekretnina. Većina objekata je građena u drugoj polovini 20. stoljeća, a renovacije su rađene u posljednjim decenijama. Kvalitet i stanje kuća su prosječni, ali s velikim rasponom ocjena. Vanjski sadržaji kao što su terase i bazeni su često odsutni, ali tamo gdje postoje mogu imati velike dimenzije. Cijene nekretnina su široko rasprostranjene, sa prosjekom oko 180 hiljada, ali i nekim skupljim objektima koji značajno utiču na ukupnu statistiku.

2.1.2. Provjera nedostajućih vrijednosti i outlier-a

U skupu podataka *train_df* pregledane su nedostajuće vrijednosti korištenjem metode koja broji prazne (NaN) vrijednosti po kolonama, a zatim su rezultati sortirani po količini nedostataka od najvećeg ka najmanjem. Fokus je bio na kolonama koje sadrže barem jednu nedostajuću vrijednost kako bi se dobila jasnija slika o obimu praznina u podacima. Najviše nedostajućih podataka je zabilježeno u kolonama koje opisuju dodatne karakteristike objekata, poput kvalitete bazena, prisustva raznih dodatnih objekata, pristupa sa strane, ograde i vrste građevinskog materijala za fasadu. Ove kolone imaju nekoliko stotina ili čak preko hiljadu nedostajućih vrijednosti, što ukazuje da ti elementi često nisu prisutni ili nisu evidentirani u velikom dijelu uzorka. Srednja količina nedostataka je primijećena u kolonama vezanim za karakteristike garaže i podruma, dok su neke kolone gotovo potpune, sa samo jednom ili vrlo malim brojem nedostajućih vrijednosti.

```
missing = train_df.isnull().sum().sort_values(ascending=False)
missing[missing > 0]
```

Rezultati provjere nedostajućih vrijednosti:

Kolona	Broj nedostajućih vrijednosti
PoolQC	1453
MiscFeature	1406
Alley	1369
Fence	1179
MasVnrType	872
FireplaceQu	690
LotFrontage	259
GarageQual	81
GarageFinish	81
GarageType	81
GarageYrBlt	81
GarageCond	81
BsmtFinType2	38
BsmtExposure	38
BsmtCond	37
BsmtQual	37
BsmtFinType1	37
MasVnrArea	8
Electrical	1

Grafikon prikazuje odnos između stambene površine iznad zemlje (GrLivArea) i prodajne cijene kuće (SalePrice). Svaka tačka na grafikonu predstavlja pojedinačnu kuću iz skupa podataka. Jasno je vidljiv pozitivan trend, kako se povećava površina kuće, tako raste i njena cijena. Ipak, mogu se uočiti i neke odstupajuće tačke, odnosno outlieri. To su kuće koje imaju

znatno veću površinu, ali ne prate očekivanu visoku cijenu. Ove anomalije mogu biti posljedica grešaka u podacima, specifičnih pojedinačnih slučajeva koji se razlikuju od općeg tržišnog trenda, ili ukazuju na potrebu za detaljnijom analizom.

```
plt.scatter(train_df["GrLivArea"], train_df["SalePrice"])
plt.title("GrLivArea vs SalePrice")
```



Slika 2. Prikaz outlier-a

2.1.3. Analiza numeričkih i kategorijskih varijabli

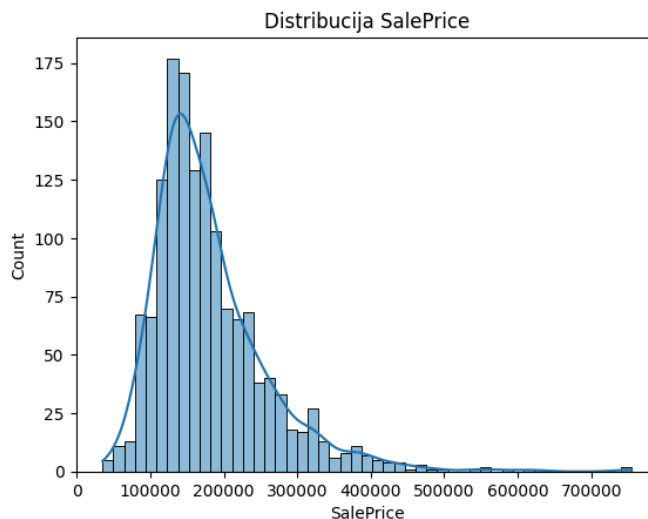
Podaci su podijeljeni u dvije osnovne kategorije: numeričke i kategorijske kolone. Numeričke kolone sadrže brojne vrijednosti (tipa int64 i float64), dok kategorijske predstavljaju tekstualne ili nominalne vrijednosti (tipa object).

U skupu podataka identificirano je ukupno 38 numeričkih kolona i 43 kategorijske kolone. Za kategorijske kolone izvršena je analiza broja jedinstvenih vrijednosti unutar svake od njih. Rezultat pokazuje da neke kategorijske kolone imaju samo dvije ili tri različite vrijednosti, što ih svrstava u binarne ili nisko-varijabilne kategorije, dok druge imaju do 25 različitih vrijednosti, ukazujući na veći stepen raznolikosti u podacima.

2.1.4. Analiza SalePrice i korelacija sa SalePrice

U ovom koraku fokus je stavljen na analizu raspodjele ciljne varijable SalePrice, koja predstavlja prodajnu cijenu kuće. Za vizualizaciju je korišten histogram sa uključenom KDE (Kernel Density Estimation) krivuljom. Histogram prikazuje koliko često se javljaju različiti rasponi vrijednosti cijena, dok KDE krivulja daje glatku procjenu distribucije podataka, što olakšava uočavanje općeg oblika raspodjele. Rezultati ove analize pokazuju da distribucija

SalePrice nije simetrična, već je desno asimetrična (right-skewed). To znači da je većina kuća koncentrisana oko nižih cijena, dok postoji manji broj kuća sa znatno višim cijenama, što je uobičajeno u tržišnim podacima nekretnina.



Slika 3. Distribucija za SalePrice

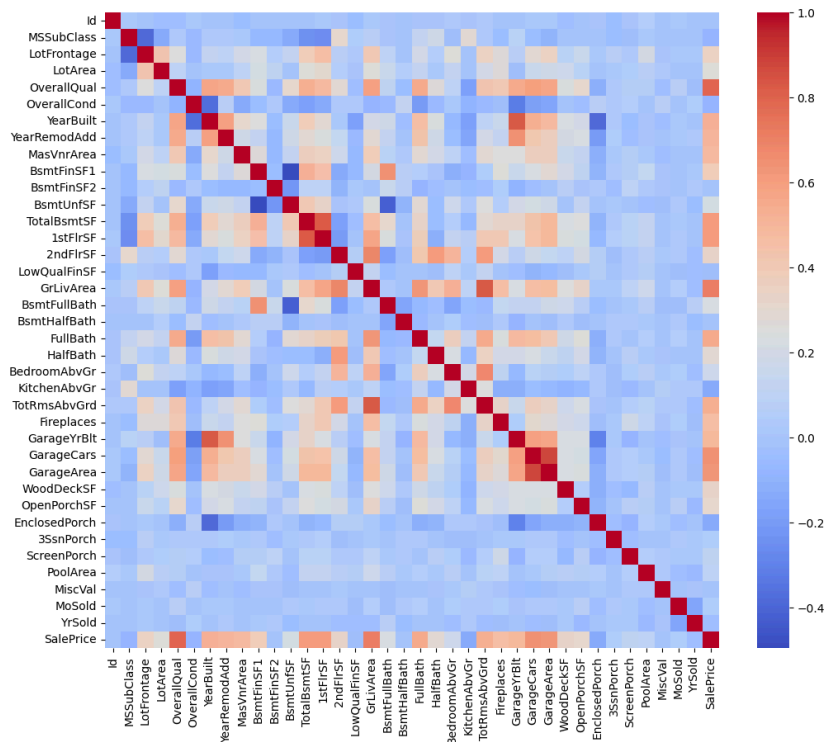
Ovdje također vršimo analizu korelacije između numeričkih varijabli i ciljne varijable SalePrice, koja predstavlja prodajnu cijenu kuće. Korelacija nam govori o jačini i smjeru linearne veze između dvije varijable. Vrijednosti korelacije kreću se od -1 do 1, gdje:

- 1 označava savršenu pozitivnu korelaciju, što znači da se obje varijable povećavaju zajedno.
- -1 označava savršenu negativnu korelaciju, gdje jedna varijabla raste, a druga opada.
- 0 znači da nema linearne veze između varijabli.

Izračunata korelacija pokazuje top 15 numeričkih varijabli koje imaju najjaču pozitivnu vezu sa SalePrice. Te varijable su potencijalno najvažnije za predviđanje cijene kuće, a među njima su:

- OverallQual (kvalitet izgradnje i materijala),
- GrLivArea (stambena površina iznad zemlje),
- GarageCars i GarageArea (kapacitet i površina garaže),
- TotalBsmtSF (ukupna površina podruma),
- 1stFlrSF (površina prvog sprata),
- te druge koje odražavaju karakteristike kuće i starost.

Pored numeričkog ispisa korelacija, napravljena je i vizualizacija u obliku heatmap-e koja prikazuje međusobne korelacije svih numeričkih varijabli. Korištena je paleta boja coolwarm koja jasno ističe pozitivne i negativne veze.



Slika 5. Heatmap-a koja prikazuje međusobne korelacije svih numeričkih varijabli

2.1.5. Unique vrijednosti za kategorijske kolone

U ovom koraku analizirali smo broj jedinstvenih vrijednosti (kardinalitet) za svaku kategorijsku varijablu u skupu podataka koristeći metodu `.nunique()`. Ova analiza pomaže u određivanju načina obrade kategorijskih podataka pri daljoj pripremi za modeliranje.

- Varijable sa malim brojem jedinstvenih vrijednosti (npr. 2 ili 3 kategorije) su jednostavne za kodiranje, te se mogu lako pretvoriti u one-hot encoding ili label encoding.
- Varijable sa velikim brojem jedinstvenih vrijednosti (npr. 30 ili više kategorija) zahtijevaju dodatnu pažnju. Za njih se često primjenjuju tehnike poput grupisanja rijetkih kategorija ili korištenja target encodinga kako bi se smanjila dimenzionalnost i poboljšala efikasnost modela.

```
train_df[categorical_cols].nunique().sort_values()
```

Ispis daje rezultat:

Varijabla	Broj jedinstvenih vrijednosti
Street	2
Alley	2
Utilities	2
CentralAir	2
MasVnrType	3
LandSlope	3
PavedDrive	3
GarageFinish	3
PoolQC	3
ExterQual	4
BsmtQual	4
KitchenQual	4
Fence	4
Electrical	5
Heating	6
HouseStyle	8
SaleType	9
Exterior1st	15
Neighborhood	25

2.1.6. Dodatna analiza ključnih feature-a

- Opis ciljne varijable i važnijih numeričkih karakteristika

Koristeći metodu `.describe()`, analizirali smo osnovne statističke pokazatelje za ciljnu varijablu `SalePrice` (prodajna cijena kuće), kao i za nekoliko ključnih numeričkih atributa:

- `SalePrice`: prikazuje osnovne informacije o cijeni kuće, uključujući minimalnu, maksimalnu, prosječnu vrijednost i kvartile, što nam daje uvid u raspon i centralnu tendenciju podataka.

- OverallQual: ocjena ukupnog kvaliteta kuće, što direktno utiče na njenu vrijednost.
- GrLivArea: površina stambenog prostora iznad tla, važan faktor za cijenu.
- GarageCars: broj automobila koje garaža može smjestiti, također relevantan za tržišnu vrijednost.
- TotalBsmtSF: ukupna površina podruma, što može dodatno povećati vrijednost nekretnine.

- Asimetrija (skewness) numeričkih varijabli

Izračunali smo asimetriju za sve numeričke varijable kako bismo utvrdili koje varijable nisu normalno raspoređene. Visoka asimetrija, bilo pozitivna ili negativna, može značiti da je potrebna transformacija podataka (npr. log-transformacija) radi bolje prilagodbe modela. Kod varijable SalePrice i povezanih atributa primećena je značajna pozitivna asimetrija, što može utjecati na efikasnost modela ako se ne adresira. Deset numeričkih varijabli sa najizraženijom asimetrijom su, između ostalog, MiscVal, PoolArea, LotArea i 3SsnPorch. Ove varijable imaju vrlo izražene ekstreme koje bi bilo korisno dodatno istražiti ili transformisati prije modeliranja.

Naziv varijable Vrijednost asimetrije (skewness)

MiscVal	24.476794
PoolArea	14.828374
LotArea	12.207688
3SsnPorch	10.304342
LowQualFinSF	9.011341
KitchenAbvGr	4.488397
BsmtFinSF2	4.255261
ScreenPorch	4.122214
BsmtHalfBath	4.103403
EnclosedPorch	3.089872

- Analiza najjače koreliranih varijabli sa SalePrice

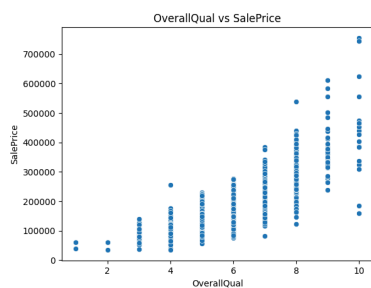
Vizualizacija odnosa između ključnih atributa i prodajne cijene

Izabrali smo pet numeričkih atributa sa najjačom pozitivnom korelacijom sa SalePrice, osim same ciljane varijable. To su:

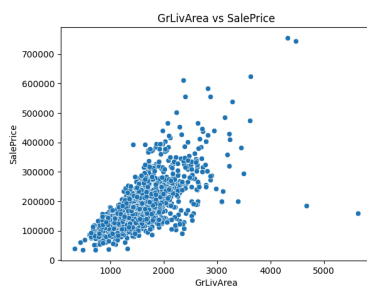
- OverallQual (kvalitet kuće),
- GrLivArea (površina stambenog prostora),
- GarageCars (broj mjesta u garaži),
- GarageArea (površina garaže),
- TotalBsmtSF (površina podruma).

Za svaki od ovih atributa napravili smo scatter plot kako bismo vizualno provjerili:

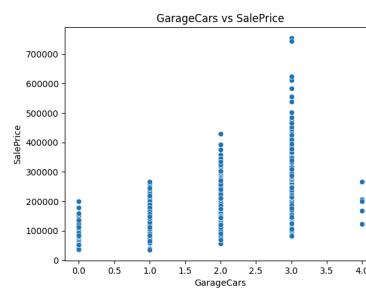
- da li je veza između atributa i cijene linearna ili ne,
- postoji li prisutnost očiglednih outliera (vrijednosti koje odstupaju),
- i kakav je opći trend koji bi mogao biti koristan za predikciju cijene.



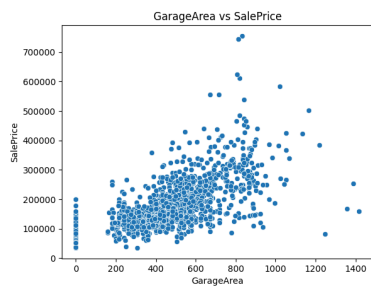
Slika 6. OverallQual



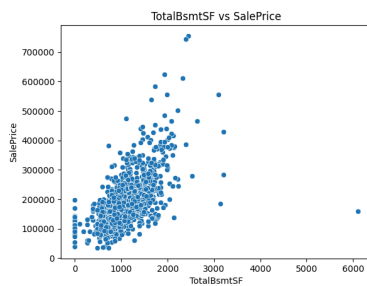
Slika 7. GrLivArea



Slika 8. GarageCars



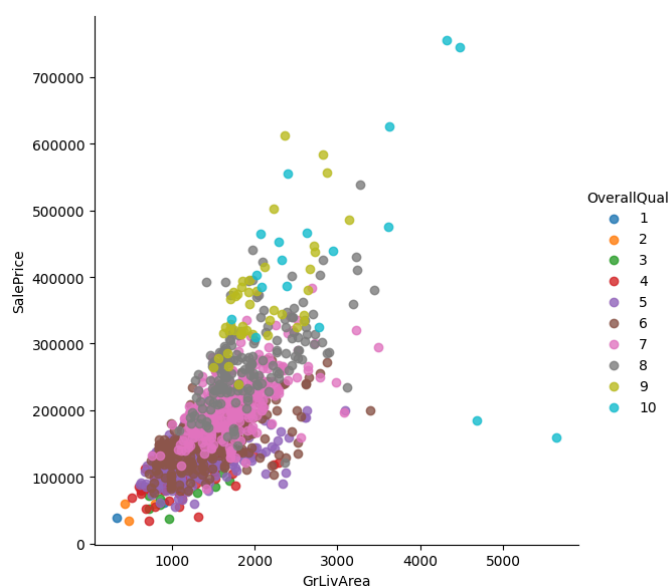
Slika 9. GarageArea



Slika 10. TotalBsmtSF

- **Višedimenzionalna analiza: Veza između GrLivArea i SalePrice uz utjecaj kvaliteta kuće (OverallQual)**

Koristeći Implot iz Seaborn biblioteke, prikazan je scatter plot koji ilustrira odnos između površine dnevnog prostora i prodajne cijene, uz dodatno grupisanje po ocjeni kvaliteta kuće. Podaci su obojeni različitim bojama prema ocjeni kvaliteta (OverallQual), što omogućava uvid u to kako kvalitet utiče na cijenu za različite veličine kuće. Regresiona linija nije crtana (fit_reg=False), fokus je na distribuciji i grupisanju podataka. Ova vizualizacija je korisna da se razumije složenost odnosa između veličine, kvaliteta i cijene nekretnine.



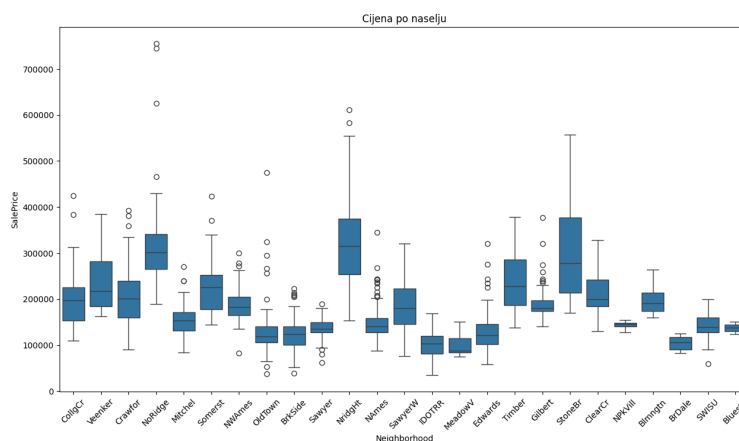
Slika 11. Višedimenzionalna analiza

- **Boxplot: Distribucija cijena kuća po naseljima (Neighborhood)**

Analizirali smo raspodjelu prodajnih cijena kroz različita naselja koristeći boxplot:

- Na x-osi su različita naselja,
- Na y-osi su cijene kuća,

Svaki boxplot prikazuje medijanu, donji i gornji kvartil, te eventualne outliere za cijene u datom naselju. Nazivi naselja su rotirani radi preglednosti, a veličina grafa je povećana da bi se bolje vidjeli detalji. Ova analiza omogućava identifikaciju naselja sa većom ili manjom prosječnom cijenom, kao i uočavanje varijabilnosti cijena unutar naselja, što može biti značajno za dalju tržišnu procjenu.



Slika 12. Boxplot: Distribucija cijena kuća po naseljima (Neighborhood)

2.1.7. Provjera duplikata i low variance

Metoda `duplicated()` se koristi da se utvrdi da li postoje duplikatni redovi u datasetu, odnosno identični podaci koji se ponavljaju.

- `train_df.duplicated().sum()` vraća ukupan broj duplikatnih redova u skupu podataka.
- U našem slučaju, rezultat je 0, što znači da ne postoje duplikati u datasetu.
- Odsustvo duplikata je važno za tačnost analize i pouzdanost modela, jer duplikati mogu iskriviti rezultate.

```
train_df.duplicated().sum()
```

Analiza za low variance ima za cilj identifikovati kolone koje imaju vrlo mali broj različitih vrijednosti (manje od 3), što znači da su gotovo svi podaci u tim kolonama isti ili se ponavljaju.

- `train_df[col].nunique()` se koristi da se prebroji koliko jedinstvenih vrijednosti postoji u svakoj koloni.
- Kolone koje imaju manje od 3 jedinstvene vrijednosti smatraju se kolona sa niskom varijabilnošću.
- Takve kolone obično nemaju značajan doprinos u treniranju modela jer ne pružaju dovoljno informacija za razlikovanje između uzoraka.
- Često se takve kolone uklanjaju tokom procesa čišćenja podataka.

U datasetu su identifikovane sljedeće kolone sa skoro istim vrijednostima:

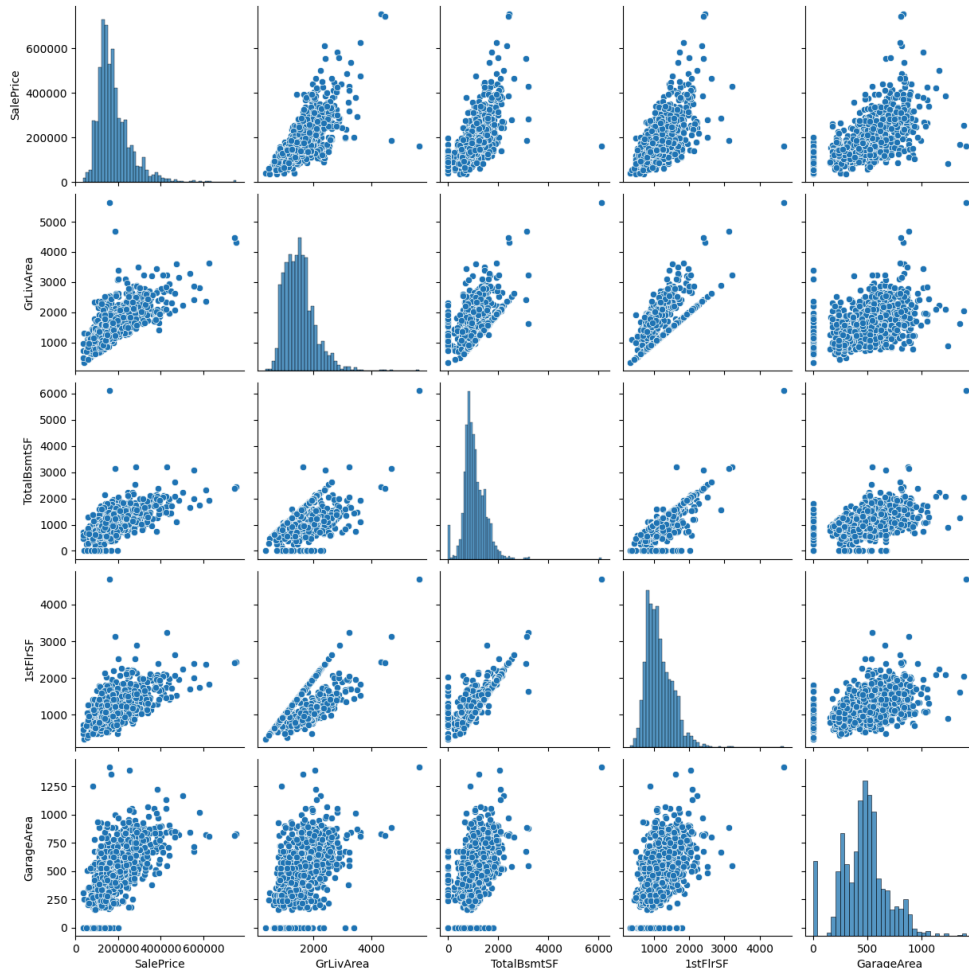
- Street
- Alley
- Utilities
- CentralAir

2.1.8. Vizualna analiza odnosa varijabli

Pairplot kreira mrežu grafikona da bi se analizirali međusobni odnosi varijabli u datasetu. Dijagonale prikazuju histogram pojedinačnih varijabli, što nam daje pregled raspodjele podataka za svaku od njih. Donja i gornja trokutasta matrica prikazuju scatter plotove koji predstavljaju odnos između parova varijabli. U ovom slučaju, fokusirali smo se na odnos ciljane varijable `SalePrice` i nekoliko ključnih karakteristika kuće koje mogu utjecati na cijenu:

- `GrLivArea` - površina glavnog stambenog prostora,

- TotalBsmntSF - ukupna površina podruma,
- 1stFlrSF - površina prvog sprata,
- GarageArea - površina garaže.



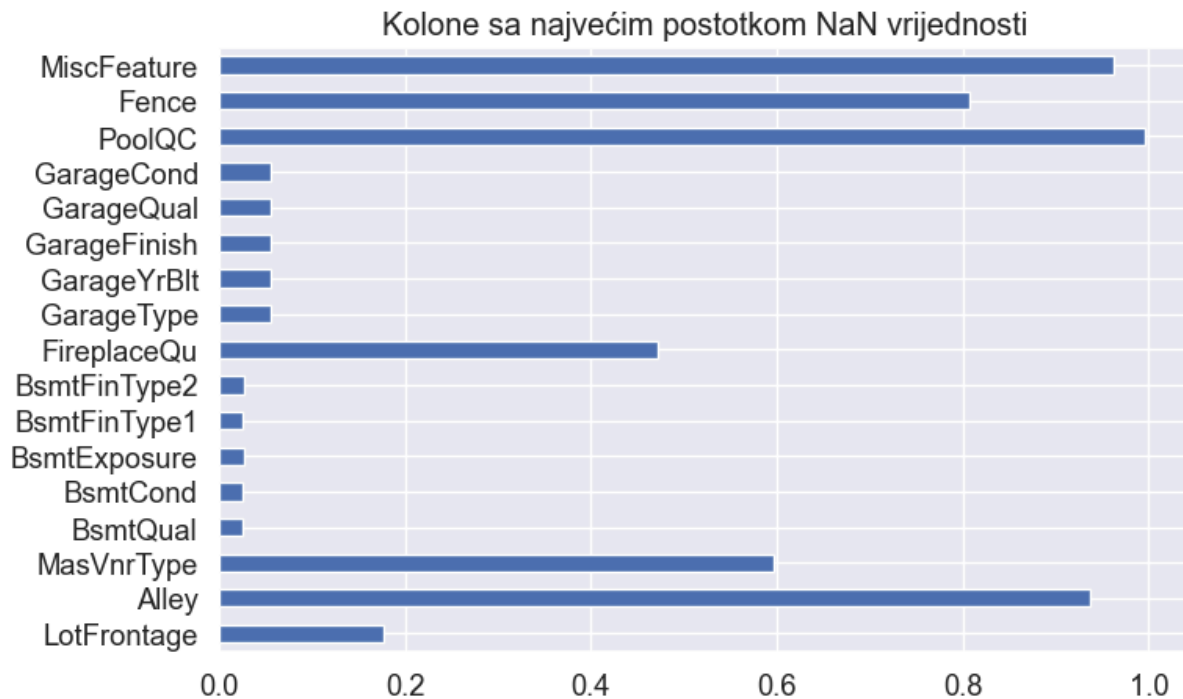
Slika 13. Vizualna analiza odnosa varijabli

2.1.9. Kolone sa NaN vrijednostima

U ovom dijelu analiziramo koliko kolona u datasetu sadrži nedostajuće vrijednosti (NaN) i koliko je njihov udio u odnosu na ukupni broj redova.

- Varijabla `col_nan` računa postotak nedostajućih vrijednosti po koloni, tako što dijeli broj NaN vrijednosti sa ukupnim brojem redova u datasetu (`train_df.shape[0]`).
- Zatim prikazujemo horizontalni bar grafikon koji uključuje samo one kolone gdje je udio NaN vrijednosti veći od 1% (tj. > 0.01).

- Ovakav graf jasno ističe koje kolone imaju značajan problem s nedostajućim podacima, što je ključni korak za odlučivanje o daljnjoj obradi tih kolona (popunjavanje, uklanjanje ili specifični pristupi).



Slika 14. Kolone sa NaN vrijednostima

2.2. Čišćenje podataka

2.2.1. Uklanjanje ID kolone, kolona sa istim vrijednostima i outlier-a

Kolona Id predstavlja jedinstveni identifikator svakog zapisa (kuće) u datasetu. Budući da ta kolona nema nikakvu korisnu prediktivnu vrijednost za modeliranje cijene nekretnina, smatra se nepotrebnom za daljnju analizu i treniranje modela. Zato se uklanja iz oba dataseta (train_df i test_df) korištenjem metode drop().

```
train_df = train_df.drop("Id", axis=1)
test_df = test_df.drop("Id", axis=1)
```

Kolone Street i Utilities su uklonjene iz oba dataset-a.

```
train_df = train_df.drop(["Street", "Utilities"], axis=1)
test_df = test_df.drop(["Street", "Utilities"], axis=1)
```

Uklanjammo redove (uzorke) kod kojih je vrijednost površine iznad zemlje (`GrLivArea`) veća od 4500 kvadratnih stopa.

- Ekstremno velike vrijednosti u `GrLivArea` su outlieri koji mogu značajno izmijeniti model i njegovu sposobnost da uči iz podataka.
- Takvi outlieri mogu uticati na lošiju generalizaciju modela, jer ne predstavljaju tipične slučajeve.
- Uklanjanjem ovih primjera smanjujemo šum i povećavamo preciznost modela na realnim, prosječnim vrijednostima.

```
train_df = train_df.drop(train_df[(train_df['GrLivArea'] > 4500)].index)
```

2.2.2. Spajanje train i test podataka u jedan skup podataka

Prvo izdvajamo ciljnu varijablu SalePrice iz trening skupa podataka, jer je ona ono što želimo predviđati. Zatim uklanjamo kolonu SalePrice iz trening skupa kako bismo dobili samo ulazne karakteristike koje ćemo koristiti za treniranje modela. Na kraju, spajamo ulazne karakteristike iz trening i test skupa u jedan zajednički skup podataka. Ovo omogućava dosljednu i uniformnu obradu podataka (kao što su imputacija nedostajućih vrijednosti, enkodiranje kategorijskih varijabli, skaliranje i slično) na oba skupa podataka. Indeksi se resetuju da bi rezultat bio uredan i neprekidan.

```
y = train_df['SalePrice']
train_features = train_df.drop(['SalePrice'], axis=1)
all_data = pd.concat([train_features, test_df]).reset_index(drop=True)
```

2.2.3. Ispravljanje grešaka u podacima

Ovdje provjeravamo vrijednosti u koloni `GarageYrBlt` (godina izgradnje garaže) koje su veće od 2017. Svima tim vrijednostima dodjeljujemo godinu 2007. Razlog za ovu korekciju može biti da su neke vrijednosti očito greške ili outlajeri, budući da je 2017. gornja granica logične godine izgradnje u datasetu (ili godina kreiranja podataka). Postavljanjem na 2007. ove vrijednosti se vraćaju na realističniji datum, što pomaže da model ne bude zbunjen nerealnim ekstremnim podacima.

```
all_data.loc[all_data['GarageYrBlt'] > 2017, 'GarageYrBlt'] = 2007
```

2.2.4. Popunjavanje LotFrontage po Neighborhood

Kolona LotFrontage predstavlja dužinu fronta zemljišta oko kuće i sadrži određeni broj nedostajućih (NaN) vrijednosti. Umjesto da ove nedostajuće vrijednosti popunimo jednom globalnom vrijednošću za cijeli skup podataka, koristimo lokalni pristup zasnovan na naseljima. Grupišemo podatke prema koloni Neighborhood (naselje) i za svaku grupu

izračunavamo medijanu vrijednosti LotFrontage. Nedostajuće vrijednosti unutar svake grupe popunjavamo upravo tom medijanom, specifičnom za to naselje. Vrijednosti LotFrontage znatno variraju između različitih naselja, pa je primjerenije koristiti medijanu unutar svake grupe nego jednu zajedničku vrijednost za cijeli dataset. Ovim pristupom sačuvamo lokalne karakteristike podataka i dobijemo realističniju i precizniju imputaciju.

- Metoda groupby('Neighborhood') dijeli podatke u grupe po naseljima.
- Zatim transform(lambda x: x.fillna(x.median())) popunjava sve nedostajuće vrijednosti u svakoj grupi medijanom te grupe.

```
all_data['LotFrontage'] =  
all_data.groupby('Neighborhood')['LotFrontage'].transform(lambda x:  
x.fillna(x.median()))
```

2.2.5. Pretvaranje numeričkih varijabli u kategorijske

- Kolone 'MSSubClass' i 'OverallCond' su u izvornom datasetu predstavljene kao numeričke vrijednosti (int ili float).
- Međutim, ove kolone predstavljaju kategorijske informacije, odnosno klase ili ocjene koje nisu numerički kontinuirane vrijednosti.
- Zato ih pretvaramo u string ('astype(str)'), kako bismo ih tretirali kao kategorijske varijable tokom daljnje analize i modeliranja.

```
all_data['MSSubClass'] = all_data['MSSubClass'].astype(str)  
all_data['OverallCond'] = all_data['OverallCond'].astype(str)
```

2.2.6. Z-score metoda za uklanjanje outlier-a

Ovdje je predstavljena funkcija remove_outliers koja služi za identifikaciju i uklanjanje redova u datasetu čije vrijednosti u određenim numeričkim kolonama značajno odstupaju od prosjeka, koristeći Z-score pristup.

- Opis funkcije:
 - Ulazni parametri:
 - dataset: DataFrame nad kojim želimo izvršiti analizu i uklanjanje outlier-a.
 - threshold (zadana vrijednost: 3): Prag Z-score vrijednosti iznad kojeg se pojedinačne vrijednosti smatraju outlierima.
 - columns: Lista kolona koje se provjeravaju na prisustvo outlier-a. Ukoliko nije specificirana, funkcija automatski obrađuje sve numeričke kolone tipa int64 i float64.

- Način rada:
 - Za odabrane kolone izračunava se Z-score — standardizirana vrijednost koja pokazuje koliko je neki podatak udaljen od srednje vrijednosti izražene u standardnim devijacijama.
 - Identifikuju se redovi u kojima je bilo koja od vrijednosti u odabranim kolonama veća od zadanog praga (tj. apsolutna vrijednost Z-score veća od 3).
 - Izračunava se broj takvih redova (outliera).
 - Funkcija vraća novi DataFrame iz kojeg su izbačeni svi ti outlieri, čime se čisti skup podataka od ekstremnih odstupanja.

Uklanjanje outliera iz kolone GrLivArea u train_df rezultiralo je brisanjem 16 redova sa ekstremno visokim vrijednostima.

2.2.7. Popunjavanje nedostajućih numeričkih vrijednosti

Ovaj dio koda se bavi tretiranjem nedostajućih (NaN) vrijednosti u skupu podataka all_data, koristeći različite pristupe za različite grupe kolona, u skladu sa značenjem i prirodom podataka.

1. Kolone koje popunjavamo nulama:

```
zero_fill = ['BsmtFinSF1', 'BsmtFinSF2', 'BsmtFullBath', 'BsmtHalfBath',
'MasVnrArea']
```

Ove kolone predstavljaju površine ili brojeve koji se logično mogu tumačiti kao nula ako nema podataka (npr. nema podruma, nema zidnog obloga ili nema dodatnih kupatila). Stoga se nedostajuće vrijednosti popunjavaju sa 0, jer odsustvo podataka često znači da ta karakteristika nije prisutna.

2. Kolone koje popunjavamo medijanom:

```
median_fill = [col for col in all_data.select_dtypes(include=['int64',
'float64']).columns if col not in zero_fill]
```

Sve ostale numeričke kolone, osim onih koje smo već odabrali za popunjavanje nulama, popunjavamo medijanom. Medijan se koristi jer je otporniji na ekstremne vrijednosti (outliere) i bolje predstavlja centralnu tendenciju u slučaju nesimetričnih podataka.

3. Popunjavanje nedostajućih vrijednosti:

```
all_data[zero_fill] = all_data[zero_fill].fillna(0)
all_data[median_fill] =
all_data[median_fill].fillna(all_data[median_fill].median())
```

Za kolone iz zero_fill grupe, svi NaN se zamjenjuju sa 0. Za kolone iz median_fill grupe, NaN vrijednosti se popunjavaju medijanom odgovarajuće kolone.

2.2.8. Popunjavanje kategorijskih kolona specifičnim vrijednostima

Ovaj dio rješava problem nedostajućih (NaN) vrijednosti u različitim kategorijskim kolonama skupa podataka all_data tako što za svaku kolonu koristi zamjensku vrijednost koja ima smislen kontekst u odnosu na prirodu podataka.

Opis postupka:

- Definisana je lista none_conversion koja sadrži parove (ime_kolone, zamjenska_vrijednost).
- Za svaku kolonu navedenu u toj listi, ako se nalazi u skupu podataka, svi NaN unosi se popunjavaju zadatom zamjenskom vrijednošću.

Primjeri zamjenskih vrijednosti i njihovo značenje:

- "MasVnrType": zamjena sa "None" znači da objekat nema oblogu od kamena.
- "BsmtQual": popunjava se sa "NA" što označava odsustvo podruma.
- "Electrical": popunjava se sa "SBrkr", što je najčešći tip električne instalacije u datasetu.
- "GarageType": popunjava se sa "No", što znači da objekat nema garažu.
- "KitchenQual": popunjava se sa "TA" (Typical/Average), što predstavlja standardnu ocjenu kvaliteta kuhinje.
- Ostale kolone popunjavaju se na sličan način, birajući najlogičniju ili najčešću kategoriju za svaku od njih.

Na ovaj način se osigurava da nedostajuće vrijednosti u kategorijskim kolonama budu popunjene na smislen i konzistentan način, što doprinosi kvaliteti i pouzdanosti podataka za daljnju analizu ili modeliranje.

2.2.9. Transformacija “skewed” kolona sa Box-Cox

U ovom dijelu koda vrši se prepoznavanje i ispravljanje asimetrije u distribuciji numeričkih podataka u datasetu `all_data` kroz primjenu Box-Cox transformacije, kako bi se podaci što bolje približili normalnoj raspodjeli.

Koraci i objašnjenja:

1. Izdvajanje numeričkih kolona: Prvo se iz skupa podataka izdvajaju sve kolone numeričkog tipa (`int64` i `float64`), koje su potencijalni kandidati za transformaciju.
2. Mjerenje asimetrije distribucije: Za svaku numeričku kolonu izračunava se asimetrija (skewness) distribucije koristeći funkciju `skew` iz biblioteke `scipy.stats`.
 - Asimetrija pokazuje koliko je raspodjela podataka iskrivljena u lijevu ili desnu stranu.
 - Vrijednosti veće od 0.5 označavaju značajnu pozitivnu asimetriju, što može negativno uticati na analizu i modele.
3. Filtriranje kolona sa značajnom asimetrijom: Kolone kod kojih je asimetrija veća od 0.5 se izdvajaju u listu `high_skew` i smatraju kandidatima za transformaciju.
4. Primjena Box-Cox transformacije: Na kolone s visokom asimetrijom primjenjuje se `boxcox1p` transformacija sa parametrom 0.15.
 - Ova transformacija pomaže da distribucija postane što normalnija.
 - `boxcox1p` je verzija Box-Cox transformacije koja može raditi i sa vrijednostima koje uključuju nulu, čime se izbjegavaju problemi sa logaritamskim transformacijama.

```
from scipy.special import boxcox1p
from scipy.stats import skew

# Izdvajanje numeričkih kolona
numeric_feats = all_data.select_dtypes(include=['int64', 'float64']).columns

# Izračunavanje asimetrije za svaku kolonu
skewed_feats = all_data[numeric_feats].apply(lambda x:
skew(x.dropna())).sort_values(ascending=False)

# Filtriranje kolona sa asimetrijom većom od 0.5
high_skew = skewed_feats[skewed_feats > 0.5].index

# Primjena Box-Cox transformacije na izabrane kolone
for feat in high_skew:
    all_data[feat] = boxcox1p(all_data[feat], 0.15)
```

2.2.10. Enkodiranje

1. Ordinalno mapiranje

Varijable koje imaju prirodan redoslijed (npr. kvaliteta: loše < izvrsno) mapiramo ručno na cijele brojeve koji očuvaju taj redoslijed. Primjer: "Po" (poor) → 1, "Ex" (excellent) → 5. Na taj način model može iskoristiti informaciju o hijerarhiji u kategorijama.

2. Label encoding

Za ostale kategorijske varijable koje nemaju redoslijed, koristi se LabelEncoder iz sklearn-a. Svaka jedinstvena kategorija dobija numeričku vrijednost, ali bez ikakvog značenja redoslijeda. Ovakav pristup je jednostavan i često dovoljan za stabla odlučivanja (Random Forest, xGBoost).

Ordinalno mapiranje omogućava modelima da razumiju "stepen" kvalitete ili funkcionalnosti (npr. TA je bolje od Fa, ali lošije od Gd). Label encoding je efikasan i dovoljan za stabla odlučivanja jer oni ne pretpostavljaju linearne odnose između kategorijskih vrijednosti. Čuvanje enkodera (label_encoders) je praktično jer omogućava da isti kod koristiš i na testnim ili budućim podacima, što sprečava curenje informacija i greške u transformaciji.

2.3. Provjera čišćenja

- X_train shape: prikazuje dimenzije trening skupa podataka, tj. broj uzoraka (redova) i broj značajki (kolona). To nam govori koliko podataka imamo za učenje i koliko atributa koristimo za predikciju.
- y shape: prikazuje dimenziju ciljne varijable, odnosno broj vrijednosti koje model treba da nauči predikciju. Broj uzoraka u y mora biti isti kao u X_train.
- X_test shape: prikazuje dimenzije testnog skupa podataka koji se koristi za evaluaciju modela. Test skup nema ciljnu varijablu jer je njegova svrha da model na njemu pravi predikcije.

Ovaj ispis potvrđuje da:

- Imamo 1458 trening uzoraka sa po 77 feature-a.
- Ciljna varijabla y ima odgovarajući broj od 1458 vrijednosti.
- Testni skup sadrži 1459 uzoraka, također sa 77 feature-a.

2.3.1. Provjera NaN vrijednosti

- NaN u X_train: prikazuje ukupan broj nedostajućih vrijednosti u trening skupu podataka. Idealno je da ova vrijednost bude nula, što znači da nema praznih ili nepopunjenih polja.

- NaN u X_test: prikazuje ukupan broj nedostajućih vrijednosti u testnom skupu podataka, koji također treba biti bez praznina.

```
print("NaN u X_train:", X_train.isnull().sum().sum())  
print("NaN u X_test:", X_test.isnull().sum().sum())
```

Ovaj rezultat:

NaN u X_train: 0

NaN u X_test: 0

potvrđuje da su podaci u oba skupa kompletni i spremni za dalju upotrebu bez potrebe za dodatnim popunjavanjem ili čišćenjem.

2.3.2. Provjera tipova podataka

- Izlaz X_train.dtypes.value_counts() prikazuje broj kolona po tipu podataka.
- U našem slučaju, sve značajke su numeričke - int64 i float64, što je poželjno za modele poput XGBoost i Random Forest koji zahtijevaju numeričke ulaze.

```
print(X_train.dtypes.value_counts())
```

Ovaj rezultat:

int64 50

float64 27

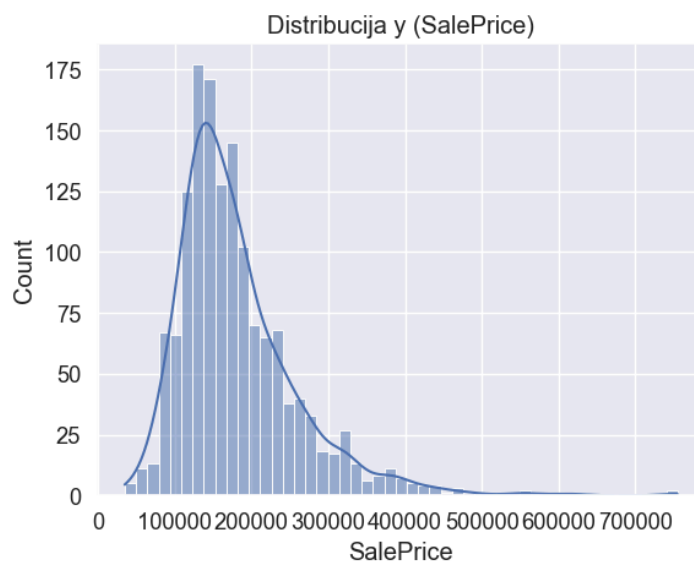
Name: count, dtype: int64

potvrđuje da su svi podaci spremni za treniranje modela, bez preostalih tekstualnih vrijednosti koje bi bilo potrebno dodatno transformisati.

2.3.3. Provjera distribucije ciljne varijable (y)

Za vizualnu analizu raspodjele ciljne varijable y nakon čišćenja (u ovom slučaju SalePrice), koristimo biblioteku Seaborn koja omogućava crtanje histogram sa KDE (Kernel Density Estimate) linijom. KDE pomaže da bolje razumijemo oblik distribucije - da li je simetrična, ima li repova ili više modova.

- sns.histplot(y, kde=True) iscrtava histogram y zajedno sa glatkom KDE linijom.
- Grafikon pomaže u procjeni prirode podataka i može ukazati na eventualnu potrebu za transformacijom (npr. log-transformacijom) ciljne varijable.



Slika 15. Distribucija SalePrice nakon čišćenja

2.3.4. Provjera korelacija

Nakon što smo izvršili čišćenje i pripremu podataka, uključujući uklanjanje outliera, popunjavanje nedostajućih vrijednosti i enkodiranje kategorijskih varijabli, kreirali smo kopiju skupa podataka za treniranje i dodali ciljnu varijablu 'SalePrice'. Zatim smo izračunali Pearsonovu korelaciju između svih numeričkih atributa i 'SalePrice' te sortirali vrijednosti od najveće do najmanje. Ispis prvih deset značajki sa najjačom korelacijom pomaže nam identificirati koje varijable najviše utiču na cijenu nekretnine, što je korisno za dalji razvoj i optimizaciju modela.

```
train_corr = X_train.copy()
train_corr["SalePrice"] = y
correlation = train_corr.corr()["SalePrice"].sort_values(ascending=False)
print(correlation.head(10))
```

Naziv varijable	Vrijednost
SalePrice	1.000000
OverallQual	0.287047
GrLivArea	0.248560
GarageCars	0.244847
GarageArea	0.243663

ExterQual	0.242166
KitchenQual	0.230058
BsmtQual	0.227074
1stFlrSF	0.218437
FireplaceQu	0.213916

2.3.5. Spremanje pripremljenih podataka

Nakon završetka svih koraka čišćenja, obrade i enkodiranja podataka, pripremljene skupove podataka spremamo za dalju upotrebu. Kreiramo direktorij `../processed_data` (ako već ne postoji) i u njega pohranjujemo:

- `X_train_processed.csv` - pripremljene značajke za trening modela,
- `X_test_processed.csv` - pripremljene značajke za testiranje modela,
- `y_train.csv` - ciljnu varijablu (cijene kuća) za trening.

Na kraju, ispisujemo potvrdu o uspješnom spremanju datoteka, kao i sadržaj foldera kako bismo provjerili da li su datoteke zaista spremljene. Ovaj korak omogućava lako upravljanje podacima i njihovu ponovnu upotrebu u narednim fazama razvoja modela.

3. ALGORITMI ZA KLASIFIKACIJU

U cilju predviđanja cijena nekretnina korišteni su nadzirani algoritmi mašinskog učenja koji su se pokazali efikasnim u regresionim problemima, gdje je cilj predvidjeti numeričku vrijednost cijene na osnovu relevantnih karakteristika nekretnina. S obzirom na kompleksnost problema i značajne varijacije u podacima (poput lokacije, veličine, godišta i stanja nekretnine), odabrani su algoritmi koji mogu precizno modelirati nelinearne odnose i interakcije između obilježja.

U ovom poglavlju detaljno su opisani korišteni algoritmi – Random Forest Regressor, XGBoost Regressor i Ansambl model – sa fokusom na njihove tehničke karakteristike, način rada te razloge zbog kojih su pogodni za rješavanje problema regresije u kontekstu predviđanja cijena nekretnina. Svaki od ovih algoritama ima svoje prednosti, od robusnosti Random Foresta, preko visoke preciznosti XGBoosta, do poboljšane generalizacije Ansambl modela.

Glavni aspekti koji su razmatrani pri odabiru algoritama uključuju njihovu sposobnost obrade velikog broja obilježja, otpornost na šum u podacima i mogućnost interpretacije rezultata, što je od posebnog značaja za donosiocje odluka u domenu nekretnina.

3.1. Random forest

Random forest su kombinacija stabla prediktora, gdje svako stablo zavisi od vrijednosti nasumičnog vektora uzorkovanog nezavisno i s istom distribucijom za sva stabla u šumi. Generalizacijska greška za šume konvergira a.s. do limita kako broj stabala u šumi postaje veliki. Generalizacijska greška šume stabala klasifikatora zavisi od snage pojedinačnih stabala u šumi i korelacije među njima. Korištenje nasumične selekcije karakteristika za podjelu svakog čvora daje stope grešaka koje su povoljnije u poređenju s Adaboost-om[8], ali su robusnije u odnosu na šum. Interni procjenitelji prate grešku, snagu i korelaciju, a ovi podaci se koriste za prikaz odgovora na povećanje broja karakteristika korištenih u podjeli. Interni procjenitelji se također koriste za mjerenje važnosti varijabli. Ove ideje su također primjenjive na regresiju.

Značajna poboljšanja u tačnosti klasifikacije postignuta su rastom skupa stabala i omogućavanjem da glasaju za najpopularniju klasu. Da bi se uzgajali ovi skupovi, često se generišu nasumični vektori koji upravljaju rastom svakog stabla u skupu. Rani primjer je bagging [9], gdje se za rast svakog stabla vrši nasumična selekcija (bez ponavljanja) iz primjera u skupu za treniranje. Još jedan primjer je nasumična selekcija podjele [10], gdje se

na svakom čvoru podjela bira nasumično među K najboljih podjela. Breiman [11] generiše nove skupove za treniranje randomizacijom izlaza u originalnom skupu za treniranje. Drugi pristup je odabir skupa za treniranje iz nasumičnog skupa težina na primjerima u skupu za treniranje.

3.2. XGboost

XGBoost je biblioteka za mašinsko učenje koja koristi optimiziranu verziju gradijentnog pojačavanja stabala odlučivanja. Dizajnirana je da bude visoko efikasna, prilagodljiva i skalabilna, omogućavajući brzu i tačnu obradu velikih i kompleksnih skupova podataka, uključujući i rad u distribuiranim sistemima poput Hadoop-a i MPI-ja. [12] XGBoost pravi model tako što dodaje nova stabla koja ispravljaju greške prethodnih, model se gradi iterativno. Svaka iteracija minimizira funkciju cilja uzimajući u obzir prethodne predikcije. Model cilja da minimizira sumu greške (loss function) i regularizacije. Ovo pomaže da se spriječi overfitting podataka. Loss funkcija i regularizacija zajedno čine *Objective Function* što je mjera kvaliteta modela tokom učenja i sastoji se od tačnosti na trenirajućim podacima plus dodatnih ograničenja za kontrolu složenosti modela. *Objective Function* - $obj(\theta)$ se predstavlja:

$$obj(\theta) = L(\theta) + \Omega(\theta)$$

gdje je L loss funkcija za trening podatke, a Ω regularizacija. Najčešće se za loss funkciju nad trening podacima koristi *mean squared error*:

$$L(\theta) = \sum_1 (y_i - \hat{y}_i)^2$$

Decision Tree Ensembles u XGBoostu odnose se na korištenje skupa CART modela (Classification and Regression Trees) kao osnovnih građevnih jedinica. Za razliku od klasičnih stabala odlučivanja, gdje listovi sadrže samo klasifikacijske odluke, kod CART-a listovi nose realne numeričke vrijednosti što omogućava fleksibilnije izražavanje i optimizaciju modela. Matematički se predstavlja:

$$obj(\theta) = \sum_{k=1}^K f_k(x_i), f_k \in \mathcal{F}$$

gdje je K broj stabala, f_k je funkcija u funkcionalnom prostoru \mathcal{F} , a \mathcal{F} je skup svih mogućih CART-ova.

XGBoost koristi ansambl stabala, gdje se predikcije više pojedinačnih stabala sabiraju kako bi se dobila konačna predikcija. Takav pristup poboljšava tačnost jer pojedinačna stabla često nisu dovoljno snažna sama po sebi. Model se formalno definiše kao suma funkcija iz prostora mogućih CART-ova, a treniranje modela podrazumijeva minimizaciju ciljne funkcije koja uključuje funkciju greške (loss) i regularizaciju složenosti stabala.

Random forest i gradient boosted trees pripadaju istoj klasi modela - ansambl stabala, i razlikuju se samo po načinu treniranja. [13]

3.3. Ansambl model

U okviru eksperimentisanja s modelima, tokom evaluacije performansi Random Forest i XGBoost regresora, uočeno je da svaki od njih pokazuje specifične prednosti u različitim aspektima problema. Random Forest se pokazao robusnijim u generalizaciji i detekciji šireg spektra obrazaca, dok je XGBoost bio precizniji u hvatanju lokalnih varijacija u podacima, što je bilo posebno korisno kod nelinearnih odnosa i izraženih anomalija u cijenama.

Na osnovu tih opažanja, došlo se do ideje da se konstruiše jednostavan ansambl model, koji kombinuje predikcije oba algoritma. Ova kombinacija, ostvarena putem prostog prosjeka predikcija, ima za cilj da iskoristi komplementarne prednosti oba pristupa – stabilnost i otpornost Random Foresta, te sposobnost XGBoosta da iterativno poboljšava tačnost kroz korekciju grešaka.

"Kombinacija bagginga (RF) i boostinga (XGBoost) može smanjiti i varijansu i pristrasnost, što je idealno za regresione probleme s kompleksnim odnosima" (Zhou, 2012).[14]

"XGBoost ima bolju preciznost na lokalnim obrascima, dok RF bolje generalizira – njihova kombinacija pokriva širi spektar obrazaca" (Sagi & Rokach, 2018).[15]

Kao što ističu Zhou [14] i Sagi & Rokach [15], kombinacija bagging (Random Forest) i boosting (XGBoost) pristupa može dovesti do smanjenja i varijanse i pristrasnosti modela, što je posebno značajno kod regresionih problema sa kompleksnim obrascima.

U kontekstu predikcije cijena kuća, gdje postoji veliki broj zavisnih faktora i međusobnih odnosa među varijablama, ovakav hibridni model pruža bolji balans između preciznosti i opšte sposobnosti generalizacije. Rezultati evaluacije ansambl modela dodatno potvrđuju opravdanost ovog pristupa, kroz poboljšane metrike performansi u poređenju sa pojedinačnim modelima.

Kao što je već navedeno u radu je korišten prosti prosjek predikcija (engl. averaging ensemble), gdje se konačna vrijednost izračunava kao:

$$y = \frac{y_{rf} + y_{xb}}{2}$$

Ovakav pristup pokazao se efektivnim u praksi, što potvrđuju i evaluacione metrike (RMSE, MAE, R^2) na validacionom skupu podataka, gdje ansambl model u prosjeku postiže bolje rezultate od pojedinačnih modela. Dodatna prednost je što ansamblovanje smanjuje varijansu greške bez značajnog povećanja pristrasnosti, što je posebno korisno u kontekstu predviđanja cijena nekretnina gdje su stabilne procjene od ključnog značaja.

4. IMPLEMENTACIJA MODELA

4.1. Uvod u implementaciju: biblioteke i učitavanje podataka

4.1.1. Biblioteke i priprema okruženja

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score,
GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
import warnings
warnings.filterwarnings('ignore')

plt.style.use('seaborn')
sns.set_palette('husl')
%matplotlib inline
```

- numpy i pandas: osnovni alati za manipulaciju numeričkim i tabelarnim podacima.
- matplotlib i seaborn: za vizualizaciju podataka i rezultata (grafovi).
- sklearn.model_selection: omogućava podjelu podataka i unakrsnu validaciju (GridSearchCV je automatska pretraga najboljih hiperparametara).
- RandomForestRegressor i XGBRegressor: dva moćna modela za regresiju koja ćemo koristiti.
- sklearn.metrics: funkcije za mjerenje tačnosti modela.
- StandardScaler i make_pipeline: za skaliranje podataka i lakšu organizaciju koraka obrade i treniranja modela.
- warnings.filterwarnings('ignore'): isključuje nepotrebne poruke upozorenja da notebook izgleda urednije.
- Stilovi i opcije za grafove podešeni su radi ljepšeg izgleda prikaza.

4.1.2. Učitavanje podataka

```
try:
    X_train = pd.read_csv('../processed_data/X_train_processed.csv')
    y_train = pd.read_csv('../processed_data/y_train.csv').values.ravel()
    X_test = pd.read_csv('../processed_data/X_test_processed.csv')
    print("Podaci uspješno učitani!")
    print(f"X_train shape: {X_train.shape}")
    print(f"y_train shape: {y_train.shape}")
```

```
print(f"X_test shape: {X_test.shape}")
except Exception as e:
    print(f"Greška pri učitavanju podataka: {str(e)}")
```

- Učitavamo prethodno očišćene i pripremljene (procesirane) podatke za trening i testiranje.
- try-except blok služi za hvatanje grešaka prilikom učitavanja datoteka.
- `y_train.values.ravel()` konvertuje ciljnu varijablu u jednodimenzionalni niz što neki sklearn modeli zahtijevaju.
- Ispisujemo oblike podataka radi provjere da li je sve ispravno učitano.

4.1.3. Podjela podataka na trening i validaciju

```
X_train, X_val, y_train, y_val = train_test_split(
    X_train, y_train, test_size=0.3, random_state=42
)
```

- Podijelili smo trening skup na dva dijela: 70% za treniranje i 30% za validaciju.
- Validacioni skup koristi se za procjenu modela dok ga još treniramo, da izbjegnemo prekomjerno prilagođavanje modela podacima za treniranje.
- `random_state=42` osigurava da je podjela reproducibilna.

4.2. Tehnike i metode za poboljšanje modela

4.2.1. Definisane evaluacijskih metrika

```
def evaluate_model(y_true, y_pred, model_name=""):
    metrics = {
        'MAE': mean_absolute_error(y_true, y_pred),
        'RMSE': np.sqrt(mean_squared_error(y_true, y_pred)),
        'RMSLE': np.sqrt(mean_squared_error(np.log1p(y_true),
np.log1p(y_pred))),
        'R2': r2_score(y_true, y_pred)
    }
    print(f"\n{model_name} Metrike:")
    for name, value in metrics.items():
        print(f"{name}: {value:.4f}")
    return metrics
```

- MAE (Mean Absolute Error): prosječna apsolutna greška.
- RMSE (Root Mean Squared Error): korijen srednje kvadratne greške, kažnjava veće greške jače.

- RMSLE (Root Mean Squared Logarithmic Error): slična RMSE, ali koristi logaritamsku skalu, što je posebno korisno kod cijena (osjetljivije na relativne greške).
- R^2 (koeficijent determinacije): pokazuje koliki dio varijacije ciljnih vrijednosti model objašnjava.

4.2.2. Podešavanje hiperparametara s GridSearchCV

GridSearchCV omogućava automatsko testiranje različitih kombinacija hiperparametara modela. Koristi unakrsnu validaciju (5-fold u ovom slučaju) da procijeni model na različitim podskupovima. Cilj je naći skup hiperparametara koji daje najmanju grešku (maksimalnu tačnost).

4.2.3. Korištenje Pipeline objekta

Pipeline omogućava da niz koraka (skaliranje, treniranje modela) tretiramo kao jedan objekt. To omogućava lakše korištenje i sprečava curenje podataka (npr. iz validacionog u trening skup).

4.3. Implementacija i analiza modela

4.3.1. Random Forest

```
rf_pipeline = make_pipeline(
    StandardScaler(),
    RandomForestRegressor(random_state=42, n_jobs=-1)
)

rf_params = {
    'randomforestregressor__n_estimators': [100, 200],
    'randomforestregressor__max_depth': [None, 10, 20],
    'randomforestregressor__min_samples_split': [2, 5],
    'randomforestregressor__min_samples_leaf': [1, 2]
}

rf_grid = GridSearchCV(
    estimator=rf_pipeline,
    param_grid=rf_params,
    cv=5,
    scoring='neg_mean_squared_error',
    n_jobs=-1,
    verbose=1
)

rf_grid.fit(X_train, y_train)
best_rf = rf_grid.best_estimator_
```

- Random Forest gradi ansambl stabala odluka i kombinuje njihove predikcije.

- Skaliramo podatke da model radi bolje.
- Testiramo različite vrijednosti broja stabala, maksimalne dubine, i minimalnog broja uzoraka za dijeljenje.
- Nakon treniranja, biramo model sa najboljim parametrima.
- Predikcije na validacionom skupu procjenjujemo pomoću ranije definisanih metrika.

4.3.2. Značajnost obilježja u Random Forest

```
rf_feature_imp = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance':
best_rf.named_steps['randomforestregressor'].feature_importances_
}).sort_values('Importance', ascending=False)

plt.figure(figsize=(12, 8))
sns.barplot(x='Importance', y='Feature', data=rf_feature_imp.head(20))
plt.title('Top 20 obilježja - Random Forest')
plt.tight_layout()
plt.show()
```

- Prikazujemo koja obilježja najviše utiču na odluke modela.
- Ovo pomaže u razumijevanju modela i može ukazati na potencijalne ključne faktore koji utiču na cijene kuća.

4.4. XGBoost model

```
xgb_pipeline = make_pipeline(
    StandardScaler(),
    XGBRegressor(random_state=42, n_jobs=-1, objective='reg:squarederror')
)

xgb_params = {
    'xgbregressor__n_estimators': [100, 200],
    'xgbregressor__learning_rate': [0.01, 0.05, 0.1],
    'xgbregressor__max_depth': [3, 5, 7],
    'xgbregressor__subsample': [0.8, 1.0],
    'xgbregressor__colsample_bytree': [0.8, 1.0]
}

xgb_grid = GridSearchCV(
    estimator=xgb_pipeline,
    param_grid=xgb_params,
    cv=5,
    scoring='neg_mean_squared_error',
    n_jobs=-1,
    verbose=1
)

xgb_grid.fit(X_train, y_train)
```

```
best_xgb = xgb_grid.best_estimator
```

- XGBoost koristi boosting tehniku, gdje se modeli nadograđuju jedan na drugi kako bi se smanjile greške.
- Podešavamo broj stabala, brzinu učenja, maksimalnu dubinu, te poduzorkovanje da spriječimo prenaučenosť.
- Nakon treniranja i odabira najboljih hiperparametara, model evaluiramo.

4.4.1. Značajnost obilježja u XGBoostu

```
xgb_feature_imp = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': best_xgb.named_steps['xgbregressor'].feature_importances_
}).sort_values('Importance', ascending=False)

plt.figure(figsize=(12, 8))
sns.barplot(x='Importance', y='Feature', data=xgb_feature_imp.head(20))
plt.title('Top 20 obilježja - XGBoost')
plt.tight_layout()
plt.show()
```

4.5. Evaluacija modela na validacionom skupu

```
y_pred_rf = best_rf.predict(X_val)
evaluate_model(y_val, y_pred_rf, "Random Forest")
y_pred_xgb = best_xgb.predict(X_val)
evaluate_model(y_val, y_pred_xgb, "XGBoost")
```

- Predikcije pravimo na podacima koje model nije vidio prilikom treniranja.
- Upoređujemo metrike da vidimo koji model bolje generalizuje.

4.6. Predikcija na test skupu i spremanje rezultata

```
y_test_pred = best_xgb.predict(X_test) # ili best_rf, ovisno koji je bolji
submission = pd.DataFrame({
    'Id': X_test.index,
    'SalePrice': y_test_pred
})
submission.to_csv('submission.csv', index=False)
```

- Nakon što smo odabrali najbolji model, koristimo ga za predikciju konačnih vrijednosti na test skupu.

- Rezultat se sprema u CSV datoteku za eventualnu dalju upotrebu (npr. za takmičenja ili poslovnu analizu).

5. ZAKLJUČAK

Predviđanje cijena nekretnina složen je problem s velikim praktičnim značenjem, kako za pojedince tako i za institucije. Ovaj rad pokazuje da moderni algoritmi mašinskog učenja poput Random Forest-a i XGBoost-a mogu postići visoku razinu tačnosti u procjeni vrijednosti nekretnina, što ih čini korisnim alatima za tržišnu analizu. Ključni doprinos ovog istraživanja leži u sistematskom pristupu – od pripreme podataka i optimizacije modela do detaljne evaluacije i interpretacije rezultata.

Eksperimenti su potvrdili da XGBoost često nadmašuje druge modele u pogledu generalizacije, dok Random Forest pruža bolju interpretabilnost. Primjena GridSearchCV omogućila je pronalaženje optimalnih parametara, a analiza značajnosti obilježja otkrila je koje varijable najviše utječu na cijene. Dodatno, kombinovanje modela kroz ansambl pokazalo je obećavajuće rezultate, sugerirajući da integracija različitih pristupa može poboljšati stabilnost predviđanja.

Ovi rezultati imaju neposrednu primjenu u stvarnom svijetu – od pomoći kupcima i prodavateljima u donošenju informiranih odluka do podrške bankama i agencijama u procjeni rizika. Budući radovi mogu se usredotočiti na proširenje skupa podataka, testiranje dodatnih algoritama ili primjenu dubokog učenja za hvatanje kompleksnijih obrazaca. Bez obzira na daljnji pravac istraživanja, jasno je da mašinsko učenje postaje sve važniji alat u analizi nekretnina, pružajući objektivnije i pouzdanije procjene u dinamičnom tržišnom okruženju.

Konačno, ovaj rad ne samo da potvrđuje efikasnost postojećih metoda već i otvara put novim istraživačkim izazovima, naglašavajući važnost interdisciplinarnog pristupa u rješavanju ekonomskih problema pomoću umjetne inteligencije.

6. REFERENCE

- [1] A. Kaushal, A. Shankar, *House Price Prediction Using Multiple Linear Regression*, Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University, Noida, Uttar Pradesh, India. [Online]. Dostupno: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3833734 [Pristupano: 09.06.2025].
- [2] S. Abdulla, S. Jeberson, *A Comparative Study of Bagging, Boosting and C4.5: The Recent Improvements in Decision Tree Learning Algorithm*, [Online]. Dostupno: https://www.researchgate.net/publication/274692934_A_Comparative_Study_of_Bagging_Boosting_and_C45_The_Recent_Improvements_in_Decision_Tree_Learning_Algorithm [Pristupano: 09.06.2025].
- [3] S. Abdul Rahman, N. H. Zulkifley, U. N. Hasbiah i I. Ibrahim, "House Price Prediction using a Machine Learning Model: A Survey of Literature," *International Journal of Modern Education and Computer Science*, vol. 12, no. 6, pp. 46–54, Dec. 2020. Dostupno: https://www.researchgate.net/publication/347584803_House_Price_Prediction_using_a_Machine_Learning_Model_A_Survey_of_Literature [Pristupano: 09.06.2025].
- [4] Kaggle, *House Prices – Advanced Regression Techniques*. [Online]. Dostupno: <https://www.kaggle.com/c/house-prices-advanced-regression-techniques> [Pristupano: 10.06.2025].
- [5] Kaggle, *House Prices – Advanced Regression Techniques*. [Online]. Dostupno: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data?select=data_description.txt [Pristupano: 10.06.2025].
- [6] GeeksforGeeks, Principal Component Analysis (PCA), 29.05.2025. [Online]. Dostupno: <https://www.geeksforgeeks.org/principal-component-analysis-pca/> [Pristupano: 29.05.2025].
- [7] GeeksforGeeks, Interquartile Range (IQR) – GeeksforGeeks, 29.05.2025, [Online]. Dostupno: <https://www.geeksforgeeks.org/interquartile-range/> [Pristupano: 29.05.2025].
- [8] Yoav Freund and Robert E. Schapire, A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, 19.12.1996, [Online]. Dostupno: https://www.face-rec.org/algorithms/Boosting-Ensemble/decision-theoretic_generalization.pdf [Pristupano: 17.05.2025].
- [9] Leo Breiman, Bagging Predictors, Septembar 1994, [Online]. Dostupno: <https://www.stat.berkeley.edu/~breiman/bagging.pdf> [Pristupano: 17.05.2025].

- [10] Thomas G, Dietterich, Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms, 30.12.1997., [Online]. Dostupno: <https://sci2s.ugr.es/keel/pdf/algorithm/articulo/dietterich1998.pdf> [Pristupano: 17.05.2025].
- [11] Leo Breiman, RANDOM FORESTS - RANDOM FEATURES, Septembar 1999, [Online]. Dostupno: <https://www.stat.berkeley.edu/~breiman/bagging.pdf> [Pristupano: 17.05.2025].
- [12] XGBoost Documentation, XGBoost: Scalable and Flexible Gradient Boosting, 2023, [Online]. Dostupno: https://xgboost.readthedocs.io/en/release_3.0.0/ [Pristupano: 29.05.2025].
- [13] XGBoost Documentation, "Learning Task and Objective," [Online]. Dostupno: https://xgboost.readthedocs.io/en/release_3.0.0/tutorials/model.html [Pristupano: 29.05.2025].
- [14] H. Zhang, S. Wu, X. Yu, Z. Wang, "Ensemble machine learning for prediction with missing data," Department of Computer Science and Engineering, Michigan State University, East Lansing, USA. [Online]. Dostupno: <https://tjzhifei.github.io/links/EMFA.pdf> [Pristupano: 10.06.2025].
- [15] S. Sagi, L. Rokach, "Ensemble learning: A survey," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 8, no. 4, e1249, 2018. [Online]. Dostupno: <https://www.sci-hub.se/10.1002/widm.1249> [Pristupano: 10.06.2025].