Usporedba naprednih regresijskih metoda kod procjene cijena kuća

Šuća, Antonela

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Antonela Šuća

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori: Student:

doc. dr. sc. Tomislav Stipančić, dipl. ing. Antonela Šuća

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svom mentoru doc. dr. sc Tomislavu Stipančiću na pruženoj prilici i pomoći. Zahvaljujem mu na uloženom vremenu i ukazanom povjerenju koje mi je pružio tijekom izrade ovog rada.

Posebno se zahvaljujem svojoj obitelji i prijateljima koji su bili uz mene kroz svaki korak mog studija bodreći me da nastavim dalje i što su vjerovali u mene kada sam imala najmanje snage, strpljenja i volje. Također zahvaljujem kolegama studentima koji su mi na bilo koji način pomogli kroz godine studija i koji su teške muke studiranja olakšali pozitivnom energijom i smijehom.

Antonela Šuća

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove: proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 22 – 6 / 1**	
Ur.broj: **15 - 1703 - 22 -**	

ZAVRŠNI ZADATAK

Student: **Antonela Šuća** JMBAG: **0035218017**

Naslov rada na hrvatskom jeziku: **Usporedba naprednih regresijskih metoda kod procjene cijena kuća**

Naslov rada na engleskom jeziku: **Comparison of advanced regression methods in estimating house prices**

Opis zadatka:

U svijetu podataka i velike količine informacija algoritmi umjetne inteligencije postaju sve značajniji. U ovisnosti o vrsti i strukturi podataka metode strojnog učenja koriste se prilikom predviđanja, odlučivanja, klasifikacije, prepoznavanja, itd.
Koristeći različite regresijske metode primijenjene na prikladnom skupu podataka koji sadrži informacije o cijenama kuća na nekom području potrebno je izvršiti analizu te usporediti rezultate primijenjenih metoda.

U radu je potrebno:

- koristiti odgovarajući skup podataka koji sadrži više kategorija podataka koje definiraju njihove ovisnosti (skup podataka je potrebno podijeliti na podatke za trening i na podatke za testiranje),
- koristeći tehnike rudarenja podataka, podatke je potrebno pripremiti i prilagoditi za daljnju obradu,
- trenirati i evaluirati više modela za predviđanje cijena kuća u ovisnosti o korištenim regresijskim metodama,
- napraviti ilustracije rezultata rada različitih modela te napisati kritički osvrt.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan: 9. 5. 2022.

Zadatak zadao: Doc. dr. sc. Tomislav Stipančić

Datum predaje rada:
2. rok (izvanredni): 6. 7. 2022.
3. rok: 22. 9. 2022.

Predviđeni datumi obrane:
2. rok (izvanredni): 8. 7. 2022.
3. rok: 26. 9. – 30. 9. 2022.

Predsjednik Povjerenstva: Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS OZNAKA

train.csv	CSV datoteka s podacima kretanja cijena kuća „train“
test.csv	CSV datoteka s podacima kretanja cijena kuća „test“
SalePrice	Cijena nekretnine u dolarima
MSSubClass	Klasa izgradnje
MSZoning	Opća klasifikacija kuća po zonama
LotFrontage	Linearna povezanost ulica s imanjem u stopama
LotArea	Veličina parcele u kvadratnim stopama
Street	Vrsta kolnog prilaza
Alley	Vrsta prilaza uličici
LotShape	Opći oblik nekretnine
LandContour	Ravnost posjeda
Utilities	Vrsta dostupnih komunalnih usluga
LotConfig	Konfiguracija parcele
LandSlope	Nagib imanja
Neighborhood	Fizičke lokacije unutar granica grada Amesa
Condition1	Blizina glavne ceste ili željezničke pruge
Condition2	Blizina glavne ceste ili željezničke pruge (ako postoji druga)
BldgType	Vrsta nastambe
HouseStyle	Stil nastambe
OverallQual	Ukupna kvaliteta materijala i završne obrade
OverallCond	Ukupna ocjena stanja
YearBuilt	Izvorni datum izgradnje
YearRemodAdd	Datum preuređenja
RoofStyle	Vrsta krova
RoofMatl	Krovni materijal
Exterior1st	Vanjski pokrov na kući
Exterior2nd	Vanjska obloga na kući (ako je krov načinje od više od jednog materijala)
MasVnrType	Tip zidanog furnira

MasVnrArea	Površina zidanog furnira u kvadratnim stopama
ExterQual	Kvaliteta vanjskog materijala
ExterCond	Trenutno stanje materijala na vanjskoj strani
Foundation	Vrsta temelja
BsmtQual	Visina podruma
BsmtCond	Opće stanje podruma
BsmtExposure	Izlaz ili podrumski zidovi na razini vrta
BsmtFinType1	Kvaliteta gotove površine podruma
BsmtFinSF1	Tip 1 gotovog podruma u kvadratnim metrima
BsmtFinType2	Kvaliteta drugog završenog područja podruma (ako postoji)
BsmtFinSF2	Tip 2 gotovog podruma u kvadratnim metrima
BsmtUnfSF	Nedovršeni kvadratni metri podrumske površine
TotalBsmtSF	Ukupni kvadratni metar površine podruma
Heating	Vrsta grijanja
HeatingQC	Kvaliteta i stanje grijanja
CentralAir	Centralni klima uređaj
Electrical	Električni sustav
1stFlrSF	Prvi kat u četvornim stopama
2ndFlrSF	Drugi kat u četvornim stopama
LowQualFinSF	Dovršena površina niske kvalitete u četvornim stopama (svi katovi)
GrLivArea	Stambeni prostor iznad razine tla u kvadratnim metrima
BsmtFullBath	Kupaonice u podrumu
BsmtHalfBath	Polukupaonice u podrumu
FullBath	Kupaonice u kući
HalfBath	Polukupaonice u kući
Bedroom	Broj spavaćih soba iznad razine podruma
Kitchen	Broj kuhinja
KitchenQual	Kvaliteta kuhinje
TotRmsAbvGrd	Ukupan broj soba iznad razine tla (ne uključuje kupaonice)

Functional	Ocjena funkcionalnosti doma
Fireplaces	Broj kamina
FireplaceQu	Kvaliteta kamina
GarageType	Lokacija garaže
GarageYrBlt	Godina izgradnje garaže
GarageFinish	Unutarnja završna obrada garaže
GarageCars	Veličina garaže prema kapacitetu automobila
GarageArea	Veličina garaže u kvadratnim stopama
GarageQual	Garažna kvaliteta
GarageCond	Stanje garaže
PavedDrive	Asfaltirani prilaz
WoodDeckSF	Površina drvene terase u kvadratnim stopama
OpenPorchSF	Otvoreni trijem u kvadratnim stopama
EnclosedPorch	Zatvorena površina trijema u kvadratnim stopama
3SsnPorch	Trijem za 3 godišnja doba (osim zime) u kvadratnim stopama
ScreenPorch	Površina trijema s ekranom u kvadratnim stopama
PoolArea	Površina bazena u kvadratnim stopama
PoolQC	Kvaliteta bazena
Fence	Kvaliteta ograde
MiscFeature	Razne značajke koje nisu obuhvaćene drugim kategorijama
MiscVal	Vrijednost raznih neobuhvaćenih značajki u dolarima
MoSold	Mjesec prodaje kuće
YrSold	Godina prodaje kuće
SaleType	Vrsta prodaje
SaleCondition	Uvjet prodaje
train1	Kopija CSV datoteke „train“
garage_cols	DataFrame svih karakteristika garaža
bsmt_cols	DataFrame svih karakteristika podruma
mas_cols	DataFrame svih karakteristika zidnog furnira
Total_House_SF	Ukupna površina kuće u četvornim stopama

Total_Home_Quality	Ukupna kvaliteta kuće
Total_Bathrooms	Ukupan broj kupaonica
Skew_cols	DataFrame kategorija koje sadrže iskrivljene značajke
Alpha	Parametar alfa kod Lasso i Ridge regresije
RMSE	Srednja kvadratna pogreška

$J(\boldsymbol{w})$	funkcija srednje kvadratne pogreške ovisna o w
$\boldsymbol{w}$	optimalna težina
x, y	ulazni podaci na „x-osi i y-osi
n	ukupan broj podataka koje unosimo

SAŽETAK

Tema ovog rada je procjena kretanja cijena kuća usporedbom naprednih regresijskih metoda. Za provedbu zadatka, koristit će se programski jezik Python. Za izvedbu cijelog koda potrebno je skinuti Python biblioteke Pandas, Numpy, Scikit i SciPy. Vrijednosti koje se koriste kroz rad su preuzete iz gotovih CSV datoteka za obilježja kuća grada Amesa. Podaci iz CSV datoteke će se najprije detaljno obraditi i očisti prije primjene regresijskih modela. Za modeliranje i uspoređivanje dobivenih vrijednosti podataka koristit će se Lasso regresija, Ridge regresija i regresija potpornih vektora. Za interpretaciju važnosti značajki će zbog svoje jednostavnosti biti upotrijebljena samo Lasso regresija.

Ključne riječi: Python, CSV datoteka, podaci, regresija

SUMMARY

The topic of this paper is the estimation of house price fluctuation by comparing advanced regression methods. Python programming language will be used to implement the task. To run the entire code, it is needed to download the Pandas, Numpy, Scikit and SciPy Python libraries. The values used throughout the paper are taken from ready-made CSV files for the characteristics of the houses of the city of Ames. The data from the CSV file will first be thoroughly processed and cleaned before applying the regression models. Lasso regression, Ridge regression and Support vector regression will be used to model and compare the obtained data values. Due to its simplicity, only Lasso regression will be used to interpret the importance of features.

Keywords: Python, CSV file, data, regression

1. Uvod

Kada bi se stranca na ulici pitalo kako bi izgledala njegova idealna kuća, zasigurno ne bi započeo s nekom nasumičnom i isprva naizgled nebitnom karakteristikom kao što je na primjer visina podruma. Razlog tomu je što je čovjek biće sa sviješću koje ima sposobnost vizualnog zapažanja i pri kupnji kuće će presuditi omjer vizualnog dojma i pristupačnosti cijene. S druge strane, računalo ne vidi ima li kuća kuhinju kakvu je baš nečija žena zamislila, nego „vidi“ samo brojeve. Ta činjenica donosi veliku prednost računala jer je ono u stanju koristiti algoritam koji će procijeniti je li željena kuća uistinu isplativa s obzirom na njezinu cijenu uzevši u obzir sve željene značajke.

2. Python biblioteke

2.1. Pandas

Pandas je softverska biblioteka za programski jezik Python. Koristi se za uređivanje i analizu podataka. Nudi strukture podataka i operacije za manipuliranje numeričkim tablicama i vremenskim serijama. Naziv pandas je izvedenica od izraza "panel data" što je zapravo skup podataka u kojima se ponašanje entiteta promatra kroz vrijeme.

Pandas se uglavnom koristi za analizu podataka i povezanu manipulaciju tabličnim podacima u DataFrame-ovima. DataFrame je podatkovna struktura koja organizira podatke u dvodimenzionalnu tablicu redaka i stupaca, slično proračunskoj tablici. DataFrame-ovi su jedna od najčešćih struktura podataka koje se koriste u modernoj analitici podataka jer predstavljaju fleksibilan i intuitivan način pohrane i rada s podacima.[2] Pandas omogućuje uvoz podataka iz različitih formata datoteka kao što su vrijednosti odvojene zarezom, JSON, Parquet, SQL tablice ili upiti baze podataka i Microsoft Excel. Pandas dopušta različite operacije za manipulaciju podacima kao što su spajanje, preoblikovanje, odabir kao i čišćenje podataka. Razvoj pandas-a uveo je u Python mnoge usporedne značajke rada s DataFrame-ovima koji su uspostavljeni u R programskom jeziku. Biblioteka Pandas izgrađena je na drugoj biblioteci NumPy koja je orijentirana na učinkovit rad s nizovima umjesto značajki rada na DataFrame-ovima. [3]

```
import pandas as pd
```

Slika 1. Naredba za implementiranje biblioteke Pandas u program Python

Na slici je prikazan kod kojim će se implementirati biblioteka Pandas u Python.

2.2. Numpy

NumPy je biblioteka za programski jezik Python koja podržava velike, višedimenzionalne nizove i matrice uz rad s velikom zbirkom matematičkih funkcija visoke razine za rad s tim nizovima. Primjena NumPy-ja u Pythonu može se usporediti s MATLAB-om jer oba omogućuju korisniku pisanje brzih programa na način da se većina operacija izvodi u obliku nizova ili matrica umjesto skalara. Međutim, MATLAB se može pohvaliti velikim brojem

dodatnih alatnih okvira, posebice Simulinkom, dok je NumPy intrinzično integriran s Pythonom, modernijim i potpunijim programskim jezikom. Štoviše, dostupni su komplementarni Python paketi; SciPy je biblioteka koja dodaje više funkcionalnosti slične MATLAB-u, a Matplotlib je paket za crtanje koji pruža funkcionalnost crtanja sličnu MATLAB-u te će se koristiti u ovome radu. Uz Matplotlib, koristit će se i biblioteka Seaborn. Seaborn je Python biblioteka za vizualizaciju podataka temeljena na Matplotlibu. Omogućuje sučelje visoke kvalitete za crtanje atraktivne i informativne statističke grafike. Interno se i MATLAB i NumPy oslanjaju na BLAS i LAPACK za učinkovita izračunavanja linearne algebre. [4]

```
import numpy as np
```

Slika 2. Naredba za implementiranje biblioteke Numpy u program Python

Na slici je prikazan kod kojim će se implementirati biblioteka NumPy u Python.

2.3. Scikit-learn

Scikit-learn (Sklearn) je najkorisnija i najsnažnija biblioteka za strojno učenje u Pythonu. Pruža širok izbor učinkovitih alata za strojno učenje i statističko modeliranje uključujući klasifikaciju, regresiju, grupiranje i smanjenje dimenzionalnosti putem konzistentnog sučelja u Pythonu. Njeno implementiranje će uvelike pomoći za prikaz usporedbe naprednih regresijskih metoda. [5]

2.4. SciPy

SciPy je znanstvena računalna biblioteka koja koristi NumPy kao bazu. Njeno ime je kratica za Scientific Python. Omogućuje više korisnih funkcija za optimizaciju, statistiku i obradu signala.Osnovna struktura podataka koju koristi SciPy je višedimenzionalno polje koji pruža NumPy modul.

3. Unos podataka o kriterijima za izbor kuća

Iako se ovakve baze podataka mogu kreirati vlastoručno, u ovome radu koristit će se stvarnom gotovom bazom podataka koja je pohranjena u obliku takozvanog CSV filea čiji je puni naziv na engleskom – *Comma-Separated Values*, što bi u prijevodu značilo „Zarezom odvojene vrijednosti“. Takav naziv nosi upravo iz razloga što je riječ o vrsti dokumenta u kojem su informacije odvojene zarezima. Međutim, iako nosi takav naziv, umjesto zarezima informacije se mogu odvajati razmakom, točka-zarezom ili nekim drugim interpunkcijskim znakom, ali je zarez nekako najčešće u uporabi. Ovakvu vrstu pohrane čini jedinstvenom je činjenica da su informacije jako dobro organizirane te ih je moguće otvoriti u različitim aplikacijama zbog fleksibilnosti formata. [6]

```
train = pd.read_csv('C:\\Users\\anton\\Desktop\\house-prices-advanced-regression-techniques\\train.csv')
test = pd.read_csv('C:\\Users\\anton\\Desktop\\house-prices-advanced-regression-techniques\\test.csv')
train.head()
   Id  MSSubClass MSZoning  ...  SaleType  SaleCondition SalePrice
0   1          60       RL  ...        WD         Normal    208500
1   2          20       RL  ...        WD         Normal    181500
2   3          60       RL  ...        WD         Normal    223500
3   4          70       RL  ...        WD        Abnorml    140000
4   5          60       RL  ...        WD         Normal    250000

[5 rows x 81 columns]
```

Slika 3. Implementiranje i ispisivanje preuzetog CSV filea

Kako je vidljivo na slici, za očitavanje vrijednosti DataFrame-a koristit će se pd.read_csv() naredbom. Budući da će skinuto polje podataka biti preveliko jer sadrži preko tisuću redaka, radi preglednosti koristit ćemo se naredbom head() koja će nam pokazati podatke za samo prvih 5 redova. S obzirom da polje ima 81 stupac, neće se svi stupci pregledno ispisati, ali je bitno da program može očitati i zapamtiti sve podatke upisane u CSV fileu koji se odnose na podatke za brojne kriterije kod biranja kuća pri njihovoj kupnji.

4. Istraživačka analiza podataka

Istraživačka analiza podataka (eng. *EDA = Exploratory Data Analysis*) odnosi se na kritički proces izvođenja početnih istraživanja podataka kako bi se otkrili obrasci, uočile anomalije, testirale hipoteze i provjerile pretpostavke uz pomoć sažete statistike i grafičkih prikaza. Prvobitno je razvijena od strane američkog matematičara Johna Tukeyja 1970-ih godina. Ova metoda je danas često upotrebljavana među znanstvenicima, a primarno se koristi kako bi se vidjelo što podaci mogu otkriti izvan zadatka formalnog modeliranja ili testiranja hipoteza i pruža bolje razumijevanje varijabli skupa podataka i odnosa između njih. Također, može pomoći u određivanju jesu li statističke tehnike koje razmatramo prikladne za analizu podataka. Glavna svrha EDA-e je pomoć pri pregledu podataka prije nego li donesemo bilo kakvu pretpostavku. Ona nam može pomoći u identificiranju očitih grešaka kao i u boljem razumijevanju obrazaca koji se nalaze unutar podataka, otkrivanju izvanrednih vrijednosti ili nenormalnih događaja te pronalaženju zanimljivih odnosa među varijablama. Znanstvenici koji se bave podacima mogu koristiti istraživačku analizu kako bi osigurali da su rezultati valjani i primjenjivi na sve željene poslovne rezultate i ciljeve. EDA također može pomoći odgovoriti na pitanja o standardnim odstupanjima, kategoričkim varijablama i intervalima pouzdanosti. Nakon što je EDA proces dovršen i dobiju se uvidi, njegove se značajke mogu koristiti za sofisticiraniju analizu podataka ili modeliranje, uključujući strojno učenje. [7]
Postoje četiri osnovne vrste prikazivanja istraživačke analize podataka:

1. **Jednovarijantni negrafički**

Ovo je najjednostavniji oblik analize podataka iz razloga što se podaci koji se analiziraju sastoje od samo jedne varijable. Budući da se radi o jednoj varijabli, ne bavi se uzrocima i međusobnim odnosima. Glavna svrha ovakve analize je opisati podatke i pronaći obrasce koji postoje unutar njih.

2. **Jednovarijantni grafički**

Negrafičke metode, kao što im i samo ime kaže, ne daju potpunu sliku podataka. Stoga su radi boljeg shvaćanja i vizualnog razmatranja potrebne grafičke metode.

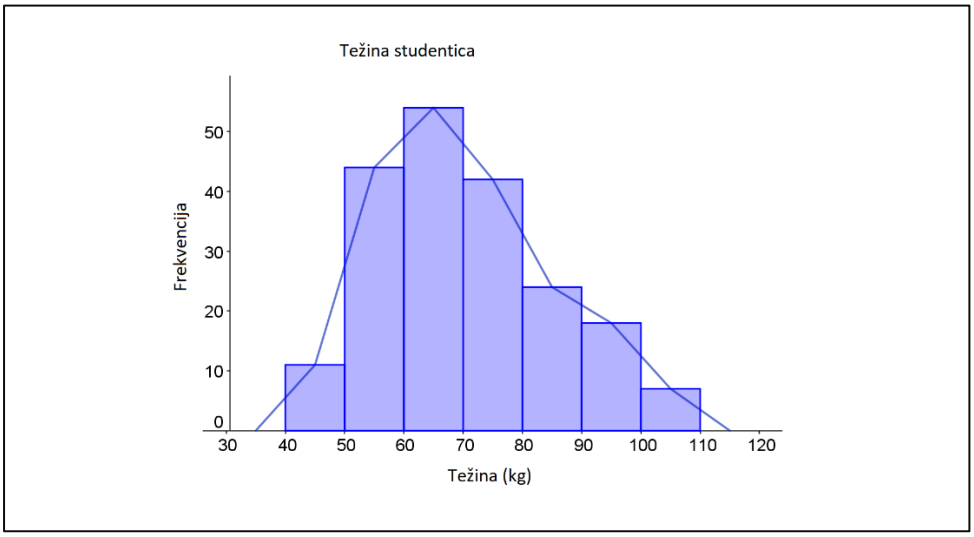
Ova vrsta metode može se prikazati na raznolike grafičke načine putem:

• Dijagrami stabljike i lišća (eng. *stem-and-leaf plots*) – vrsta dijagrama koji pokazuju sve vrijednosti podataka kao i oblik distribucije.

Class A		Class B
Leaves	**Stem**	**Leaves**
8 0	6	0 0
5 0	7	0 1 3 3 5 6 7
6 4	8	4 5 6
6 4 4 2 1 0	9	1 2
0 0	10	

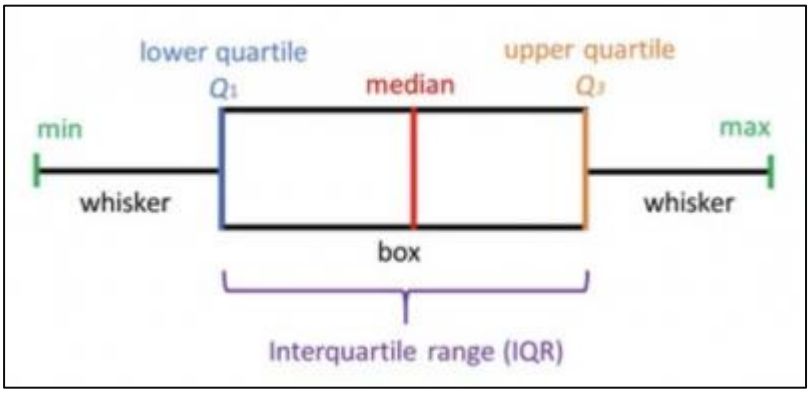
Slika 4. Primjer dijagrama „stabljike i lišća“

• Histogrami – stupčasti dijagram u kojem svaki stupac predstavlja brojčanu učestalost ili udio slučajeva za raspon vrijednosti.



Slika 5. Primjer histograma

• Okvirni dijagrami (eng. *box plot*) – vrsta dijagrama koji grafički prikazuju sažetak od pet brojeva – minimuma, prvog kvartila, medijana, trećeg kvartila i maksimuma.



Slika 6. Oblik okvirnog dijagrama

3. Multivarijantni negrafički

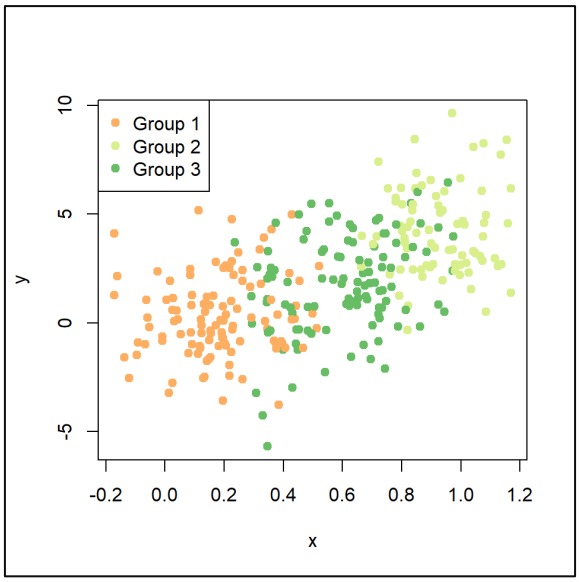
Multivarijantni podaci proizlaze iz više od jedne varijable. Multivarijatne negrafičke EDA tehnike općenito pokazuju odnos između dvije ili više varijabli podataka kroz unakrsnu tablicu ili statistiku.

4. Multivarijantni grafički

Multivarijantni podaci koriste grafičke prikaze odnosa među dva ili više skupova podataka. Grafički prikaz koji se najčešće koristi je grupirani stupčasti dijagram ili stupčasti dijagram gdje svaka grupa predstavlja jednu razinu jedne od varijabli, a svaka traka unutar grupe predstavlja razine druge varijable.

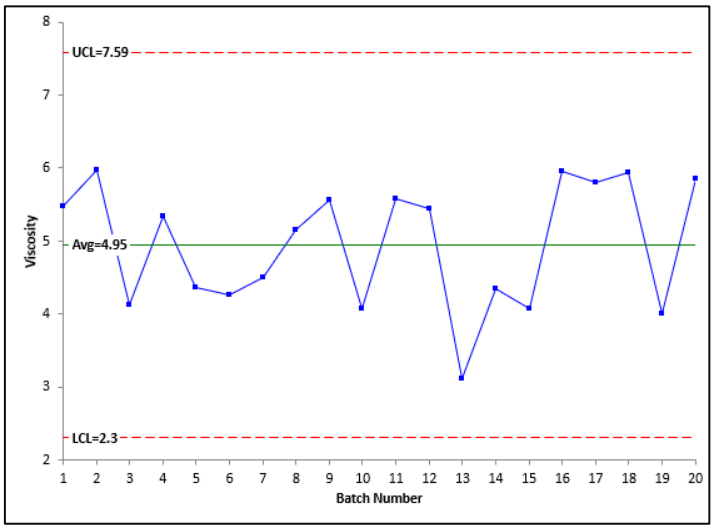
Multivarijantni grafički dijagrami uključuju:

• Dijagram raspršenosti (eng. *scatter plot*) koji se koristi za iscrtavanje podatkovnih točaka na vodoravnoj i okomitoj osi kako bi se pokazalo koliko jedna varijabla utječe na drugu.



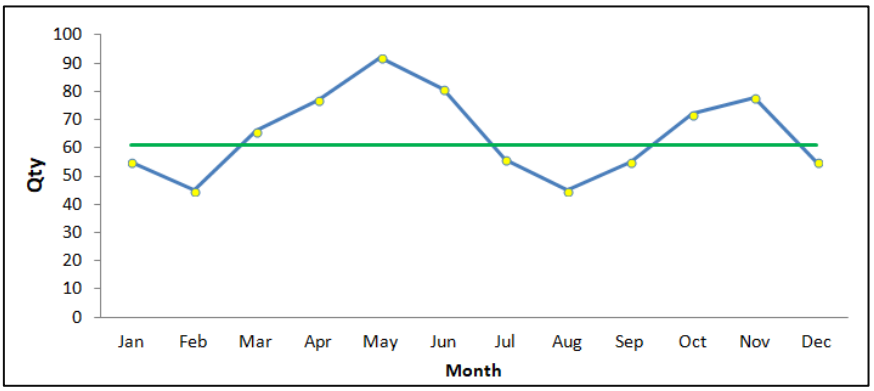
Slika 7. Primjer dijagrama raspršenosti

• Multivarijantni grafikon, koji je grafički prikaz odnosa između faktora i odgovora.



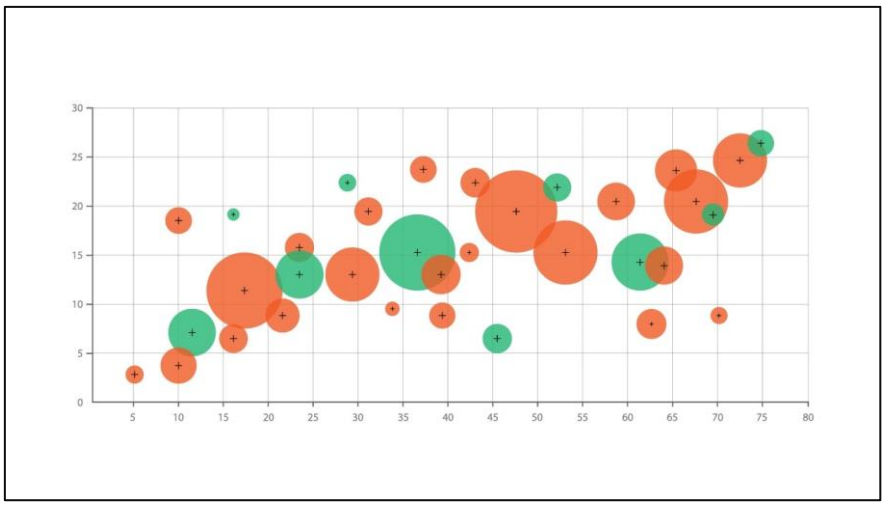
Slika 8. Primjer multivarijantnog grafikona

• Grafikon pokreta (eng. *run chart*), koji je linijski grafikon podataka iscrtanih tijekom vremena.



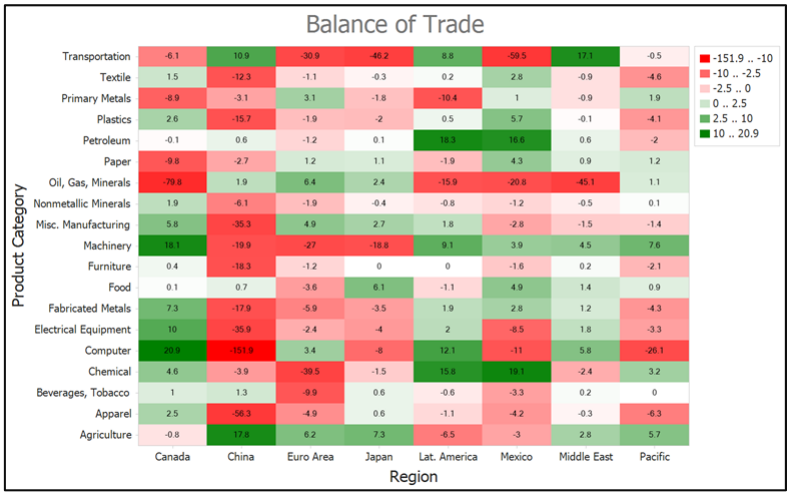
Slika 9. Primjer grafikona pokreta

• Mjehurićasti grafikon (eng. *bubble chart*), koji je vizualizacija podataka koja prikazuje više krugova (mjehurića) u dvodimenzionalnom dijagramu.



Slika 10. Primjer mjehurićastog grafikona

• Toplinska karta, koja je grafički prikaz podataka gdje su vrijednosti prikazane bojom.



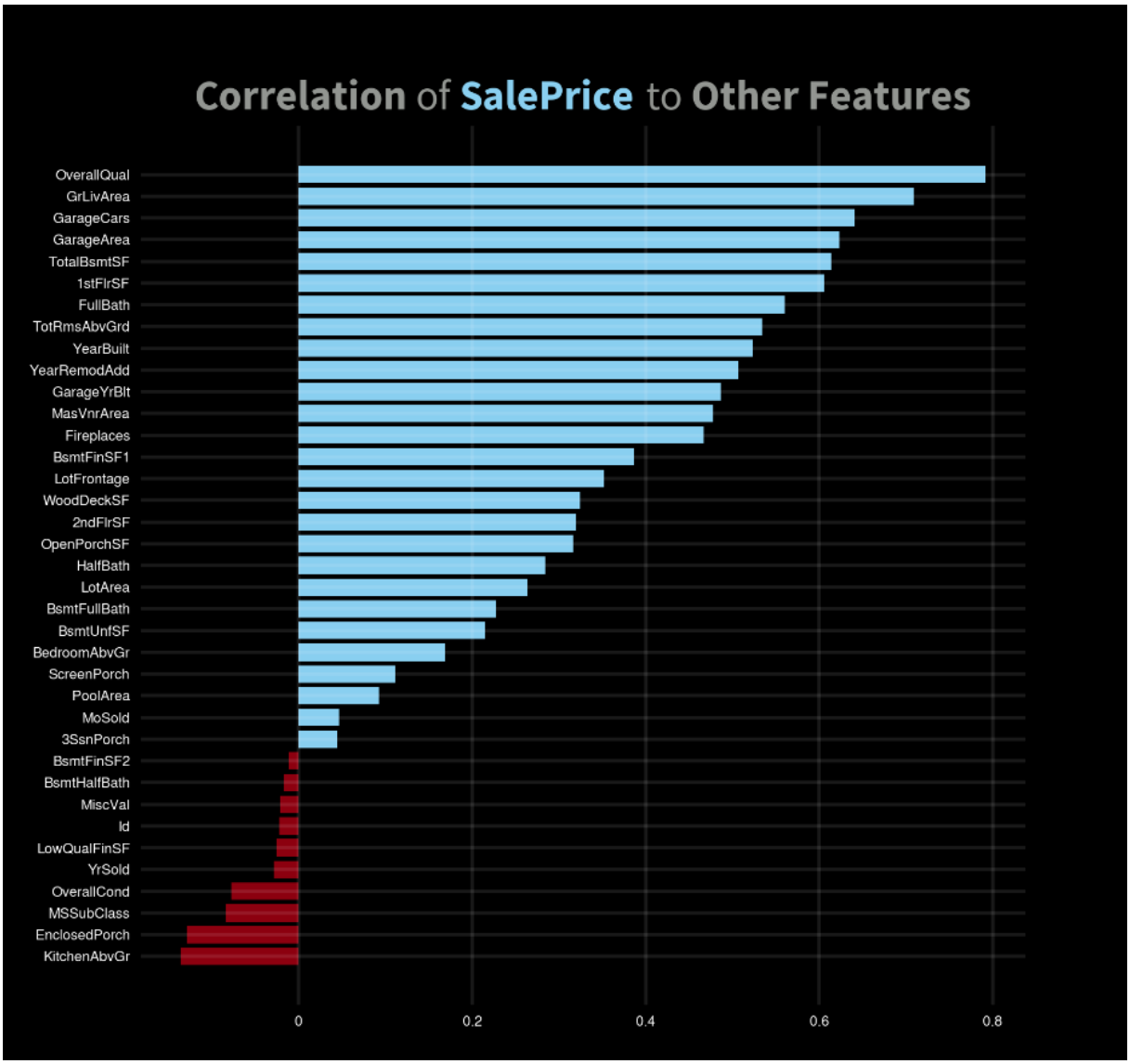
Slika 11. Primjer toplinske karte

Kako bi se lakše analizirali podaci za kretanje cijena kuća, najbolje je napraviti grafikon koji će korisniku lakše vizualno dočarati na koje kriterije treba obratiti pažnju kod izbora kuće. Na sljedećoj slici je prikaz koda za crtanje dijagrama u Pythonu korištenjem Matplotliba.

```
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
import proplot as pplt
def plots_design():
    fig.patch.set_facecolor('black')
    ax.patch.set_facecolor('black')
...     ax.tick_params(axis='both', which='major', labelsize=8)
...     ax.yaxis.set_label_coords(0, 0)
...     ax.grid(color='white', linewidth=2)
...     ax.xaxis.set_ticks_position('none')
...     ax.yaxis.set_ticks_position('none')
...     for i in ['top', 'bottom', 'left', 'right']:
...         ax.spines[i].set_visible(False)
...     ax.tick_params(axis='x', colors='white')
...     ax.tick_params(axis='y', colors='white')
...     mpl.rcParams['font.family'] = 'Source Sans Pro'
...
...
>>> corr = train[train.columns].corr()['SalePrice'][:].sort_values(ascending=True).to_frame()
>>> corr = corr.drop(corr[corr.SalePrice > 0.99].index)
>>> fig, ax = plt.subplots(figsize =(9, 9))
>>> ax.barh(corr.index, corr.SalePrice, align='center', color = np.where(corr['SalePrice'] < 0, 'crimson', '#89CFF0'))
<BarContainer object of 37 artists>
>>> plots_design()
>>> plt.text(-0.12, 39, "Correlation", size=24, color="grey", fontweight="bold");
Text(-0.12, 39, 'Correlation')
>>> plt.text(0.135, 39, "of", size=24, color="grey");
Text(0.135, 39, 'of')
>>> plt.text(0.185, 39, "SalePrice", size=24, color="#89CFF0", fontweight="bold");
Text(0.185, 39, 'SalePrice')
>>> plt.text(0.4, 39, "to", size=24, color="grey");
Text(0.4, 39, 'to')
>>> plt.text(0.452, 39, "Other Features", size=24, color="grey", fontweight="bold");
Text(0.452, 39, 'Other Features')
>>> plt.text(0.9, -7, "Antonela Suca", fontsize=11, ha="right", color='grey');
Text(0.9, -7, 'Antonela Suca')
>>> top_corr = corr['SalePrice'].sort_values(ascending=False).head(10).index
>>> top_corr = top_corr.union(['SalePrice'])
...
>>> sns.pairplot(train[top_corr]);
<seaborn.axisgrid.PairGrid object at 0x000001AC2032F010>
>>> plt.show()
```

Slika 12. Kod za crtanje dijagrama korelacije cijene kuće i željenih značajki

Nakon upisivanja navedenog koda, program crta sljedeći dijagram.



Slika 13. Dijagram korelacije cijene kuće i ostalih značajki

Na ovom dijagramu vidljiva je korelacija prodajne cijene s ostalim numeričkim značajkama. Plavom bojom su označene pozitivne korelacije cijene kuće sa svakom od željenih značajki, a crvenom bojom su označene negativne. Kao što je vidljivo najveće pozitivne korelacije su:

• OverallQual: Ukupna kvaliteta materijala i završne obrade

• GrLivArea: Stambeni prostor iznad razine tla u kvadratnim stopama

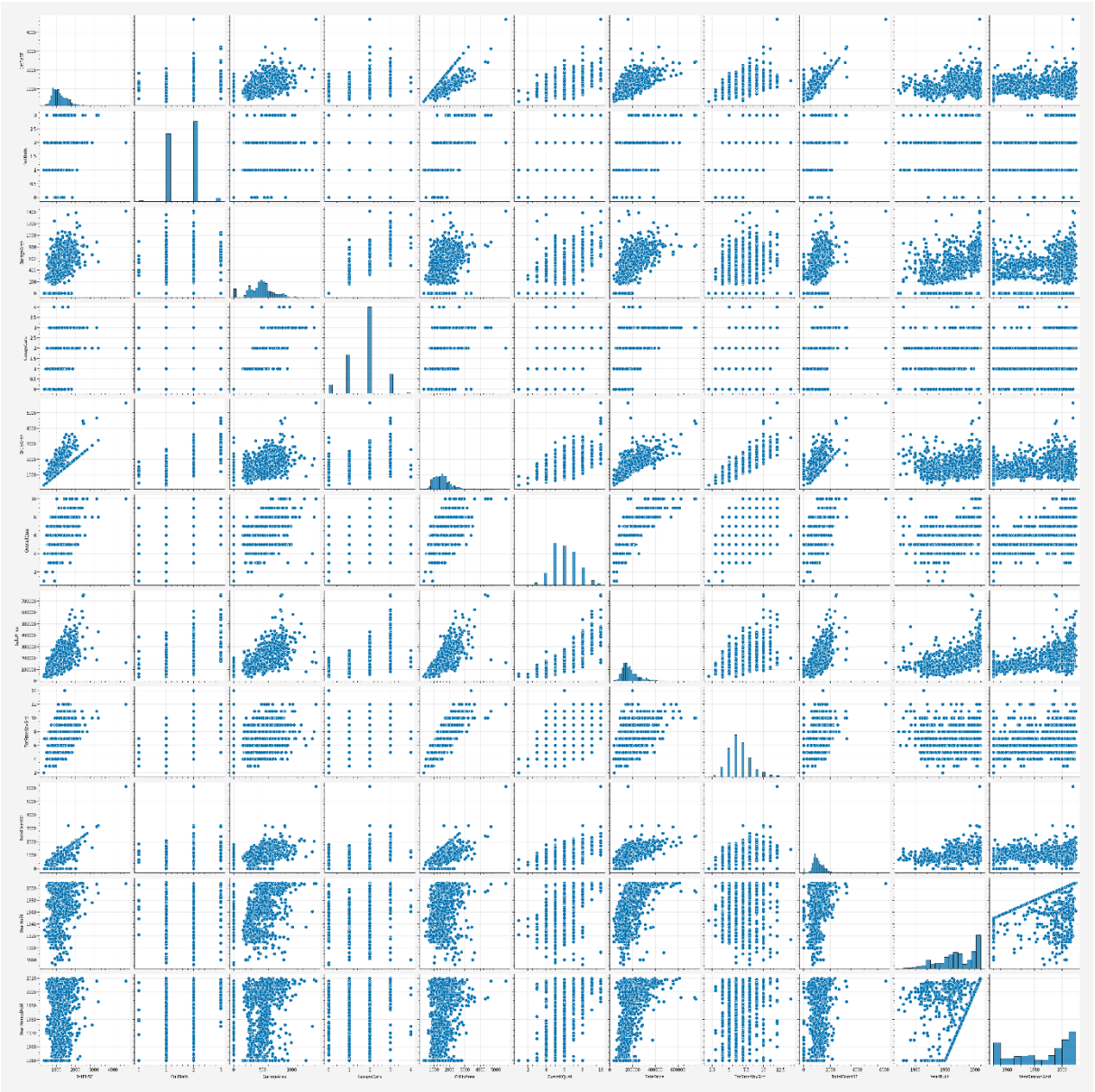
• GarageCars: Veličina garaže prema kapacitetu automobila

• GarageArea: Veličina garaže u kvadratnim stopama

• TotalBsmtSF: Ukupni kvadratni metar površine podruma

• 1stFlrSF: Prvi kat četvornih stopa

Prema zadanim postavkama, naredba sns.pairplot() biblioteke Seaborn će stvoriti mrežu osi tako da će se svaka numerička varijabla u podacima dijeliti preko y-osi u jednom retku i x-osi u jednom stupcu. Najjednostavniji prikaz vrši se na način da koristi dijagram raspršenosti za svako uparivanje različitih varijabli iz različitih redaka i stupaca, dok uspoređivanje iste varijable odnosno varijable koje se poklapaju na mjestu matrice istog retka i stupca se prikazuje histogramom. To znači kada se to cijelo polje podataka prikaže grafički, po dijagonali će biti histogrami, a na ostalim mjestima će biti dijagrami raspršenosti.



Slika 14. Prikaz korelacije cijene kuće i ostalih značajki putem sistema matričnog poklapanja

5. Čišćenje i obrada podataka

5.1. Obrada podataka (eng. *Data processing*)

Obrada podataka je definirana kao prikupljanje, manipulacija i obrada prikupljenih podataka za upotrebu. Glavna svrha ovog procesa je pretvaranje podataka iz početnog oblika u puno upotrebljiviji oblik čineći ga smislenijim i informativnijim. Korištenjem algoritama strojnog učenja, matematičkog modeliranja i statističkog znanja cijeli se ovaj proces može automatizirati. Proces obrade podataka se izvodi u šest faza prema sljedećem redoslijedu.

1. Prikupljanje podataka

Prikupljanje podataka je prvi korak prilikom obrade podataka. Podaci se izvlače iz dostupnih izvora za koje je važno da su dostupni, pouzdani i dobro izgrađeni kako bi prikupljeni podaci koji će kasnije biti korišteni kao konkretne točne informacije bili najviše moguće kvalitete.

2. Priprema podataka

Nakon što su podaci prikupljeni u prvoj fazi, oni tada ulaze u sljedeću fazu pripreme podataka. Priprema podataka je faza u kojoj se neobrađeni podaci čiste i organiziraju za sljedeću fazu. Tijekom pripreme, neobrađeni podaci se temeljito provjeravaju radi eventualnih pogrešaka. Svrha ovog koraka je eliminirati loše podatke (suvišne, nepotpune ili netočne podatke) i početi stvarati visokokvalitetne podatke.

3. Unos podataka

Očišćeni podaci se zatim unose na svoje odredište i prevode na jezik koji mogu razumjeti. Unos podataka prva je faza u kojoj neobrađeni podaci počinju poprimati oblik upotrebljivih informacija.

4. Obrada

Tijekom faze obrade, podaci uneseni u računalo u prethodnoj fazi, obrađuju se za tumačenje. Obrada se vrši pomoću algoritama strojnog učenja, iako se sam postupak može neznatno razlikovati ovisno o izvoru podataka koji se obrađuju i njihovoj namjeni.

5. Izlaz/tumačenje podataka

Faza izlaza/tumačenja je faza u kojoj su podaci konačno upotrebljivi ljudima koji se smatraju laicima za ovakva područja obrade podataka. Podaci su prevedeni, čitljivi i često u obliku grafikona, videa, slika, običnog teksta itd.

6. Pohrana podataka

Završna faza obrade podataka je pohrana. Nakon što se svi podaci obrade, pohranjuju se za buduću upotrebu. Dok se neke informacije mogu odmah upotrijebiti, većina će kasnije poslužiti svrsi. Ako su podaci ispravno pohranjeni, članovi organizacije mogu im brzo i lako pristupiti kada god im je to potrebno. [8]

5.2. Čišćenje podataka (eng. *Data Cleaning*)

Čišćenje podataka je postupak popravljanja ili uklanjanja netočnih, oštećenih, nepravilno formatiranih, dupliciranih ili nepotpunih podataka unutar njihovog skupa. To je jedan od važnih dijelova strojnog učenja koji igra značajnu ulogu u izgradnji modela. Čišćenje podataka se odvija u sljedeće četiri faze.

1. Uklanjanje dvostrukih ili nevažnih opažanja

Dupliciranе datoteke su česta pojava kad je riječ o fazi prikupljanja podataka. Prilikom kombiniranja skupova podataka s više različitih lokacija, uvijek postoji mogućnost za nenamjerno stvaranje duplih podataka. Dvostruke datoteke je logično potrebno ukloniti pa se u ovoj fazi vrši deduplikacja.

Irelevantna odnosno nebitna opažanja su lako uočljiva jer je riječ o podacima koji se ne uklapaju u konkretan problem koji pokušavamo analizirati. Njihovo uklanjanje može učiniti analizu učinkovitijom i značajno smanjiti smetnje pri nalaženju rješenja primarnog problema kao i poticanje stvaranja skupa podataka kojim se lakše upravlja i koji ima veću učinkovitost.

2. Popravljanje strukturnih pogrešaka

Strukturne pogreške su lako uočljive jer je riječ o greškama koje se javljaju kod primjetnih čudnih konvencija imenovanja odnosno to su pogreške koje nastaju pri pogrešnom upisivanju nekih riječi ili netočnoj upotrebi velikih i malih slova. Ove nedosljednosti mogu uzrokovati pogrešno označene kategorije ili klase.

3. Filtriranje neželjenih ekstrema (eng. *outliers*)

Često se javljaju jednokratna opažanja za koja se na prvi pogled vidi da se ne uklapaju u podatke koji se analiziraju jer se nalaze u obliku ekstremnih izvanrednih vrijednosti. Iz tog razloga je potrebno utvrđivanje valjanosti iznosa takvih vrijednosti. Ako se taj izuzetan iznos pokaže irelevantnim za analizu ili je pogreška, ponekad ga je najbolje ukloniti.

4. Obrada podataka koji nedostaju

Podatke koji nedostaju nije pametno ignorirati jer mnogi algoritmi ne mogu prihvatiti vrijednosti koje nedostaju i javljaju grešku. Postoji nekoliko načina rješavanja podataka koji nedostaju, ali ni jedno nije potpuno učinkovito.

Kao prva opcija, opažanja kojima nedostaju vrijednosti mogu se ukloniti, ali time će se ispustiti ili izgubiti informacije, stoga to treba imati na umu ovo prije njihovog uklanjanja.

Kao druga opcija, vrijednosti koje nedostaju mogu se naknadno unijeti na temelju drugih opažanja, ali s tim postupkom i dalje postoji mogućnost da se izgubi cjelovitost podataka jer se možda radi na temelju pretpostavki, a ne stvarnih opažanja.

Kao treća opcija, moguće je preinačiti način na koji se podaci koriste za učinkovito kretanje nultim vrijednostima. [8]

5.3. Primjena obrade podataka i čišćenja

Da bi se provjerilo kojom veličinom polja se raspolaže, koristi se naredba shape().

```
print('Training Shape:', train.shape)
Training Shape: (1460, 81)
print('Test Shape:', test.shape)
Test Shape: (1459, 80)
```

Slika 15. Provjera veličine polja podataka

Kao što je vidljivo sa slike, CSV datoteka pod nazivom „train“ sadrži 1460 redaka i 81 stupac dok CSV datoteka pod nazivom „test“ sadrži 1459 redaka i 80 stupaca.

5.3.1. Ekstremne vrijednosti (eng. *outliers*)

Ako se skrene pozornost na ciljanu varijablu – SalePrice (prodajna cijena), vidljivo je da postoje neki ekstremne vrijednosti (eng. *outliers*) u značajkama kao što su GarageArea, GrLivArea i TotalBsmtSF koje znatno odskaču od ostalih vrijednosti stoga ih je radi bolje provedbe regresije (koja će se provesti kasnije) potrebno ukloniti.

Upotrebom Python naredbe copy() izrađuje se kopija već postojećeg popisa „train“ i proizvoljno mu se dodjeljuje naziv „train1“. Naredba copy() dodaje se na kraj objekta popisa te

se stoga ne prihvaćaju novi parametri što daje sigurnost da će nova lista biti identična originalnoj.

Svaku od navedenih značajki će se usporediti s ciljanom varijablom u obliku uvjeta s određenim brojčanim iznosima i kao povratna poruka se očekuje vrijednost nekog brojčanog iznosa zbog čega je na kraj svakog uvjeta potrebno staviti naredbu index().

```
train1 = train.copy()
train1 = train1.drop(train1[(train1['GarageArea']>1200) & (train1['SalePrice']<300000)].index)
train1 = train1.drop(train1[(train1['GrLivArea']>4000) & (train1['SalePrice']<300000)].index)
train1 = train1.drop(train1[(train1['TotalBsmtSF']>5000)].index)
print('Outliers removed =' , train.shape[0] - train1.shape[0])
Outliers removed = 5
```

Slika 16. Uklanjanje ekstremnih vrijednosti

Kako je prikazano kodom na slici, nakon što se postavljeni uvjeti provedu, program javlja da je uklonjeno 5 izuzetnih odnosno ekstremnih značajki koje se odbacuju.

5.3.2. Razdvajanje vrijednosti redaka (x) i stupaca (y)

Prije nego li se povede proces čišćenja podataka, potrebno je spojiti datoteke „train“ i „test“ čemu prethodi razdvajanje vrijednosti redaka od vrijednosti stupaca. Cijeli proces prikazan je kodom na sljedećoj slici.

```
X = train1.drop('SalePrice', axis=1)
y = train1['SalePrice'].to_frame()
X['train'] = 1
test['train'] = 0
df = pd.concat([test, X])
print('Count of Features per Data Type:')
Count of Features per Data Type:
df.dtypes.value_counts()
object     43
int64      26
float64    11
dtype: int64
```

Slika 17. Spajanje CSV datoteka „train“ i „test“

Naredba drop() odbacuje navedeni redak ili stupac. Određivanjem osi stupca (axis='columns'), naredba drop() odbacuje navedeni stupac. Određivanjem osi retka (axis='index'), naredba drop() odbacuje navedeni redak. U ovom slučaju bit će potrebno izbrisati cijeli prvi redak i stupac kako bi izbacili nebrojčane string vrijednosti. To se radi iz razloga da program može dalje obrađivati samo brojeve bez da ga zbunjuju unesene riječi u datotekama.

Naredba to_frame() koristi se za pretvaranje serije u DataFrame što znači da će naša y-os iz serije prijeći u DataFrame.

Naredba pd.concat() obavlja sav težak posao izvođenja operacija ulančavanja zajedno s osi objekata dok izvodi logiku skupa (uniju ili presjek) indeksa (ako postoje) na drugim osima. Drugim riječima, nakon dodavanja novih varijabli ukrstit će se liste „train“ i „test“ prije nego li se provede čišćenje podataka.

Naredba df.dtypes.value_counts() kao odgovor vraća seriju koja sadrži broj jedinstvenih redaka u DataFrameu. Odgovor programa nalaže da su pronađena 43 takva objekta od kojih su 26 cijeli, a 11 decimalni brojevi.

Nakon što su datoteke uspješno ukrštene, provodi se provjera postoje li ikakve duplicirane vrijednosti kao i vrijednosti podataka koji nedostaju.

```
print('Number of Duplicates:', len(df[df.duplicated()]))

Number of Duplicates: 0
print('Number of Missing Values:', df.isnull().sum().sum())
Number of Missing Values: 13945
```

Slika 18. Broj duplih značajki i značajki koje nedostaju

Kao što je vidljivo, nakon provjere ukrštenih datoteka program je vratio povratnu informaciju da ne postoje nikakve dvostruke vrijednosti, ali isto tako da postoji čak 13945 vrijednosti koje nedostaju.

6. Inženjering značajki

Inženjering značajki (eng. *Feature engineering*) je pojam širokog značenja. Naime, ono se odnosi se na manipulaciju – dodavanje, brisanje, kombinaciju, mutaciju skupa podataka kako bi se poboljšala obuka modela strojnog učenja, što dovodi do boljih performansi i veće točnosti. Učinkovit inženjering značajki temelji se na dobrom poznavanju poslovnog problema i dostupnih izvora podataka. Stvaranje novih značajki vodi do dubljeg razumijevanja podataka i rezultira kvalitetnijim uvidima u podatke. Kada se izvede ispravno, inženjering značajki jedna je od najkorisnijih tehnika znanosti o podacima, ali je također jedna od najizazovnijih. Kako vi se razumjelo što je inženjering značajki na intuitivnoj razini i zašto je neophodan, za početak valja razmotriti kako ljudi shvaćaju podatke. Ljudi imaju sposobnost pronaći složene obrasce ili odnose, toliko da ih mogu vidjeti čak i kada zapravo ne postoje što čovjeka stavlja u veliku prednost nad strojem. To je razlog zašto je inženjering značajki neizostavan korak u procesu strojnog učenja. [9]

6.1. Vrijednosti koje nedostaju (eng. *missing values*)

Budući da je ranije program javio povratnu informaciju kako ne postoje duplicirane datoteke, potrebno je dalje skrenuti pažnju na vrijednosti koje nedostaju kojih je čak 13945. Da bi se vidjelo u kojim kategorijama se nalaze vrijednosti koje nedostaju, potrebno ih je za početak ispisati.

```
df.isnull().sum().sort_values(ascending=False).head(25)
PoolQC          2905
MiscFeature     2810
Alley           2716
Fence           2343
FireplaceQu     1419
LotFrontage      485
GarageCond       159
GarageQual       159
GarageYrBlt      159
GarageFinish     159
GarageType       157
BsmtCond          82
BsmtExposure      82
BsmtQual          81
BsmtFinType2      80
BsmtFinType1      79
MasVnrType        24
MasVnrArea        23
MSZoning           4
BsmtHalfBath       2
Functional         2
BsmtFullBath       2
Utilities          2
BsmtUnfSF          1
KitchenQual        1
dtype: int64
```

Slika 19. Prikaz vrijednosti koje nedostaju

Funkcija df. isnull(). sum() vraća broj vrijednosti koje nedostaju u skupu podataka. Dio koda sort_values(ascending=False) označava da će program ispisivati vrijednosti od najveće prema najmanjoj, a budući da datoteka i dalje ima prevelik broj redova, proizvoljno će se ispisati samo prvih 25 što predstavlja dio koda .head(25). Nakon što je program ispisao koje kategorije sadrže značajke koje nedostaju, potrebno je posvetiti se analiziranju tih kategorija kako bi se riješili takvih značajki na jedan od načina opisanih ranije.

PoolQC odnosi se na kvalitetu bazena kuće. Opis podataka kaže da NaN u ovoj kategoriji znači da kuća nema bazen i takvih je 2905 značajki. NaN je kratica za Not A Number što u prijevodu znači „nije broj“ i jedan je od uobičajenih načina predstavljanja vrijednosti koja nedostaje u podacima. To je posebna vrijednost s pomičnim zarezom i ne može se pretvoriti u bilo koji drugi tip osim float (decimalan broj).

MiscFeature odnosi se na raznovrsnost značajki kuće. Opis podataka kaže da NaN u ovoj kategoriji znači da kuća ne posjeduje nikakve raznovrsne značajke i takvih je 2810 značajki.

Alley se odnosi na vrstu uličice pristupa nekretnini. Opis podataka kaže da postojanje NaN u ovoj kategoriji znači da kuća ne posjeduje uličice za pristup nekretnini i takvih je 2716 značajki.

Fence se odnosi na vrstu ograde oko imanja. Opis podataka kaže da postojanje NaN u ovoj kategoriji znači da kuća nema ogradu i takvih je 2343 značajki.
FireplaceQu se odnosi na kvalitetu kamina. Opis podataka kaže da postojanje NaN u ovoj kategoriji znači da kuća nema kamin i takvih je 1419 značajki.

Naredba fillna() zamjenjuje NaN (odsutnost vrijednosti) vrijednosti s određenom konkretnom vrijednošću i vraća novi objekt u DataFrame. U ovom slučaju, NaN vrijednosti koje nedostaju će se ručno zamijeniti stringom „None“ što bi u prijevodu značilo „nema“ odnosno da kuća ne posjeduje navedenu karakteristiku te će se tim postupkom eliminirati vrijednosti koje nedostaju koje zbunjuju program.

```
df['PoolQC'] = df['PoolQC'].fillna('None')
df['MiscFeature'] = df['MiscFeature'].fillna('None')
df['Alley'] = df['Alley'].fillna('None')
df['Fence'] = df['Fence'].fillna('None')
df['FireplaceQu'] = df['FireplaceQu'].fillna('None')
```

Slika 20. Pretvaranje NaN u „None“

Nadalje, LotFrontage se odnosi na udaljenost u stopama između ulice i nekretnine. Vrijednosti koje nedostaju, kojih je 485, imputirat će se s medijanom susjedstva.

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

Slika 21. Imputacija vrijednosti koje nedostaju u kategoriji LotFrontage

Naredba groupby omogućuje podjelu podataka u zasebne skupine kako bi se izvršili izračuni za bolju analizu. Pythonova naredba transform vraća okvir podataka s transformiranim vrijednostima te će se u ovom slučaju koristiti izraz lambda pomoću kojeg se mogu programirati uvjeti transformacije. Tako će se u ovoj kategoriji vrijednosti koje nedostaju transformirati na način da će se na njihovom mjestu ispisati medijan između LotFrontage i Neighbourhood. Naredba median() izračunava medijan odnosno srednju vrijednost zadanog skupa podataka čiji su podaci razvrstani ulaznim redoslijedom prije samog računanja. Matematička formula za medijan je: medijan = {(n + 1) / 2}-ta vrijednost, gdje je n broj vrijednosti u skupu podataka.

Kako je više značajki s prefiksom Garage, sve značajke koje počinju s Garage i sadrže NaN znači da te kuće nemaju garažu. S druge strane, kuće koje imaju garaže imaju zasebne karakteristike za svaku od garaža. Budući da je takvih značajki više, najbolje je ih je izdvojiti u zaseban DataFrame.

```
garage_cols = [col for col in df if col.startswith('Garage')]
df[garage_cols]
     GarageType  GarageYrBlt GarageFinish  ...  GarageArea  GarageQual GarageCond
0        Attchd       1961.0          Unf  ...       730.0          TA          TA
1        Attchd       1958.0          Unf  ...       312.0          TA          TA
2        Attchd       1997.0          Fin  ...       482.0          TA          TA
3        Attchd       1998.0          Fin  ...       470.0          TA          TA
4        Attchd       1992.0          RFn  ...       506.0          TA          TA
...         ...          ...          ...  ...         ...         ...         ...
1455     Attchd       1999.0          RFn  ...       460.0          TA          TA
1456     Attchd       1978.0          Unf  ...       500.0          TA          TA
1457     Attchd       1941.0          RFn  ...       252.0          TA          TA
1458     Attchd       1950.0          Unf  ...       240.0          TA          TA
1459     Attchd       1965.0          Fin  ...       276.0          TA          TA

[2914 rows x 7 columns]
```

Slika 22. DataFrame karakteristika garaža

Može se primijetiti da su neka obilježja kategorička, a druga numerička. Iz tog razloga je potrebno zamijeniti NaN vrijednosti koje nedostaju s „None“ u kategoričkim značajkama, a u numeričkim značajkama s 0.

```
for i in df[garage_cols].select_dtypes(exclude='object').columns:
    df[i] = df[i].fillna(0)

for i in df[garage_cols].select_dtypes(include='object').columns:
    df[i] = df[i].fillna('None')
```

Slika 23. Zamjena NaN u kategoriji garaža

Sve značajke koje počinju s Bsmt i sadrže NaN znače da te kuće nemaju podrum pa je potrebno napraviti isti dio zamijene vrijednosti koje nedostaju kao i kod garaža.

```
bsmt_cols = [col for col in df if col.startswith('Bsmt')]
df[bsmt_cols]
     BsmtQual BsmtCond BsmtExposure  ... BsmtUnfSF  BsmtFullBath BsmtHalfBath
0          TA       TA           No  ...     270.0           0.0          0.0
1          TA       TA           No  ...     406.0           0.0          0.0
2          Gd       TA           No  ...     137.0           0.0          0.0
3          TA       TA           No  ...     324.0           0.0          0.0
4          Gd       TA           No  ...    1017.0           0.0          0.0
...       ...      ...          ...  ...       ...           ...          ...
1455       Gd       TA           No  ...     953.0           0.0          0.0
1456       Gd       TA           No  ...     589.0           1.0          0.0
1457       TA       Gd           No  ...     877.0           0.0          0.0
1458       TA       TA           Mn  ...       0.0           1.0          0.0
1459       TA       TA           No  ...     136.0           1.0          0.0

[2914 rows x 10 columns]

for i in df[bsmt_cols].select_dtypes(exclude='object').columns:
    df[i] = df[i].fillna(0)

for i in df[bsmt_cols].select_dtypes(include='object').columns:
     df[i] = df[i].fillna('None')
```

Slika 24. Zamjena NaN u kategoriji podruma

Sve značajke koje počinju s Mas i sadrže NaN znače da te kuće nemaju zidani furnir pa je potrebno napraviti isti dio zamijene vrijednosti koje nedostaju kao i kod garaža i bazena.

```
mas_cols = [col for col in df if col.startswith('Mas')]
df[mas_cols]
     MasVnrType  MasVnrArea
0          None         0.0
1       BrkFace       108.0
2          None         0.0
3       BrkFace        20.0
4          None         0.0
...         ...         ...
1455       None         0.0
1456      Stone       119.0
1457       None         0.0
1458       None         0.0
1459       None         0.0

[2914 rows x 2 columns]

for i in df[mas_cols].select_dtypes(exclude='object').columns:
    df[i] = df[i].fillna(0)

for i in df[mas_cols].select_dtypes(include='object').columns:
    df[i] = df[i].fillna('None')
```

Slika 25. Zamjena NaN u kategoriji zidnog furnira

MSZoning se odnosi na opću klasifikaciju prodaje po zonama. Potrebno je imputirati vrijednosti koje nedostaju s najčešćom kategorijom susjedstva.

```
df['MSZoning'] = df.groupby('Neighborhood')['MSZoning'].transform(lambda i: i.fillna(i.value_counts().index[0]))
print('Missing Values left:',df.isnull().sum().sort_values(ascending=False).head(10))
Missing Values left: Functional      2
Utilities       2
Electrical      1
TotalBsmtSF     1
KitchenQual     1
Exterior1st     1
Exterior2nd     1
SaleType        1
BsmtHalfBath    0
BsmtFullBath    0
dtype: int64
```

Slika 26. Vrijednosti koje nedostaju u klasifikaciji prodaje kuća po zonama susjedstva

Imputacija se vrši na sličan način kao kod kategorije LotFrontage. Međutim, umjesto upotrebe medijana, koristiti će se naredba value_counts(). To je funkcija koja vraća objekt koji sadrži brojeve jedinstvenih vrijednosti. Nakon što je program ispisao vrijednosti od najveće prema najmanjoj, očito je da su preostale vrijednosti koje nedostaju minimalne i njih ćemo zamijeniti korištenjem naredbe mode(). Mode() je statistička funkcija koja se uglavnom koristi u financijskim sektorima za usporedbu trenutnih vrijednosti/cijena s prethodnim te za izračunavanje/predviđanje vjerojatnih budućih cijena iz skupa distribucije cijena. Stoga će ta funkcija odraditi potreban posao zamijene vrijednosti umjesto da ih mijenjamo ručno.

```
df = df.fillna(df.mode().iloc[0])
```

Slika 27. Zamijena vrijednosti koje nedostaju funkcijom mode()

6.2. Pretvaranje nekih od numeričkih kategorija u kategoričke kategorije

Naredba describe() koristi se za izračunavanje nekih statističkih podataka kao što su percentil, srednja vrijednost i std numeričkih vrijednosti Serije ili DataFramea. Analizira i numeričke i objektne serije, kao i skupove stupaca DataFrame mješovitih tipova podataka. Tako će rezultat nakon prethodno provedenih koraka izgledati ovako:

```
df.describe().T
               count          mean           std  ...      50%       75%       max
MSSubClass    2914.0     57.112217     42.474217  ...     50.0     70.00     190.0
LotFrontage   2914.0     69.406829     21.191130  ...     70.0     80.00     313.0
LotArea       2914.0  10128.200755   7798.584415  ...   9450.0  11546.25  215245.0
OverallQual   2914.0      6.087509      1.405287  ...      6.0      7.00      10.0
OverallCond   2914.0      5.566232      1.113182  ...      5.0      6.00       9.0
YearBuilt     2914.0   1971.291352     30.286886  ...   1973.0   2001.00    2010.0
YearRemodAdd  2914.0   1984.254633     20.887641  ...   1993.0   2004.00    2010.0
MasVnrArea    2914.0    100.879204    178.071569  ...      0.0    162.75    1600.0
BsmtFinSF1    2914.0    438.919012    444.059991  ...    368.0    732.75    4010.0
BsmtFinSF2    2914.0     49.650309    169.311762  ...      0.0      0.00    1526.0
BsmtUnfSF     2914.0    560.042210    438.937719  ...    467.0    802.75    2336.0
TotalBsmtSF   2914.0   1048.611531    429.273260  ...    988.0   1301.50    5095.0
1stFlrSF      2914.0   1157.320178    384.986979  ...   1082.0   1383.75    5095.0
2ndFlrSF      2914.0    336.207275    428.204291  ...      0.0    704.00    2065.0
LowQualFinSF  2914.0      4.702471     46.436218  ...      0.0      0.00    1064.0
GrLivArea     2914.0   1498.229925    496.930960  ...   1443.0   1743.00    5095.0
BsmtFullBath  2914.0      0.428964      0.523985  ...      0.0      1.00       3.0
BsmtHalfBath  2914.0      0.061428      0.245805  ...      0.0      0.00       2.0
FullBath      2914.0      1.567605      0.552491  ...      2.0      2.00       4.0
HalfBath      2914.0      0.379890      0.502811  ...      0.0      1.00       2.0
BedroomAbvGr  2914.0      2.860329      0.823228  ...      3.0      3.00       8.0
KitchenAbvGr  2914.0      1.044612      0.214638  ...      1.0      1.00       3.0
TotRmsAbvGrd  2914.0      6.447495      1.564767  ...      6.0      7.00      15.0
Fireplaces    2914.0      0.596088      0.644924  ...      1.0      1.00       4.0
GarageYrBlt   2914.0   1870.144132    450.040352  ...   1977.0   2001.00    2207.0
GarageCars    2914.0      1.763898      0.760680  ...      2.0      2.00       5.0
GarageArea    2914.0    471.363075    213.086144  ...    479.0    576.00    1488.0
WoodDeckSF    2914.0     93.575154    126.412139  ...      0.0    168.00    1424.0
OpenPorchSF   2914.0     47.291009     67.141235  ...     26.0     70.00     742.0
EnclosedPorch 2914.0     23.137955     64.292224  ...      0.0      0.00    1012.0
3SsnPorch     2914.0      2.606726     25.209546  ...      0.0      0.00     508.0
ScreenPorch   2914.0     16.089911     56.228619  ...      0.0      0.00     576.0
PoolArea      2914.0      2.090940     34.579098  ...      0.0      0.00     800.0
MiscVal       2914.0     50.721002    567.807459  ...      0.0      0.00   17000.0
MoSold        2914.0      6.213452      2.713410  ...      6.0      8.00      12.0
YrSold        2914.0   2007.792725      1.315727  ...   2008.0   2009.00    2010.0
train         2914.0      0.499314      0.500085  ...      0.0      1.00       1.0

[37 rows x 8 columns]
```

Slika 28. Polje podataka prije pretvorbe kategorija

Čitanje opisa podataka vrlo jasno pokazuje da neke numeričke značajke predstavljaju određenu kategoriju, stoga je sljedeći cilj numeričke kategorije kao što su godina i mjesec prodaje pretvoriti u kategoričke kategorije.

```
df['MSSubClass'] = df['MSSubClass'].astype(str)
df['MoSold'] = df['MoSold'].astype(str)
df['YrSold'] = df['YrSold'].astype(str)
```

Slika 29. Pretvorba numeričkih kategorija u kategoričke

Naredba astype() vraća novi DataFrame gdje su postojeći tipovi podataka promijenjeni u nanovo odabrani tip što će u ovom slučaju biti string.

6.3. Zbrajanje relevantnih značajki

Prilikom odabira kuće, postoje neke relevantne značajke koje su raspoređene po različitim kategorijama stoga ih je poželjno povezati u zajedničku cjelinu kako bi se dodatno smanjilo polje kategorija. Također, zbrajanje relevantnih značajki može povećati točnost predviđanja. Tako ćemo uvesti nove kategorije: Total_House_SF (ukupna površina kuće u četvornim stopama), Total_Home_Quality (ukupna kvaliteta kuće) i Total_Bathrooms (ukupni broj kupaonica).

```
df['Total_House_SF'] = df['TotalBsmtSF'] + df['1stFlrSF'] + df['2ndFlrSF']
df['Total_Home_Quality'] = (df['OverallQual'] + df['OverallCond'])/2
df['Total_Bathrooms'] = (df['FullBath'] + (0.5 * df['HalfBath']) + df['BsmtFullBath'] + (0.5 * df['BsmtHalfBath']))
```

Slika 30. Zbrajanje relevantnih značajki

6.4. Iskrivljene značajke (eng. *skewed features*)

Podaci se smatraju iskrivljenima (eng. *skewed data*) kada je njihova krivulja distribucije asimetrična u usporedbi s krivuljom normalne distribucije koja je savršeno simetrična. Asimetrija za normalnu distribuciju iznosi 0 stoga je optimalno odabrati značajke koje imaju nagib veći od 0,5. Te iste značajke će radi boljeg shvaćanja biti prikazane grafički.

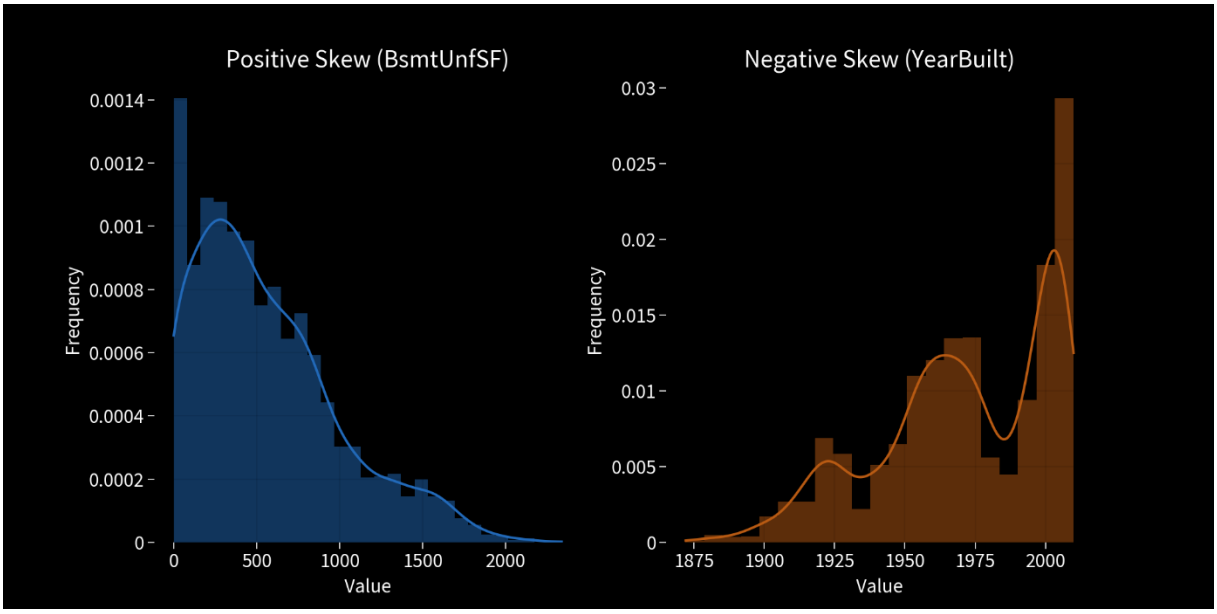
```
numeric_cols = df.select_dtypes(exclude='object').columns
skew_limit = 0.5
skew_vals = df[numeric_cols].skew()
skew_cols = (skew_vals
             .sort_values(ascending=False)
             .to_frame()
             .rename(columns={0:'Skew'})
             .query('abs(Skew) > {0}'.format(skew_limit)))
skew_cols
                      Skew
MiscVal          21.949442
PoolArea         17.688586
LotArea          13.168399
LowQualFinSF     12.084424
3SsnPorch        11.371955
KitchenAbvGr      4.300206
BsmtFinSF2        4.144176
EnclosedPorch     4.002083
ScreenPorch       3.944742
BsmtHalfBath      3.929621
MasVnrArea        2.624455
OpenPorchSF       2.530314
WoodDeckSF        1.847494
1stFlrSF          1.260200
LotFrontage       1.105971
GrLivArea         1.070551
Total_House_SF    1.010874
BsmtFinSF1        0.981793
BsmtUnfSF         0.916129
2ndFlrSF          0.860449
TotRmsAbvGrd      0.750440
Fireplaces        0.726331
HalfBath          0.696987
TotalBsmtSF       0.671945
BsmtFullBath      0.622594
OverallCond       0.569084
Total_Home_Quality -0.565021
YearBuilt         -0.600470
GarageYrBlt       -3.904268
```

Slika 31. Iznosi nagiba kod iskrivljenih značajki

Naredba skew() vraća nagib iskrivljenosti preko tražene osi. Iskrvljenost je mjera asimetrije distribucije vjerojatnosti stvarne slučajne varijable u odnosu na njezinu srednju vrijednost. Navedenom naredbom će se izračunava nagib iskrivljenosti – skew za svaki stupac. Za vizualni prikaz kao primjer će se uzeti BsmUnfSF pozitivnog nagiba iskrivljenosti u iznosu od 0,916129 i YearBuilt negativnog nagiba iskrivljenosti u iznosu od -0,600470.

```
mpl.rcParams['font.family'] = 'Source Sans Pro'
mpl.rcParams['font.size'] = 12
fig, (ax_positive, ax_negative) = plt.subplots(1, 2, figsize=(10, 5))
fig.patch.set_facecolor('black')
ax_positive.patch.set_facecolor('black')
ax_negative.patch.set_facecolor('black')
sns.histplot(df['BsmtUnfSF'],kde=True, stat='density', linewidth=0, color = '#236AB9', ax=ax_positive)
<AxesSubplot:xlabel='BsmtUnfSF', ylabel='Density'>
sns.histplot(df['YearBuilt'], kde=True, stat='density', linewidth=0,color='#B85B14', ax=ax_negative)
<AxesSubplot:xlabel='YearBuilt', ylabel='Density'>
ax_positive.tick_params(axis='x', colors='white')
ax_positive.tick_params(axis='y', colors='white')
ax_negative.tick_params(axis='x', colors='white')
ax_negative.tick_params(axis='y', colors='white')
ax_positive.set(ylabel='Frequency', xlabel='Value');
[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Value')]
ax_negative.set(ylabel='Frequency', xlabel='Value');
[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Value')]
ax_positive.xaxis.label.set_color('white')
ax_positive.yaxis.label.set_color('white')
ax_negative.xaxis.label.set_color('white')
ax_negative.yaxis.label.set_color('white')
ax_positive.set_title('Positive Skew (BsmtUnfSF)', color='white', fontsize= 15)
Text(0.5, 1.0, 'Positive Skew (BsmtUnfSF)')
ax_negative.set_title('Negative Skew (YearBuilt)', color='white', fontsize= 15)
Text(0.5, 1.0, 'Negative Skew (YearBuilt)')
for i in ['top', 'bottom', 'left', 'right']:
    ax_positive.spines[i].set_visible(False)

for i in ['top', 'bottom', 'left', 'right']:
    ax_negative.spines[i].set_visible(False)
```

Slika 32. Kod za crtanje iskrivljenih značajki



Slika 33. Grafički prikaz dviju iskrivljenih značajki s pozitivnim i negativnim nagibom

Na slici koja jasno prikazuje grafički prikaz iskrivljenih značajki na lijevoj strani je graf koji prikazuje primjer iskrivljene značajke s pozitivnim nagibom iskrivljenosti odnosno riječ je o jednoj karakteristici podruma, a na desnoj primjer iskrivljene značajke s negativnim nagibom iskrivljenosti odnosno riječ je o karakteristici godina izgradnje.

6.4.1. Transformacija iskrivljene u normalnu distribuciju

Distribucije u stvarnom svijetu obično su iskrivljene kao što je vidljivo u prethodnim primjerima. Međutim, prisutnost pretjerane asimetrije u podacima može uzrokovati da statistički modeli ne rade učinkovito. U iskrivljenim podacima neke vrijednosti mogu djelovati kao ekstremi (outlieri) za statistički model, a kao što je ranije objašnjeno oni nepovoljno utječu na izvedbu modela, posebno na modele temeljene na regresiji. Iz tog razloga je potrebna transformacija iskrivljenih asimetričnih značajki u simetričnu normalnu (Gaussovu) distribuciju.

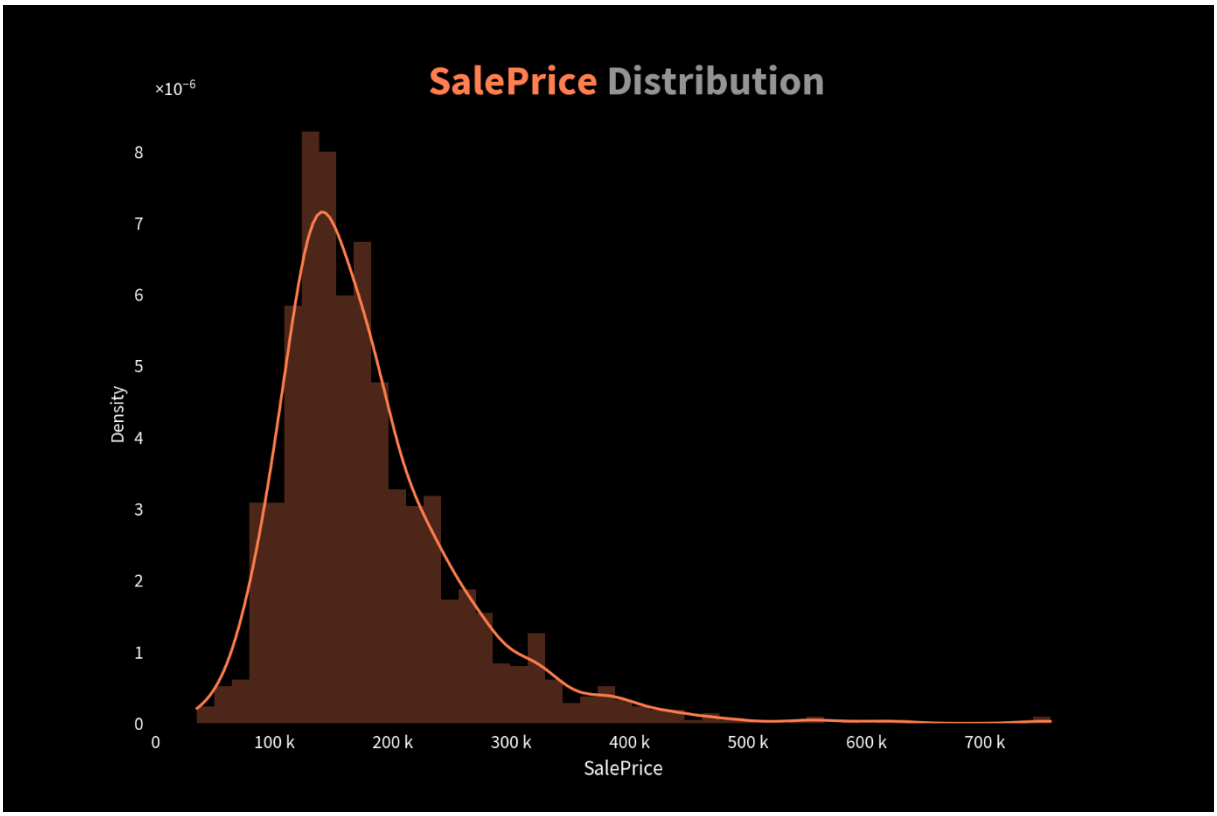
```
from scipy.special import boxcox1p
from scipy.stats import boxcox_normmax
for col in skew_cols.index:
    df[col] = boxcox1p(df[col], boxcox_normmax(df[col] + 1))
```

Slika 34. Transformacija iskrivljene u normalnu distribuciju

Kako bi se transformacija iskrivljenih značajki u normalne uspješno provela, koristit će se tzv. Box-Cox transformacija Python biblioteke SciPy.

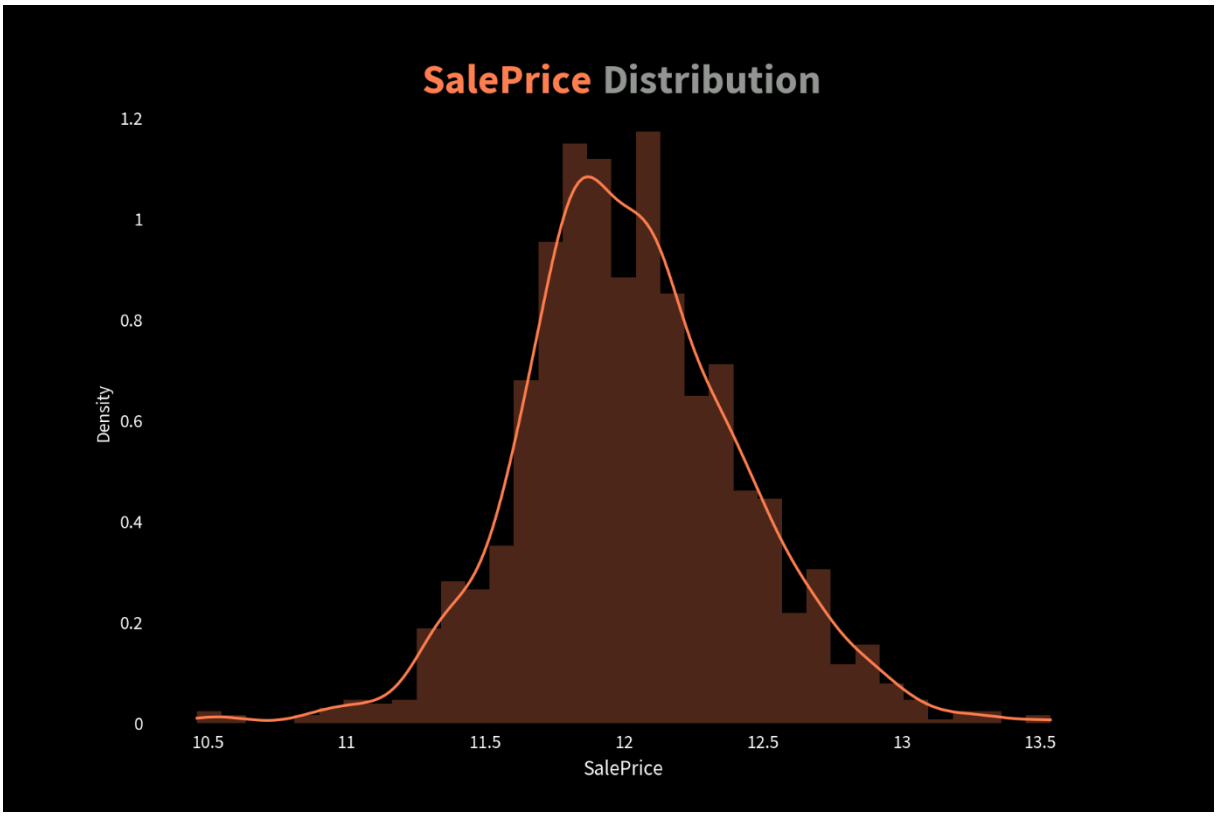
```
import matplotlib.ticker as ticker
mpl.rcParams['font.family'] = 'Source Sans Pro'
mpl.rcParams['font.size'] = 10
fig, ax = plt.subplots(figsize =(9, 6))
fig.patch.set_facecolor('black')
ax.patch.set_facecolor('black')
sns.histplot(y['SalePrice'], stat='density', linewidth=0, color = '#ff7f50', kde=True, alpha=0.3);
<AxesSubplot:xlabel='SalePrice', ylabel='Density'>
ax.xaxis.set_ticks_position('none')
ax.yaxis.set_ticks_position('none')
for i in ['top', 'bottom', 'left', 'right']:
    ax.spines[i].set_visible(False)

ax.xaxis.set_major_formatter(ticker.EngFormatter())
ax.tick_params(axis='x', colors='white')
ax.tick_params(axis='y', colors='white')
ax.xaxis.label.set_color('white')
ax.yaxis.label.set_color('white')
mpl.rcParams['font.family'] = 'Source Sans Pro'
plt.xlabel('SalePrice', fontsize=11);
Text(0.5, 0, 'SalePrice')
plt.text(11.27, 1.25, "SalePrice", size=22, color="#ff7f50", fontweight="bold");
Text(11.27, 1.25, 'SalePrice')
plt.text(11.92, 1.25, "Distribution", size=22, color="grey", fontweight="bold");
Text(11.92, 1.25, 'Distribution')
```

Slika 35. Kod za crtanje transformirane normalne distribucije



Slika 36. Normalna distribucija prodajne cijene

Iako je izvršena transformacija iskrivljenih značajki u normalne, grafički prikaz funkcije i dalje ne izgleda simetrično. Kako bi se taj problem iskrivljenih podataka riješio, potrebno je primijeniti logaritamsku transformaciju na cijeli skup vrijednosti kako bi se eliminirala mogućnost postojanja bilo kakvih ekstrema (ouliera) koji bi negativno utjecali na performanse modela. Logaritamska transformacija je metoda transformacije podataka u kojoj na podatke primjenjujemo logaritamsku funkciju. Zamjenjuje svaku vrijednost x s log(x). Nakon provedbe logaritamske transformacije, jasnije je vidljiva simetrija normalne raspodjele.



Slika 37. Normalna distribucija prodajne cijene u logaritamskom mjerilu

Nakon primjene logaritamske transformacije naredbom log1p(), kao što je prikazano na slici jasno su vidljive određene vrijednosti koje je isprva bilo teško očitati jer su bile u prevelikom brojčanom rasponu. Prije primjene logaritmiranja postojalo je previše ekstremnih vrijednosti, što bi negativno utjecalo na kasniju izvedbu regresijskog modela.

7. Modeliranje

U statističkom modeliranju, regresijska analiza je skup statističkih procesa za procjenu odnosa između zavisne varijable (varijabla nekog ishoda u jeziku strojnog učenja) i jedne ili više nezavisnih varijabli (koje predstavljaju značajke). Najčešći oblik regresijske analize je linearna regresija, čije je obilježje jedna linija (ili složenija linearna kombinacija) koja najviše odgovara podacima prema određenom matematičkom kriteriju. Iz specifičnih matematičkih razloga, ovo omogućuje istraživaču procjenu uvjetnog očekivanja zavisne varijable kada nezavisne varijable poprime zadani skup vrijednosti.

Regresijska analiza prvenstveno se koristi u dvije konceptualno različite svrhe. Jedna od njih je da se regresijska analiza naširoko koristi za predviđanje i procjenu, gdje se njezina upotreba značajno preklapa s područjem strojnog učenja. S druge strane, u nekim se situacijama regresijska analiza može koristiti za zaključivanje uzročno-posljedičnih odnosa između nezavisnih i zavisnih varijabli.

Kod linearne regresije za minimizaciju greške potrebno je izvršiti optimizaciju prema sljedećoj formuli (koja je ujedno i formula za funkciju greške kod standardnog učenja):

$$J(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2$$

Gdje je:

$J(\boldsymbol{w})$ = funkcija srednje kvadratne pogreške ovisna o w

$\boldsymbol{w}$ = optimalna težina

x, y = ulazni podaci na „x-osi i y-osi

n = ukupan broj podataka koje unosimo

Prije provedbe regresijskih modela, bitno je provjeriti jesu li vrijednosti koje nedostaju uklonjene iz podataka. Nakon provjere, upotrebom naredbe get_dummies() svi kategorički stupci će se pretvoriti u stupce indikatora (stupce od 0 i 1). Drugim riječima, sve varijable koje su pohranjene u obliku stringa će se transformirati indikator 1 (True) ako odabrana vrijednost postoji ili indikator 0 (False) ako odabrana vrijednost ne postoji za određeni formirani redak.

```
categ_cols = df.dtypes[df.dtypes == np.object]
categ_cols = categ_cols.index.tolist()

df_enc = pd.get_dummies(df, columns=categ_cols, drop_first=True)

X = df_enc[df_enc['train']==1]
test = df_enc[df_enc['train']==0]
X.drop(['train'], axis=1, inplace=True)
test.drop(['train'], axis=1, inplace=True)
```

Slika 38. Pretvorba kategoričkih stupaca u stupce indikatora

Nakon opisane pretvorbe, stupce s nezavisnom varijablom i stupce za koje smo izradili tzv. „lažne varijable“ (eng. *dummy* = lažan, umjetan) potrebno je odbaciti.

```
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=12345)
def rmse(ytrue, ypredicted):
    return np.sqrt(mean_squared_error(ytrue, ypredicted))
```

Slika 39. Implementiranje Lasso i Ridge regresije iz Python datoteka

Prije početka vršenja regresije, osim što se trebaju unijeti Lasso i Ridge modeli iz Sklearn biblioteke, također je potrebno unijeti train_test_split te mean_squared_error.

Funkcija train_test_split dijeli nizove ili matrice u nasumične podskupove za train i test podatke.

Srednja kvadratna pogreška (RMSE) ili srednja kvadratna devijacija (RMSD) procjenitelja mjeri prosjek kvadrata pogreške, tj. prosječnu kvadratnu razliku između procijenjenih vrijednosti i prave vrijednosti. To je funkcija rizika, koja odgovara očekivanoj vrijednosti kvadrata gubitka pogreške. RMSE je uvijek pozitivna vrijednost, a vrijednost 0 (koja se gotovo nikad ne postiže u praksi) značila bi savršeno uklapanje podataka. Iz toga možemo zaključiti da je poželjna što manja RMSE.

7.1. Lasso regresija i unakrsna provjera

Lasso regresija je tehnika regularizacije. Riječ "LASSO" je kratica na engleskom od *Least Absolute Shrinkage and Selection Operator* što bi u prijevodu značilo „operater najmanjeg apsolutnog skupljanja i odabira“. Naime, riječ je o linearnom modelu koji minimizira svoju funkciju troška.

Lasso regresija pomaže smanjiti prekomjerno uklapanje i odabir značajki. Koristi se za točnije predviđanje. Ovaj model koristi takozvano „skupljanje“ što označava mjesto gdje se vrijednosti podataka skupljaju prema središnjoj točki kao srednjoj vrijednosti. Lasso postupak je pogodan za jednostavne modele odnosno modele s manje parametara.

Ako regresijski model koristi L1 regularizacijsku tehniku, takva vrsta regresije se naziva Lasso regresija. Ako se koristi L2 tehnika regulacije, riječ je o Ridge regresiji.

Funkcija ima parametar regularizacije -L1 penal- odnosno alfa koja podešava intenzitet ovog penala. Ovaj penal smanjuje neke značajke na nulu, što olakšava razumijevanje i tumačenje predviđanja. Što je veća vrijednost alfe, to je više koeficijenata prisiljeno postati nula te onda bivaju eliminirani iz modela što je idealno za izradu jednostavnijih modela. Ovaj tip regulacije može rezultirati rijetkim modelima s malo koeficijenata.

Formula za funkciju greške kod Lasso regresije izgleda ovako:

$$J(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2 + \lambda\|\boldsymbol{w}\|$$

Gdje je:

$$\|\boldsymbol{w}\| = \sum_{i=0}^{d}\|w_i\| \quad \text{i} \quad \lambda \geq 1$$

Znak λ označava količinu skupljanja. Ako je λ = 0 implicira se da su sve značajke uzete u obzir i taj slučaj je ekvivalentan linearnoj regresiji gdje se samo rezidualni zbroj kvadrata uzima u obzir za izgradnju predviđenog modela. Ako je λ = ∞ implicira se na to da se nijedna značajka ne razmatra, tj. kako se λ približava beskonačnosti, eliminira se sve više značajki. [10]

```
lasso = Lasso(max_iter = 100000, normalize = True)

lassocv = LassoCV(alphas = None, cv = 10, max_iter = 100000, normalize = True)
lassocv.fit(X_train, y_train)

lasso.set_params(alpha=lassocv.alpha_)
lasso.fit(X_train, y_train)

print('The Lasso I:')
print("Alpha =", lassocv.alpha_)
print("RMSE =", rmse(y_test, lasso.predict(X_test)))
The Lasso I:
Alpha = 6.872779832960424e-05
RMSE = 0.12072477839971668
```

Slika 40. Odabir najbolje vrijednosti alfe za Lasso regresiju

Prvi korak za izradu Lasso modela je pronalaženje optimalne lambda vrijednosti. Nakon vršenja deseterostruke unakrsne provjere za odabir najbolje alfe, potrebno je ponovno prilagodili model.

```
alpha = np.geomspace(1e-5, 1e0, num=6)
lasso_cv_model = LassoCV(alphas = alpha, cv = 10, max_iter = 100000, normalize = True).fit(X_train,y_train)
lasso_tuned = Lasso(max_iter = 100000, normalize = True).set_params(alpha = lasso_cv_model.alpha_).fit(X_train,y_train)
print('The Lasso II:')
print("Alpha =", lasso_cv_model.alpha_)
print("RMSE =", rmse(y_test, lasso_tuned.predict(X_test)))
The Lasso II:
Alpha = 0.0001
RMSE = 0.11962451145557704
```

Slika 41. Prilagođavanje Lasso regresije sa određenom alfom

Model se najbolje prilagođava prelaskom na logaritamske vrijednosti korištenjem naredbe geomspace(). Iako se vrijednost alfe značajnije povećala s $6{,}8728*10^{-5}$ na vrijednost od 0,0001, vrijednost RMSE se smanjila s 0,1207 na 0,1196 (zaokruženo na prve 4 znamenke).

7.2. Ridge regresija i unakrsna provjera

Ridge regresija je metoda podešavanja modela koja se koristi za analizu svih podataka koji pate od multikolinearnosti. Ova metoda izvodi L2 regulaciju koja za razliku od L1 regulacije, ne rezultira nikakvom eliminacijom rijetkih modela ili koeficijenata. Što je niža vrijednost alfe, to će model biti linearniji. Ridge regresija nosi ovakav naziv jer se dijagonala jedinica u korelacijskoj matrici može opisati kao greben (eng. *ridge* = greben). Kada se pojavi problem multikolinearnosti, najmanji kvadrati su nepristrani, a varijance velike, to rezultira time da su predviđene vrijednosti daleko od stvarnih vrijednosti.[11]
Formula za funkciju greške kod Ridge regresije izgleda ovako:

$$J(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2 + \lambda\|\boldsymbol{w}\|^2$$

Gdje je:

$$\|\boldsymbol{w}\|^2 = \sum_{i=0}^{d}\|w_i\|^2 \quad \text{i} \quad \lambda \geq 1$$

Kao i kod Lasso regresije, polje podataka će prijeći na logaritamske vrijednosti zbog transformacije u normalnu distribuciju.

```
alphas = np.geomspace(1e-9, 5, num=100)

ridgecv = RidgeCV(alphas = alphas, scoring = 'neg_mean_squared_error', normalize = True)
ridgecv.fit(X_train, y_train)

ridge = Ridge(alpha = ridgecv.alpha_, normalize = True)
ridge.fit(X_train, y_train)

print('Ridge Regression:')
print("Alpha =", ridgecv.alpha_)
print("RMSE =", rmse(y_test, ridge.predict(X_test)))
Ridge Regression:
Alpha = 0.21251935471767855
RMSE = 0.12350180903194746
```

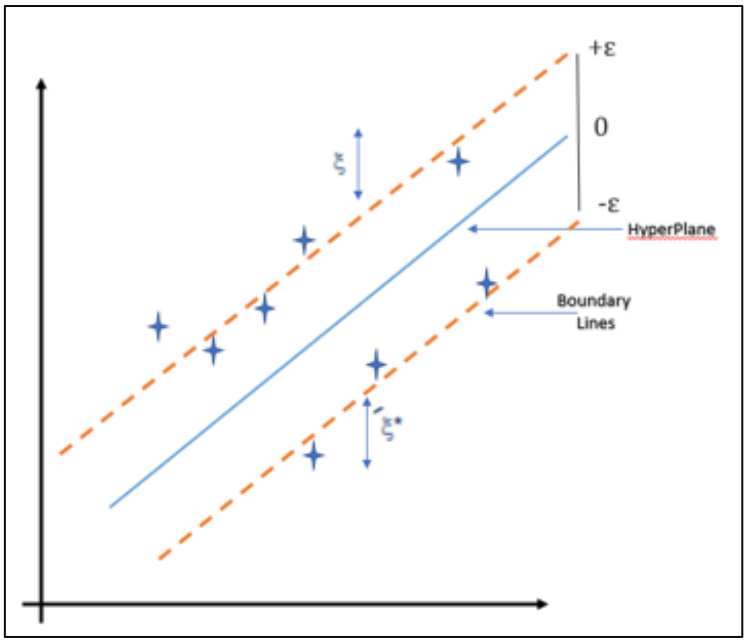
Slika 42. Ridge regresija u Pythonu

Provođenje Ridge regresije ispisuje vrijednost alfe od zaokruženo 0,2125 što je poprilično veća vrijednost od one dobivene Lasso regresijom, dok se vrijednost RMSE malo povećala na 0,1235.

7.3. Regresija potpornih vektora i unakrsna provjera

Regresija potpornih vektora (eng. *SVR = Support Vector Regression*) je moćan algoritam koji omogućuje odabir mjere tolerancije greškaka kroz prihvatljivu marginu pogreške i kroz podešavanje tolerancije. Regresija potpornih vektora, kao što joj ime govori, je regresijski algoritam koji podržava i linearne i nelinearne regresije jer ova metoda radi na principu potpornih vektora. Koristi se za predviđanje kontinuiranih uređenih varijabli.

U jednostavnoj regresiji ideja je minimizirati stopu pogreške, dok je u SVR-u ideja uklopiti pogrešku unutar unaprijed određenih granica tolerancije, što znači da je riječ o aproksimaciji najbolje vrijednosti unutar zadane margine koja se naziva ε-cijev.



Slika 43. ε-cijev

Kao što je vidljivo sa slike, ε-cijev se sastoji od nekoliko različitih dijelova:

• **Hiperravnina**

Hiperravnina je linija razdvajanja između dvije klase podataka u višoj dimenziji od stvarne dimenzije. U SVR-u se definira kao linija koja pomaže u predviđanju ciljane vrijednosti.

• **Kernel**

U SVR-u regresija se izvodi na višoj dimenziji. Kako bi se to ostvarilo, potrebna je funkcija koja bi trebala mapirati podatkovne točke u svoju višu dimenziju. Ta funkcija se zove kernel.

• **Granične linije**

Ovo su dvije linije koje su povučene oko hiperravnine na udaljenosti od ε (epsilon). Koriste se za stvaranje margine između podatkovnih točaka koju možemo razmatrati kao tolerancijsko polje čije granične vrijednosti se ne smiju prijeći.

• **Potporni vektor**

Potporni vektor je vektor koji se koristi za definiranje hiperravnine ili možemo reći da su to ekstremne podatkovne točke u skupu podataka koje pomažu u definiranju hiperravnine. Ove podatkovne točke leže blizu granice.

SVR radi na principu pokušavanja nalaženja pripadajuće krivulje za zadane točke podataka. No, budući da je to regresijski algoritam, umjesto da koristi krivulju kao granicu odluke, on koristi krivulju za pronalaženje podudaranja između vektora i položaja krivulje. U tom naumu pomaže uporaba potpornih vektora koji pomažu u određivanju najbližeg podudaranja između podatkovnih točaka i funkcije koja se koristi za njihovo predstavljanje. [12]

```
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import RobustScaler
from sklearn.model_selection import KFold
from sklearn.svm import SVR


kf = KFold(shuffle=True, random_state=1234, n_splits=10)

X_train_scale = RobustScaler().fit_transform(X_train)
X_test_scale = RobustScaler().fit_transform(X_test)

parameters = {'C':[20, 30, 40], 'gamma': [1e-4, 3e-4, 5e-4],'epsilon':[0.1, 0.01, 0.05]}
svr = SVR(kernel='rbf')
clf = GridSearchCV(svr, parameters, cv=kf)
clf.fit(X_train_scale,y_train)
clf.best_params_
{'C': 20, 'epsilon': 0.01, 'gamma': 0.0003}

svr = SVR(kernel ='rbf', C= 20, epsilon= 0.01, gamma=0.0003)
svr.fit(X_train_scale,y_train)

print('SVR Regression:')
print("RMSE =", rmse(y_test, svr.predict(X_test_scale)))
SVR Regression:
RMSE = 0.12708575951074244
```

Slika 44. Regresija potpornih vektora u Pythonu

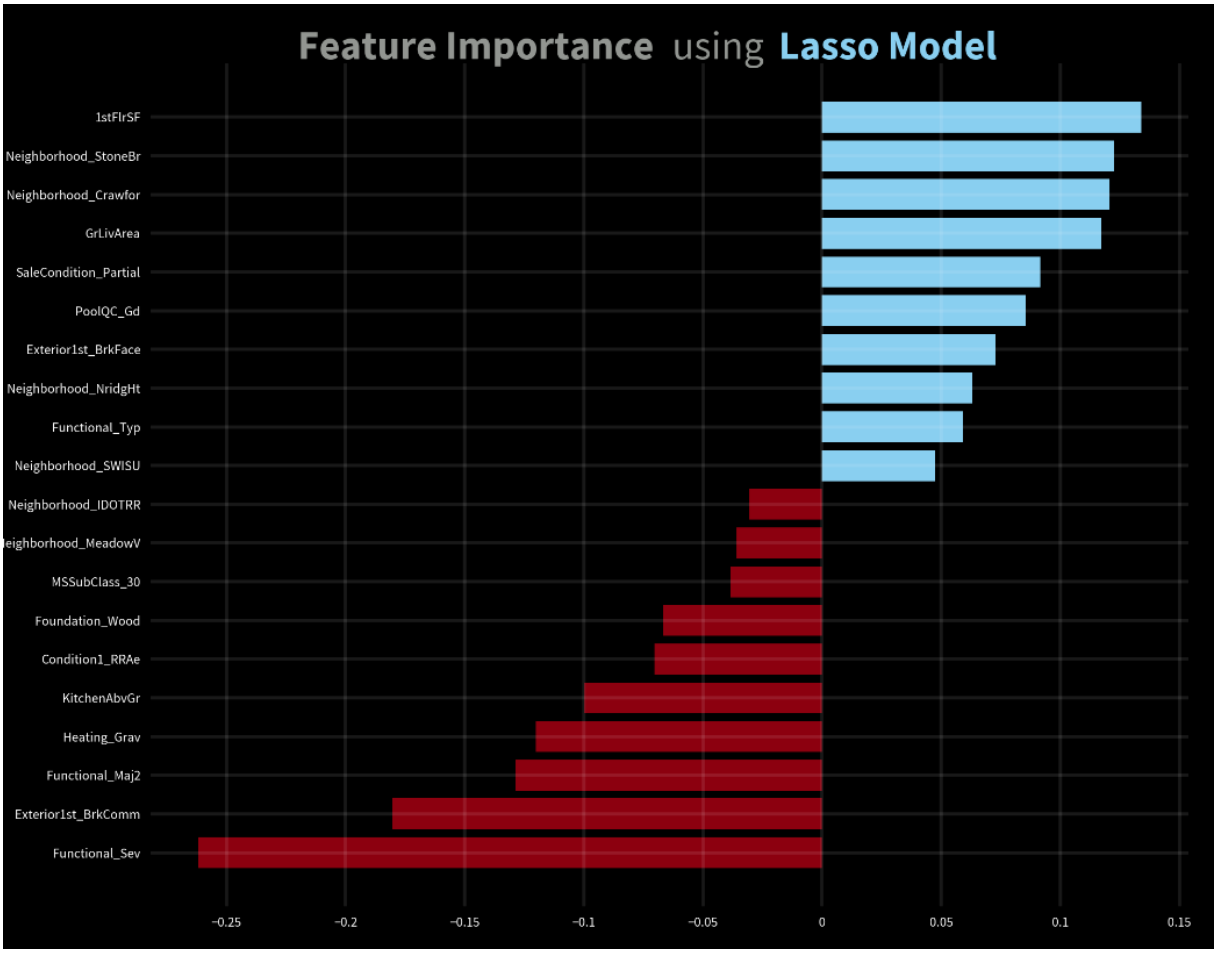
U Python kodu za SVR, oznaka C se odnosi na parametar regularizacije. Snaga regularizacije obrnuto je proporcionalna C i mora biti strogo pozitivna. Parametar epsilon određuje epsilon-cijev, a parametar gamma odnosi se na koeficijent kernela.

Provedbom SVR regresije u Pythonu na zadanom modelu vrijednost RMSE iznosi 0,1271.

Kao što je ranije rečeno, idealan slučaj bi bio kada bi RMSE iznosio 0, no to je u praksi gotovo nemoguće. Vidljivo je da se RMSE svih provedenih regresijskih metoda gotovo malo razlikuje po decimalama no ipak u krajnjem slučaju Lasso regresija daje njegovu najmanju vrijednost.

8. Interpretacija

Kao što je ranije spomenuto, Lasso regresija je vrlo jednostavan model za tumačenje, stoga je najbolja opcija interpretaciju važnosti značajki odraditi na Lasso regresijskom modelu. Ispisivanjem koda, dobije se da od 288 koeficijenata razrađenih Lasso regresijom, 116 nisu nule.



Slika 45. Važnost značajki prema Lasso regresiji

Sve karakteristike kuće koje su na grafikonu označene plavom bojom pozitivno utječu na prodajnu cijenu kuće, što znači da te karakteristike povećavaju cijenu kuće. Suprotno tome, sva karakteristike koje su na grafikonu označene crvenom bojom negativno utječu na prodajnu cijenu kuće, što znači da smanjuju cijenu kuće.

9. Zaključak

Na početnom dijagramu je omjer korelacije prodajne cijene i željenih značajki pokazao da je dakako puno više značajki koje povećavaju cijenu kuće naspram onih koje smanjuju njenu novčanu vrijednost. Međutim, primjenom regresijskih metoda dokazalo se upravo suprotno odnosno da je prema važnosti značajki veći skok kod karakteristika koje smanjuju prodajnu cijenu kuće spram karakteristika koje tu cijenu povećavaju. Drugim riječima, računalo je obavilo bolji posao od ljudske intuicije i to je upravo jedan od razloga zašto umjetna inteligencija stalno napreduje kako bi pomogla čovjeku u njegovim suvremenim dilemama i problemima.

Literatura

[1] What Is the Definition of Machine Learning? | Expert.ai | Expert.ai

[2] DataFrames – Databricks

[3] pandas (software) - Wikipedia

[4] NumPy - Wikipedia

[5] sklearn.linear_model.Lasso — scikit-learn 1.1.2 documentation

[6] Comma-separated values - Wikipedia

[7] What is Exploratory Data Analysis? | IBM

[8] Difference between Data Cleaning and Data Processing - GeeksforGeeks

[9] 8 Feature Engineering Techniques for Machine Learning (projectpro.io)

[10] What is LASSO Regression Definition, Examples and Techniques (mygreatlearning.com)

[11]Ridge Regression Definition & Examples | What is Ridge Regression? (mygreatlearning.com)

[12] Support Vector Regression | Learn the Working and Advantages of SVR (educba.com)

[13] House Price Prediction: Lasso, Ridge and More | Kaggle

[14] House Prices - Advanced Regression Techniques | Kaggle

[15] Python Tutorial (w3schools.com)

PRILOZI

```
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
import pandas as pd
import numpy as np
train = pd.read_csv('C:\\Users\\anton\\Desktop\\house-prices-advanced-regression-techniques\\train.csv')
test = pd.read_csv('C:\\Users\\anton\\Desktop\\house-prices-advanced-regression-techniques\\test.csv')
train.head()
   Id  MSSubClass MSZoning  ...  SaleType  SaleCondition SalePrice
0   1          60       RL  ...        WD         Normal    208500
1   2          20       RL  ...        WD         Normal    181500
2   3          60       RL  ...        WD         Normal    223500
3   4          70       RL  ...        WD        Abnorml    140000
4   5          60       RL  ...        WD         Normal    250000

[5 rows x 81 columns]
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
import proplot as pplt
def plots_design():
    fig.patch.set_facecolor('black')
    ax.patch.set_facecolor('black')
...     ax.tick_params(axis='both', which='major', labelsize=8)
...     ax.yaxis.set_label_coords(0, 0)
...     ax.grid(color='white', linewidth=2)
...     ax.xaxis.set_ticks_position('none')
...     ax.yaxis.set_ticks_position('none')
...     for i in ['top', 'bottom', 'left', 'right']:
...         ax.spines[i].set_visible(False)
...     ax.tick_params(axis='x', colors='white')
...     ax.tick_params(axis='y', colors='white')
...     mpl.rcParams['font.family'] = 'Source Sans Pro'
...
...
>>> corr = train[train.columns].corr()['SalePrice'][:].sort_values(ascending=True).to_frame()
>>> corr = corr.drop(corr[corr.SalePrice > 0.99].index)
>>> fig, ax = plt.subplots(figsize =(9, 9))
>>> ax.barh(corr.index, corr.SalePrice, align='center', color = np.where(corr['SalePrice'] < 0, 'crimson', '#89CFF0'))
<BarContainer object of 37 artists>
>>> plots_design()
>>> plt.text(-0.12, 39, "Correlation", size=24, color="grey", fontweight="bold");
Text(-0.12, 39, 'Correlation')
>>> plt.text(0.135, 39, "of", size=24, color="grey");
Text(0.135, 39, 'of')
>>> plt.text(0.185, 39, "SalePrice", size=24, color="#89CFF0", fontweight="bold");
Text(0.185, 39, 'SalePrice')
>>> plt.text(0.4, 39, "to", size=24, color="grey");
```

```
Text(0.135, 39, 'of')
>>> plt.text(0.185, 39, "SalePrice", size=24, color="#89CFF0", fontweight="bold");
Text(0.185, 39, 'SalePrice')
>>> plt.text(0.4, 39, "to", size=24, color="grey");
Text(0.4, 39, 'to')
>>> plt.text(0.452, 39, "Other Features", size=24, color="grey", fontweight="bold");
Text(0.452, 39, 'Other Features')
>>> plt.text(0.9, -7, "Antonela Suca", fontsize=11, ha="right", color='grey');
Text(0.9, -7, 'Antonela Suca')
>>> top_corr = corr['SalePrice'].sort_values(ascending=False).head(10).index
>>> top_corr = top_corr.union(['SalePrice'])
...
>>> sns.pairplot(train[top_corr]);
<seaborn.axisgrid.PairGrid object at 0x000001AC2032F010>
>>> plt.show()
```

```
print('Training Shape:', train.shape)
Training Shape: (1460, 81)
print('Test Shape:', test.shape)
Test Shape: (1459, 80)
train_id = train['Id']
test_id = test['Id']
del train['Id']
del test['Id']
train1 = train.copy()
train1 = train1.drop(train1[(train1['GarageArea']>1200) & (train1['SalePrice']<300000)].index)
train1 = train1.drop(train1[(train1['GrLivArea']>4000) & (train1['SalePrice']<300000)].index)
train1 = train1.drop(train1[(train1['TotalBsmtSF']>5000)].index)
print('Outliers removed =' , train.shape[0] - train1.shape[0])
Outliers removed = 5
X = train1.drop('SalePrice', axis=1)
y = train1['SalePrice'].to_frame()
X['train'] = 1
test['train'] = 0
df = pd.concat([test, X])
print('Count of Features per Data Type:')
Count of Features per Data Type:
df.dtypes.value_counts()
object     43
int64      26
float64    11
dtype: int64
print('Count of Features per Data Type:')
df.dtypes.value_counts()
SyntaxError: multiple statements found while compiling a single statement
print('Number of Duplicates:', len(df[df.duplicated()]))

Number of Duplicates: 0
print('Number of Missing Values:', df.isnull().sum().sum())
Number of Missing Values: 13945
print('Missing Values per Column:')
Missing Values per Column:
df.isnull().sum().sort_values(ascending=False).head(25)
```

```
Missing Values per Column:
df.isnull().sum().sort_values(ascending=False).head(25)
PoolQC          2905
MiscFeature     2810
Alley           2716
Fence           2343
FireplaceQu     1419
LotFrontage      485
GarageCond       159
GarageQual       159
GarageYrBlt      159
GarageFinish     159
GarageType       157
BsmtCond          82
BsmtExposure      82
BsmtQual          81
BsmtFinType2      80
BsmtFinType1      79
MasVnrType        24
MasVnrArea        23
MSZoning           4
BsmtHalfBath       2
Functional         2
BsmtFullBath       2
Utilities          2
BsmtUnfSF          1
KitchenQual        1
dtype: int64
df['PoolQC'] = df['PoolQC'].fillna('None')
df['MiscFeature'] = df['MiscFeature'].fillna('None')
df['Alley'] = df['Alley'].fillna('None')
df['Fence'] = df['Fence'].fillna('None')
df['FireplaceQu'] = df['FireplaceQu'].fillna('None')
df['LotFrontage'] = df.groupby('Neighborhood')['LotFrontage'].transform(lambda i: i.fillna(i.median()))
garage_cols = [col for col in df if col.startswith('Garage')]
```

```
garage_cols = [col for col in df if col.startswith('Garage')]
df[garage_cols]
     GarageType  GarageYrBlt GarageFinish  ...  GarageArea  GarageQual GarageCond
0        Attchd       1961.0          Unf  ...       730.0          TA          TA
1        Attchd       1958.0          Unf  ...       312.0          TA          TA
2        Attchd       1997.0          Fin  ...       482.0          TA          TA
3        Attchd       1998.0          Fin  ...       470.0          TA          TA
4        Attchd       1992.0          RFn  ...       506.0          TA          TA
...         ...          ...          ...  ...         ...         ...         ...
1455     Attchd       1999.0          RFn  ...       460.0          TA          TA
1456     Attchd       1978.0          Unf  ...       500.0          TA          TA
1457     Attchd       1941.0          RFn  ...       252.0          TA          TA
1458     Attchd       1950.0          Unf  ...       240.0          TA          TA
1459     Attchd       1965.0          Fin  ...       276.0          TA          TA

[2914 rows x 7 columns]
for i in df[garage_cols].select_dtypes(exclude='object').columns:
    df[i] = df[i].fillna(0)

for i in df[garage_cols].select_dtypes(include='object').columns:
    df[i] = df[i].fillna('None')

bsmt_cols = [col for col in df if col.startswith('Bsmt')]
df[bsmt_cols]
     BsmtQual BsmtCond BsmtExposure  ... BsmtUnfSF  BsmtFullBath BsmtHalfBath
0          TA       TA           No  ...     270.0           0.0          0.0
1          TA       TA           No  ...     406.0           0.0          0.0
2          Gd       TA           No  ...     137.0           0.0          0.0
3          TA       TA           No  ...     324.0           0.0          0.0
4          Gd       TA           No  ...    1017.0           0.0          0.0
...       ...      ...          ...  ...       ...           ...          ...
1455       Gd       TA           No  ...     953.0           0.0          0.0
1456       Gd       TA           No  ...     589.0           1.0          0.0
1457       TA       Gd           No  ...     877.0           0.0          0.0
1458       TA       TA           Mn  ...       0.0           1.0          0.0
1459       TA       TA           No  ...     136.0           1.0          0.0

[2914 rows x 10 columns]

for i in df[bsmt_cols].select_dtypes(exclude='object').columns:
    df[i] = df[i].fillna(0)

for i in df[bsmt_cols].select_dtypes(include='object').columns:
     df[i] = df[i].fillna('None')

for i in df[bsmt_cols].select_dtypes(include='object').columns:
    df[i] = df[i].fillna('None')
```

```
mas_cols = [col for col in df if col.startswith('Mas')]
df[mas_cols]
     MasVnrType  MasVnrArea
0          None         0.0
1       BrkFace       108.0
2          None         0.0
3       BrkFace        20.0
4          None         0.0
...         ...         ...
1455       None         0.0
1456      Stone       119.0
1457       None         0.0
1458       None         0.0
1459       None         0.0

[2914 rows x 2 columns]

for i in df[mas_cols].select_dtypes(exclude='object').columns:
    df[i] = df[i].fillna(0)

for i in df[mas_cols].select_dtypes(include='object').columns:
    df[i] = df[i].fillna('None')

df['MSZoning'] = df.groupby('Neighborhood')['MSZoning'].transform(lambda i: i.fillna(i.value_counts().index[0]))
```

```
print('Missing Values left:')
Missing Values left:
df.isnull().sum().sort_values(ascending=False).head(10)
Functional      2
Utilities       2
Electrical      1
TotalBsmtSF     1
KitchenQual     1
Exterior1st     1
Exterior2nd     1
SaleType        1
BsmtHalfBath    0
BsmtFullBath    0
dtype: int64
df = df.fillna(df.mode().iloc[0])
df.describe().T
```

```
df = df.fillna(df.mode().iloc[0])
df.describe().T
                count          mean          std  ...      50%       75%       max
MSSubClass     2914.0     57.112217    42.474217  ...     50.0     70.00     190.0
LotFrontage    2914.0     69.406829    21.191130  ...     70.0     80.00     313.0
LotArea        2914.0  10128.200755  7798.584415  ...   9450.0  11546.25  215245.0
OverallQual    2914.0      6.087509     1.405287  ...      6.0      7.00      10.0
OverallCond    2914.0      5.566232     1.113182  ...      5.0      6.00       9.0
YearBuilt      2914.0   1971.291352    30.286886  ...   1973.0   2001.00    2010.0
YearRemodAdd   2914.0   1984.254633    20.887641  ...   1993.0   2004.00    2010.0
MasVnrArea     2914.0    100.879204   178.071569  ...      0.0    162.75    1600.0
BsmtFinSF1     2914.0    438.919012   444.059991  ...    368.0    732.75    4010.0
BsmtFinSF2     2914.0     49.650309   169.311762  ...      0.0      0.00    1526.0
BsmtUnfSF      2914.0    560.042210   438.937719  ...    467.0    802.75    2336.0
TotalBsmtSF    2914.0   1048.611531   429.273260  ...    988.0   1301.50    5095.0
1stFlrSF       2914.0   1157.320178   384.986979  ...   1082.0   1383.75    5095.0
2ndFlrSF       2914.0    336.207275   428.204291  ...      0.0    704.00    2065.0
LowQualFinSF   2914.0      4.702471    46.436218  ...      0.0      0.00    1064.0
GrLivArea      2914.0   1498.229925   496.930960  ...   1443.0   1743.00    5095.0
BsmtFullBath   2914.0      0.428964     0.523985  ...      0.0      1.00       3.0
BsmtHalfBath   2914.0      0.061428     0.245805  ...      0.0      0.00       2.0
FullBath       2914.0      1.567605     0.552491  ...      2.0      2.00       4.0
HalfBath       2914.0      0.379890     0.502811  ...      0.0      1.00       2.0
BedroomAbvGr   2914.0      2.860329     0.823228  ...      3.0      3.00       8.0
KitchenAbvGr   2914.0      1.044612     0.214638  ...      1.0      1.00       3.0
TotRmsAbvGrd   2914.0      6.447495     1.564767  ...      6.0      7.00      15.0
Fireplaces     2914.0      0.596088     0.644924  ...      1.0      1.00       4.0
GarageYrBlt    2914.0   1870.144132   450.040352  ...   1977.0   2001.00    2207.0
GarageCars     2914.0      1.763898     0.760680  ...      2.0      2.00       5.0
GarageArea     2914.0    471.363075   213.086144  ...    479.0    576.00    1488.0
WoodDeckSF     2914.0     93.575154   126.412139  ...      0.0    168.00    1424.0
OpenPorchSF    2914.0     47.291009    67.141235  ...     26.0     70.00     742.0
EnclosedPorch  2914.0     23.137955    64.292224  ...      0.0      0.00    1012.0
3SsnPorch      2914.0      2.606726    25.209546  ...      0.0      0.00     508.0
ScreenPorch    2914.0     16.089911    56.228619  ...      0.0      0.00     576.0
PoolArea       2914.0      2.090940    34.579098  ...      0.0      0.00     800.0
MiscVal        2914.0     50.721002   567.807459  ...      0.0      0.00   17000.0
MoSold         2914.0      6.213452     2.713410  ...      6.0      8.00      12.0
YrSold         2914.0   2007.792725     1.315727  ...   2008.0   2009.00    2010.0
train          2914.0      0.499314     0.500085  ...      0.0      1.00       1.0

[37 rows x 8 columns]
```

```
df['MSSubClass'] = df['MSSubClass'].astype(str)
df['MoSold'] = df['MoSold'].astype(str)
df['YrSold'] = df['YrSold'].astype(str)

df['Total_House_SF'] = df['TotalBsmtSF'] + df['1stFlrSF'] + df['2ndFlrSF']
df['Total_Home_Quality'] = (df['OverallQual'] + df['OverallCond'])/2
df['Total_Bathrooms'] = (df['FullBath'] + (0.5 * df['HalfBath']) + df['BsmtFullBath'] + (0.5 * df['BsmtHalfBath']))

numeric_cols = df.select_dtypes(exclude='object').columns
skew_limit = 0.5
skew_vals = df[numeric_cols].skew()
skew_cols = (skew_vals
             .sort_values(ascending=False)
             .to_frame()
             .rename(columns={0:'Skew'})
             .query('abs(Skew) > {0}'.format(skew_limit)))
skew_cols
                        Skew
MiscVal            21.949442
PoolArea           17.688586
LotArea            13.168399
LowQualFinSF       12.084424
3SsnPorch          11.371955
KitchenAbvGr        4.300206
BsmtFinSF2          4.144176
EnclosedPorch       4.002083
ScreenPorch         3.944742
BsmtHalfBath        3.929621
MasVnrArea          2.624455
OpenPorchSF         2.530314
WoodDeckSF          1.847494
1stFlrSF            1.260200
LotFrontage         1.105971
GrLivArea           1.070551
Total_House_SF      1.010874
BsmtFinSF1          0.981793
BsmtUnfSF           0.916129
2ndFlrSF            0.860449
TotRmsAbvGrd        0.750440
Fireplaces          0.726331
HalfBath            0.696987
TotalBsmtSF         0.671945
BsmtFullBath        0.622594
OverallCond         0.569084
Total_Home_Quality -0.565021
YearBuilt          -0.600470
GarageYrBlt        -3.904268
```

```
mpl.rcParams['font.family'] = 'Source Sans Pro'
mpl.rcParams['font.size'] = 12
fig, (ax_positive, ax_negative) = plt.subplots(1, 2, figsize=(10, 5))
fig.patch.set_facecolor('black')
ax_positive.patch.set_facecolor('black')
ax_negative.patch.set_facecolor('black')
sns.histplot(df['BsmtUnfSF'],kde=True, stat='density', linewidth=0, color = '#236AB9', ax=ax_positive)
<AxesSubplot:xlabel='BsmtUnfSF', ylabel='Density'>
sns.histplot(df['YearBuilt'], kde=True, stat='density', linewidth=0,color='#B85B14', ax=ax_negative)
<AxesSubplot:xlabel='YearBuilt', ylabel='Density'>
ax_positive.tick_params(axis='x', colors='white')
ax_positive.tick_params(axis='y', colors='white')
ax_negative.tick_params(axis='x', colors='white')
ax_negative.tick_params(axis='y', colors='white')
ax_positive.set(ylabel='Frequency', xlabel='Value');
[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Value')]
ax_negative.set(ylabel='Frequency', xlabel='Value');
[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Value')]
ax_positive.xaxis.label.set_color('white')
ax_positive.yaxis.label.set_color('white')
ax_negative.xaxis.label.set_color('white')
ax_negative.yaxis.label.set_color('white')
ax_positive.set_title('Positive Skew (BsmtUnfSF)', color='white', fontsize= 15)
Text(0.5, 1.0, 'Positive Skew (BsmtUnfSF)')
ax_negative.set_title('Negative Skew (YearBuilt)', color='white', fontsize= 15)
Text(0.5, 1.0, 'Negative Skew (YearBuilt)')
for i in ['top', 'bottom', 'left', 'right']:
    ax_positive.spines[i].set_visible(False)

for i in ['top', 'bottom', 'left', 'right']:
    ax_negative.spines[i].set_visible(False)
```

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


for col in skew_cols.index:
    df[col] = boxcox1p(df[col], boxcox_normmax(df[col] + 1))
import matplotlib.ticker as ticker


mpl.rcParams['font.family'] = 'Source Sans Pro'
mpl.rcParams['font.size'] = 10
fig, ax = plt.subplots(figsize =(9, 6))
fig.patch.set_facecolor('black')
ax.patch.set_facecolor('black')

sns.histplot(y['SalePrice'], stat='density', linewidth=0, color = '#ff7f50', kde=True, alpha=0.3);
ax.xaxis.set_ticks_position('none')
ax.yaxis.set_ticks_position('none')

for i in ['top', 'bottom', 'left', 'right']:
    ax.spines[i].set_visible(False)

plt.grid(b=None)
ax.xaxis.set_major_formatter(ticker.EngFormatter())
ax.tick_params(axis='x', colors='white')
ax.tick_params(axis='y', colors='white')
ax.xaxis.label.set_color('white')
ax.yaxis.label.set_color('white')

mpl.rcParams['font.family'] = 'Source Sans Pro'

plt.xlabel('SalePrice', fontsize=11);

plt.text(230000, 0.0000088, "SalePrice", size=22, color="#ff7f50", fontweight="bold");
plt.text(380000, 0.0000088, "Distribution", size=22, color="grey", fontweight="bold");
y["SalePrice"] = np.log1p(y["SalePrice"])

import matplotlib.ticker as ticker

mpl.rcParams['font.family'] = 'Source Sans Pro'
mpl.rcParams['font.size'] = 10

fig, ax = plt.subplots(figsize =(9, 6))
fig.patch.set_facecolor('black')
ax.patch.set_facecolor('black')
```

```
sns.histplot(y['SalePrice'], stat='density', linewidth=0, color = '#ff7f50', kde=True, alpha=0.3);

ax.xaxis.set_ticks_position('none')
ax.yaxis.set_ticks_position('none')

for i in ['top', 'bottom', 'left', 'right']:
    ax.spines[i].set_visible(False)

plt.grid(b=None)

ax.xaxis.set_major_formatter(ticker.EngFormatter())

ax.tick_params(axis='x', colors='white')
ax.tick_params(axis='y', colors='white')
ax.xaxis.label.set_color('white')
ax.yaxis.label.set_color('white')

mpl.rcParams['font.family'] = 'Source Sans Pro'

plt.xlabel('SalePrice', fontsize=11);

plt.text(11.27, 1.25, "SalePrice", size=22, color="#ff7f50", fontweight="bold");
plt.text(11.92, 1.25, "Distribution", size=22, color="grey", fontweight="bold");

categ_cols = df.dtypes[df.dtypes == np.object]
categ_cols = categ_cols.index.tolist()

df_enc = pd.get_dummies(df, columns=categ_cols, drop_first=True)
X = df_enc[df_enc['train']==1]
test = df_enc[df_enc['train']==0]
X.drop(['train'], axis=1, inplace=True)
test.drop(['train'], axis=1, inplace=True)

from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=12345)
def rmse(ytrue, ypredicted):
    return np.sqrt(mean_squared_error(ytrue, ypredicted))
```

```
lasso = Lasso(max_iter = 100000, normalize = True)

lassocv = LassoCV(alphas = None, cv = 10, max_iter = 100000, normalize = True)
lassocv.fit(X_train, y_train)

lasso.set_params(alpha=lassocv.alpha_)
lasso.fit(X_train, y_train)

print('The Lasso I:')
print("Alpha =", lassocv.alpha_)
print("RMSE =", rmse(y_test, lasso.predict(X_test)))
The Lasso I:
Alpha = 6.872779832960424e-05
RMSE = 0.12072477839971668


alpha = np.geomspace(1e-5, 1e0, num=6)
lasso_cv_model = LassoCV(alphas = alpha, cv = 10, max_iter = 100000, normalize = True).fit(X_train,y_train)
lasso_tuned = Lasso(max_iter = 100000, normalize = True).set_params(alpha = lasso_cv_model.alpha_).fit(X_train,y_train)
print('The Lasso II:')
print("Alpha =", lasso_cv_model.alpha_)
print("RMSE =", rmse(y_test, lasso_tuned.predict(X_test)))
The Lasso II:
Alpha = 0.0001
RMSE = 0.11962451145557704

alphas = np.geomspace(1e-9, 5, num=100)

ridgecv = RidgeCV(alphas = alphas, scoring = 'neg_mean_squared_error', normalize = True)
ridgecv.fit(X_train, y_train)

ridge = Ridge(alpha = ridgecv.alpha_, normalize = True)
ridge.fit(X_train, y_train)

print('Ridge Regression:')
print("Alpha =", ridgecv.alpha_)
print("RMSE =", rmse(y_test, ridge.predict(X_test)))
Ridge Regression:
Alpha = 0.21251935471767855
RMSE = 0.12350180903194746
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import RobustScaler
from sklearn.model_selection import KFold
from sklearn.svm import SVR


kf = KFold(shuffle=True, random_state=1234, n_splits=10)

X_train_scale = RobustScaler().fit_transform(X_train)
X_test_scale = RobustScaler().fit_transform(X_test)

parameters = {'C':[20, 30, 40], 'gamma': [1e-4, 3e-4, 5e-4],'epsilon':[0.1, 0.01, 0.05]}
svr = SVR(kernel='rbf')
clf = GridSearchCV(svr, parameters, cv=kf)
clf.fit(X_train_scale,y_train)
clf.best_params_
{'C': 20, 'epsilon': 0.01, 'gamma': 0.0003}
svr = SVR(kernel ='rbf', C= 20, epsilon= 0.01, gamma=0.0003)
svr.fit(X_train_scale,y_train)

print('SVR Regression:')
print("RMSE =", rmse(y_test, svr.predict(X_test_scale)))
SVR Regression:
RMSE = 0.12708575951074244

print('Out of {} coefficients, {} are non-zero with Lasso.'
     .format(len(lasso_tuned.coef_), len(lasso_tuned.coef_.nonzero()[0])))
Out of 288 coefficients, 116 are non-zero with Lasso.

coefs = pd.Series(lasso_tuned.coef_, index = test.columns)

lasso_coefs = pd.concat([coefs.sort_values().head(10),
                         coefs.sort_values().tail(10)])

lasso_coefs = pd.DataFrame(lasso_coefs, columns=['importance'])

fig, ax = plt.subplots(figsize =(11, 9))

ax.barh(lasso_coefs.index, lasso_coefs.importance, align='center',
        color = np.where(lasso_coefs['importance'] < 0, 'crimson', '#89CFF0'))

plots_design()

plt.text(-0.22, 20.5, "Feature Importance", size=24, color="grey", fontweight="bold");
plt.text(-0.063, 20.5, "using", size=24, color="grey");
plt.text(-0.0182, 20.5, "Lasso Model", size=24, color="#89CFF0", fontweight="bold");
```