Aplikacija za predikciju rezervacija otkazanih na temelju vremenske prognoze

Buhin, Kristian

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

KRISTIAN BUHIN

APLIKACIJA ZA PREDIKCIJU REZERVACIJA OTKAZANIH NA TEMELJU VREMENSKE PROGNOZE

Završni rad

Pula, rujan, 2020. godine

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

KRISTIAN BUHIN

APLIKACIJA ZA PREDIKCIJU REZERVACIJA OTKAZANIH NA TEMELJU VREMENSKE PROGNOZE

Završni rad

JMBAG: 0303071183, izvanredni student

Studijski smjer: Informatika

Predmet: Programsko inženjerstvo

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: informacijski sustavi i informatologija

Mentor: izv. prof. dr. sc. Tihomir Orehovački

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Kristian Buhin, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

Buhin K

U Puli, 23. rujna, 2020. godine

IZJAVA

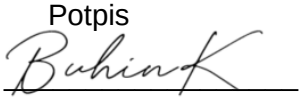
o korištenju autorskog djela

Ja, Kristian Buhin dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom APLIKACIJA ZA PREDIKCIJU REZERVACIJA OTKAZANIH NA TEMELJU VREMENSKE PROGNOZE koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 23.rujna, 2020. (datum)

Potpis



Sadržaj:

Sažetak

Model strojnog učenja za predviđanje otkazanih rezervacija na temelju vremenske prognoze iz ovog projekta kao i sučelje aplikacije razvijeni su u svrhu maksimiziranja dobiti pri upravljanju prihodima u sektoru turizma i ugostiteljstva. Web aplikacija služi za prikaz rezultata modela te osnovne operacije nad određenim skupom podataka koji se koriste u samom radu, dok se prikupljanje, analiza, obrada i procesiranje podataka periodički izvršavaju u pozadinskom dijelu te samim time isključuju mogućnost pogreške od strane korisnika.

Krajnje točke aplikacije razvijene su tako da se model strojnog učenja lako integrira u bilo koji sustav, bilo da je to neki centralni rezervacijski sustav kao kanal ulaznih podataka ili druga baza podataka kao spremište za sve rezultate obrade i analize podataka. U odnosu na ostala, trenutno dostupna programska rješenja koja se koriste u toj poslovnoj domeni, njegova prednost jest da njegov rad ne zahtijeva poveću količinu resursa te nema potrebe za instalacijom dodatnih struktura za podršku.

Da rješenje bude u obliku web aplikacije dodatan je bonus jer je za korištenje dovoljan samo internetski preglednik. Važno je i spomenuti da srodna programska rješenja, razvijena od strane veći tvrtki, procesiraju mnogo veći broj ulaznih značajki, odnosno uzimaju puno širi kontekst pri optimizaciji upravljanja prihoda. Gledajući s te strane, tu leži najveće ograničenje trenutnog modela jer je on usko specificiran za predviđanja otkaza rezervacija isključivo na temelju vremenske prognoze.Međutim, ta činjenica može svakako poslužiti kao dobra smjernica pri dodatnom razvoju modela.

Ključne riječi: strojno učenje, upravljanje prihodima, predikcije, optimizacija, vremenska prognoza

Abstract

A machine learning model for predicting canceled reservations based on the weather forecast from this project as well as an application interface were developed in order to maximize profits in revenue management in the tourism and hospitality sector. The web application is used to display the results of the model and for the basic operations on a particular set of data used in the work process, while data collection, analysis, and processing are periodically performed in the background and thus exclude the possibility of user errors.

Application endpoints have been developed so that the machine learning model can be easily integrated into any system, whether it is a central reservation system as an input data channel or another database as a repository for all data processing and analysis results. Compared to other, currently available software solutions used in this business domain, its advantage is that its operation does not require a larger amount of resources and there is no need to install any additional support structures.

To make the solution in the form of a web application, is an additional bonus, because only an internet browser is enough to use it. It is also important to mention that related software solutions, developed by larger companies, process a much larger number of input features, that is, they take a much broader context in optimizing revenue management. From this point of view, this is the biggest limitation of the current model because it is narrowly specified for forecasting cancellations based solely on the weather forecast. However, this fact can certainly serve as a good guide in further model development.

Keywords: machine learning, revenue management, predictions, optimization, weather forecast

1.Uvod

Upravljanje prihodima je u današnje vrijeme ključan koncept u poslovanju bilo koje tvrtke. Naročito dolazi do izražaja u ugostiteljstvu i turizmu iz razloga što hotelijeri imaju potrebu „boriti“ se s unaprijed određenim troškovima poput utroška rada, tekuće imovine, različitih nameta i amortizacije kako bi ostvariti maksimalni prihod i održali ili u određenom postotku čak povećali profitabilnost vlastite tvrtke.

U tu svrhu koriste se različiti alati i sustavi koji uključuju korištenje velikog broja podataka za analizu radi predviđanja stupnja potražnje za određenim uslugama koje pripadaju portfelju te iste tvrtke ili pak za donošenje nekih više ili manje važnih strateških odluka. Konačni cilj korištenja takvih alata jest da omoguće informiranije odluke kako bi se poslovna praksa maksimalno optimizirala i samim time osigurala pokrivanje svih prethodno navedenih troškova i u konačnici maksimizirala profit.

Generalni naziv za takve sustave je *Revenue management system* (RMS) te ukoliko bi se jedan takav sustav podijelio na jednostavne funkcije, one bi bile sljedeće:

- Sposobnost brzog i preciznog izračuna optimalnih cijena smještajnih jedinica
- Pružanje ključnih pokazatelja o učinku u fazi analize i planiranja
- Procjena prihoda za različite cjenovne strategije

Zašto su sve prethodne informacije bitne i kako ih povezati sa domenom strojnog učenja. Odgovor je veoma jednostavan. Kada se pobliže pogledaju osnovne funkcije sustava za upravljanje prihodima, vidljivo je da je većina njih definirana na temelju nekih povijesnih podataka, bilo da su to nedavni trendovi ili događaji unazad nekoliko godina, dok konačan cilj stvara projekciju i daje uvid u neko buduće vrijeme. Ista definicija se može primijeniti i na veliki postotak modela za strojno učenje, specijaliziranih za predikciju, u današnje doba. Ova činjenica dolazi do još većeg izražaja kod modela za nadgledano strojno učenje gdje se na temelju ulaznih parova različitih značajki i klasnog atributa pokušava shvatiti njihov međusobni odnos te u

konačnici razviti model koji za buduće ulazne značajke može sa velikom sigurnošću zaključiti koja će biti vrijednost klasnog, odnosno ciljnog atributa.

U idućim poglavljima ovog rada primarni fokus biti će baš na takvom tipu modela za strojno učenje. Pravilna artikulacija samog problema biti će temelj na kojem će se bazirati ostale faze razvoja samog modela poput definiranja izvora podataka, reprezentativnosti podataka, njihove povezanosti sa ciljnim atributom te da li će model u konačnici biti sposoban pružiti odgovor na postavljeni problem, odnosno u kojoj mjeri će taj odgovor biti pouzdan. Naravno, svaka faza razvoja modela biti će zasebno popraćena i opisana u nastavku samog rada sa posebnim naglaskom na fazu pretprocesiranja podataka potrebnih za funkcioniranje modela koja će se popratiti nizom dijagrama i koji služe za što bolju identifikaciju relevantnih ulaznih značajki te fazu ocjenjivanja preciznosti donesenih odluka modela analizom dobivenih preliminarnih rezultata.Za kraj, popis osnovnih alata koji će se koristiti za razvoj cijele aplikacije čine: Visual Studio Code (programski jezik Python), Postgresql baza podataka te Javascript knjižnica „Vue.js”.

2. Opis problema

Nastavno na temu upravljanja prihoda, vrijeme da se definira problematika kojom će se model strojnog učenja baviti. Kao dodatni mali uvod, važno je spomenuti da je ključna osobina kvalitetnog upravljanja prihodima anticipacija. Preciznije rečeno, što je predviđanje kretanja potražnje za dani period točnije, kvaliteta upravljanja prihodima je viša te se ostvaruje veći prihod. Međutim, velika segmentacija i granulacija tržišta veoma otežava kvalitetnu anticipaciju. Potrebna je visoka razina razumijevanja tržišnih segmenata, distribucijskih kanala kao i detalja vlastitog proizvoda ili usluge kako bi se prepoznali ključni faktori za što uspješnije upravljanje. Iako su navedene kategorije kod upravljanja prihodima dosta kompleksne, one se u velikom postotku mogu kvantificirati i uz pravilnu analizu postati čvrst temelj za donošenje odluka pozitivnih za poslovanje.

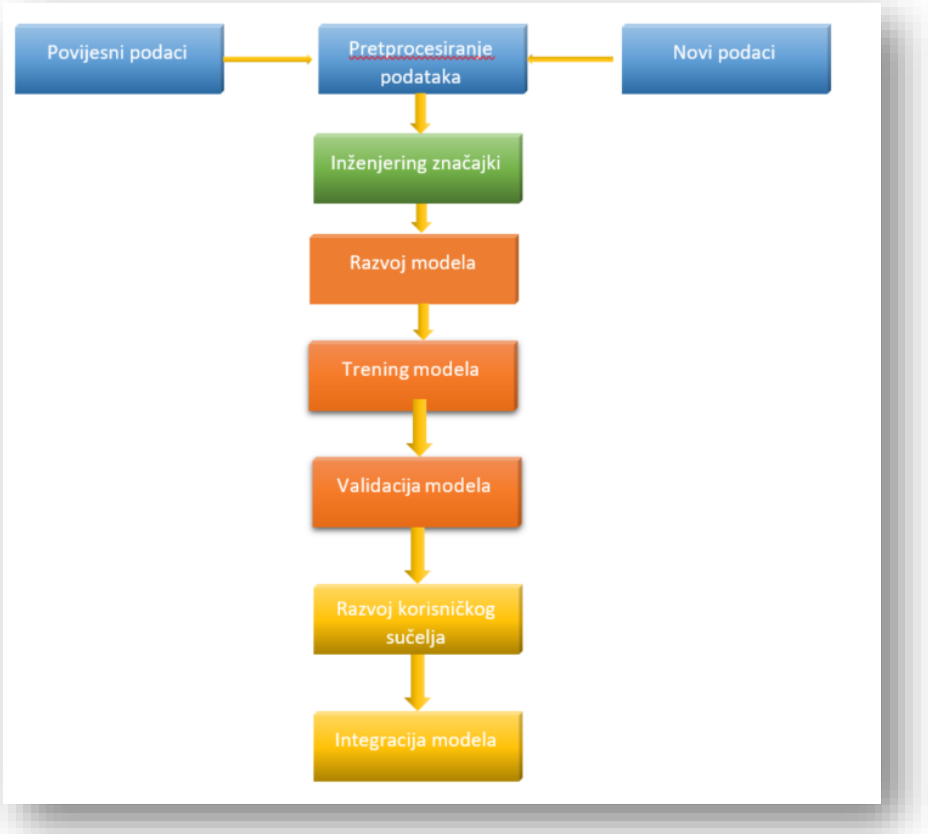
S druge pak strane, osim navedenih elemenata, postoje i elementi koji su u puno većem postotku nepravilniji i teži za analizu. Jedan od tih elemenata je kretanje popunjenosti objekata za vrijeme lošijih vremenskih uvjeta. Taj scenarij je puno više izražen u turističkim destinacijama koje se nalaze blizu obala mora ili zimskih odmarališta ili skijališta te samim time igra veliku ulogu u upravljanju prihodima. Taj scenarij ujedno je i uvod u problematiku modela za strojno učenje. Naime, zadatak modela, kojeg ovaj rad opisuje, jeste predvidjeti broj jedinica ili rezervacija koje će se otkazati u slučaju najave određenih vremenskih uvjeta, odnosno prema vremenskoj prognozi u nadolazećem vremenskom periodu od 8 dana. Još preciznije rečeno, model će, na temelju svojstava rezervacija te detalja u samoj vremenskoj prognozi, klasificirati rezervaciju kao otkazanu ili neotkazanu. Samim time, pod pretpostavkom da je pouzdanost modela veoma visoka, upravljanje prihodima postaje za razinu jednostavnije i olakšava donošenje specifičnih odluka, poput koliko smještajnih jedinica preko kapaciteta smještajnog objekta je potrebno prodati da bi se uz predviđeni broj otkazanih jedinica, objekt u potpunosti napunio ili koliko mora biti optimalna cijena određenih tipova smještajnih jedinica da se ostvari maksimalan prihod.

Iz prethodnog se da naslutiti da se u ovom scenariju radi o problemu binarne klasifikacije s obzirom da klasni, odnosno ciljni atribut može imati samo dva stanja, otkazano ili neotkazano. U svrhu lakšeg praćenja, potrebno je samo napomenuti i da će se u daljnjem tekstu za neotkazane rezervacije koristiti i izrazi stornirane ili nerealizirane rezervacije. Binarna klasifikacija je vjerojatno najčešći problem s kojim se svatko susreće u svakodnevnom životu. Posebice to vrijedi za diskretizaciju, definiranu kao proces u kojem se kontinuirane vrijednosti i varijable pretvaraju u njihove diskretne ekvivalente. Primjer toga je čitanje poruke e-pošte gdje se poruke prema sadržaju ili pošiljatelju klasificiraju kao relevantne ili nerelevantne, odnosno da li je poruka neželjena pošta ili ne. U medicini kod dijagnostike da li je bolest prisutna ili ne na temelju rezultata različitih pretraga, ili pak kod jednostavnog odlaska u trgovinu i odluke da li će se neki proizvod kupiti ili ne na temelju svojih svojstava. Isti princip primijeniti će se kod razvoja dotičnog modela te će se početna heuristika bazirati na istim temeljima kao prethodno navedene svakodnevne odluke, odnosno putem određenog seta ulaznih informacija tražiti će se precizan konačni rezultat.

Cilj ovog prvog stadija jeste razviti jednostavan logički model kao solidnu osnovnu bazu za daljnje faze razvoja ciljanog modela. Početni jednostavni model u velikoj mjeri može pomoći da se utvrdi je li složeni model uopće potreban, odnosno da li postoji dovoljno veliko poboljšanje kvalitete modela da opravda daljnji razvoj. Prvi korak ka tome biti će identifikacija početnog skupa podataka koji će prolaziti kroz faze obrade podataka (eng. *data wrangling*), odnosno etape filtriranja kako bi se dobio konačni set podataka koji će služiti kao ulazni atributi za izgradnju modela kod navedenog problema binarne klasifikacije. Te faze sastoje se od filtriranja provjere varijancija, odbacivanja nepotrebnih nizova podataka, obrade podataka sa praznim (eng. *null)* vrijednostima, tj. praznim zapisima, otkrivanja odstupanja od standardnih vrijednosti kod pojedinih ulaznih značajki kao i odstranjivanja šumova u samim podacima.

Nakon navedenih koraka, ulazne značajke promatrati će se kroz prizmu tipova podataka. Numeričke, kao i kategoričke varijable proći će kroz transformacije koje uključuju kodiranje, spajanje, skaliranje te smanjenje dimenzija. Normalizacija i standardizacija značajki u skupu podataka potrebna je prije pokretanja bilo kakvog algoritma obrade nad tim istim skupom podataka. Kada se kroz analizu uvidi da je obrađeni skup podataka pouzdan i može služiti kao ulaz modelu, idući stadij je razvoj samo modela.

Definiranje setova podataka za treniranje i evaluaciju modela, neophodan je uvodni korak u razvoju modela. Prilikom treniranja, odnosno „učenja" modela koristiti će se niz hiperparametara i metrika kako bi se odabrao najoptimalniji skup koji daje najpouzdaniji rezultat. Rezultat će se promatrati kroz metrike poput *receiver operating characteristic* krivulje (ROC) i *precision-recall* krivulje (PR), konfuzijske matrice te izvještaja klasifikacije. Nakon što metrike potvrde da je model zadovoljavajući krenuti će se u finaliziranje istog i izgradnju korisničkog sučelja. Za sučelje će se koristiti „Vue.js" razvojni okvir te će se i taj proces dokumentirati. Konačni izgled aplikacije biti će prikazan nizom snimki zaslona koje će prikazati mogućnosti i opcije te otkriti koje sve funkcionalnosti stoje na raspolaganju korisniku. Za kraj ovog uvodnog dijela, Na slici 1 se može vidjeti vizualizacija osnovnog dijagrama toka iznad navedenih stadija razvoja modela .



Slika 1. Osnovni dijagram toka razvoja modela i aplikacije

3. Opis početnog skupa podataka

Kada se gleda iz perspektive stvarnog svijeta, podaci koji su dostupni su u različitim oblicima te nisu nužno uvijek potpuni. Čak naprotiv, podaci s kojima se raspolaže često imaju dosta nedefiniranih vrijednosti u sebi. Ovo je scenarij i u setu podataka s kojim se raspolaže za ovaj razvoj ovog modela. U pripremnoj fazi analizirat će se baš taj aspekt seta podataka te utvrditi u kojoj mjeri su sami podaci cjeloviti te koliko je korekcija potrebno da on postanu reprezentativni u smislu ulaznih značajki u procesu nadgledanog učenja. S obzirom da je zadatak modela strojnog učenja klasifikacija rezervacija kao otkazanih ili neotkazanih u kraćem nadolazećem vremenskom intervalu na temelju informacija samim tih rezervacija i vremenske prognoze, u ovom slučaju prvenstveno se radi o dva skupa podataka. Prvi skup su podaci o rezervacijama u proteklom periodu od 3 godine, odnosno sve rezervacije dotičnog hotela od početka 2017. godine pa do kraja 2019. godine. Drugi skup podataka su povijesni podaci vremenskih prognoza za taj isti period. Kako su ta dva

skupa odvojena jer dolaze iz različitih izvora analizirati će se svaki zasebno te u konačnom stadiju adekvatno ujediniti kako bi činili cjelinu i potpun set ulaznih značajki.

3.1 Skup podataka o rezervacijama

Prije samog učitavanja seta podataka o rezervacijama, važno je spomenuti tehničke detalje o samom hotelu koji je promatran. Hotel u svom sklopu ima 100 soba različitih kategorija na tri etaže te pruža različite usluge poput noćenja s doručkom, polupansiona ili punog pansiona. Maksimalan dnevni broj gostiju koji hotel može ugostiti je 214 te raspolaže jednim restoranom, unutarnjim bazenom i vanjskim parkingom.

Nakon što se, u obliku tablice pomoću *pandas* knjižnice, učita set podataka o samim rezervacijama u periodu koji je prethodno naveden može se vidjeti da se set podataka sastoji od 25067 zapisa, odnosno redaka te 40 stupaca, odnosno značajki za svaku rezervaciju kao što je vidljivo na slici 2 ispod.

```
from Domain import *
from Models import *
import pandas as pd

lista = Rezervacije.listaj()
df = pd.DataFrame(lista)
df.shape
```

```
(25067, 40)
```

Slika 2. Dimenzije skupa podataka s podacima o rezervacijama

Svaki redak opisuje jednu rezervaciju te sadrži stupce koji su specificirani na slici ispod. Na slici 3 vidljivo je da postoji nekoliko kolona koje služe isključivo za numeraciju i identifikaciju same rezervacije poput stupaca *id*, *sif_hotela*, *sif_rezervacije*. Odmah se može zaključiti da su ti stupci nepotrebni za daljnje procesiranje te ih se može eliminirati za daljnje faze obrade podataka.

```
df.columns
```

```
Index(['id', 'sif_hotela', 'godina', 'sif_rezervacije', 'rbr_stavke',
       'vrijeme_kreiranje', 'datum_kreiranja', 'datum_od', 'datum_do',
       'broj_dana', 'sif_usluge', 'status_rezervacije',
       'datum_do_potvrde_opcije', 'sif_drzave', 'sif_agencije', 'tip_ro',
       'obaveza_akontacije', 'iznos_akontacije', 'storno', 'vrijeme_storna',
       'datum_storna', 'broj_osoba', 'broj_djece', 'broj_soba', 'nocenja',
       'jedinice', 'cijena_pans_usl', 'iznos_bruto', 'valuta', 'tecaj',
       'lead_time_dani', 'dat_storna_do_dat_dolaska', 'tip_garancije',
       'postotak_akontacije', 'mjesec', 'godina_rezervacije', 'vrsta_sobe',
       'kanal', 'nacin_rezervacije', 'vrsta_sobe_naplata'],
      dtype='object')
```

Slika 3. Popis stupaca u setu podataka

Idući prikaz je statistički opis numeričkih stupaca u cijelom setu podataka, pomoću naredbe *df.describe(),* gdje se može vidjeti broj zapisa u svakom stupcu, srednja vrijednost, standardna devijacija, te vrijednosti po kvantilima , odnosno preciznije rečeno po kvartilima 0% - 25%, 25% - 50%. 50% - 75% te 75% - 100%. U tablici 1 dodatno se mogu vidjeti i minimalna i maksimalna vrijednost za svaki stupac.

Tablica 1. Statistički podaci stupaca

	rbr_stavke	broj_dana	iznos_akontacije	broj_osoba	broj_djece	broj_soba	nocenja	jedinice
count	25067	25067	2919	25067	25067	25067	25067	25067
mean	1.632265528	4.191885746	3184.573436	2.404196753	0.04049148283	1.235927714	10.00454781	5.093110464
std	3.665152782	4.805068891	7603.696112	3.661757608	0.2294711991	1.897714797	19.39777453	10.49973817
min	1	1	0	0	0	0	0	0
25%	1	2	0	2	0	1	4	2
50%	1	3	0	2	0	1	6	3
75%	1	6	2923.035	2	0	1	12	6
max	93	317	73968.92	86	4	49	816	408

	cijena_pans_usl	iznos_bruto	tecaj	lead_time_dani	dat_storna_do_dat_dolaska	postotak_akontacije	mjesec	godina_rezervacije
count	25067	25067	25067	25067	6771	23541	25067	25067
mean	1191.685859	3607.723219	7.591477241	59.7374636	51.36168956	0.6325134871	6.268999082	2018.158974
std	1429.006828	4831.863468	0.03347526764	78.33806406	63.83406436	6.808092398	2.834887824	0.9087350779
min	0	0	7.46	-145	-12	0	1	2017
25%	385.455	776.6	7.6	5	6	0	4	2017
50%	744.76	1988.84	7.6	23	23	0	7	2018
75%	1298.52	5029.875	7.6	92	77	0	8	2019
max	20452.17	127693.53	7.6	566	508	100	12	2021

Ono što se može zaključiti iz prikaza jeste da u skupu podataka postoje velike razlike u obujmu u kojem se vrijednosti pojedinačnih stupaca kreću, Dobar primjer jesu stupac *iznos_akontacije* kojemu se vrijednosti kreću u rasponu od 0 do 73968.92 te stupac *postotak_akontacije* koji ima raspon od 0 do 100. Takva struktura podataka govori da će u budućim koracima biti potrebna neka vrsta standardizacije kako bi se podaci mogli što vjerodostojnije tumačiti od strane modela.

Idući korak jeste naredba *df.isna().sum()* kojom se prikazuje zbroj praznih (*null*) zapisa u svakom stupcu seta podataka. Identifikacija broja praznih vrijednosti veoma je važan korak pri procesiranju podataka jer većina modela strojnog učenja nije u mogućnosti obrađivati podatke gdje postoji veliki postotak takvih zapisa. Iz tog razloga biti će potrebna dodatna obrada kako bi se ti podaci mogli ispravno interpretirati. Taj proces biti će opisan u kasnijem poglavlju ovom rada. Pokrenuvši gore navedenu naredbu, u dobivenom formatu na slici 4 mogu se vidjeti dva stupca, prvi sa nazivom stupca te drugi koji sadrži broj praznih vrijednosti koji se nalazi u tom istom stupcu.

```
df.isna().sum()
```

```
id                              0
sif_hotela                      0
godina                          0
sif_rezervacije                 0
rbr_stavke                      0
vrijeme_kreiranje               0
datum_kreiranja                 0
datum_od                        0
datum_do                        0
broj_dana                       0
sif_usluge                      0
status_rezervacije              0
datum_do_potvrde_opcije     22780
sif_drzave                      0
sif_agencije                    0
tip_ro                          0
obaveza_akontacije              0
iznos_akontacije            22148
storno                          0
vrijeme_storna              18296
datum_storna                18296
broj_osoba                      0
broj_djece                      0
broj_soba                       0
nocenja                         0
jedinice                        0
cijena_pans_usl                 0
iznos_bruto                     0
valuta                          0
tecaj                           0
lead_time_dani                  0
dat_storna_do_dat_dolaska   18296
tip_garancije                 198
postotak_akontacije          1526
mjesec                          0
godina_rezervacije              0
vrsta_sobe                      0
kanal                           0
nacin_rezervacije               0
vrsta_sobe_naplata              0
dtype: int64
```

Slika 4.
Rezervacije - detalji o broju praznih zapisa

Daljnjom analizom, može se vidjeti da je velika većina značajki bez praznih vrijednosti te nad njima neće biti potrebno raditi kompenzaciju za iste.

Značajke koje pak odskaču su sljedeće:

* datum_do_potvrde opcije
* iznos_akontacije
* vrijeme_storna
* datum_storna
* datum_storna_do_datuma_dolaska
* tip_garancije

* postotak_akontacije

Kako bi se moglo pravilno nastaviti s obradom podataka kod ovih značajki potrebno je objašnjenje poslovnog procesa iz same domene poslovanja u svrhu identifikacije što te značajke opisuju kod same rezervacije.

Kod stupca *datum_do_potvrde_opcije* sam naziv stupca ne otkriva puno dok se ne objasni da svaka rezervacija može imati dva statusa nakon što se izvrši. Prvi je status takozvane opcije. To znači da u trenutku rezervacije, sama rezervacija nije finalno potvrđena potrebnim sredstvima te je nositelju rezervacije dana mogućnost da u nekom roku potvrdi ili otkaže rezervaciju u tom statusu. Takve rezervacije imati će kao vrijednost u promatranom stupcu datum do kojega je moguće finalizirati proces i potvrditi rezervaciju. Drugi status je „potvrđeno", što samo po sebi govori da je u trenutku same rezervacije proces potvrde finaliziran. Kod tog scenarija, vrijednost u stupcu biti će prazna. Što to znači za gore navedenu kolonu? Govori da se u jako velikom postotku, u ovom slučaju brojevi pokazuju preko 90,8% (kada se broj praznih zapisa podijeli sa ukupnim brojem zapisa), rezervacija odmah potvrdi i proces potvrde bude finaliziran.

Drugi promatrani stupac pod nazivom *iznos_akontacije* je dosta lako za objasniti. Nastavno na detalje iznad, jedno od mogućih načina potvrde rezervacije jest uplata avansa ili akontacije. U slučaju da nositelj rezervacije odabere ovaj način potvrde rezervacije, polje će kao vrijednost imati iznos koji je potrebno uplatiti da se rezervacija potvrdi.

Iduće dvije značajke usko su vezane sa klasnim atributom koji će konačni model morati klasificirati. Naime oba stupca, *vrijeme_storna* i *datum_storna,* biti će prazna ako rezervacija nije otkazana. Ukoliko je rezervacija otkazana, njihove vrijednosti biti će vrijeme i točan datum kada se to dogodilo.

Za stupac *tip_garancije* bitno je spomenuti da je to zapis koji govori kako je rezervacija bila ili trebala biti potvrđena dok se značajka *postotak_akontacije* koristi kao dodatni alat kada se, umjesto točno definiranog novčanog iznosa avansa, pri procesu rezervacije definira da će se uplatiti određeni postotak, npr. 15%, od ukupnog iznosa boravka. Kod stupaca gdje ne postoje prazni zapisi, sami nazivi otkrivaju što koja

značajka definira kod same rezervacije. Najvažnije je da se spomene stupac *storno* koji je ustvari cilj konačnog modela i cijelog projekta.

```
df['storno'].unique()

array(['DA', 'NE'], dtype=object)
```

Slika 5. Klasni atribut klasifikacije

Na slici 5 je vidljivo da ta značajka potvrđuje da je odabir binarne klasifikacije ispravno rješenje za opisani problem jer postoje samo dvije moguće vrijednosti za nju.

3.2. Skup podataka o prognozama

Drugi skup podataka je onaj o podacima vremenskih prognoza. Isti princip analize primijeniti kao i maloprije primijeniti će se i ovaj set podataka.

```
from Domain import *
from Models import *
import pandas as pd

lista = VremePrognoze.listaj_iz_baze()
df = pd.DataFrame(lista)
df.shape


(17457, 19)
```

Slika 6. Dimenzije skupa podataka s podacima o prognozama

Slika 6 pokazuje da kod ovog skupa podataka postoji 17457 redaka te 19 značajki, odnosno stupaca. Za uvod je bitna informacija da vrijeme ima šest glavnih komponenata koje ga opisuju. To su temperatura, atmosferski tlak, vjetar, vlaga, oborine i oblačnost. Slika 7 prikazuje nazive značajki i vidljivo je da podaci sadrže svih šest navedenih komponenata uz još dodatne atribute.

```
df.columns
```

```
Index(['id', 'datum', 'datum_prognoze', 'temp_prosjek', 'temp_max', 'temp_min',
       'vidljivost', 'smjer_vjetra_stupnjevi', 'brzina_vjetra',
       'nalet_vjetra_brzina', 'relativna_vlaznost', 'tlak_zraka', 'uv_index',
       'oblaci_pokrice', 'oborine_akumulirano', 'dubina_snijega',
       'rosa_prosjek', 'prognoza', 'preostalo_dana'],
      dtype='object')
```

Slika 7. Popis stupaca u setu podataka s prognozama

Ono što je zapisano u samoj tablici (eng. *dataframe(df)*) jesu povijesni podaci vremenskih prognoza za svaki dan od 01.01.2017. godine pa sve do 31.12.2019. godine. Ukupno su dokumentirana 1094 dana, međutim, postoji više od 17450 zapisa u promatranoj tablici. To je iz razloga što je svakom datumu unutar navedenog perioda pridružena prognoza za svaki od 16 do 0 dana prije tog datuma, odnosno svaki datum ima 16 pojedinačnih prognoza. Koja prognoza je bila važeća za koliko dana prije tog datuma je vidljivo u stupcu *preostalo_dana* kao što to prikazuje tablica 2.

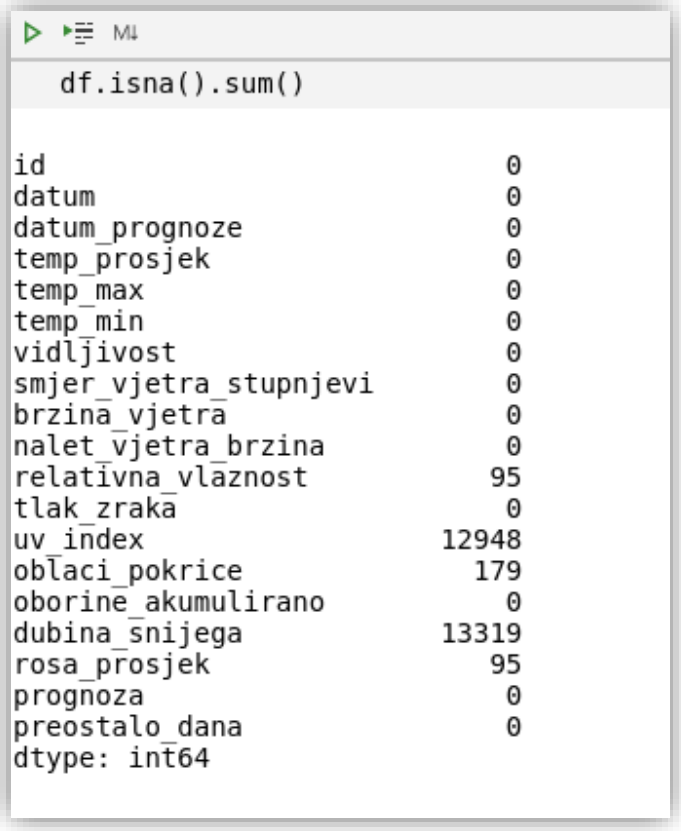
Tablica 2. Prognoze - statistički podaci stupaca

	temp_prosjek	temp_max	temp_min	vidljivost	smjer_vjetra_stupnjevi	brzina_vjetra	nalet_vjetra_brzina	relativna_vlaznost
count	17457	17457	17457	17457	17457	17457	17457	17362
mean	14,5019	19,3534	9,8369	16,6667	150,6136	16,0759	2,0289	70,0100
std	7,4466	7,9656	7,2008	6,4139	37,2302	6,3342	9,0019	13,3004
min	-3,9	-1,1	-10,1	0	21	0,65643	0	23,38
25%	8,7	12,6	4,3	11,6	128,84	12,4	0	60,43
50%	14,1	19,1	10,1	15,8	153	15	0	70,885
75%	21	26,4	16	21,1	175,67	18,4	0	80,81
max	30,1	37,3	25,1	33,4	302	47,7	93,6	96,04

	relativna_vlaznost	tlak_zraka	uv_index	oblaci_pokrice	oborine	dubina_snijega	rosa	preostalo_dana
count	17362	17457	4509	17278	17457	4138	17362	17457
mean	70,0100	1015,4274	27,2030	32,9691	4,0981	0,5777	8,4545	7,8506
std	13,3004	7,8298	9,0407	26,3204	11,5133	1,2505	7,1591	4,3354
min	23,38	973,982	0,679864	0	0	0	-19,4	0
25%	60,43	1011,8	27,8	10,8	0	0	3,5	4
50%	70,885	1015,5	29,5	26,3	0	0	9,4	8
75%	80,81	1019,9	31,9	51,875	1,4	1	14,4	12
max	96,04	1036,4	38,7	100	160	8	22,9	16

I u ovom setu podataka ponavlja se priča kao u prethodnom, odnosno ponovno su rasponi u kojima se vrijednosti stupaca kreću u potpuno drugačijim domenama te će biti potrebna normalizacija.

Ovdje se to može najbolje primijetiti kod stupaca *tlak_zraka* te *temp_prosjek*.



```
df.isna().sum()

id                          0
datum                       0
datum_prognoze              0
temp_prosjek                0
temp_max                    0
temp_min                    0
vidljivost                  0
smjer_vjetra_stupnjevi      0
brzina_vjetra               0
nalet_vjetra_brzina         0
relativna_vlaznost         95
tlak_zraka                  0
uv_index                12948
oblaci_pokrice            179
oborine_akumulirano         0
dubina_snijega          13319
rosa_prosjek               95
prognoza                    0
preostalo_dana              0
dtype: int64
```

Slika 8. Prognoze - detalji o broju praznih zapisa

Provjerom broja praznih zapisa ponovno iskaču određeni stupci sa velikim postotkom istih. Prema slici 8, ovdje je to slučaj sa stupcima *relativna_vlažnost*, *uv_index*, *dubina_snijega* te *rosa_prosjek*. Kod stupca *dubina_snijega* lako je zaključiti da se u polje ne upisuje nikakva vrijednost ako snijega nije bio najavljen dok je kod ostalih jedini mogući zaključak da skup podataka nije potpun te da će biti potrebna dodatna obrada ili odbacivanje istih kako se ne bi narušio integritet samog modela prilikom razvoja.

Uvodnom analizom, na raspolaganju su nove informacije te se može prijeći na idući stadij, a to je obrada samih podataka i kreiranje jednog jedinstvenog skupa značajki.

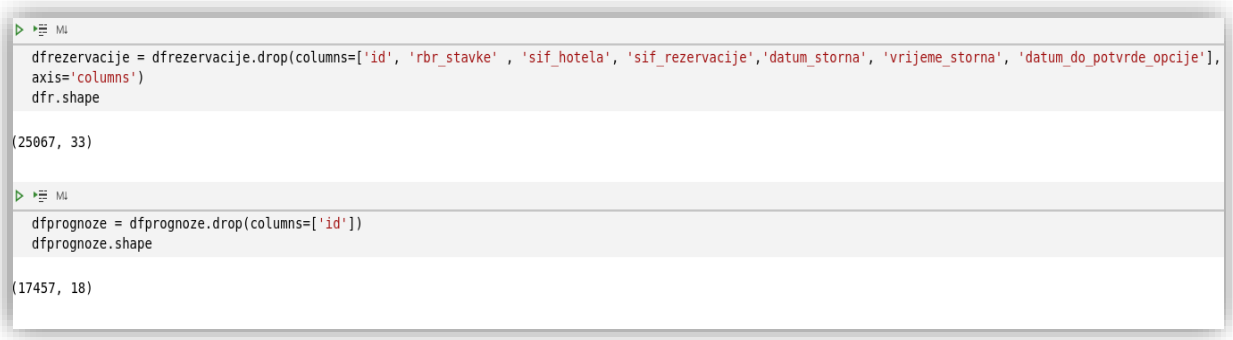
4.Obrada podataka

4.1. Značajke sa slučajnim vrijednostima

Uzimajući u obzir informacije dobivene iz uvodne analize, mogu se odmah primijeniti početni koraci u obradi samih podataka. Ranije spomenuti stupci koji služe samo kao identifikacijske oznake rezervacije su suvišni te ih se naredbom *df.drop* odmah miče iz postojećeg skupa podataka. Razlog micanja iz postojećeg skupa podataka jest činjenica da se vrijednosti u tim stupcima u potpunosti baziraju na slučajnosti. Na primjer, šifra rezervacije se dodjeljuje na način da je to ustvari broj zadnje upisane rezervacije u bazi podataka uvećan za jedan. Kako taj broj nema značenje za samu rezervaciju te je moguća redundantnost (od početka iduće godine, brojač za dodjelu šifri rezervacijama se resetira na nulu), on se može slobodno maknuti iz skupa podataka. Tim korakom skup podataka s rezervacijama smanjen je za šest stupaca te skup podataka sa vremenskim prognozama za jednu (stupac *id*).

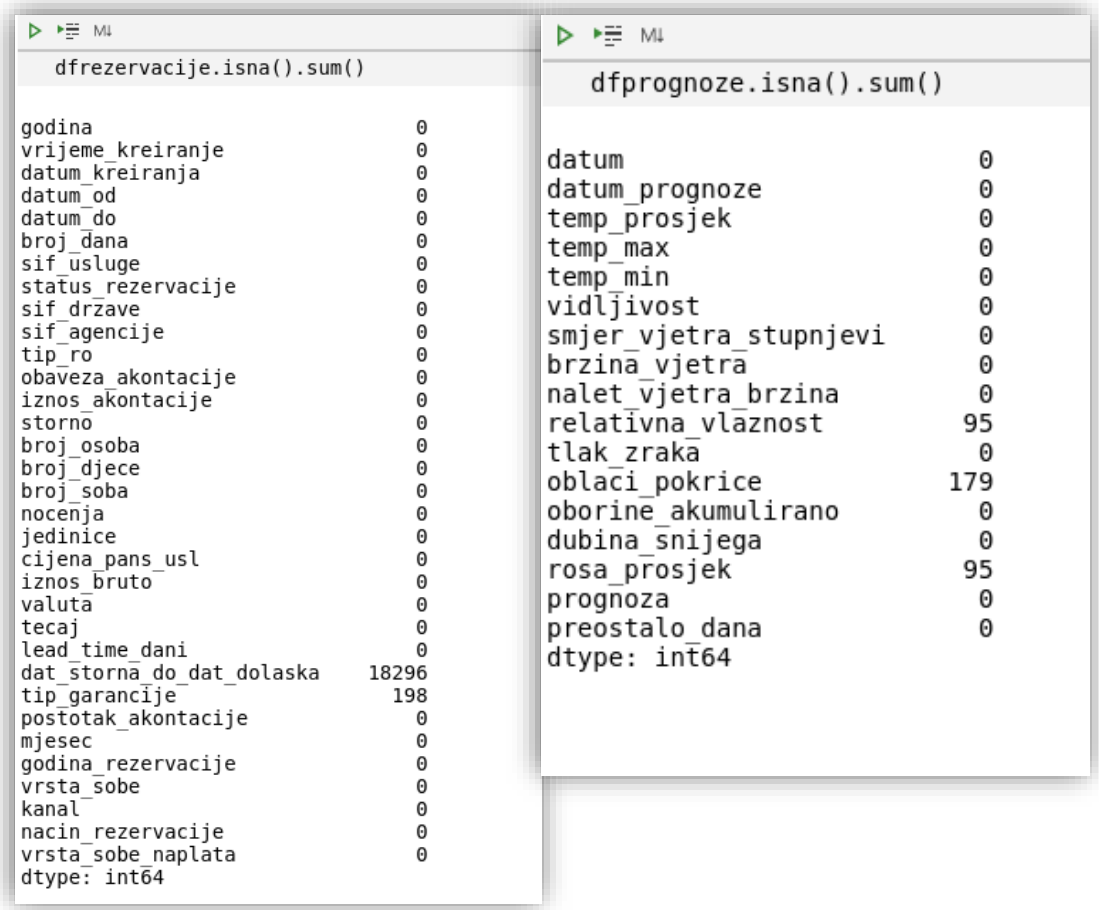
4.2. Prazne vrijednosti

„*Kvaliteta podataka i količina korisnih informacija koje posjeduju, ključni su faktori koji odlučuju koliko dobro model strojnog učenja može učiti*“ *(Raschka i Mirjalili, 2018, str. 107, vlastiti prijevod).* U ovoj fazi, moguće je iz skupa podataka maknuti još dvije značajke. Prva je stupac *datum_do_potvrde_opcije* koji je imao preko 90% praznih zapisa. Kako je to postotak koji je teško kompenzirati tako da on ima smisleno značenje za model, njega se također može ukloniti iz skupa podataka. Druga značajka je iz skupa podataka o vremenskim prognozama, a to je *UV_index*. Nakon odrađenih akcija, dimenzije skupova su reducirane te su prikazane na slici 9 ispod.



Slika 9. Nove dimenzije skupova podataka

Sada su na redu preostali stupci koji su u prethodnoj analizi sadržavali veliki broj praznih zapisa. Postoji nekoliko načina kako obraditi takve zapise, a načini ovise o postotku u kojem se oni nalaze unutar cijelog skupa. Prvi način jest zamjena praznih zapisa nekom vrijednosti koja neće imati negativan utjecaj na model prilikom treniranja te neće uvelike utjecati na konačan rezultat. Za primjer uzmimo stupac *iznos_akontacije*. Prethodna analiza je pokazala da taj stupac ima preko 90% praznih zapisa. Uzimajući u obzir kontekst i značenje tog stupca, odmah se kao rješenje nameće prijedlog da se prazni zapisi u tom stupcu zamijene s vrijednošću nula. Objašnjenje je veoma jednostavno. Ukoliko je zapis prazan, potreban iznos akontacije za potvrdu rezervacije bio je ravan nuli te je to opravdana zamjenska vrijednost.



Slika 10. Suma praznih zapisa nakon obrade

Isti način razmišljanja te rješenje možemo primijeniti i na stupac *dubina_snijega* u skupu podataka s vremenskim prognozama. Ukoliko snijeg nije najavljen, dubina

snijega iznosila je nula. Također se isto može primijeniti na stupce *rosa_prosjek* te *postotak_akontacije*. Nakon primijenjenih akcija, broj praznih zapisa prema stupcima u oba skupa podataka može se vidjeti na slici 10.

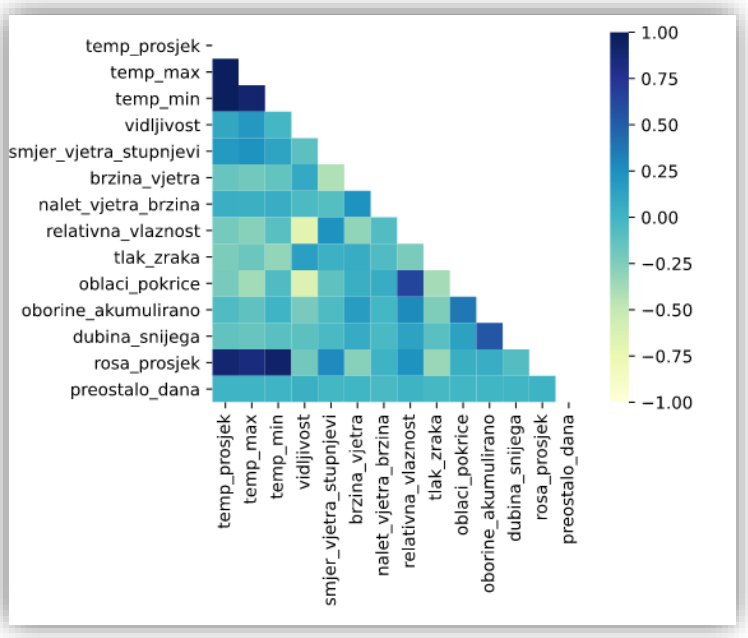
Izuzev namjerno ostavljenog stupca *dat_storna_do_dat_dolaska*, preostalo je još samo nekoliko značajki sa manjim postotkom praznih zapisa. Prije nego uklonimo ili zamijenimo te zapise nekom drugom vrijednošću, veoma je bitno spomenuti da sa brisanjem tih redaka postoji mogućnost da se obriše neka potencijalno relevantna informacija iz nekog drugog stupca. Prije nego se odlučimo na akciju brisanja potrebna je detaljnija analiza.

4.3. Matrice korelacija

Idući korak je promatranje korelacija između stupaca u dva skupa podataka. Korelacija je ustvari međusobni odnos te je veoma praktičan način da se razumije ovisnost između više varijabli i atributa u samom skupu podataka. Koristeći korelaciju može se dobiti uvid da li jedan ili više atributa ovise o nekom drugom atributu unutar samog skupa podataka. Iz tog razloga koristi se kao osnovna veličina za mnoge tehnike modeliranja. Dvije su osnovne kategorije korelacija, pozitivna i negativna. Ukoliko postoji pozitivna korelacija, atribut koji ovisi o nekom drugom atributu rasti ili padati će paralelno s njim dok je kod negativna korelacija situacija suprotna. S rastom osnovnog atributa, ovisni atribut će padati ili obrnuto. Raspon vrijednosti u kojem se te dvije kategorije kreću je 0 do 1 za pozitivnu korelaciju te 0 do -1 za negativnu korelaciju.

Zašto je korelacija bitna za trenutne skupove podataka? Ako skup podataka ima atribute u gotovo savršenoj pozitivnoj ili negativnoj korelaciji, postoji velika mogućnost da će na izvedbu modela utjecati problem zvan *multikolinearnost*. Taj scenarij se događa kada se jedna varijabla u modelu može na temelju ostalih predvidjeti s veoma visokim stupnjem točnosti. U konačnici to može dovesti do iskrivljenih rezultata.

U nastavku prema slikama 11 i 12 slijede vizualni i tabularni prikazi matrica korelacija za trenutna dva skupa podataka.



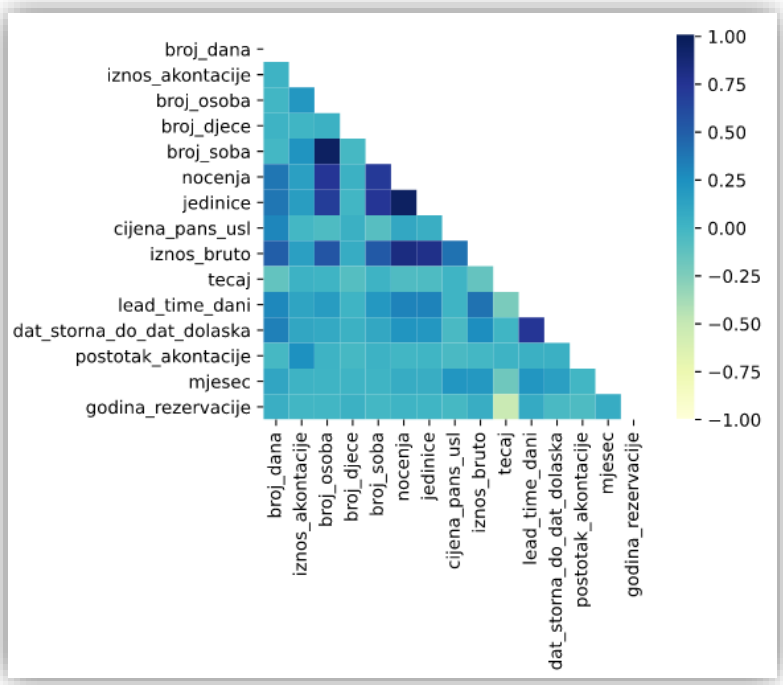
Slika 11. Matrica korelacija za prognoze

	temp_prosjek	temp_max	temp_min	vidljivost	smjer_vjetra_stupnjevi	brzina_vjetra	nalet_vjetra_brzina	relativna_vlaznost	tlak_zraka	oblaci_pokrice	oborine_akumulirano	dubina_snijega	rosa_prosjek	preostalo_dana
temp_prosjek	1,00	0,98	0,97	0,10	0,19	-0,15	0,05	-0,20	-0,24	-0,22	-0,06	-0,13	0,90	0,01
temp_max	0,98	1,00	0,91	0,20	0,24	-0,20	0,04	-0,27	-0,17	-0,37	-0,12	-0,15	0,85	0,01
temp_min	0,97	0,91	1,00	-0,01	0,13	-0,13	0,06	-0,09	-0,32	-0,06	0,01	-0,10	0,92	0,01
vidljivost	0,10	0,20	-0,01	1,00	-0,11	0,09	-0,04	-0,68	0,16	-0,64	-0,22	-0,11	-0,20	0,04
smjer_vjetra_stupnjevi	0,19	0,24	0,13	-0,11	1,00	-0,42	-0,08	0,24	0,04	-0,12	-0,05	-0,03	0,28	-0,01
brzina_vjetra	-0,15	-0,20	-0,13	0,09	-0,42	1,00	0,24	-0,31	0,07	0,05	0,17	0,08	-0,28	0,01
nalet_vjetra_brzina	0,05	0,04	0,06	-0,04	-0,08	0,24	1,00	-0,07	-0,06	0,07	-0,01	-0,03	0,02	-0,03
relativna_vlaznost	-0,20	-0,27	-0,09	-0,68	0,24	-0,31	-0,07	1,00	-0,23	0,64	0,28	0,14	0,23	0,02
tlak_zraka	-0,24	-0,17	-0,32	0,16	0,04	0,07	-0,06	-0,23	1,00	-0,38	-0,24	-0,10	-0,34	-0,02
oblaci_pokrice	-0,22	-0,37	-0,06	-0,64	-0,12	0,05	0,07	0,64	-0,38	1,00	0,37	0,14	0,05	0,00
oborine_akumulirano	-0,06	-0,12	0,01	-0,22	-0,05	0,17	-0,01	0,28	-0,24	0,37	1,00	0,55	0,06	0,01
dubina_snijega	-0,13	-0,15	-0,10	-0,11	-0,03	0,08	-0,03	0,14	-0,10	0,14	0,55	1,00	-0,07	0,00
rosa_prosjek	0,90	0,85	0,92	-0,20	0,28	-0,28	0,02	0,23	-0,34	0,05	0,06	-0,07	1,00	0,02
preostalo_dana	0,01	0,01	0,01	0,04	-0,01	0,01	-0,03	0,02	-0,02	0,00	0,01	0,00	0,02	1,00

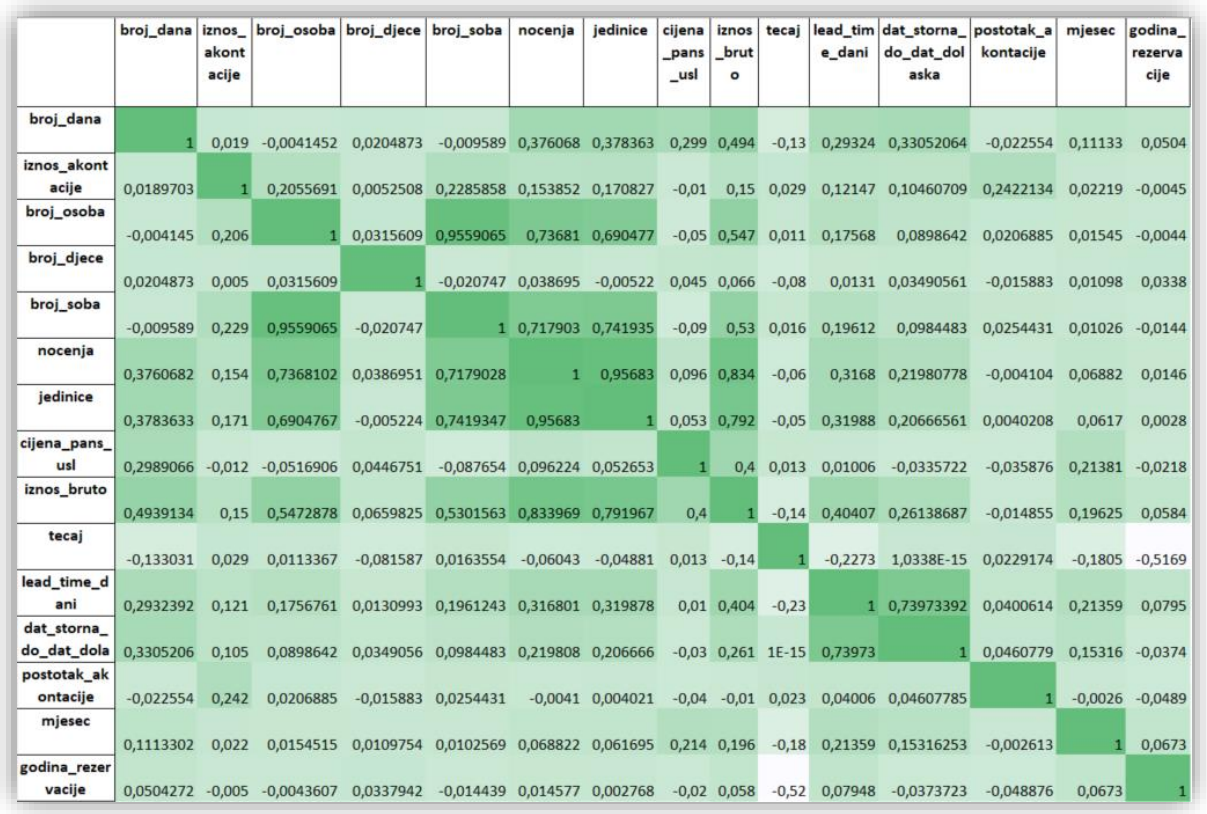
Slika 12. Matrica korelacija za prognoze s vrijednostima

Ono što je odmah vidljivo iz prikaza matrice korelacija jeste da postoje atributi koji imaju visoku razinu pozitivne ili negativne korelacije. Dijagonalne vrijednosti se mogu zanemariti jer prikazuju odnos atributa sa samim sobom. Tako na primjer, atribut *rosa_prosjek* ima visok stupanj pozitivne korelacije sa stupcima vezanima za temperaturu (*temp_prosjek*, *temp_max* i *temp_min*). Kako su ova tri stupca jedan od osnovnih elemenata koji opisuju vrijeme, logički je da se atribut vezan za najavljenu količinu rose makne iz skupa podataka. Dodatno, atribut *dubina_snijega* ima visok stupanj korelacije sa atributom *oborine_akumulirano* te se i on može izbrisati jer nam atribut s najavljenom količinom oborina pokriva i odbačeni atribut. Postoji još jedan visoki stupanj korelacije između atributa *vidljivost* i atributa *oblaci_pokrice*. U ovom slučaju riječ je o negativnoj korelaciji. Kako nam atribut s vrijednostima o oblačnosti važniji od vidljivosti, idući korak je micanje potonjeg iz skupa podataka. Kao krajnji korak, iz skupa podataka se miču atributi *nalet_brzina_vjetra* te *smjer_vjetra_stupnjevi* jer su direktno ovisni o atributu *brzina_vjetra*.

Atribut *relativna_vlaznost* također ima visoki stupanj negativne korelacije s atributom *oblaci_pokrice*, međutim, taj atribut će se kasnije koristiti za izračun dodatnog atributa te zasada ostaje u samom skupu podataka. Slijedi analiza matrice korelacija za skup podataka o rezervacijama gdje će se primijeniti slična logika, koju prikazuje slika 13.



Slika 13. Matrica korelacija za rezervacije



	broj_dana	iznos_akontacije	broj_osoba	broj_djece	broj_soba	nocenja	jedinice	cijena_pans_usl	iznos_bruto	tecaj	lead_time_dani	dat_storna_do_dat_dolaska	postotak_akontacije	mjesec	godina_rezervacije
broj_dana	1	0,019	-0,0041452	0,0204873	-0,009589	0,376068	0,378363	0,299	0,494	-0,13	0,29324	0,33052064	-0,022554	0,11133	0,0504
iznos_akontacije	0,0189703	1	0,2055691	0,0052508	0,2285858	0,153852	0,170827	-0,01	0,15	0,029	0,12147	0,10460709	0,2422134	0,02219	-0,0045
broj_osoba	-0,004145	0,206	1	0,0315609	0,9559065	0,73681	0,690477	-0,05	0,547	0,011	0,17568	0,0898642	0,0206885	0,01545	-0,0044
broj_djece	0,0204873	0,005	0,0315609	1	-0,020747	0,038695	-0,00522	0,045	0,066	-0,08	0,0131	0,03490561	-0,015883	0,01098	0,0338
broj_soba	-0,009589	0,229	0,9559065	-0,020747	1	0,717903	0,741935	-0,09	0,53	0,016	0,19612	0,0984483	0,0254431	0,01026	-0,0144
nocenja	0,3760682	0,154	0,7368102	0,0386951	0,7179028	1	0,95683	0,096	0,834	-0,06	0,3168	0,21980778	-0,004104	0,06882	0,0146
jedinice	0,3783633	0,171	0,6904767	-0,005224	0,7419347	0,95683	1	0,053	0,792	-0,05	0,31988	0,20666561	0,0040208	0,0617	0,0028
cijena_pans_usl	0,2989066	-0,012	-0,0516906	0,0446751	-0,087654	0,096224	0,052653	1	0,4	0,013	0,01006	-0,0335722	-0,035876	0,21381	-0,0218
iznos_bruto	0,4939134	0,15	0,5472878	0,0659825	0,5301563	0,833969	0,791967	0,4	1	-0,14	0,40407	0,26138687	-0,014855	0,19625	0,0584
tecaj	-0,133031	0,029	0,0113367	-0,081587	0,0163554	-0,06043	-0,04881	0,013	-0,14	1	-0,2273	1,0338E-15	0,0229174	-0,1805	-0,5169
lead_time_dani	0,2932392	0,121	0,1756761	0,0130993	0,1961243	0,316801	0,319878	0,01	0,404	-0,23	1	0,73973392	0,0400614	0,21359	0,0795
dat_storna_do_dat_dola	0,3305206	0,105	0,0898642	0,0349056	0,0984483	0,219808	0,206666	-0,03	0,261	1E-15	0,73973	1	0,0460779	0,15316	-0,0374
postotak_akontacije	-0,022554	0,242	0,0206885	-0,015883	0,0254431	-0,0041	0,004021	-0,04	-0,01	0,023	0,04006	0,04607785	1	-0,0026	-0,0489
mjesec	0,1113302	0,022	0,0154515	0,0109754	0,0102569	0,068822	0,061695	0,214	0,196	-0,18	0,21359	0,15316253	-0,002613	1	0,0673
godina_rezervacije	0,0504272	-0,005	-0,0043607	0,0337942	-0,014439	0,014577	0,002768	-0,02	0,058	-0,52	0,07948	-0,0373723	-0,048876	0,0673	1

Slika 14. Matrica korelacija za rezervacije s vrijednostima

Kod ovog skupa podatak vidljivo je više atributa koji imaju viši stupanj korelacije. Atribut *iznos_bruto* raste proporcionalno s atributima *broj_soba*, j*edinice* i *nocenja* dok atribut *nocenja* raste kako raste *broj_osoba* koji je također u pozitivnoj korelaciji s prethodno navedenim atributima. Kako za osnovni opis rezervacije svi ti atributi nisu nužno potrebni, mogu se svi, osim atributa „broj_soba“, maknuti iz skupa podataka i samim time se izbjegne problem multikolinearnosti. Dodatno ostaje za riješiti samo atribute *lead_time_dani* koji govori koliko dana prije dolaska je rezervacija napravljena, a ima viši stupanj korelacije sa atributom *dat_storna_do_dat_dolaska* za kojeg je spomenuto da je potreban u kasnijoj fazi obrade podataka. Logički je izbaciti atribut *lead_time_dani* iz skupa podataka. Za kraj, na temelju vizualnog prikaza na slici 14, iz skupa podataka mogu se izbaciti i atributi *tecaj*, koji je jedinstven za sve rezervacije, atribut *cijena_pans_usluge* koja je ustvari atribut *iznos_bruto* podijeljen sa brojem dana te atribut *obaveza_akontacije* koja je direktno vezana uz atribut *iznos_akontacije*.

Važno je napomenuti da su u matricama korelacija prikazani samo atributi čija je vrijednost numeričke prirode. Za kategoričke varijable primijeniti će se drugačiji postupak obrade podataka.

4.4. Jedinstveni skup podataka

Da bi se moglo vršiti daljnju analizu, potrebno je adekvatno pripojiti jedan skup podataka drugome te promatrati sve informacije koje stoje na raspolaganju kao jedinstveni set podataka. „*Kategoriziranje skupa podataka i primjena funkcije na svaku skupinu, bilo da se radi o agregaciji ili transformaciji, često je ključna komponenta tijeka analize podataka*“ *(McKinney, 2017:287, vlastiti prijevod)*. Korak koji prethodi tome jeste redizajn skupa podataka s rezervacijama tako da se stupci *datum_od* i *datum_do* zamijene samo jednim stupcem koji će imati naziv *dan_boravka*. Time se pokriva mogućnost gdje se neka rezervacija može otkazati kada su gosti već u hotelu, odnosno ukoliko nekoliko dana već borave, a za preostali dio predviđenog boravka je vremenska prognoza loša, te se oni odluče na otkazivanje ostatka boravka. Funkcija predviđena za to se nalazi u datoteci„*BAZA_RazlomiInicijalneRezervacijeNaDane.py*”. Ono što funkcija radi jest da za raspon od datuma dolaska do datuma odlaska u rezervaciji, kreira zapis za svaki dan unutar tog raspona te taj zapis spremi u bazu podataka zajedno sa svim ostalim značajkama rezervacije. Za daljnju analizu se koristi novi format tog skupa podataka.

Funkcija spajanja pod nazivom *„ucitaj_podatke_i_spoji_za_analizu()”* se nalazi u datoteci „*PreprocessingListe_final.py*”. Ono šta ta konkretna funkcija radi jest da u preliminarnu listu iz baze podataka učita sve datume u periodu od 2017. do 2019. godine na koji postoji barem jedna otkazana rezervacija u promatranom hotelu. Zatim se drugoj listi učitaju podaci svih rezervacija te se na temelju datuma (prethodno spomenutog dana boravka) iz prve liste formira skup svih rezervacija koje će služiti modelu za treniranje i razvoj. Drugi dio spajanja odvija se pomoću treće liste koja sadrži sve prognoze iz baze podataka za promatrani period. Lista rezervacija uspoređuje sa listom prognoza datum za koji je prognoza napravljena te ga, ovisno da li je rezervacija

otkazana ili nije adekvatno pripoji i stvori novi zapis sa značajkama iz oba skupa podataka. Uvjeti da se prognoza spoji detaljima rezervacije su sljedeći:

1) *Rezervacija nije stornirana* – ukoliko dan boravka odgovara datumu za koji je vremenska prognoza važeća, rezervaciji pripoji vremenske prognoze za taj dan boravka koje su bile aktualne jedan, tri i sedam dana prije samog dana boravka.

2) *Rezervacija je stornirana* - ukoliko dan boravka odgovara datumu za koji je vremenska prognoza važeća, rezervaciji pripoji vremenske prognoze za taj dan koje su bile aktualne jedan, tri i sedam dana prije dana kada je rezervacija otkazana.

Kao parametar prepoznavanja ovdje je iskorišten atribut *dat_storna_do_dat_dolaska* kojemu je vrijednost koliko dana prije dolaska je sama rezervacija otkazana. Nakon što je taj atribut poslužio funkciji, može se maknuti iz skupa podataka s obzirom da je direktno vezan uz atribut *storno* koji se pokušava predvidjeti.

Navedena funkcija je dodatno proširena u svrhu daljnje analize te kreirana druga inačica iste pod nazivom *„ucitaj_podatke_i_spoji_u_jedinstven_set()“.* Ova funkcija dodatno proširuje domenu prethodne na načina da za svaki zapis u skupu podataka izračuna indeks vrućine prema formuli ispod.

Konstante indexa vrućine

*c1, c2, c3, c4, c5, c6, c7, c8, c9 = -42.379, 2.04901523, 10.14333127, -0.22475541, -6.83783 * 10**-3, -5.481717 * 10**-2, 1.22874*10**-3, 8.5282*10**-4, -1.99*10**-6*

Temperatura i Relativna vlažnost

T = *(stavka['temp_max'] * 1.8) + 32*

V = *stavka['relativna_vlaznost']*

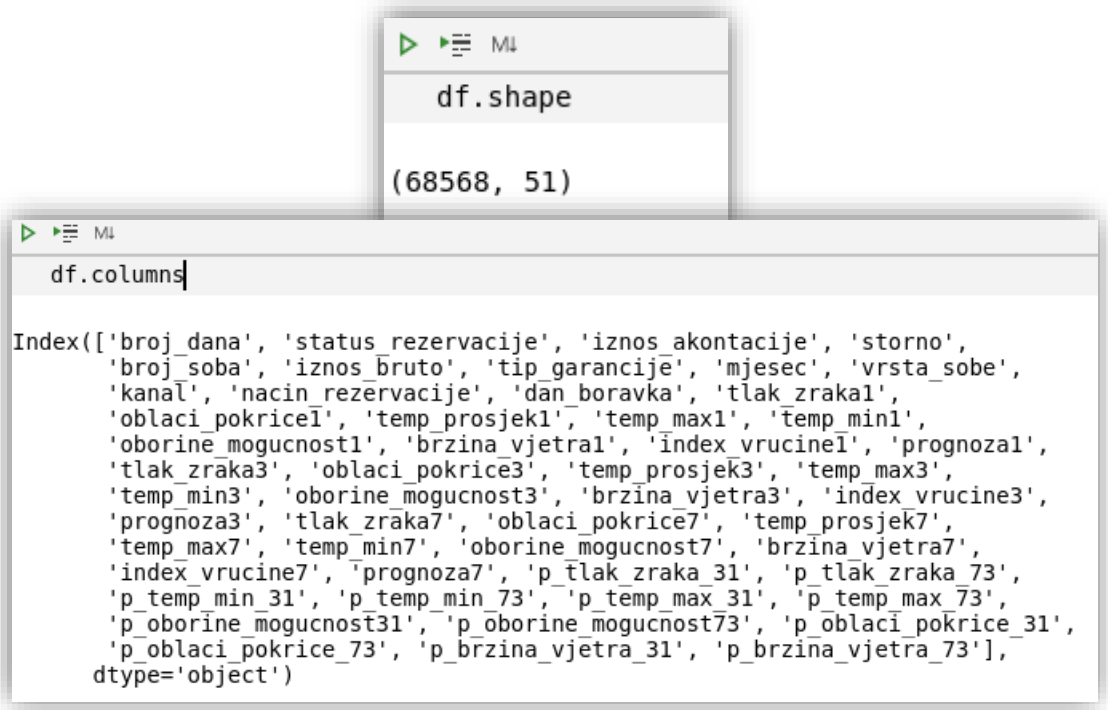
Indeks vrućine

*stavka['index_vrucine'] = c1 + c2*T + c3*V + c4*T*V + c5*T**2 + c6*V**2 + c7*T**2*V + c8*T*V**2 + c9*T**2*V**2*

Dodatno, uzimajući u obzir matrice korelacija, bilo je vidljivo da su atributi vezani uz temperaturu usko povezani jedan s drugim. Gornja funkcija ima i drugu namjenu, a to je da te atribute pretvori u atribute razlika od jedne vremenske prognoze do druge. Samim time, korelacija između tih atributa nestaje kada se originalni atributi maknu iz skupa podataka, a novi izračunati atributi pridruže. Ispod se nalazi primjer programskog koda prema kojem je taj dio funkcije implementiran i u konačnici izvršen u projektu.

> „ *for stavka in finalnaLista:*
> ***stavka['p_tlak_zraka_31'] =*** *stavka['tlak_zraka3'] - stavka['tlak_zraka1']*
> ***stavka['p_temp_min_31'] =*** *stavka['temp_min3'] - stavka['temp_min1']*
> ***stavka['p_temp_max_31'] =*** *stavka['temp_max3'] - stavka['temp_max1']*
> ***stavka['p_oborine_mogucnost31'] =*** *stavka['oborine_mogucnost3'] - stavka['oborine_mogucnost1']* ***stavka['p_brzina_vjetra_31'] =*** *stavka['brzina_vjetra3'] - stavka['brzina_vjetra1']* „

Ovim pristupom kreiran je jedinstven set podataka koji se može dalje obrađivati, a čiji format se može vidjeti na slici 15.



Slika 15. Detalji jedinstvenog skupa podataka

4.5. Filtriranje rezervacija

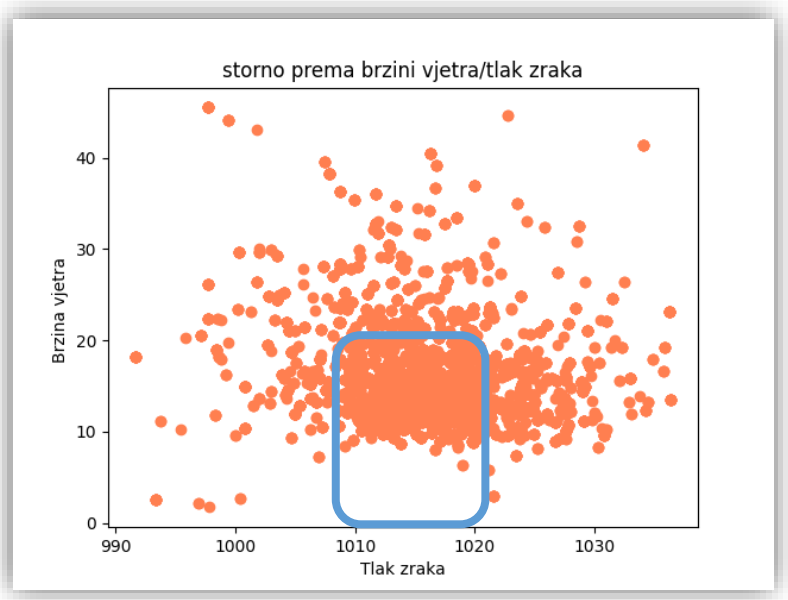
Nakon što je sada skup podataka sveden na format koji je već u velikoj mjeri adekvatan kao ulaz model strojnog učenja, dolazi se do pitanja kako pronaći rezervacije koje su otkazane zbog vremenske prognoze. Naime, sve operacije koje su se dosad odradile nad podacima bile su na skupu koji sadrži sve otkazane rezervacije, odnosno razlozi otkazivanja su razni. Kako bi se odgovorilo na to pitanje potrebno je primijeniti logiku iz same domene poslovanja i postepenim odbacivanjem doći do realnih podataka s kojima se može dalje raditi. Prvi korak jest identificirati one rezervacije koje su otkazane, a razlog otkaza sigurno nije najava loših vremenskih uvjeta. To su rezervacije sa sljedećim parametrima:

- Rezervacije putem kanala „Grupe“ jer imaju posebno definirane uvjete te razlog otkazivanja nikad nije vremenska prognoza jer se za taj scenarij uvijek priprema rezervna opcija kod ugovaranja poslova

- Rezervacije putem kanala „Mice“ koje se odnose na poslovni turizam te njihovo otkazivanje također nije podložno vremenskoj prognozi

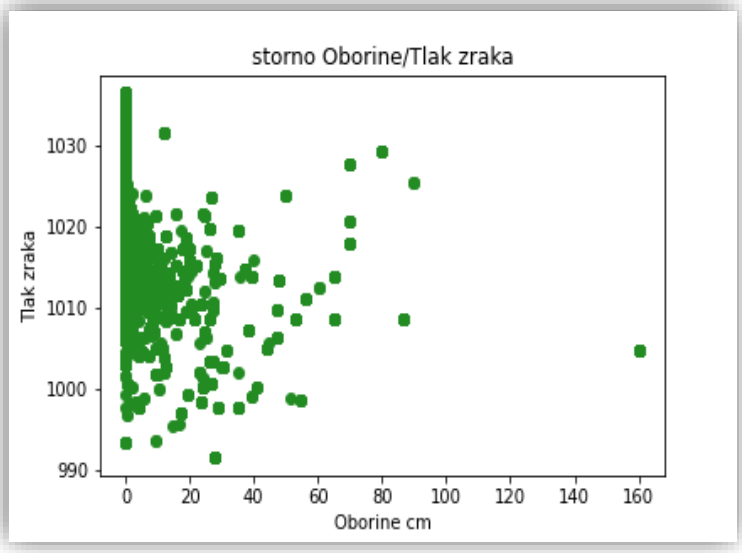
- Sve rezervacije otkazane više od 8 dana prije dolaska s obzirom da će model raditi klasifikaciju za period do 8 dana unaprijed

U idućem koraku će se koristiti vizualni prikazi distribucije storniranih rezervacija u svrhu daljnje identifikacije zapisa koji mogu zadovoljiti kriterij da razlog otkaza nije vremenska prognoza. Svaki od prikaza vezan je uz određene atribute te je inicijalni skup podataka dodatno filtriran tako da prikazu isključivo zapise sa otkazanih rezervacijama. *„Temeljni dio alata znanstvenika podataka je vizualizacija podataka. Iako je vrlo jednostavno stvoriti vizualizacije, puno je teže stvoriti dobre“ (Grus, 2019:37, vlastiti prijevod).*



Slika 16. Distribucija storna prema brzini vjetra i tlaku zraka

Iz prikaza na slici 16 je vidljivo da je većina pozitivne klase (vrijednosti 1 u klasnom atributu *storno*) distribuirana u rasponu tlaka zraka od 1010 - 1022 hektopaskala (hPa) te brzini vjetra u rasponu od 10 – 20 kilometara na sat (km/h). Kada se tome doda činjenica da su te vrijednosti brzine vjetra prema Beaufortovoj skali, koju prikazuje slika 18, definirane kao tišina, lahor i slab povjetarac te da je navedeni raspon tlaka zraka karakterističan za stabilno vrijeme, može se odmah zaključiti da se za sve pozitivne klase u toj zoni na grafu može opravdano reći da razlog otkaza rezervacije u njihovom slučaju nije vremenska prognoza. Slika 17 je prikaz distribucije *storna* prema količini najavljenih oborina te tlaku zraka.



Slika 17. Distribucija storna prema količini oborina i tlaku zraka

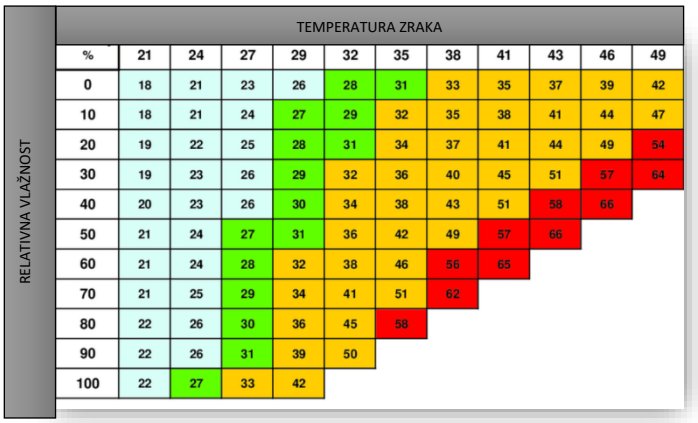
Jačina (Bf)	Naziv	Brzina			Max visina vala	
		km/h	m/s	čvor	unutrašnje more blizu obale	otvoreno more
0	Tišina	<1	0-0,2	<1	---	---
1	Lahor	1-5	0,3-1,5	1-3	0,1	0,1
2	Povjetarac	6-11	1.6-3.3	4-6	0,2	0,3
3	Slabi vjetar	12-19	3.4-5.4	7-10	0,6	1
4	Umjereni vjetar	20-28	5.5-7.9	11-16	1	1,5
5	Umjereno jaki vjetar	29-38	8.0-10.7	17-21	2	2,5
6	Jaki vjetar	39-49	10.8-13.8	22-27	3	4
7	Žestoki vjetar	50-61	13.9-17.1	28-33	4	5,5
8	Olujni vjetar	62-74	17.2-20.7	34-40	5,5	7,5
9	Jaki olujni vjetar	75-88	20.8-24.4	41-47	7	10
10	Orkanski vjetar	89-102	24.5-28.4	48-55	9	12,5
11	Jaki orkanski vjetar	103-117	28.5-32.6	56-63	11,5	16
12	Orkan	>118	>32.7	>64	14	---

Slika 18. Beaufortova skala[1]

I ovaj slika na prvu identificira određenu distribuciju pozitivne klase za koju se sa sigurnošću može reći da ne spada u kategoriju rezervacija otkazanih zbog vremenske prognoze. Na slici je vidljivo da je većina pozitivna klase smještena kod vrijednosti između 0-10 na X osi, odnosno kada nema nikakvih ili postoje minimalne oborine. Ako se tome doda i prethodno definirani raspon tlaka za stabilno vrijeme, mogu se sa sigurnošću odbaciti i svi zapisi pozitivne klase u gore označenom području.

Sa dva prethodna koraka eliminiran je velika broj zapisa pozitivne klase iz skupa podataka, međutim, postoji još načina filtriranja koji se mogu iskoristiti. Jedan od tih se izvodi uz pomoć prethodno kalkulirane značajke *index_vrucine*. Prema vrijednostima indeksa vrućine postoji normativna tablica koja klasificira kombinaciju vrijednosti relativne vlažnosti i temperatura zraka kao razinu pogodnog odnosno nepogodnog vremena. Prikaz te tablice nalazi se na slici 19.

[2] Beaufortova ljestvica (po Francisu Beaufortu), je međunarodno prihvaćena iskustvena ljestvica za procjenjivanje jakosti vjetra prema učincima, u 13 stupnjeva (0 do 12 bofora). Hrvatska enciklopedija, mrežno izdanje. Leksikografski zavod Miroslav Krleža, 2020.



RELATIVNA VLAŽNOST	TEMPERATURA ZRAKA											
%	21	24	27	29	32	35	38	41	43	46	49
0	18	21	23	26	28	31	33	35	37	39	42
10	18	21	24	27	29	32	35	38	41	44	47
20	19	22	25	28	31	34	37	41	44	49	54
30	19	23	26	29	32	36	40	45	51	57	64
40	20	23	26	30	34	38	43	51	58	66	
50	21	24	27	31	36	42	49	57	66		
60	21	24	28	32	38	46	56	65			
70	21	25	29	34	41	51	62				
80	22	26	30	36	45	58					
90	22	26	31	39	50						
100	22	27	33	42							

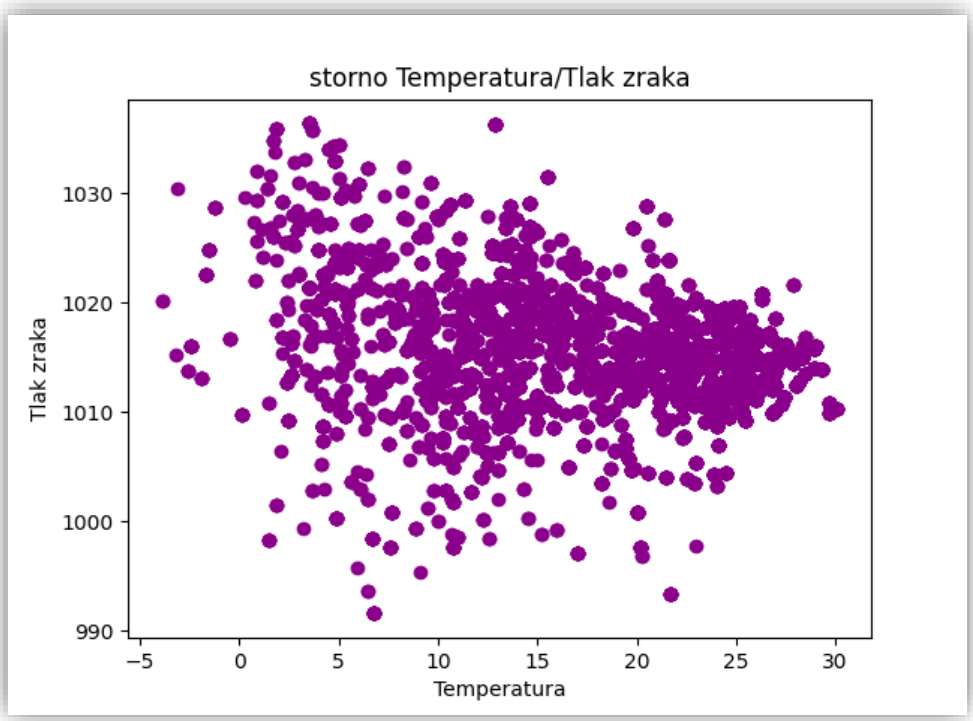
Slika 19. Tablica indeksa vrućina

U samoj tablici, zelene vrijednosti označavaju vrijeme neriskantno za zdravlje. Žuto su označene vrijednosti gdje prilikom dužeg izlaganja vremenskim uvjetima može doći do srčanog udara, dok su crvene vrijednosti veoma opasne po zdravlje. Nakon svih ovih vizualnih prikaza i analize istih, mogu se izdvojiti sljedeći skupovi koji ne spadaju u domenu rezervacija otkazanih zbog najavljenih vremenskih uvjeta.

- Rezervacije otkazane na vedar ili sunčan dan (koristeći definiciju vedrog dana te vrijednosti iz Beaufortove ljestvice)
- Rezervacije otkazane prilikom normalnih vrijednosti tlaka zraka i brzine vjetra
- Rezervacije otkazane kada nisu najavljene oborine te normalnih vrijednosti tlaka zraka
- Rezervacije otkazane prilikom povoljnog indeksa vrućine te prognoze vedro ili malo oblaka

Filtriranje prema pravilima koja su navedena tijekom ove analize izvedeno je u programskom kodu u datoteci „*PreprocessingListe_final.py*“ u funkciji „*ucitaj_i_spoji()*“ te se tim postupkom broj pozitivne klase u klasnom atributu *storno* sa 31011 smanjio na samo 7292. Dodatno vrijedi napomenuti da se, primjenom istih filtera te uzimajući u obzir informacije dobivene iz analize podataka prema slici 20, broj otkazanih rezervacija smanjio, prema procjeni unutar same domena poslovanja, na broj koji u

velikoj mjeri realno predstavlja broj rezervacije otkazanih na temelju vremenske prognoze.



Slika 20. Distribucija storna prema maksimalnoj temperaturi i tlaku zraka

4.6. Kategoričke varijable

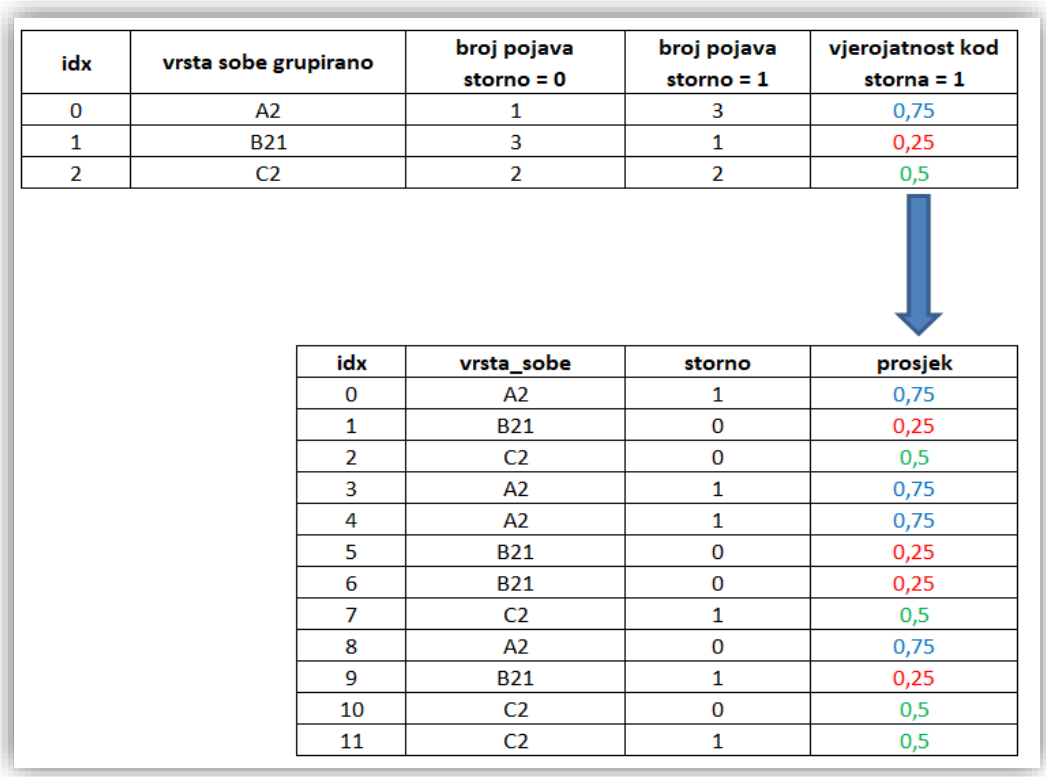
4.6.1.Target encoder

Nakon što su se numeričke varijable obradile te sada imamo jedinstven skup podataka, na redu je obrada kategoričkih varijabli. To su varijable opisnog tipa poput *nacin_rezervacije*, *vrsta_sobe*, *kanal* ili *prognoza* koji dodatno opisuju sam redak, odnosno zapis.

U trenutnom skupu podataka postoji 9 različitih kategoričkih varijabli koje se trebaju normalizirati jer model strojnog učenja radi preciznije s numeričkim tipovima varijabli. Postoji više načina na koji se to može učiniti poput korištenja predefiniranih modula iz „*scikit-learn*“ (sklearn) knjižnice (moduli poput *LabelEncoder*-a ili *OrdinalEncoder*-a), modula iz „*category-encoders*“ knjižnice (moduli kao *OneHotEncoder* ili *TargetEncoder*) ili pak posebno kreirane funkcije.

Za ovaj model primijenjeno je kodiranje pomoću *Target encoder*-a. Taj proces zamjenjuje vrijednosti u kategoričkim varijablama vrijednostima koje su kombinacija vjerojatnosti određene vrijednosti ciljnog atributa s obzirom na određenu vrijednost u kategoričkoj varijabli te vjerojatnosti određene vrijednosti unutar ciljnog atributa nad svim podacima o skupu podataka za treniranje.

Logika procesa vizualno je objašnjena na slici 21.



idx	vrsta sobe grupirano	broj pojava storno = 0	broj pojava storno = 1	vjerojatnost kod storna = 1
0	A2	1	3	0,75
1	B21	3	1	0,25
2	C2	2	2	0,5

idx	vrsta_sobe	storno	prosjek
0	A2	1	0,75
1	B21	0	0,25
2	C2	0	0,5
3	A2	1	0,75
4	A2	1	0,75
5	B21	0	0,25
6	B21	0	0,25
7	C2	1	0,5
8	A2	0	0,75
9	B21	1	0,25
10	C2	0	0,5
11	C2	1	0,5

Slika 21. Primjer target encoding-a

Ovim procesom sljedeće kategoričke varijable su pretvorene u numeričke:

- *status_rezervacije*
- *tip_garancije*
- *vrsta_sobe*
- *kanal*
- *nacin_rezervacije*
- *prognoza1*
- *prognoza3*
- *prognoza7*

4.6.2. Vremenske varijable

U ovu kategoriju varijabli ulaze varijable datuma i vremena. U ovom skupu podataka to su varijable *vrijeme_kreiranja*, *datum_kreiranja*, koje se odnose na točan trenutak u vremenu kada je rezervacija napravljena, te varijable *dan_boravka* i varijable *prognoza1*, *prognoza3* i *prognoza7* koje su tekstualni opis svih numeričkih varijabli u skupu podataka sa vremenskim prognozama. Kako bi model dobio reprezentativne podatke kao ulaz, potrebno je iz vremenskih varijabli napraviti smislene numeričke varijable. Na primjer, varijabla *dan_boravka*, koja je u formatu datuma (eng. *datetime*) može se pretvoriti u nekoliko novih varijabli poput dana u tjednu, dana u godini i mjeseca u godini kako je prikazano ispod.

```
„ df['dan_boravka']= pd.to_datetime(df['dan_boravka'])
df['mjesec'] = df['dan_boravka'].dt.month
df['dan_u_tjednu'] = df['dan_boravka'].dt.dayofweek
df['dan_u_godini'] = df['dan_boravka'].dt.dayofyear
df['tjedan'] = df['dan_boravka'].dt.week “
```

Taj proces stvaranja novih varijabli je dodan u novu funkciju „*ucitaj_podatke_i_spoji()*“ koja će odraditi i funkcionalnost spajanja dva skupa podataka u jedan jedinstveni skup te ih pripremiti za daljnju obradu.

4.7. Simple Imputer

Nastavno na poglavlje o praznim zapisima u određenim varijablama, postoji još jedna tehnika obrade podataka izuzev micanja svih redaka u kojima postoji takav zapis. To je imputacija nedostajućih vrijednosti. Taj proces je ustvari zamjena praznih vrijednosti sa nekim zamjenskim vrijednostima. Prema mišljenju mnogih postoje tri glavna problema koje nedostatak podataka uzrokuje:

- nastanak značajne količinu pristranosti u modelu
- rukovanje i analizu podataka postaju mnogo zahtjevniji
- smanjenje učinkovitosti

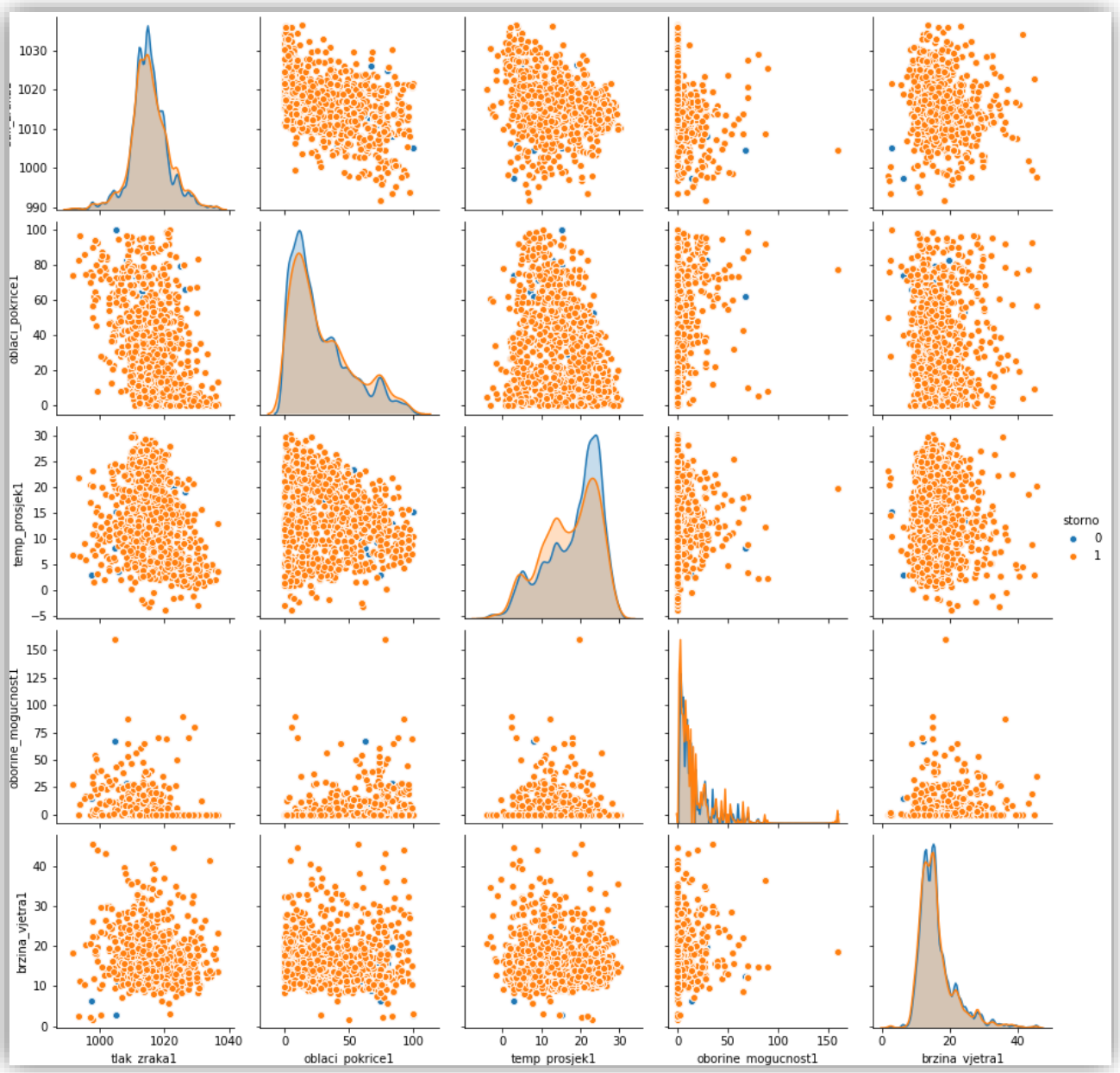
Kako bi se ti problemi izbjegli, imputacijom se prazni zapisi zamjenjuju vrijednostima na temelju nekih statističkih podataka iz tih istih varijabli ili na temelju podataka iz ostalih varijabli. Za primjer se može uzeti varijabla *oblaci_pokrice1* koja trenutno u još uvijek sadrži 139 praznih zapisa. Te zapise moguće je zamijeniti aritmetičkom sredinom ili medijanom preostalih vrijednosti u cijelom stupcu kako se pristranost modela ne bi uvelike povećala, odnosno da model ne izgubi na učinkovitosti. Opisani proces moguće je implementirati na nekoliko načina, bilo posebno kreirano funkcijom ili korištenjem postojećih alata. U ovom slučaju koristiti će se modul *SimpleImputer* iz knjižnice „*sklearn.compute*“. Nakon što se podaci obrade navedenom metodom u skupu podataka više ne postoji praznih zapisa.

4.8. Standardizacija ili normalizacija

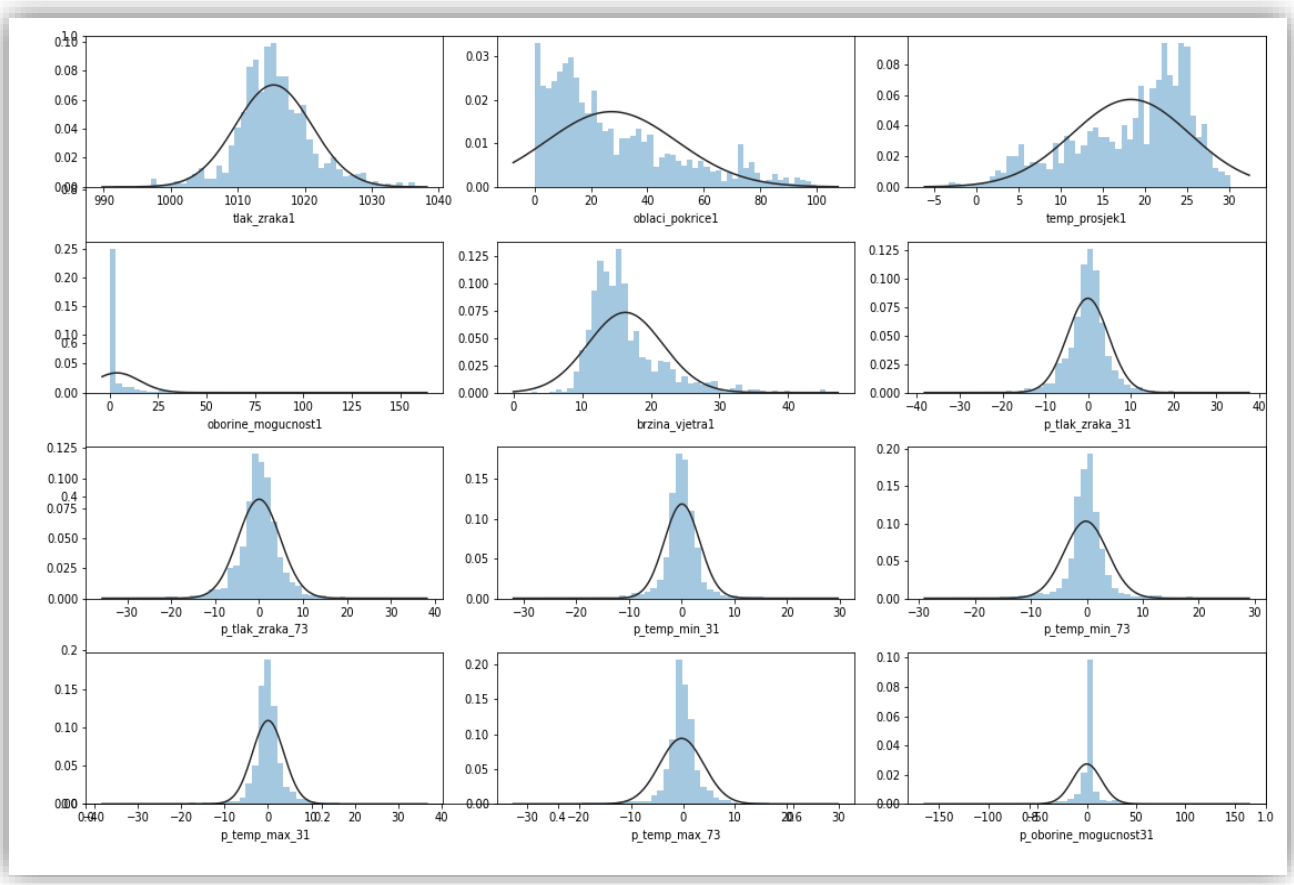
Prilikom uvodne analize, spomenuto je da atributi, odnosno varijable imaju vrijednosti u različitim rasponima. Kako bi se nastavilo s pravilnom obradom podataka, potrebno je određeno skaliranje kako bi sve te vrijednosti bile u jednom rasponu. Time će se modelu omogućiti još lakše tumačenje podataka nad kojima radi. Naravno, ovaj proces ovisi o širem kontekstu i treba ga primjenjivati samo u situacijama kada standardizacija ili normalizacija može samo pozitivno pridonijeti učinkovitosti modela. Koja je razlika između standardizacije i normalizacije? Normalizaciju je dobro koristiti kada se raspodjela samih podataka unutar varijabli ne slijedi Gaussovu krivulju[2] raspodjele. S druge strane, standardizacija se koristi baš u suprotnim slučajevima, odnosno kada podaci slijede Gaussovu krivulju raspodjele.

[2] Gaussova krivulja [gạus~] (krivulja vjerojatnosti) (po Carlu Friedrichu Gaussu), ravninska transcendentna krivulja u obliku zvona, u pravokutnom Kartezijevu koordinatnom sustavu određena jednadžbom y = e–x, Hrvatska enciklopedija, mrežno izdanje. Leksikografski zavod Miroslav Krleža, 2020.

Dakako, postoje i odstupanja od ove teze, međutim, za potrebe ovog projekta sama, definicija će biti oslonac. Na slici 22 se može vidjeti distribucija vrijednosti klasnog atributa *storno* prema osnovnim komponentama koje opisuju vrijeme. Pikaz na slici je kreiran putem funkcije „*pairplot()*“ iz knjižnice „*Seaborn*“.



Slika 22. Distribucija vrijednosti klasnog atributa



Slika 23. Distribucija vrijednosti atributa vremenske prognoze

Na temelju grafičkih prikaza koji se nalaze na slici 23 može se zaključiti sljedeće:

1) većina vrijednosti ulaznih atributa prati standardnu krivulju normalne distribucije te će u ovom slučaju primijeniti funkcija skaliranja sukladno tome. U programskom kodu koristiti će se *StandardScaler* (iz *sklearn* knjižnice).

2) Vrijednosti klasnog atributa su u velikoj neravnoteži te ta činjenica potkrjepljuje da pri izboru modela fokus mora biti na modelima koji mogu adekvatno procesirati takav skup podataka

5. XGBoost model

U ranijem poglavlju kod matrica korelacija spomenut je problem multikolinearnosti. Iako je obrada podataka ciljano bila usmjerena kako se taj problem ne bi pojavio u ovom slučaju, dodatno je potrebno selekcijom adekvatnog tipa modela anulirati bilo kakvu mogućnost istog problema. Postoje određeni algoritmi koji su imuni na problem multikolinearnosti, a to su algoritmi stabala odluka i algoritmi pojačavanja gradijenta. Jedan od modela koji koristi takav algoritam je *XGBoost* model. Ono što separira *XGBoost* model od ostalih su sljedeće značajke:

- Pametna penalizacija stabala
- Proporcionalno smanjivanje lisnih čvorova
- Dodatni parametri randomizacije
- Mogućnost obrade podataka uz prazne zapise
- Ugrađena unakrsna validacija
-

Ovaj tip modela ima veliku primjenu u praksi te se otpočetka nameće kao najoptimalnije izbor za model kod ovog projekta. Ono dodatno krasi model strojnog učenja baziran na ovakvom tipu algoritma jest mogućnost podešavanja velikog broja parametara kako bi treniranje modela bilo što efektnije. Ispod je navedeno nekoliko najbitnijih parametara:

- *Learning rate* – brzina učenja kojom je moguće spriječiti pojavu prekomjernog prilagođavanja[3] (eng. *overfitting*)
- *Max_depth* – maksimalna dubina stabla – također regulira *overftting*
- *Gamma* – određuje minimalno smanjenje gubitka potrebno za podjelu čvora
- *Subsample* – određuje udio zapisa koji će se koristiti kao uzorak za svako stablo
- *Scale_pos_weight* – koristi se kada u distribuciji binarnih vrijednosti ciljnog atributa postoji velika neravnoteža, odnosno njezina vrijednost je omjer negativne i pozitivne klase kod binarne klasifikacije

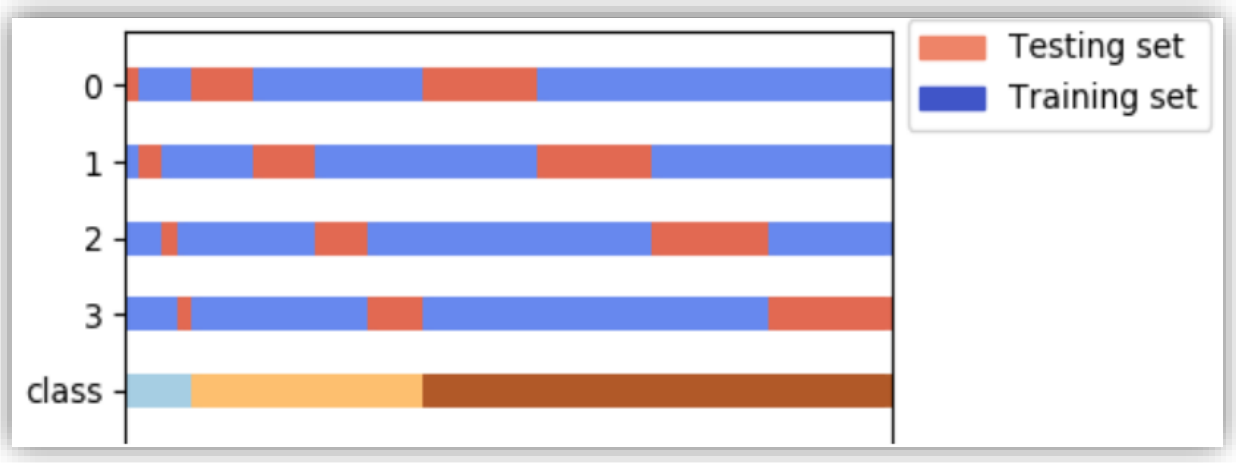
[3] Prekomjerno prilagođavanje (eng. *overftting*) modela je proces analize podataka od strane modela strojnog učenja gdje njegova analiza preusko ili točno odgovara ulaznom skupu podataka

5.1. Unakrsna validacija

Unakrsna validacija je postupak ponovnog uzorkovanja unutar skupa podataka koji se koristi za procjenu modela strojnog učenja na ograničenom uzorku podataka. „*Prije nego primijenimo model novim mjerenjima, moramo znati da li on stvarno funkcionira, odnosno, da li možemo vjerovati njegovim predikcijama*“ *(Müller i Guido 2017, 17, vlastiti prijevod).* Umjesto klasičnog razdvajanja skupa podataka na skupove za treniranje, test i validaciju, unakrsna validacija razdvaja cijeli skup podataka na sljedeći način:

1) Nasumično miješanje cijelog skupa podataka
2) Podjela skupa podataka na, od korisnika specificirani, broj manjih skupova
3) Jedan skup (eng.*fold*) podataka definira kao skup za testiranje
4) Ostale skupove definira kao setove za treniranje
5) Uklapa modelu skup za treniranja i vrši testiranje na predviđenom setu za testiranje
6) Zadržava rezultate testiranja i prelazi na idući skup podataka

Za ovaj projekt koristiti će se metoda stratifikacije uz unakrsnu validaciju uz pomoć „*StratifiedKfold*“ modula iz knjižnice „*sklearn.model_selection*“. Vizualni prikaz procesa nalazi se na slici 24 te je popraćen programskim kodom, na slici 25, koji uključuje i definiranje parametara samog modela iz prethodnog poglavlja kao i stvaranje cjevovoda (eng. *Pipeline*) za standardizaciju, imputaciju i kodiranje skupa podataka.



Slika 24. Stratified K-fold

```
# K FOLD-----------------------------------------------------------------------------------------------------------------------
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=7)
brojac = 1
for train, test in kfold.split(X,y):
    # model = XGBClassifier(objective='binary:logistic', learning_rate=0.05, max_depth=12,
    #                       n_estimators=1000, subsample=1, colsample_bytree=0.9, gamma=0.5, scale_pos_weight=2)
    model = XGBClassifier(objective='binary:logistic', learning_rate=0.03, max_depth=14,
                          n_estimators=1000, subsample=1, colsample_bytree=0.9, gamma=0.5, scale_pos_weight=2)
    X_train, y_train = X.iloc[train], y.iloc[train]
    X_test, y_test = X.iloc[test], y.iloc[test]
    print("Ovo je broj ne-storna / storna u testu", np.bincount(y_test))

    # PROCESIRANJE PODATAKA
    # DEFINIRAJ LISTE ZA NUMERIČKE I ZA KATEGORIČKE VARIJABLE
    numericke = ['index_vrucine1', 'temp_prosjek1', 'broj_soba', 'temp_max1', 'temp_min1', 'brzina_vjetra1', 'tlak_zraka1', 'oblaci_pokrice1',
                 'oborine_mogucnost1', 'p_temp_min_31', 'p_temp_min_73', 'p_temp_max_31',  'p_oborine_mogucnost31', 'p_brzina_vjetra_73']
    kategoricke = ['nacin_rezervacije', 'status_rezervacije', 'vrsta_sobe', 'kanal', 'tip_garancije']

    #UCITAJ PIPELINE SA VRIJEDNOSTIMA SVOJSTVENIM ZA CIJELI DATASET
    PrepPipeline = pickle.load(open('./Pipeline/Pipeline_Total.pkl', 'rb'))

    # OBRADI PODATKE
    X_train_fit = PrepPipeline.transform(X_train)
    X_test_fit = PrepPipeline.transform(X_test)

    # SET ZA EVALUACIJU
    eval_metrics = ["auc", "aucpr", "map", "error", "logloss"]
    eval_set = [(X_train_fit, y_train), (X_test_fit, y_test)]

    # TRENIRAJ MODEL
    model.fit(X_train_fit, y_train, early_stopping_rounds=15, eval_metric=eval_metrics, eval_set=eval_set, verbose=2)
```

Slika 25. Programski kod za StratifiedKfold i definiciju parametara modela

Cjevovod je veoma praktično rješenje jer se u jednom koraku izvrše sve akcije koje skup podataka standardiziraju i pripreme da bude adekvatan za početak treniranja modela. Dodatna pozitivna stvar kod cjevovoda jest da, s obzirom da se, nakon što model bude spreman za predikcije, novi, odnosno neviđeni podaci moraju procesirati na isti način kao i podaci koji su služili za treniranje modela pa se može iskoristiti isti cjevovod bez da se gradi novi. U programskom kodu iznad, cijeli cjevovod je, nakon obrade podataka, spremljen u jednu datoteku kako bi se vrijednosti koje su se prilikom imputiranja, standardizacije i kodiranja koristile, bile spremljene te se mogle iskoristiti na novim podacima kod predikcija. Naime, nove podatke je potrebno obraditi na način da se sva tri koraka procesiranja odviju na istoj razini, odnosno u istom rasponu kao i podaci za treniranje.

5.2. Recursive Feature Elimination (RFE)

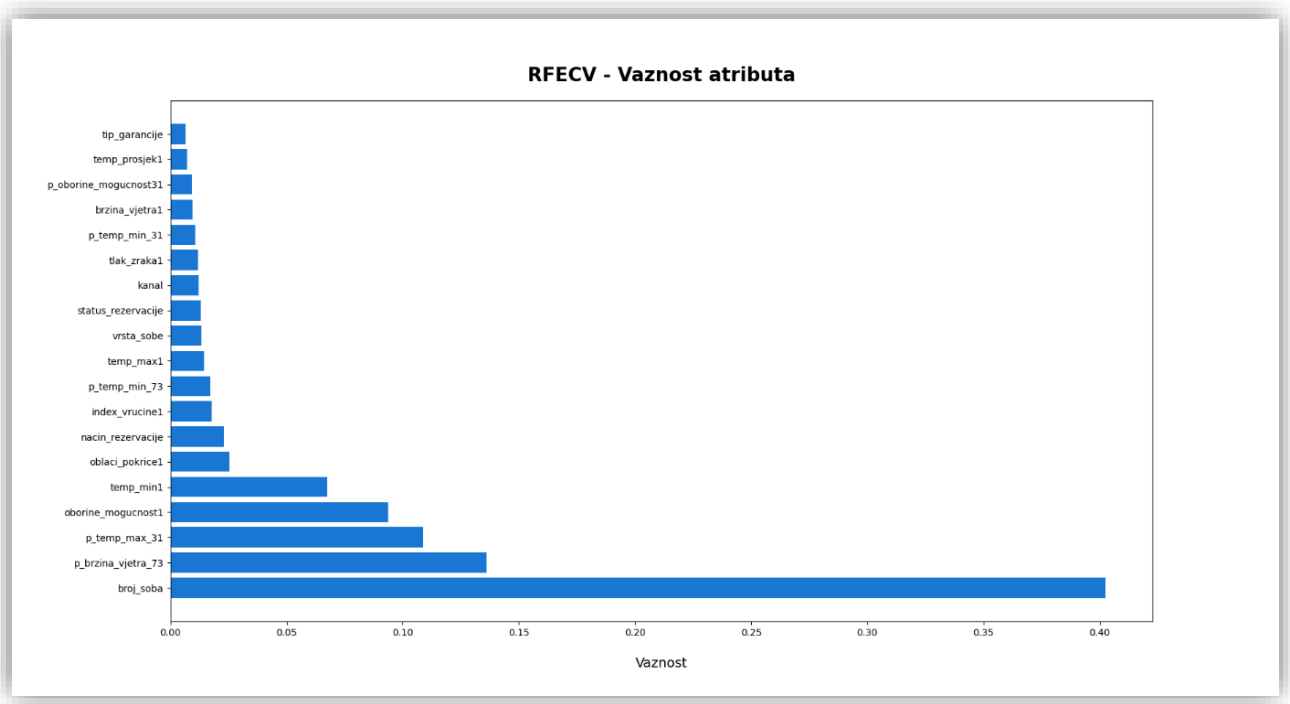
RFE metoda je proces filtracije ulaznih značajki koja često bude veoma zamoran proces. *„Pokušaj i pogreška obično su potrebni za određivanje najboljih ulaza za model“ (Bowles 2015:11, vlastiti prijevod).* Ta metoda je veoma popularna jer algoritam je građen na način da ga je lako konfigurirati i koristiti. Također je veoma

učinkovit u odabiru onih atributa u skupu podataka za treniranje koje su više ili najrelevantnije u predviđanju klasnog atributa.

Konfiguracija se bazira na dvije najvažnije stavke:

1) izbor broja značajki za odabir
2) izbor algoritma koji se koristi za odabir značajki

Za ovaj model korištena je dorađena inačica RFE metode, preciznije rečeno koristila se metoda „*Recursive Feature Elimination Cross Validation*“ **(**RFECV) koja u sebi ima i parametar za unakrsnu validaciju. Cijeli programski kod se nalazi u datoteci *RFE_selekcija_FINAL.py*, a isječak tog koda može se vidjeti na slici 27. Slika 26 s prikazuje rezultata same selekcije, odnos prikaz atributa prema važnosti.



Slika 26. Popis atributa prema važnosti

Ono što se odmah može primijetiti jest da je model za najvažnije značajke klasifikacije odabrao atribute iz skupa podataka sa vremenskim prognozama. To je potvrda je da je prethodno filtriranje rezervacija bilo uspješno i pravilno usmjereno k tome da u skupu

podataka, pozitivna klasa predstavlja samo one rezervacije koje su otkazane na temelju vremenske prognoze.

```
rfc = XGBClassifier()
rfecv = RFECV(estimator=rfc, step=1, cv=StratifiedKFold(5), scoring='f1', min_features_to_select=1, verbose=1)
rfecv.fit(X_fit, y)

print('Optimalan broj značajki je: {}'.format(rfecv.n_features_))
print(np.where(rfecv.support_ == False)[0])
X.drop(X.columns[np.where(rfecv.support_ == False)[0]], axis=1, inplace=True)


plt.figure(figsize=(16, 9))
plt.title('Recursive Feature Elimination sa unakrsnom validacijom', fontsize=18, fontweight='bold', pad=20)
plt.xlabel('Broj odabranih značajki', fontsize=14, labelpad=20)
plt.ylabel('% pravilne klasifikacije', fontsize=14, labelpad=20)
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_, color='#303F9F', linewidth=3)
plt.show()


# PLOT PO VAŽNOSTI
dset = pd.DataFrame()
dset['atribut'] = X.columns
dset['vaznost'] = rfecv.estimator_.feature_importances_
dset = dset.sort_values(by='vaznost', ascending=False)
plt.figure(figsize=(16, 14))
plt.barh(y=dset['atribut'], width=dset['vaznost'], color='#1976D2')
plt.title('RFECV - Vaznost atributa', fontsize=20, fontweight='bold', pad=20)
plt.xlabel('Vaznost', fontsize=14, labelpad=20)
plt.show()
```

Slika 27. Programski kod za RFE selekciju i prikaz rezultata prema važnosti

5.3. Parametri modela (CVGridSearch funkcija)

Pronalazak pravih parametara za treniranje modela najčešća je problematika kod razvoja modela. Ručno podešavanje obično znači veliki utrošak vremena te bude podosta zamorno za svakog razvojnog programera. *„Slijepa primjena xgboost-a može dovesti do nestabilnih modela kao rezultat prekomjerne prilagodbe podacima o treningu“ (Bruce, 2020:409, vlastiti prijevod).* Kako bi se ta problematika svela na minimum, postoje funkcije koji mogu uskočiti u pomoć. Jedna od njih je *GridSearchCv* funkcija iz knjižnice *sklearn.model_selection*. Ovaj modul je veoma intuitivan i jednostavan za korištenje jer dopušta da se raspon parametara definira u obliku listi. Algoritam zatim prolazi kroz svaki zapis u listi te ga uzima kao parametar za novu iteraciju treniranja. Algoritam na kraju iteracije bilježi željene metrike rezultata i prelazi na novu iteraciju. Taj proces se ponavlja dok se ne izvede zadnja iteracija treniranja. U ovom slučaju pretraživanje parametara navedenom tehnikom spojeno je s unakrsnom validacijom. Kako sam proces ne bi trajao duže nego što je potrebno te da model ne bi napravio *overfitting,* dodatno je specificiran broj epoha u svakom treniranju

nakon kojega trenutno aktualna iteracija treniranja prestaje ukoliko se rezultati treniranja ne poboljšaju. U nastavku slika 29 prikazuje programski kod za definiranje mreže parametara te implementaciju *GridSearchCV* metode. Slika 28 prikazuje mrežu parametara za ovaj konkretan model.

```
# PARAMETRI
params = {
        'learning_rate':[0.01, 0.1, 0.2],
        'min_child_weight': [0.75, 1, 3],
        'gamma': [0.001, 0.1, 0.8, 1],
        'subsample': [0.5, 0.8],
        'colsample_bytree':[0.8, 1],
        'max_depth': [4,6,8],
        'n_estimators': [100, 200, 350]
        'scale_pos_weight': [0.06, 1, 5, 15, 30]
}
```

Slika 28. Mreža parametara

```
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=7)
brojac = 1
for train, test in kfold.split(X,y):
    model = XGBClassifier(objective='binary:logistic')
    X_train, y_train = X.iloc[train], y.iloc[train]
    X_test, y_test = X.iloc[test], y.iloc[test]
    print("Ovo je broj ne-storna / storna u testu", np.bincount(y_test))

    # PROCESIRANJE PODATAKA
    # DEFINIRAJ LISTE ZA NUMERIČKE I ZA KATEGORIČKE VARIJABLE
    numericke = ['index_vrucine1', 'temp_prosjek1', 'broj_soba', 'temp_max1', 'temp_min1', 'brzina_vjetra1', 'tlak_zraka1', 'oblaci_pokrice1',
                 'oborine_mogucnost1', 'p_temp_min_31', 'p_temp_min_73', 'p_temp_max_31',  'p_oborine_mogucnost31', 'p_brzina_vjetra_73']
    kategoricke = ['nacin_rezervacije', 'status_rezervacije', 'vrsta_sobe', 'kanal', 'tip_garancije']

    #UCITAJ PIPELINE SA VRIJEDNOSTIMA SVOJSTVENIM ZA CIJELI DATASET
    PrepPipeline = pickle.load(open('./Pipeline/Pipeline_Total.pkl', 'rb'))

    # OBRADI PODATKE
    X_train_fit = PrepPipeline.transform(X_train)
    X_test_fit = PrepPipeline.transform(X_test)

    # Validacija
    eval_set = [(X_train_fit, y_train), (X_test_fit, y_test)]

    # PARAMETRI
    params = {
            'learning_rate':[0.01, 0.1, 0.2],
            'min_child_weight': [0.75, 1, 3],
            'gamma': [0.001, 0.1, 0.8, 1],
            'subsample': [0.5, 0.8],
            'colsample_bytree':[0.8, 1],
            'max_depth': [4,6,8],
            'n_estimators': [100, 200, 350],
            'scale_pos_weight': [0.06, 1, 5, 15, 30]
    }

    # GRID
    grid = GridSearchCV(estimator=model, param_grid=params, scoring='f1', verbose=3)
    grid.fit(X_train_fit, y_train, early_stopping_rounds=15, eval_set=eval_set, eval_metric="error", verbose=2)

    print(grid.best_params_)
    print(grid.best_score_)
```

Slika 29. GridSearchCV programski kod

Programski kod za *GridSearchCV* metodu se nalazi u datoteci *GridSearchCV_XGBoost_FINAL.py.*

6. Rezultati treninga modela

6.1. Odabir metrika

U programskom kodu koja obuhvaća i unakrsnu validaciju, zadnji korak je bila funkcija *fit()* koja je ustvari početak treniranja modela. U poglavlju koje slijedi glavni fokus će biti na analizi rezultata pojedinačnih treniranja i validacije. Rezultati će se promatrati kroz različite krivulje te izvještaje specifične za skupove podataka s visokim stupnjem neravnoteže između vrijednosti klasnog atributa. Prolazak kroz svaki *fold* skupa podataka zabilježiti će i prikazati rezultate za svaki parametar mjerenja učinkovitosti modela.

Za početak, analizirati će se lista metrika za evaluaciju treninga modela. Naime, kod grafičkih prikaza distribucije vrijednosti klasnog atributa, bilo je moguće vidjeti da ovaj skup podataka ima veliku neravnotežu između pozitivne i negativne klase klasnog atributa. Ta činjenica pomaže jer automatski odbacuje klasična mjerenja preciznosti i točnosti. Klasične metrike dobro rade na većini problema, međutim, sve metode mjerenja pretpostavljaju problem ili ono što je važno u problemu. Stoga se mora odabrati metrika evaluacije koja najbolje bilježi ono što se u projektu smatra relevantnim što u konačnici pravi izbor metrike evaluacije modela čini izazovnijim.

Klasične metode mjerenja poput preciznosti (eng. *accuracy*) u obzir uzima cijeli skup podataka za treniranja te evaluira model na prosječnoj preciznosti, ne uzimajući u obzir neravnotežu u skupu podataka. To znači da će, ukoliko je omjer pozitivne i negativne klase u skupu podataka 1:100 (10000 zapisa negativne klase naspram 100 zapisa pozitivne), i model pravilno klasificira 9900 zapisa negativne klase točno bez da radi točne predikcije po pitanju pozitivne klase, preciznost modela iznositi 98,01%. Taj rezultat je sjajan, međutim, daljnja analiza bi otkrila da je preciznost predikcije pozitivne klase, na kojoj je naglasak, podosta niža i ustvari model nije adekvatan za problem koji mu je postavljen.

Zbog ovakvih slučajeva potrebno je odabrati što adekvatnije metode mjerenja kako bi se dobio pravi uvid u učinkovitost modela. Za početak, slika 30 prikazuje dio programskog koda koji definira metode mjerenja i set za evaluaciju istih u ovom modelu.

```
eval_metrics = ["auc", "aucpr", "map", "error", "logloss"]
eval_set = [(X_train_fit, y_train), (X_test_fit, y_test)]
```

Slika 30. Definiranje metoda mjerenja

Vidljivo je da je odabran skup sa nekoliko različitih metoda mjerenja učinkovitosti modela. Lista s nazivima metrika nalazi se ispod uz odgovarajuće objašenjenje.

- *auc* – (eng. *Area Under the curve*) je metoda mjerenja performansi kod problema klasifikacije pri različitim vrijednostima pragova. Prag je granica na kojoj model odlučuje kojoj klasi će trenutni zapis pripasti. Računa se na način da se kao temelj uzme ROC krivulja vjerojatnosti, koja prikazuje omjer točno klasificiranih vrijednosti naspram netočno klasificiranih vrijednosti klase koja se promatra. AUC u tom slučaju predstavlja stupanj ili mjeru odvojenosti, odnosno sposobnost modela da što jasnije razlikuje dvije klase kod binarne klasifikacije. Što je rezultat veći to je model pouzdaniji po pitanju separiranja klasa.
- *aucpr* – (eng. *Area Under the curve, Precision and Recall*) – ova metoda pri mjerenju kao temelj ima PR krivulju. To je krivulja koja na X osi ima stupanj preciznosti, odnosno u kolikom je postotku model uspio točno prepoznati određenu klasu, a na Y osi ima stupanj osjetljivosti (eng. *Sensitivity*), odnosno omjer točno predviđene klase naspram zbroju točno predviđenih klasa i lažno predviđenih klasa
- *map* – (eng. *Mean Average Precision*) izračunava se na način da uzima srednju vrijednost prosječne preciznosti za sve klase te ukupne pragove preklapanja (eng. *Intersection over Union (*IoU)). IoU prag je razina preklapanja predviđene klase i stvarne klase. Za razliku od klasične preciznosti to nije prosječna vrijednost za sve klase u klasnom atributu već se uzima u obzir svaka klasa pojedinačno.

- *error* – stopa pogreške kod binarne klasifikacije. Izračunava se kao postotak pogrešnih slučajeva / % svih slučajeva. Kod predviđanja se kao prag koristi vjerojatnost od 50%, odnosno pozitivna klasa se predviđa ako je vjerojatnost 50% i više, dok se negativna klasa predviđa ako je vjerojatnost manje od 50%
- *logloss* – (eng. *negative log-likelihood*) Negativno u ovom slučaju znači množenje rezultata mjerenja sa -1 iz razloga što su kod ove metrike niže vrijednosti bolje, odnosno što je manja vrijednost u rezultatu mjerenja to je model optimiziraniji i učinkovitiji. *Log* dio se odnosi na logaritam u koji je ustvari umnožak predviđenih vjerojatnosti samog modela umotan radi lakše reprezentacije od strane računala. *Likelihood* je vjerojatnost da neki izračunati parametri proizvedu neke poznate podatke.

6.2. Matrica konfuzije

Kako bi se do kraja objasnili neki pojmovi kod definicije metoda mjerenja, potrebno je uvesti još jednu metodu provjere rezultata modela, a to je matrica konfuzije. Ova matrica je kod binarne klasifikacije ustvari tablica sa četiri različite kombinacije zapisa. Općeniti oblik matrice prikazuje slika 31.

	Pozitivna klasa (1)	**Pozitivna klasa (1)**
Pozitivna klasa (1)	TP	FP
Pozitivna klasa (1)	FN	TN

Slika 31. Opći oblik matrice konfuzije

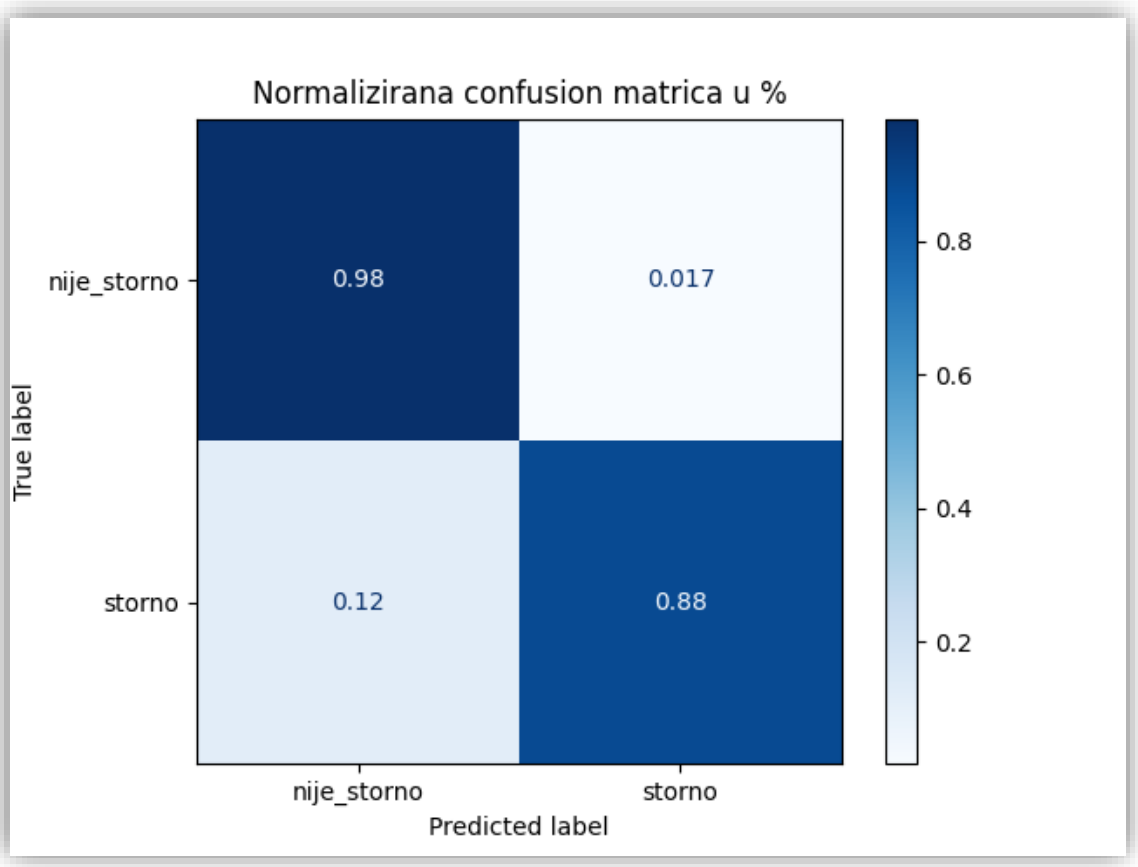
Objašnjenje vrsta vrijednosti koje se nalaze u samoj matrici je sljedeće:

- eng. *True Positive* (TP) – predikcija klase od strane modela je bila pozitivna klasa, te je klasa u stvarnosti bila pozitivna, odnosno model je predvidio 1 koja je i stvarna vrijednost
- eng. *True Negative* (TN) – predikcija klase od strane modela je bila negativna klasa, te je klasa u stvarnosti bila negativna, odnosno model je predvidio 0 koja je i stvarna vrijednost
- eng. *False Positive* (FP) – predikcija klase od strane modela je bila pozitivna klasa, a klasa je u stvarnosti bila negativna, odnosno model je predvidio 1 dok je stvarna vrijednost 0
- eng. *False Negative* (FN) – predikcija klase od strane modela je bila negativna klasa, a klasa je u stvarnosti bila pozitivna, odnosno model je predvidio 0 dok je stvarna vrijednost 1

Matrica konfuzije kod izračuna vrijednosti koristi preciznost (eng. *precision*), odnosno broj stvarno pozitivnih klasa od svih točno predviđenih vrijednosti, te osjetljivost(eng. *recall*) koji je broj točno predviđenih pozitivnih klasa od svih pozitivnih klasa u skupu podataka.

6.3. Analiza rezultata treniranja

Kako je navedeno, model je započeo svoje treniranje te je prva iteracija završila i zabilježila prve rezultate. Za početak će se promatrati matrica konfuzije koju prikazuje slika 32. Važno je napomenuti da je za ispis matrice konfuzije korištena *sklearn* knjižnica koja pri kalkulaciji koristi uzlazeći poredak vrijednosti u klasnom atributu te će u matrici klase biti zamijenjene jer je 1 veće 0. Negativna klasa će odgovarati vrijednosti 1, a pozitivna vrijednosti 0. Programski kod za crtanje matrice konfuzije je vidljiv na slici 33.



Slika 32. Normalizirana matrica konfuzije

```
# CONFUSION MATRICA
class_names = ['nije_storno', 'storno']
titles_options = [("Confusion matrica, apsolutne vrijednosti", None),
                  ("Normalizirana confusion matrica u %", 'true')]
for title, normalize in titles_options:
    disp = plot_confusion_matrix(model, X_test_fit, y_test,
                                 display_labels=class_names,
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)

    print(title)
    print(disp.confusion_matrix)
    plt.show()
```

Slika 33. Programski kod za crtanje matrice konfuzije

Prema rezultatima zabilježenim u matrici konfuzije vidimo da model ima dosta visok stupanj točnih predikcija. Predikcije negativne klase (vrijednosti 0) su 98,3% točne dok se pozitivna klasa (u obzir se uzima velika neravnoteža u podacima) predvidjela točno u 88% slučajeva. Ta činjenica se može potkrijepiti i izvještajem klasifikacije koji je vidljiv

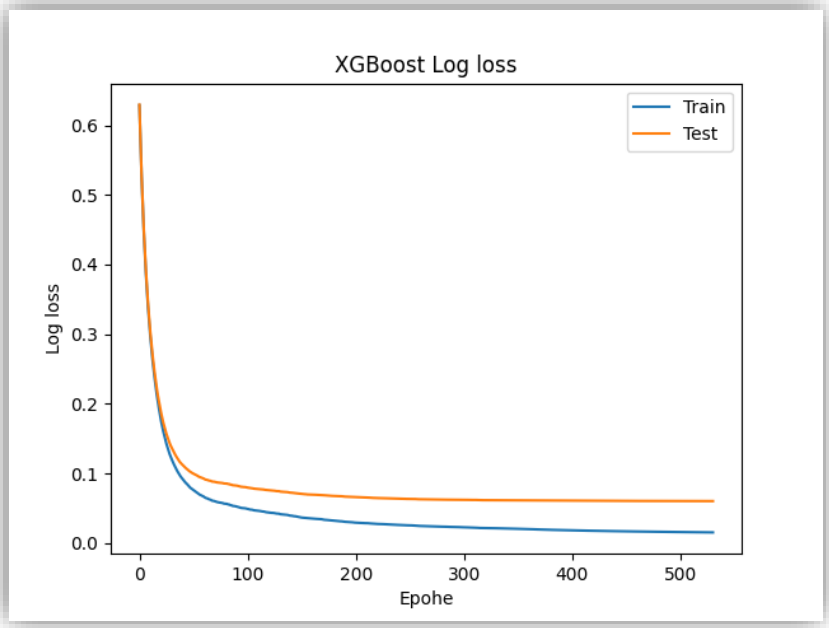
na slici 34, a pokazuje postotke preciznosti vezane uz pojedinu klasu kao i prosječne vrijednosti mjerenja za cijeli skup podataka.

	precision	recall	f1-score	support
0.0	0.99	0.99	0.99	11222
1.0	0.90	0.84	0.87	893
accuracy			0.98	12115
macro avg	0.94	0.92	0.93	12115
weighted avg	0.98	0.98	0.98	12115

```
# Classification report
y_pred = model.predict(X_test_fit)
report = classification_report(y_test, y_pred)
print(report)
```

Slika 34. Rezultati i programski kod izvještaja klasifikacije

U izvještaju je vidljivo da je preciznost kod pozitivne klase manja nego kod negativne klase zbog velike neravnoteže i velike razlike u distribuciji tih vrijednosti u skupu podataka, međutim, točnosti prema pojedinačnoj klasi (eng. *macro avg*) od 93% je jako dobar rezultat s obzirom da se kao mjera za kompenzaciju neravnoteže koristio samo parametar *scale_pos_weight* u XGBoost modelu. Analiza se nastavlja s tumačenjem rezultata ispod krivulje prema slici 35.



Slika 35. Krivulja prema *Log loss* metrici

Programski kod za crtanje prethodne krivulje prikazan je na slici 36.

```
# Log loss
fig, ax = plt.subplots()
ax.plot(x_axis, rezultati['validation_0']['logloss'], label='Train')
ax.plot(x_axis, rezultati['validation_1']['logloss'], label='Test')
ax.legend()
plt.xlabel('Epohe')
plt.ylabel('Log loss')
plt.title('XGBoost Log loss')
plt.show()
```

Slika 36. Programski kod za krivulju *Log loss* rezultata

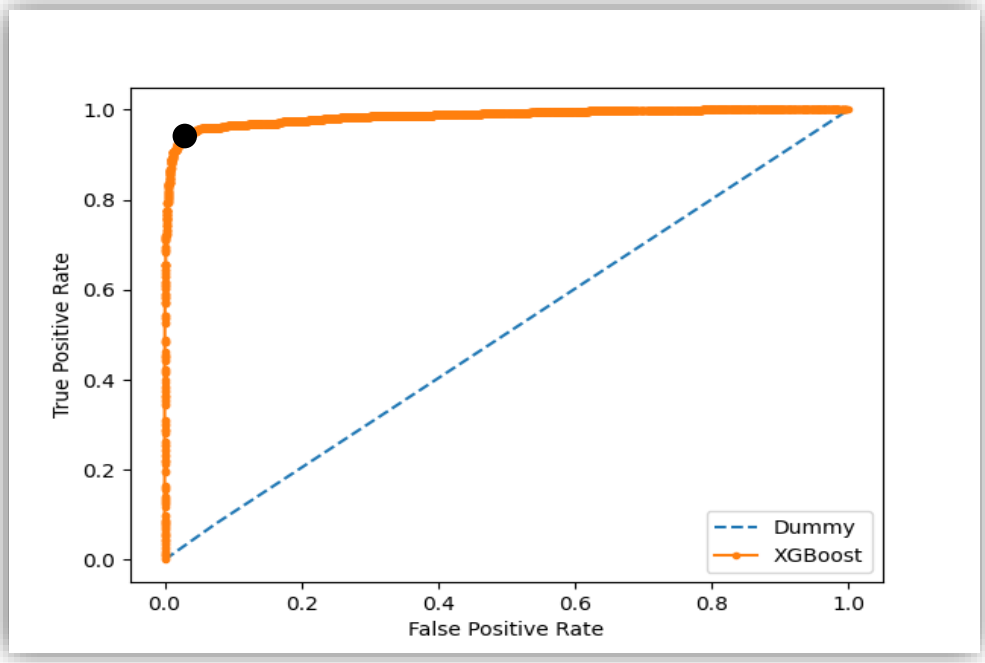
Iduća na redu je krivulja prema slici 37 koja pokazuje gubitak prema metrici opisanoj u prethodnom poglavlju. Kod ove krivulje, što je manje odstupanje u rezultatu skupa za testiranje od rezultata u skupu za treniranje to je model precizniji. Također, što manje vrijednosti, znače što učinkovitiji model. Ovdje se može vidjeti da odstupanje rezultata seta za testiranje od seta za treniranje iznosi oko 0,04 te je stabilno od 500 epohe treniranja nadalje. Slika 38 prikazuje programski kod za istu.

Slika 37. Krivulja prema mjerenju pogreške klasifikacije

```
# pogreška klasifikacije
fig, ax = plt.subplots()
ax.plot(x_axis, rezultati['validation_0']['error'], label='Train')
ax.plot(x_axis, rezultati['validation_1']['error'], label='Test')
ax.legend()
plt.xlabel('Epohe')
plt.ylabel('Class error')
plt.title('XGBoost Class error')
plt.show()
```

Slika 38. Programski kod za krivulju rezultata pogreške kod klasifikacije

Na krivulji sa slike 39 može se vidjeti da je odstupanje kod pogreške klasifikacija pozitivne klase između seta za treniranje i seta za testiranje oko 0,7% te da je pogreška kod klasifikacije te iste klase na validacijskim podacima 0,73% što je veoma zadovoljavajući rezultat s obzirom na malu zastupljenost te klase u samom skupu podataka. Kako bi se analiza još temeljitije napravila slijede ROC i PR krivulje.



Slika 39. ROC krivulja

Prema ROC krivulji može se također odrediti učinkovitost samog modela. Preciznije rečeno, što je površina ispod krivulje veća, model je pouzdaniji. "*Za crtanje ROC*

krivulje trebaju vam rezultati, a ne vjerojatnosti. Jednostavno rješenje je koristiti vjerojatnosti pozitivne klase kao rezultat“ *(Géron 2019:99, vlastiti prijevod).*

Ovdje je riječ o krivulji vjerojatnosti te je stoga za njezin grafički prikaz potrebno napraviti predikciju vjerojatnosti putem naredbe *model.predict_proba(X_test).* Zatim se definira dodatno da se želi predikcija samo pozitivne klase u nastavku prethodne naredbe sa *[:,1].* Za ovu konkretnu krivulju AUC rezultat je 0.928 koji se dobio pomoću funkcije *roc_auc_score()* iz *sklearn* knjižnice. Programski kod se nalazi na slici 40 ispod.

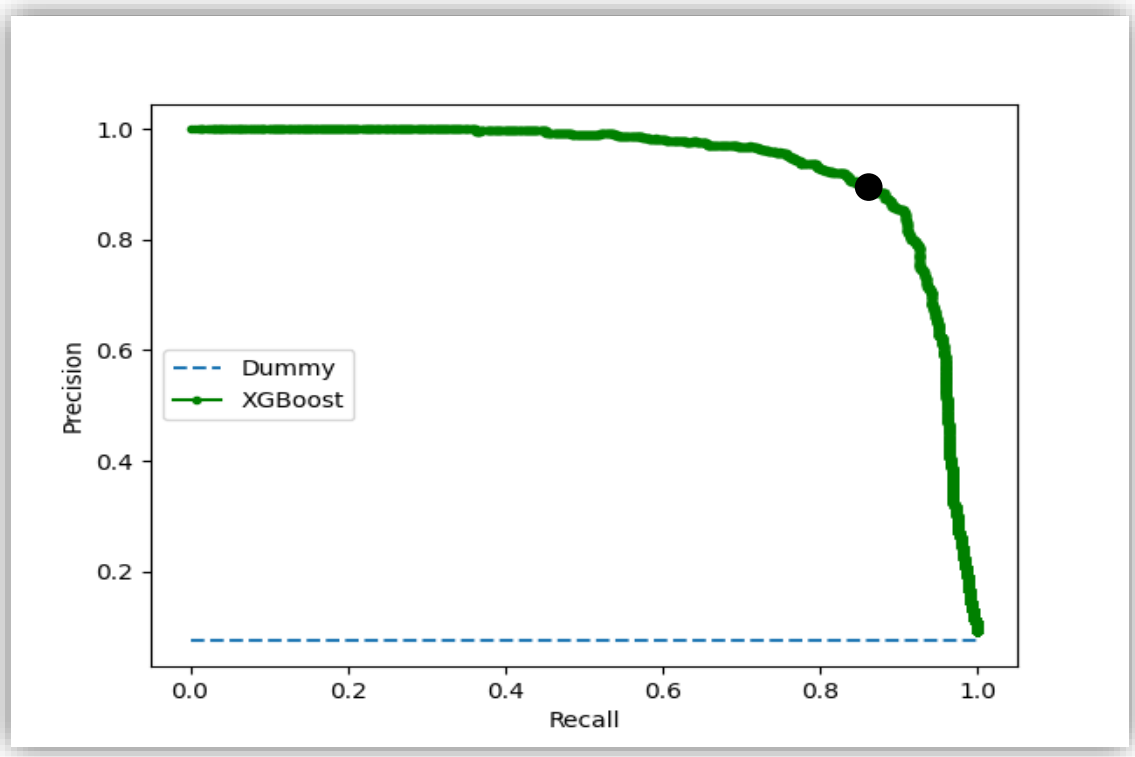
```
# ROC I AUC
y_hat = model.predict_proba(X_test_fit)
model_vjerojatnosti = y_hat[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, model_vjerojatnosti)
# IZRAČUNAJ NAJBOLJI THRESHOLD ZA ROC KRIVULJU- prema Youden's J
J = tpr - fpr
ix = np.argmax(J)
najbolji_thresh = thresholds[ix]
print('Najoptimalnija vrijednost za ROC =%f' % (najbolji_thresh))
roc_auc = roc_auc_score(y_test, model.predict(X_test_fit))
print('ROC AUC SCORE: %.3f' % roc_auc)
# PLOTAJ ROC KRIVULJU
```

```
# ROC-AUC
def plot_roc_curve(y_test, naive_vjerojatnosti, model_vjerojatnosti):
    # plotaj dummy
    fpr1, tpr1, _1 = roc_curve(y_test, naive_vjerojatnosti)
    plt.plot(fpr1, tpr1, linestyle='--', label='Dummy')
    # plotaj model
    fpr2, tpr2, _2 = roc_curve(y_test, model_vjerojatnosti)
    plt.plot(fpr2, tpr2, marker='.', label='XGBoost')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.legend()
    plt.show()
```

Slika 40. programski kod za plot ROC krivulje

Na krivulji je dodatno označena maksimalna točka koju model ima po pitanju točno klasificiranih vrijednosti. Najlakša definicija jest da je to točka najbliža gornjem lijevom kutu, odnosno točki na krivulji koju bi imao savršeni klasifikator. Iako ROC krivulja daje naslutiti da je općenita preciznost modela dosta pouzdana, za skupove podataka gdje je distribucija klasa u klasnom atributu neravnomjerna možda je bolje promatrati sve kroz PR krivulju koja je vidljiva slici 41.

„Točke su gotovo nevidljive, ako ne i za rubove na grafikonu. Međutim, oni su stvarni generatori zapleta jer točke označavaju položaje“ (Tosi, 2019:45, vlastiti prijevod).

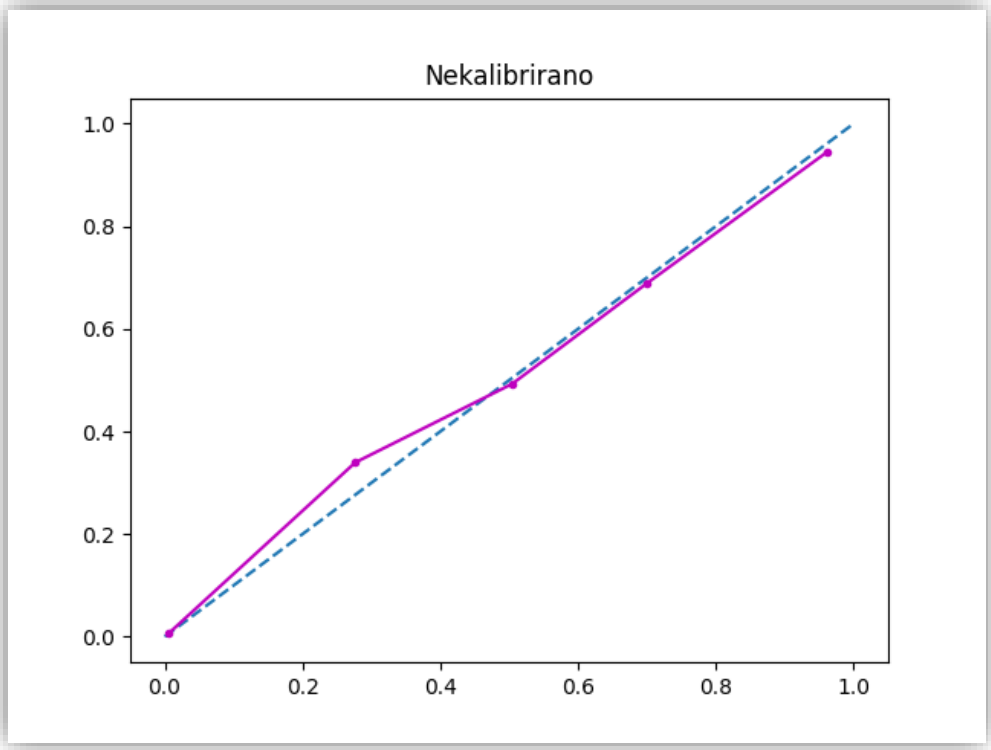


Slika 41. PR krivulja

Ova krivulja prikazuje odnos između preciznosti i osjetljivosti modela. U potpunosti zanemaruje lažne negativne vrijednosti iz matrice konfuzije. Sama krivulja dobije se tako da se na grafu prikažu pragovi odluke (eng. *threshold*s) koji prikazuju omjer dvije navedene vrijednosti. Točnije, ukoliko bi se modelu zadala neka određena vrijednost, odnosno prag sa krivulje, na temelju koje bi odlučivao kada je klasa pozitivna, a kada negativna, ima se uvid kakva bi bila preciznost, a kakva osjetljivost modela. Najoptimalnija točka je ona točka koja je najbliža gornjem desnom kutu, tj. točka gdje je preciznost = 1 i osjetljivost = 1. (savršeni klasifikator). Dodatno kod ove krivulje vidimo da njezina zadnja vrijednost nije 0, već je vrijednost koju bi imao klasifikator slučajnosti. Ta vrijednost je obično postotak pozitivnih klasa u cijelom skupu podataka koji se promatra. Za kraj, površina ispod navedene krivulje kao konačni rezultat ima vrijednost 0.904.

Postoji još jedna krivulja koja će se promatrati u svrhu analize, a to je krivulja kalibracije koja je vidljiva na slici 42. To je linijski tip prikaza gdje je predikcije vjerojatnosti stavljaju

u određene okvire takozvanih koševa (eng. *bin*). Svaki *bin* tada predstavlja jednu točku na grafičkom prikazu te se te točke spajaju jednom linijom. Za svaku točku vrijednost y je udio točnih rezultata dok je vrijednost X srednja vrijednost predikcije. Dobro kalibrirani modeli imati će ravniju liniju koja kreće od donje lijeve točke 0,0 pa sve do gornje desne točke 1,1. Na prikazu ispod iscrtan je savršeno kalibrirani model (plava iscrtana linija) te model koji se trenutno razvio (ljubičasta puna linija).



Slika 42. Krivulja kalibracije

6.4. Finaliziranje modela

Nakon što su mjerenja u prethodnoj fazi pokazala zadovoljavajuće rezultate, preostaje zadnji korak pri razvoju modela. To je, s obzirom da je unakrsna validacija demonstrirala da bi se model trebao adekvatno ponašati na neviđenim podacima, treniranje modela na cijelom skupu podataka te spremanje modela u datoteku kako bi se mogao koristiti za predikcije. Programski kod gdje je to implementirano se nalazi na slici 43.

```
# FINALNI MODEL

numericke = ['index_vrucine1', 'temp_prosjek1', 'broj_soba', 'temp_max1', 'temp_min1', 'brzina_vjetra1', 'tlak_zraka1',
             'oblaci_pokrice1', 'oborine_mogucnost1',  'p_temp_min_31', 'p_temp_min_73', 'p_temp_max_31',  'p_oborine_mogucnost31', 'p_brzina_vjetra_73']
kategoricke = ['nacin_rezervacije', 'status_rezervacije', 'vrsta_sobe', 'kanal', 'tip_garancije']

NumPipeline = Pipeline([('impute', SimpleImputer(strategy='mean', missing_values=np.nan)), ('scale', StandardScaler())])
KatPipeline = Pipeline([('impute', SimpleImputer(strategy='most_frequent', missing_values=np.nan)), ('encode', TargetEncoder()), ('scale', StandardScaler())])

FinalModelPipeline = ColumnTransformer([('prep_numericke', NumPipeline, numericke), ('prep_kategoricke', KatPipeline, kategoricke)], remainder='passthrough')
X_fit = FinalModelPipeline.fit_transform(X)
# možda s y na kraju

# Spremi pipeline
filename = 'Pipeline_Final.pkl'
direktorij = './Pipeline/'
putanja = os.path.join(direktorij, filename)
pickle.dump(te, open(putanja, 'wb'))
print("Pipeline spremljen !!")

# FINALNI MODEL
final_model = XGBClassifier(objective='binary:logistic', learning_rate=0.15, max_depth=6, min_child_weight=5, n_estimators=1000,
                            subsample=1, colsample_bytree=0.9, gamma=0.5, scale_pos_weight=2)
final_model.fit(X_fit, y, verbose=1)

# Spremi model nakon treninga
filename = 'Model_Final.pkl'
direktorij = './Modeli/'
putanja = os.path.join(direktorij, filename)
pickle.dump(model, open(putanja, 'wb'))
print("Model spremljen !!")
```

Slika 43. Programski kod za razvoj konačnog modela te spremanje istog

Iz primjera se može vidjeti da je korištena ista metoda za spremanje modela kao i za spremanje cjevovoda kod obrade podataka prije treniranja. Na taj način se olakšava učitavanje gotovog modela prilikom predikcija. Programski kod za finalni model strojnog učenja se nalazi u datoteci *XGBOOST_model_FINAL.py*.

7. Predikcije

Zadnje poglavlje vezano uz razvoj i korištenje modela opisuje postupak predikcije na neviđenim podacima. Kompletan programski kod je podijeljen na dvije datoteke. Prva je pod nazivom „*Funkcije_Final.py*“, a druga ima naziv „*03_ucitaj_i_prediciraj.py*“. Glavni dio koda je smještene u prvoj datoteci dok se druga koristi samo za pozivanje funkcija iz prve datoteke.

Funkcije uključuju postupak obrade nad novim podacima prema istom principu kao kod treniranja modela, odnosno poziva se prethodno spremljeni cjevovod, podaci se obrade te pošalju u model za predikciju. U datoteci Funkcije_Final.py postoje dvije

funkcije, jedna pod nazivom „*napravi_predikcije_za_danas()*“ za predikciju otkazanih rezervacija za trenutni dan, te druga naziva „*napravi_predikcije_za_ostale_dane()*“ za predikciju za preostale dane. Razlog tome jest što druga funkcija u sebi ima implementiranu metodu da na početku provjerava koje rezervacije iz predikcije za danas su klasificirane kao otkazane kako ne bi došlo do paradoksa da je jedan dan rezervacije klasificiran kao *storno*, dok drugi dan nije (rezervacije su u prethodnoj fazi razlomljene na dane). Isti proces se onda ponavlja za svaki idući dan. Ključni atribut prema kojem se rezervacije i predikcije uspoređuju jest šifra rezervacije.

Slika 44 prikazuje programski kod koji uključuje pozivanje cjevovoda, obradu podataka, pozivanje modela, te predikciju vjerojatnosti pozitivne klase uz pomoć prethodno izračunatog praga odluke kako bi model bio što precizniji.

```
# Definicija tipova kolona za daljnje procesiranje
numericke = ['index_vrucine1', 'temp_prosjek1', 'broj_soba', 'temp_max1', 'temp_min1', 'brzina_vjetra1', 'tlak_zraka1', 'oblaci_pokrice1',
             'oborine_mogucnost1',  'p_temp_min_31', 'p_temp_min_73', 'p_temp_max_31',  'p_oborine_mogucnost31', 'p_brzina_vjetra_73']
kategoricke = ['nacin_rezervacije', 'status_rezervacije', 'vrsta_sobe', 'kanal', 'tip_garancije']

# LOAD PIPELINE-A
PrepPipeline = pickle.load(open('./Pipeline/Pipeline_CV1.pkl', 'rb'))
X_fit = PrepPipeline.transform(df)

# učitaj model za predikciju
prag_odluke = 0.609095
model = pickle.load(open('./Modeli/Model_Final.pkl', 'rb'))
predikcije = (model.predict_proba(X_fit)[:,1]>=prag_odluke).astype(int)
```

Slika 44. Proces predikcija modela

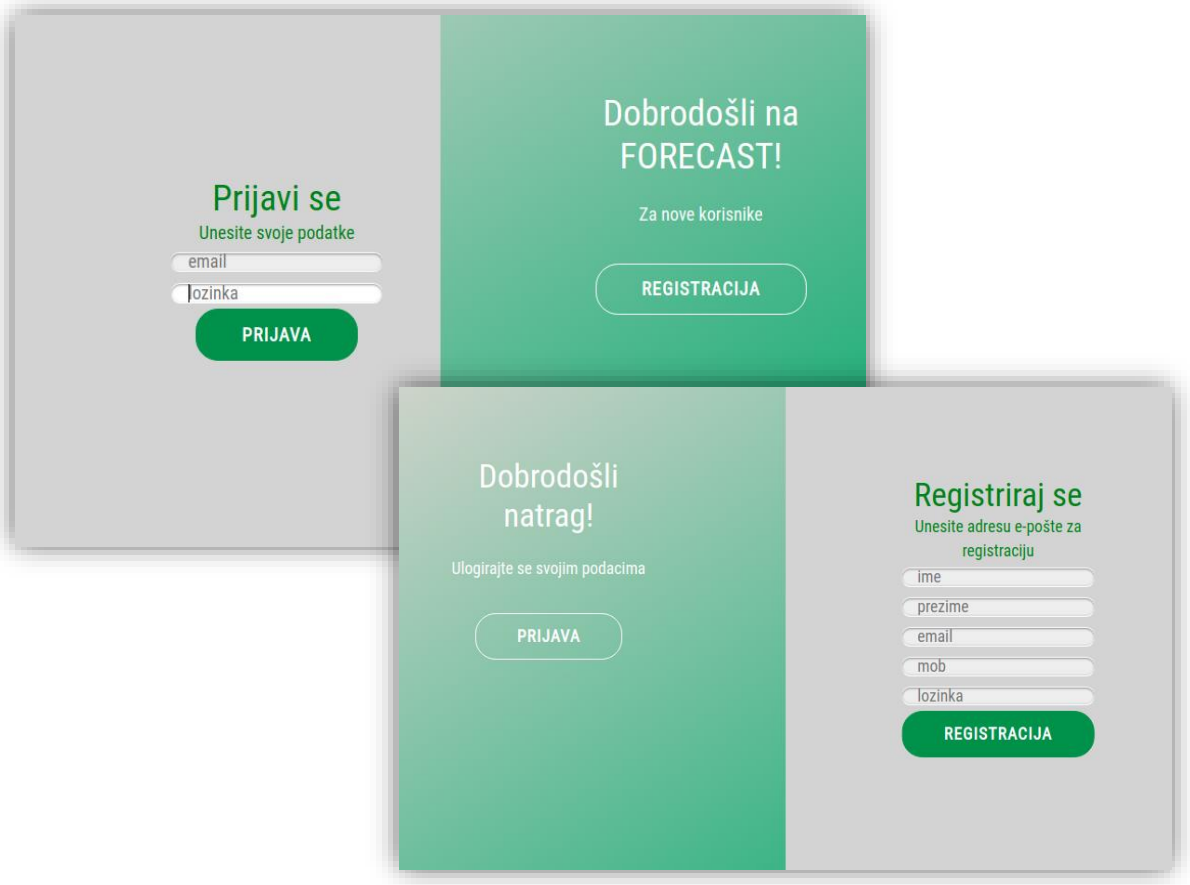
Na samom kraju, predikcije se povezuju s ostalim atributima rezervacije poput broja jedinica i bruto iznosa i spremaju u bazu podataka sa današnjim datumom. Slika 45 prikazuje proces nastanka konačnog zapisa sa predikcijom kao zadnjim atributom te spremanje tog zapisa u bazu podataka.

```
# SPREMI U DB
listaPredikcija = df_pred.to_dict('records')
conn = psycopg2.connect(baza)
for stavka in listaPredikcija:
    novi = {"id":stavka['id'], "dan_boravka":stavka['dan_boravka'], "sif_rezervacije":stavka['sif_rezervacije'],  "vrsta_sobe":stavka['vrsta_sobe'],
            "kanal":stavka['kanal'], "jedinice":stavka['jedinice'], "iznos_bruto":stavka['iznos_bruto'], "predikcije":stavka['predikcije'],
            "datum_predikcije":stavka['datum_predikcije']}
    cur = conn.cursor()
    cur.execute("""INSERT INTO predikcije(id, dan_boravka, sif_rezervacije, vrsta_sobe, kanal, jedinice, iznos_bruto, predikcije, datum_predikcije)
                VALUES (%(id)s, %(dan_boravka)s, %(sif_rezervacije)s, %(vrsta_sobe)s, %(kanal)s, %(jedinice)s, %(iznos_bruto)s, %(predikcije)s,
                %(datum_predikcije)s)""", novi)
    conn.commit()
```

Slika 45. Proces spremanja predikcija modela

8. Sučelje aplikacije

U nastavku će se ukratko opisati sučelje web aplikacije uz vizualizaciju njezinih dijelova. Sama web aplikacija sastoji se on nekoliko komponenata koje se mogu pronaći u mapi *src* unutar mape *FRONTEND* na samo repozitoriju. Kompletan programski kod za izgradnju ove web aplikacije također se nalazi u istoj mapi. Ulaz u aplikaciju jest putem „Login“ forme na početnom ekranu koja se može vidjeti na slici 46.



Slika 46. Login forma

Sama forma se sastoji od dva dijela. Jedan dio se koristi za prijavu postojeći korisnika, dok se drugi koristi za registraciju novih korisnika. Forma se ovisno o potrebi mijenja jednostavnim klikom na gumbe *Prijava* ili *Registracija*. Ukoliko se novi korisnik želi registrirati, potrebno je popuniti sva polja na formi te se pritiskom na tipku *Registracija*, korisnik sa svojim podacima spremi u bazu podataka i može napraviti prvu prijavu. Funkcija registracije iz *FRONTEND* mape se nalazi u nastavku i prikazana je na slici

47 dok su odgovarajuće funkcije u pozadinski sustavu smještene u datotekama *Main.py* te *Domain.py*.

```
registrirajMe() {
  axios.post('http://127.0.0.1:8080/useri', {
    ime: this.imeUser,
    prezime: this.prezimeUser,
    email: this.emailUser,
    mob: this.mobUser,
    rola: 'user',
    lozinka: this.lozinkaUser,
  }).then((response) => {
    console.log(response.data);
  })
    .catch((e) => {
      console.error(e);
    });
},
```

Slika 47. Funkcija registracije

Prilikom prijave, nakon unosa korisničkog imena i lozinke, vrši se prijava putem funkcije *ProvjeriPodatke(),* koju se može vidjeti na slici 49, te se, u slučaju pozitivne provjere pamte podaci trenutku korisnika i spremaju unutar lokalnog spremišta, prema slici 48, u samom pregledniku. Ova funkcija je implementirana ukoliko će se aplikacija naknadno nadograđivati gdje će se morati bilježiti promjene od strane samog korisnika.

```
zapamtiPodatkeTrenutnog(email, pass) {
  axios
    .post('http://127.0.0.1:8080/useri/ulogiran', {
      data: {
        email,
        pass,
      },
    }).then((response) => {
      console.log(response.data);
      localStorage.setItem('detalji', response.data);
      this.$router.push({ name: 'Useri2' });
    });
},
```

Slika 48. Funkcija spremanja podataka trenutnog korisnika

```
provjeriPodatke(email, pass) {
  console.log(email, pass);
  axios.post('http://127.0.0.1:8080/useri/login', {
    data: {
      email,
      pass,
    },
  }).then((response) => {
    // console.log(response.data);
    // console.log('Upisao si pravi mail');
    const validate = document.querySelectorAll('.user-validate');
    const wrongInputText = document.querySelector('.wrong-input');
    wrongInputText.style.display = 'none';
    let i;
    // console.log(validate);
    for (i = 0; i < validate.length; i += 1) {
      validate[i].style.border = '2px solid green';
    }
    localStorage.setItem('ulaz', 'OK');
  })
    .catch((e) => {
      // console.log('Upsss, probaj ponovno, nešto nije ispravno!!');
      const validate = document.querySelectorAll('.user-validate');
      const wrongInputText = document.querySelector('.wrong-input');
      let i;
      console.log(wrongInputText);
      wrongInputText.style.display = 'block';
      wrongInputText.style.color = 'red';
      for (i = 0; i < validate.length; i += 1) {
        validate[i].style.border = '1px solid red';
      }
      console.error(e);
    });
},
```

Slika 49. Funkcija prijave i provjere podataka

Nakon uspješne prijave, korisnik je preusmjeren na početni ekran aplikacije gdje su ispisani njegovi podaci te se sa strane nalazi izbornik za navigaciju kroz aplikaciju. Izbornik je moguće proširiti u svrhu lakšeg snalaženja jednostavni klikom na strelicu gore. Početni ekran aplikacije prikazan je na slici 50.

„Cascading Style Sheets (CSS) moćan je alat koji transformira prezentaciju dokumenta ili zbirku dokumenata, a proširio se na gotovo svaki kutak weba i mnoštvo tobože ne-web okruženja“ (Meyer i Weyl, 2018:1, vlastiti prijevod.)

Slika 50. Početni ekran aplikacije

U ovoj komponenti korisnik ima i mogućnost izmijeniti vlastite podatke te ih spremiti klikom na gumb SPREMI NOVE PODATKE. Nakon promjene podataka ispisuje se poruka da je potrebna ponovna prijava s novim podacima kako bi se isti ponovno mogli registrirati u pregledniku. „U pozadinskom dijelu aplikacije je ova funkcionalnost dodatno podržana funkcijom pod nazivom *daj_trenutnog()* čiji je kod prikazan na slici 51. Radi se o jednostavnoj implementaciji tokena za autentifikaciju (eng. *JSON*[4] *Web Token* (jwt)).

[4] *JavaScript Object Notation* (JSON) otvoreni je standardni format razmjene podataka temeljen na podskupini sintakse *JavaScript* programskog jezika (Crockford, 2009:1, vlastiti prijevod).

```python
#METODA ZA VRAĆANJE PODATAKA TRENUTNOG USERA IZ BAZE NA TEMELJU EMAIL ADRESE I LOZINKE
@db_session
def daj_trenutnog(data):
    key = 'ovo je tajni kljuc za projekt iz ML-a'
    userMail = data['data']['email']
    userPass = data['data']['pass']
    q = select(s for s in User if s.email == userMail and s.lozinka == userPass)
    rez = [x.to_dict() for x in q]
    token = jwt.encode({'id':rez[0]['id'], 'ime':rez[0]['ime'], 'prezime':rez[0]['prezime'],
                        'email':rez[0]['email'], 'mob':rez[0]['mob'], 'rola':rez[0]['rola'], 'lozinka':rez[0]['lozinka']},
                        key, algorithm='HS256')
    return token
```

Slika 51. Funkcija jwt tokena

Ukoliko se slijedi raspored na navigacijskoj traci sa strane, iduća komponenta aplikacije jeste lista korisnika gdje su omogućene četiri osnovne operacije sa podacima (eng. *Create, Read, Update, Delete (*CRUD)) te je ista vidljiva na slici 52.

Slika 52. Lista korisnika

Za CRUD operacije se koristi *AXIOS* aplikacijsko programsko sučelje (eng. *application programming interface* (API)) te slika 53 prikazuje implementaciju njegovih HTTP[5] metoda pod nazivima GET, POST, PUT i DELETE. Funkcije u pozadinskom dijelu se također nalaze u datotekama *Main.py* i *Domain.py* te su prikazane slikom 54.

[5] HTTP (engl. Hyper Text Transfer Protocol) je najčešća metoda prijenosa informacija na Webu. Osnovna namjena ovog protokola je omogućavanje objavljivanja i prezentacije HTML dokumenata, odnosno web stranica

```
methods: {
  getData() {
    axios
      .get('http://127.0.0.1:8080/useri')
      .then((response) => {
        const detalji = this;
        detalji.useri = response.data.data;
      });
  },
  createUser() {
    axios.post('http://127.0.0.1:8080/useri', {
      id: this.idUser,
      ime: this.imeUser,
      prezime: this.prezimeUser,
      email: this.emailUser,
      mob: this.mobUser,
      rola: this.rolaUser,
      lozinka: this.lozinkaUser,
    }).then((response) => {
      console.log(response.data);
      this.getData();
      this.idUser = '';
      this.imeUser = '';
      this.prezimeUser = '';
      this.emailUser = '';
      this.mobUser = '';
      this.rolaUser = '';
      this.lozinkaUser = '';
    })
      .catch((e) => {
        console.error(e);
      });
  },
```

```
azurirajUser() {
  axios.put('http://127.0.0.1:8080/useri', {
    id: this.trenutniId,
    ime: this.trenutnoIme,
    prezime: this.trenutnoPrezime,
    email: this.trenutniEmail,
    mob: this.trenutniMob,
    rola: this.trenutnaRola,
    lozinka: this.trenutnaLozinka,
  }).then((response) => {
    console.log(response.data);
    this.getData();
  })
    .catch((e) => {
      console.error(e);
    });
},
brisiUser(tid) {
  console.log(tid);
  axios.delete('http://127.0.0.1:8080/useri', {
    params: { id: tid },
  })
    .then((response) => {
      console.log(response.data);
      this.getData();
    })
    .catch((e) => {
      console.error(e);
    });
},
```

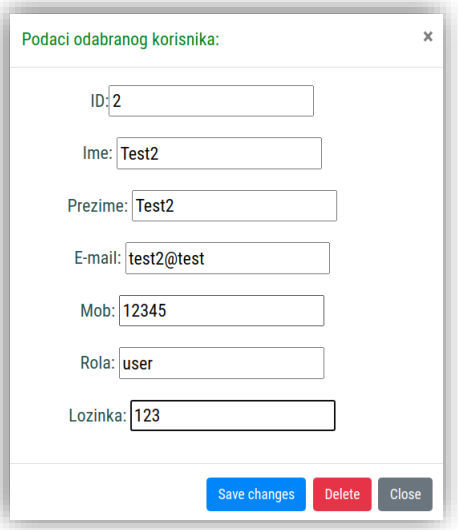
Slika 53. CRUD metode na sučelju

```
@app.route("/useri", methods=["GET"])
def handle_useri():
    useri = Useri.listaj()
    print(useri)
    return jsonify({"data": useri})

@app.route("/useri", methods=["POST"])
def handle_useri_dodaj():
    status, greske = Useri.dodaj(request.get_json())
    if status:
        return Response(status=201)
    else:
        r = Response(status=500)
        r.set_data(greske)
        return r

@app.route('/useri', methods = ['PUT'])
def azuriraj_usera():
    status, greske = Useri.user_update(request.get_json())
    if status:
        return Response(status=201)
    else:
        r = Response(status=500)
        r.set_data(greske)
        return r

@app.route('/useri', methods = ['DELETE'])
def brisi_usera():
    status, greske = Useri.user_brisi(request.args.get('id'))
    if status:
        return Response(status=204)
    else:
        r = Response(status=500)
        r.set_data(greske)
        return r
```

```
class Useri:
    @db_session()
    def listaj():
        q = select(s for s in User)
        data = [x.to_dict() for x in q]
        return data

    @db_session
    def dodaj(s):
        try:
            s["id"] = str(gid())
            s = User(**s)
            return True, None
        except Exception as e:
            return False, str(e)

    @db_session
    def user_update(data):
        try:
            User[data['id']].ime = data['ime']
            User[data['id']].prezime = data['prezime']
            User[data['id']].email = data['email']
            User[data['id']].mob = data['mob']
            User[data['id']].rola = data['rola']
            User[data['id']].lozinka = data['lozinka']
            return True, None
        except Exception as e:
            return False, str(e)

    @db_session
    def user_brisi(data):
        try:
            User[data].delete()
            return True, None
        except Exception as e:
            return False, str(e)
```

Slika 54. CRUD metode u pozadini

Brisanje i ažuriranje korisnika se vrši putem modala koji se prikazuje pritiskom na zelenu tipku na kraju svakog retka u tablici dok se kreiranje novog korisnika vrši preko forme koja se na slici nalazi desno od tablice. Slika 55 prikazuje primjer modala iz aplikacije.



Slika 55. Modal za ažuriranje ili brisanje korisnika

„*Bez okvira, završili bismo s neredom neodrživog koda, od kojeg bi se većina bavila stvarima koje okvir otklanja od nas*“ *(Macrae, 2018:1, vlastiti prijevod).* Ista implementacija navedenih operacija proteže se kroz cijelu aplikaciju pa u nastavku slijedi samo vizualni prikaz nekoliko idućih komponenata web aplikacije. Za početak slika 56 prikazuje komponentu sa listom država.

Slika 56. Komponenta s listom država

Slijedi prikaz komponente s listom vrsta soba na slici 57 koja također sadrži formu za unos podataka. „*Izvlačenje korisničkog unosa iz obrasca u varijablu kojom možemo upravljati jedno je od glavnih prednosti upotrebe razvojnog okvira poput Vue-a*“ *(Nelson, 2018:85, vlastiti prijevod)*

Slika 57. Komponenta s listom vrsta soba

Komponenta s listom hotela vidljiva je na slici 58.

Slika 58. Komponenta s listom hotela

Idući dio je središnji dio web aplikacije, a u njemu se nalaze komponente koje sadrže informacije o predikcijama od strane modela. Napravljene su tri zasebne komponente koje prikazuje broj predviđenih otkazanih rezervacija po vrsti sobe po danu, broj

otkazanih jedinica za svaku vrstu sobe po danu te ukupan bruto iznos rezervacija koje su klasificirane kao otkazane. Sva tri prikaza su važeća za period od 8 dana. Aplikacija funkcionira na način da, kada korisnik otvori komponentu, pokreće se funkcija koja poziva upit da se iz baze podataka učitaju podaci i prikaže ih u tablici na komponenti. Za ovaj upit koristio se *Psycopg2*[6] adapter za baze podataka. Sam upit prikazan je na slici 59.

```
def ucitaj_predikcije():
    listaPredikcija = []
    start = date.today()
    conn = psycopg2.connect("dbname=Mlprojekt user=Kris password=bbforlife host=localhost")
    for i in range(0,8):
        cur = conn.cursor()
        datum = (start + timedelta(days=i)).strftime('%Y-%m-%d')
        cur.execute("""select sum (predikcije) from predikcije where dan_boravka = %(datum)s and vrsta_sobe like %(A21)s
                    and datum_predikcije = %(datump)s
                    union all
                    select sum (predikcije) from predikcije where dan_boravka = %(datum)s and (vrsta_sobe = %(A2)s or vrsta_sobe like %(A2M)s
                    or vrsta_sobe like %(A220)s  or vrsta_sobe like %(A210)s) and datum_predikcije = %(datump)s
                    union all
                    select sum (predikcije) from predikcije where dan_boravka = %(datum)s and vrsta_sobe like %(B2)s
                    and datum_predikcije = %(datump)s
                    union all
                    select sum (predikcije) from predikcije where dan_boravka = %(datum)s and vrsta_sobe like %(E1)s
                    and datum_predikcije = %(datump)s
                    union all
                    select sum (predikcije) from predikcije where dan_boravka = %(datum)s and vrsta_sobe like %(D1)s
                    and datum_predikcije = %(datump)s
                    union all
                    select sum (predikcije) from predikcije where dan_boravka = %(datum)s and vrsta_sobe like %(HA)s
                    and datum_predikcije = %(datump)s""",
                    {'datum': datum, 'datump': start.strftime('%Y-%m-%d'), 'A21':'A21%', 'A2':'A2%', 'A2M':'A2M%', 'A220':'A220%', 'A210':'A210%',
                    'B2':'B2%', 'E1':'E1%','D1':'D1%','HA':'HA%',})
        podaci = cur.fetchall()
        # pretvori None u nulu
        jed_list = [x[0] for x in podaci]
        jed_list = [0 if v is None else v for v in jed_list]
        zapis = {'datum':datum, 'A21':jed_list[0], 'A2':jed_list[1], 'B2':jed_list[2], 'E1':jed_list[3], 'D1':jed_list[4], 'HA':jed_list[5]}
        print(zapis)
        listaPredikcija.append(zapis)
    cur.close()
    conn.close ()
    for zapis in listaPredikcija:
        zapis['id'] = str(gid())
    return listaPredikcija
```

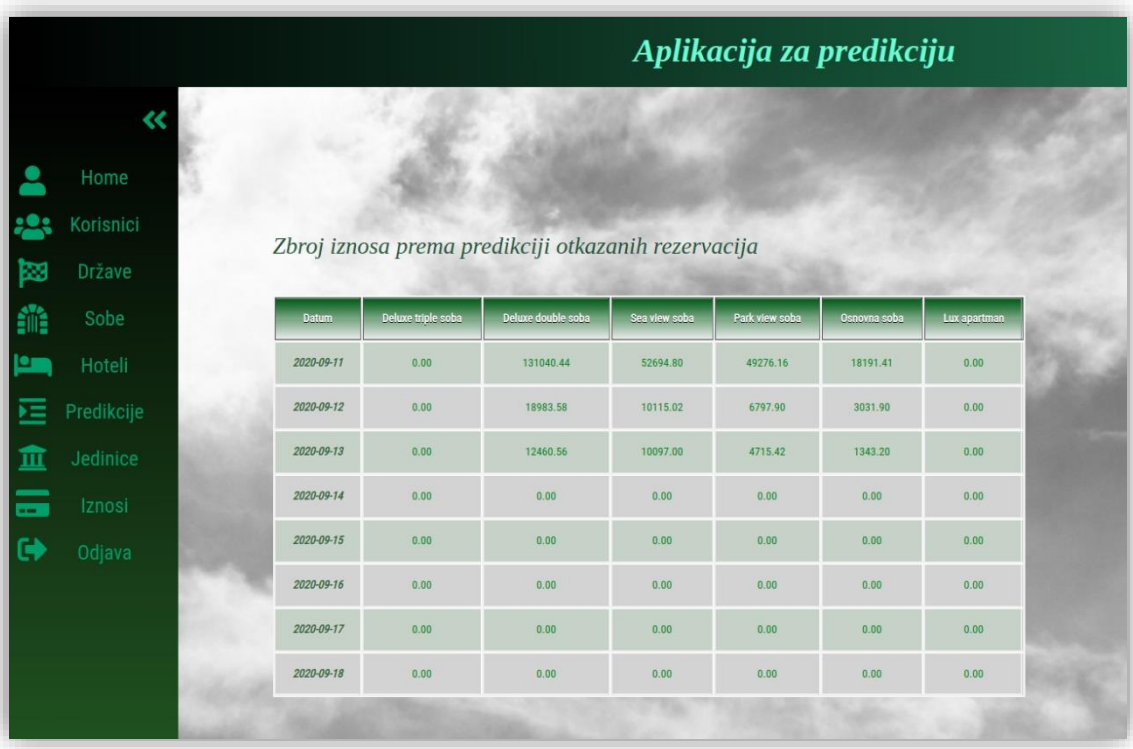
Slika 59. Upit za učitavanje broja previđenih otkazanih rezervacije

Isti princip je primijenjen i na komponente sa sumama jedinica te bruto iznosa prema predikcijama otkazanih rezervacija. Slijedi vizualni prikaz sve tri komponente nad demonstracijskim podacima. Slika 60 prikazuje komponentu s brojem predviđenih otkazanih rezervacija, slika 61 s brojem jedinica na temelju predviđenih otkazanih rezervacija te slika 62 prikazuje sumirane bruto iznose za svaku rezervaciju koju je model klasificirao kao otkazanu.

[6] Psycopg je jedan od najpopularnijih adaptera za PostgreSQL baze podataka za programski jezik Python

Slika 60. Komponenta s predikcijom otkazanih rezervacija

Slika 61. Komponenta sa zbrojem otkazanih jedinica



Slika 62. Komponenta sa zbrojem otkazanih jedinica

9. Zaključak

Prolazeći kroz sva poglavlja ovog rada dobio se uvid u proces razvoja modela strojnog učenja namijenjenog za problem binarne klasifikacije kao i razvoj adekvatne web aplikacije. Cilj projekta bio je, kao što je navedeno na samom početku, razviti početni model koji može poslužiti kao temelj za daljnje nadogradnje, a opet biti dovoljno razvijen da se njegove predikcije uzimaju u obzir kod određenih poslovnih procesa u samoj poslovnoj domeni. U konačnici to i jest postignuto te se razvijeni model već sada može koristiti u poslovnoj domeni kao dodatan alat kod upravljanja prihoda. Svakodnevni rezultati koje model daje na raspolaganje korisniku trebaju se uzeti u obzir te se daljnje poslovno odluke mogu adaptirati u skladu s njima. Uzmimo za primjer činjenicu da model specificira broj rezervacija i jedinica za koje smatra da će se otkazati te tako direktno korisniku govori koliko jedinica mora biti dostupno na tržištu za nadolazeći period. Samim time automatski se povećava mogućnost ostvarivanja maksimalnog prihoda.

Nastavno na temu procesa razvoja modela, može se zaključiti da je najvažniji stadij razvoja bila obrada podataka s kojima će model raditi. U ovom stadiju svakako postoji mjesta za dodatni napredak u smislu dodatne filtracije podataka kako bi se još preciznije izdvojio skup podataka koji još bolje definira rezervacije otkazane zbog vremenske prognoze. Iako vjerodostojni u velikoj mjeri, podaci kao takvi mogu se još dodatno obraditi pronalaženjem i micanjem vrijednosti koje u većoj mjeri odstupaju od normalne distribucije pojedine ulazne značajke. U ovom slučaju tom procesu nije pridodana veća važnost jer se kod rezervacija otkazanih na temelju vremenskih prognoza traži baš suprotno. Traže se anomalije i vrijednosti koje odstupaju od klasičnih kako bi se prepoznao potencijalni uzorak u svemu tome.

Što se tiče operativnog ciklusa, sve operacije u pozadinskom dijelu projekta se izvršavaju periodički, jednom unutar 24 sata. Za periodičko izvršavanje potrebnih funkcija korišteni su planeri zadataka iz knjižnica programskog jezika Python. Sam ciklus započinje učitavanjem dnevne vremenske prognoze koristeći API te dobivene podatke sprema u bazu podataka. Gotovo paralelno s tim procesom učitavaju se sve nove rezervacije kreirane u zadnja 24 sata te se također pohranjuju u istu bazu. Proces se nastavlja izdvajanjem aktivnih rezervacija za idućih 8 dana te adekvatnim

pripajanjem vremenskih prognoza istima. Potom se koriste cjevovodi kreirani kod treniranja modela kako bi obradili podatke. Obrađeni podaci se zatim koriste kao ulaz treniranom modelu koji ih analizira te daje svoje predikcije. Predikcije se na kraju pripajaju izdvojenom dijelu skupa podataka te se sve zajedno sprema u izdvojenu tablicu u bazi podataka. Ti rezultati se zatim prikazuju na sučelju web aplikacije u tabularnom obliku.

Kod analize rezultata raznih mjerenja, pokazalo se da učinkovitost modela svakako ima potencijala da se dodatno poveća i tu leže najveća ograničenja modela. Neki od načina za povećanje učinkovitosti su prikupljanje dodatnih povijesnih zapisa ili implementacija nekih dodatnih tehnika za ispravljanje visokog stupnja neravnoteže kod distribucije klasa u klasnom atributu poput takozvanih ponovnih uzorkovanja tehnika kao što su *Synthetic Minority Oversampling TEchnique* (SMOTE) ili SMOTE uzorkovanja uz čišćenje podataka pomoću Tomek veza[7].

Također, iako je za ovaj projekt odabran samo jedan tip modela, ne treba isključiti mogućnost implementacije ansambla (eng. *ensemble*) modela koji bi sigurno u nekoj mjeri još dodatno povećali preciznost predikcija. Ukratko sumirano, proces razvoja modela strojnog učenja mora imati kvalitetnu pripremu te veoma detaljnu obradu podataka koji će služiti kao ulaz.

Izuzev svim metoda i programskog koda koji je korišten kod gradnje dotičnog modela strojnog učenja, mogle su se vidjeti i tehnike razvoja web aplikacije putem Vue.js razvojnog okvira (eng. *framework*) koji je itekako progresivan kada je riječ o razvoju korisničkih sučelja ili jednostavnijih aplikacija gdje se traži visoka razina ponovne iskoristivosti komponenata. Također, lako ga je uklopiti u bilo koje pozadinsko razvojno okruženje što je bilo demonstrirano i kroz isječke programskog koda u ovom radu.

Nastavno na sve dosad navedeno, važno je spomenuti i da se daljnje nadogradnje modela mogu napraviti i proširenjem dodavanjem nekih funkcionalnosti poput mogućnosti da prilikom korištenja korisnik osobno korigira ponašanje modela dodavanjem niza ulaznih informacija poput koliko je model bio točan u dosadašnjem

[7] Tomek veze uklanjaju neželjeno preklapanje između klasa gdje se uklanjaju veze većinske klase dok svi minimalno udaljeni najbliži susjedski parovi nisu iste klase

radu ili stvaranjem još detaljnijeg ulaza po pitanju informacija o vremenskoj prognozi na način da model koristi još manje raspone vremena kod učitavanja prognoze, poput prognoze za svaki sat dana. Još jedna moguća nadogradnja jest da se razvije model koji će raditi predikcije za više smještajnih objekata, što je posebice korisno za hotelske kuće.

Ovaj cijeli proces je svakako potvrdio da su modeli strojnog učenja veliki doprinos bilo kojoj poslovnoj domeni te postotak njihove efikasnosti ovisi isključivo o razvojnom programeru.

Za kraj neka se spomene samo da je kompletan programski kod za ovaj projekt dostupan na Github repozitoriju na sljedećoj poveznici :

https://github.com/KristianB2020/Aplikacija_za_predikciju.git

Literatura i izvori

[1] “**Python machine learning**“, Sebastian Raschka & Vahid Mirjalili, Packt Publishing Ltd, 2019. godina

[2] „**Introduction to Machine Learning with Python**“, Andreas C. Műller & Sarah Guido, O’Reilly Media Inc, 2017. godina

[3] "**Machine Learning In Python W/Ws**", 1st Edition, Michael Bowles, John Wiley & Sons, Inc, 2015. godina

[4] "**Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**", 2nd Edition, Aurélien Géron, O’Reilly Media Inc, 2019. godina

[5] "**Data Science from Scratch: First Principles with Python**", 2nd Edition, Joel Grus, O’Reilly Media Inc, 2019. godina

[6] "**Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython**", 2nd Edition, Wes McKinney, O’Reilly Media Inc, 2017. godina

[7] "**Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python**", 2nd Edition, Peter Bruce, O’Reilly Media Inc, 2020. godina

[8] "**Vue.js: Up and Running: Building Accessible and Performant Web Apps**", 1st Edition, Callum Macrae, O’Reilly Media Inc, 2018. godina

[9] "**Getting to Know Vue.js: Learn to Build Single Page Applications in Vue from Scratch**", 1st Edition, Brett Nelson, Springer Science+Business Media, 2018. godina

[10] „**Matplotlib for Python Developers**“, Sandro Tosi, Packt Publishing Ltd, 2009. godina

[11] "**CSS: The Definitive Guide: Visual Presentation for the Web**", 4th Edition, Eric A. Meyer & Estelle Weyl, O’Reilly Media Inc, 2017. godina

[12] " **Hrvatska enciklopedija**“, mrežno izdanje. Leksikografski zavod Miroslav Krleža, 2020. godina

[13] "**Introducing JSON**", Crockford Douglas, json.org, 2009. godina

Popis slika:

Popis tablica: