Obrada i analiza prometnog opterećenja u svrhu povećanja učinkovitosti pozivnog centra

Hrsto, Matea

Sveučilište u Zagrebu

Fakultet prometnih znanosti

DIPLOMSKI RAD

OBRADA I ANALIZA PROMETNOG OPTEREĆENJA U SVRHU POVEĆANJA UČINKOVITOSTI POZIVNOG CENTRA

PROCESSING AND ANALYSIS OF TRAFFIC LOAD FOR THE PURPOSE OF INCREASING THE EFFICIENCY OF THE CALL CENTER

Mentor: dr. sc. Tomislav Erdelić

Studentica: Matea Hrsto

JMBAG: 0135246126

Zagreb, rujan 2024.

Zahvaljujem mentoru dr. sc. Tomislavu Erdeliću na mentorstvu tijekom izrade diplomskog rada i na razumijevanju.

Zahvaljujem telekomunikacijskoj firmi i kolegama koji su mi ustupili podatke iz pozivnog centra.

Zahvaljujem svojim prijateljicama, prijateljima, a posebice svojoj obitelji na neprestanoj podršci tijekom cijelog školovanja. Ovaj rad posvećujem svojoj majci koja je bila moja najveća podrška.

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI
POVJERENSTVO ZA DIPLOMSKI ISPIT

Zagreb, 22. travnja 2024.

Zavod: **Zavod za inteligentne transportne sustave**
Predmet: **Napredne baze podataka**

DIPLOMSKI ZADATAK br. 7657

Pristupnik: **Matea Hrsto (0135246126)**
Studij: Promet
Smjer: Informacijsko-komunikacijski promet

Zadatak: **Obrada i analiza prometnog opterećenja u svrhu povećanja učinkovitosti pozivnog centra**

Opis zadatka:

Diplomski rad istražuje obradu i analizu podataka prikupljenih iz call centra s ciljem poboljšanja učinkovitosti informacijsko-komunikacijskog sustava. Prikupljeni podaci obuhvaćaju količinu poziva, njihovo trajanje, broj aktivnih agenata te vremenski okvir završetka poziva. Kroz dizajniranje relacijske sheme baze podataka, podatci će biti pohranjeni radi daljnje obrade. Cilj istraživanja je analizirati povijesne podatke kako bi se identificirali ključni obrasci, poput vremenskih intervala s najvećim opterećenjem, te pronašle strategije za smanjenje opterećenja. Također, istraživanje će se fokusirati na izradu strojnog modela za predviđanje budućeg opterećenja sustava. Konačno, rezultati će biti prikazani kroz web-aplikaciju radi lakše analize prikupljenih i obrađenih podataka.

Mentor:

Tomislav Erdelić

dr. sc. Tomislav Erdelić

Predsjednik povjerenstva za diplomski ispit:

_____

SAŽETAK

U ovom radu obrađuju se i analiziraju prikupljeni podaci radi provjere prometnog opterećenja pozivnog centra odnosno svrha ovog rada je analiza povijesno prikupljenih podataka iz pozivnog centra radi povećanja učinkovitosti pozivnog centra. Podaci koji su prikupljeni su: količina primljenih poziva, duljina trajanja poziva, broj aktivnih agenata te podatak u kojem vremenskom periodu je završen poziv. Dizajnirana je relacijska shema baze podataka u programu *SQL Server Management Studio*, u koju su spremljeni podaci. Cilj je utvrditi u kojem periodu se odvija najveći broj poziva, odnosno kad je prometno opterećenje pozivnog centra na vrhuncu, kako bi se pronašao način da se smanji prometno opterećenje pozivnog centra. Nakon analize podataka, dobiveni rezultati prikazani su u web-aplikaciji, a strojnim modelom predviđeno je buduće prometno opterećenje rada pozivnog centra.

KLJUČNE RIJEČI: prometno opterećenje, relacijska shema baze podataka, podaci, SQL, web aplikacija, strojno učenje

SUMMARY

In this work, the collected data is processed and analyzed for the purpose of checking the traffic load of the call center, that is, the purpose of this work is to analyze the historically collected data from the call center in order to increase the efficiency of the call center. The data collected are: the number of calls received, the length of the call, the number of active agents and the information in which time period the call ended. A relational schema of the database was designed in the program SQL Server Management Studio, in which the data was saved. The goal is to determine in which period the largest number of calls takes place, that is, when the traffic load of the call center is at its peak, in order to find a way to reduce the traffic load of the call center. After the data analysis, the obtained results are presented in the web application, and the machine model predicts the future traffic load of the call center.

KEY WORDS: traffic load, relational database schema, data, SQL, web application, machine learning

Sadržaj

1. Uvod

Gotovo svaka tvrtka koja nudi neku uslugu na tržištu, u većini slučajeva, unutar svojih odjela ima i odjel pozivnog centra. Pozivni centar služi kako bi korisnici mogli komunicirati s konkretnom tvrtkom u svrhu prijave problema sa uslugom, interakcije koliko su korisnici zadovoljni s kupljenom uslugom, prijave problema sa plaćanjem usluge kao i potvrde o plaćanju usluge, kupovine dodatne usluge itd. U današnje vrijem pozivni centri su uglavnom unaprijeđeni sa IVR-om (eng. Interactive Voice Response) koji svojim radom smanjuje radno opterećenje agenata pozivnog centra. IVR je interaktivni glasovni odgovor koji omogućuje korisnicima (pozivateljima) da se usmjere na odgovarajući odjel unutar samog pozivnog centra. Slušanjem IVR izbornika pozivatelj pritiskom na tipku za biranje na svom telefonu, odabire željenu značajku koja mu je ponuđena.

U današnjem digitalnom dobu, pozivni centri predstavljaju ključnu komponentu u pružanju kvalitetne korisničke podrške i održavanju pozitivnog odnosa s korisnicima. Pozivni centri omogućuju tvrtkama da efikasno upravljaju velikim količinama poziva, odgovaraju na upite korisnika, te rješavaju probleme s kojima se korisnici suočavaju. Kako bi se osigurala visoka razina usluge i zadovoljstvo korisnika, potrebno je neprekidno analizirati i optimizirati rad pozivnog centra. Stoga, imajući navedeno u vidu ovaj rad se bavi analizom podataka iz pozivnog centra s ciljem povećanja učinkovitosti njegovog rada.

Rad je usmjeren na analizu povijesnih podataka prikupljenih iz pozivnog centra kako bi se identificirali ključni trenuci u kojima dolazi do najvećeg opterećenja sustava. Prikupljeni podaci uključuju broj primljenih poziva, duljinu trajanja svakog poziva, broj aktivnih agenata, te vremenski okvir u kojem su pozivi završeni. Ovi podaci predstavljaju temelj za daljnju analizu i izradu modela koji će pomoći u razumijevanju obrazaca ponašanja unutar pozivnog centra, te u predviđanju budućih prometnih opterećenja.

U prvom dijelu rada, fokus je na definiranju osnovnih pojmova vezanih uz relacijske baze podataka. Razmatrat će se koncepti i prednosti relacijskog podatkovnog modela, te uloga sustava za upravljanje bazama podataka (SUBP). Ova teoretska podloga biti će ključna za razumijevanje kasnije praktične primjene SQL Server Management Studia u izradi baze podataka koja će služiti za pohranu i obradu podataka iz pozivnog centra.

Drugi dio rada odnosi se na izradu web-aplikacije koja će prikazivati rezultate analize. Web aplikacija će biti razvijena u MVC (eng. Model-View-Controller) okruženju, što

omogućava modularni pristup u razvoju aplikacija, gdje se poslovna logika, podaci i prikaz podataka razdvajaju u zasebne dijelove. Pomoću ove aplikacije, podaci prikupljeni iz pozivnog centra bit će vizualizirani, što će omogućiti voditeljima pozivnog centra da jednostavno interpretiraju rezultate analize i donesu odgovarajuće i konkretne odluke u svrhu optimizacije rada centra.

Konačno, primjenom metode strojnog učenja, pokušat će se predvidjeti buduće opterećenje pozivnog centra. Različiti algoritmi strojnog učenja, uključujući linearnu i logističku regresiju, kao i neuronske mreže, bit će korišteni za izradu modela predikcije. Ovi modeli omogućit će da se identificiraju periodi s visokim prometnim opterećenjem, te da se unaprijed poduzmu koraci za optimizaciju resursa, kao što su raspodjela agenata ili prilagodba radnog vremena.

Zaključno, cilj ovog rada je pružiti detaljan pregled procesa analize i optimizacije rada pozivnog centra, koristeći moderne tehnologije i metode. Očekuje se da će rezultati ovog istraživanja pomoći tvrtkama u poboljšanju kvalitete usluge, smanjenju vremena čekanja, te povećanju ukupne učinkovitosti njihovih pozivnih centara. Prikazani rezultati poslužit će kao osnova za daljnje istraživanje i razvoj u području optimizacije korisničke podrške.

2. Relacijska baza podatka

Informacijski sustav je sustav koji prikuplja, pohranjuje, čuva, obrađuje i distribuira informacije koje su važne za društvo i organizaciju, s ciljem da budu dostupne i korištene za svakog tko ih želi koristiti, uključujući osoblje, klijente, poslovodstvo i druge. Informacijski sustav aktivni je društveni sustav koji može, iako ne mora, koristiti suvremenu informacijsko-komunikacijsku tehnologiju. Sustav je skup povezanih komponenata koje zajednički rade zbog ispunjena nekog cilja. Svi poslovni podaci spremljeni su u bazi podataka informacijskog sustava, [1].

Model podataka poslovnog informacijskog sustava, odnosno segmenata stvarnog svijeta, predstavlja baza podataka sa sadržajem svojih podatka. Ona je organizirana serija međusobno povezanih podataka, koja je pohranjena na sredstvima kao što su poslužitelj, mreža ili oblak, informacijsko komunikacijske tehnologije. Baza podataka skup je međusobno ovisnih podataka, spremljenih bez redundancije (zalihosti), koji služe jednoj ili više aplikacija na optimalan način, gdje su podaci neovisni od programa kojima se obrađuju i gdje postoji kontrolirani pristup do podatka, [2].

Tehnologija baze podataka brzo se razvila u zadnja tri desetljeća te se od tada događa uspon i dominacija sustava relacijskih baza podataka. Iako su mnogi specijalizirani sustavi baza podataka (objektno orijentirani, prostorni, multimedijski, itd.) pronašli značajne korisničke zajednice u znanosti i inženjerstvu, relacijske baze podataka ostaju dominantna tehnologija za poslovna poduzeća. Mnoga od ovih pomagala za dizajn pojavila su se kao komponenta baze podataka, odnosno kao alati za računalno potpomognuto programsko inženjerstvo (CASE eng. Computer-Aided Software Engineering) te mnogi od njih nude mogućnost interaktivnog modeliranja pomoću pojednostavljenog modeliranja podataka, [3].

Logički dizajn - to jest, struktura odnosa osnovnih podataka i njihova definicija u određenom sustavu baze podataka - uglavnom je domena dizajnera aplikacija. Ovi dizajneri mogu učinkovito raditi s alatima kao što su ERwin Data Modeler ili Rational Rose s UML-om, kao i s čisto ručnim pristupom. Fizički dizajn, izrada učinkovitih mehanizma za pohranu i dohvaćanje podataka na računalnoj platformi koja se koristi, obično je domena administratora baze podataka (DBA eng. DataBase Administrator), [3].

2.1. Sustav za upravljanje bazom podataka

Programski sustav (softver) koji omogućuje rad s bazom podataka je sustav za upravljanje bazom podatka (SUBP, eng. DBMS – Database Management System*)*. Taj sustav sadržava:

- Upravljačke funkcije npr.:
  - funkcije sigurnosti što znači zaštitu od neovlaštenog korištenja, korisnicima baze podatka se definira koje operacije mogu obavljati nad njom;
  - funkciju koja čuva integritet baze podataka npr. zaštita od mogućih oštećenja, primjer kako očuvati bazu podatka je uzimanje sigurnosnih kopija (eng. backup) , a nakon oštećenja radi se postupak oporavka (eng. recovery);
  - funkcija statističkog praćenja rada baze podataka;
- Funkcije za definiranje baze podataka (eng. Data Definition):
  - ostvaruje se naredbama za definiranje podataka (eng. DDL – Data Definition Language), jezika za rad s bazom podataka, primjerice SQL-a (eng. Structured Query Language);
  - s njima se definira fizički opis odnosno fizička organizacija baze podatka, te logički opis baze podatka odnosno naziv, tip i opis svakog podatka i odnos prema drugim elementima podataka;
- Funkcije za manipulaciju podacima koje se nalaze u bazi podataka:
  - postiže se naredbama za manipulaciju podacima (eng. DML – Data Manipulation Language) jezika za rad s bazom podataka, [1].

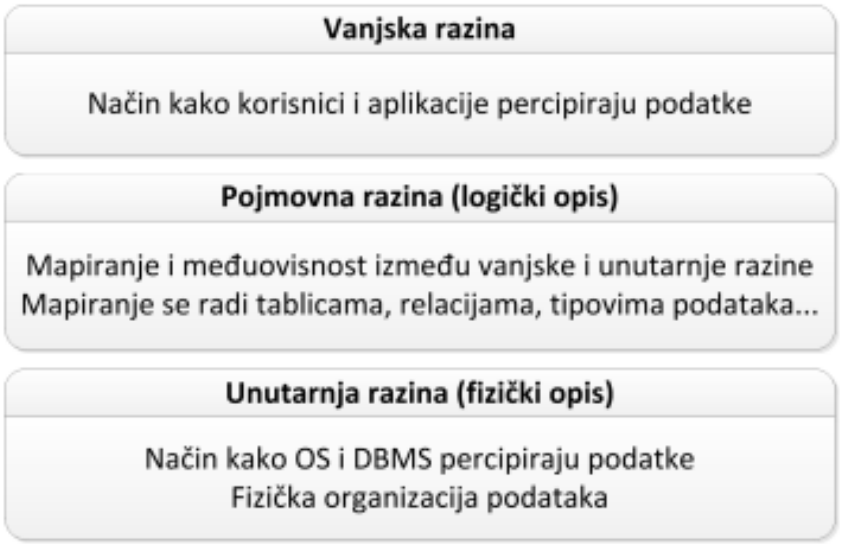
Svaki SUBP ima osnovne zadaće koje mora obavljati, a to su:

- očuvati integritet podataka u bazi,
- omogućavati konkurentnosti, odnosno omogućiti da više korisnika ima istovremeni pristup istim podacima,
- identificirati optimalne strukture u svrhu boljeg upravljanja podacima,
- rukovanje i opis podataka,
- omogućiti opis podataka metapodacima,
- u slučaju gubitka podataka omogućiti obnovu istih,
- zaštiti bazu podataka od neovlaštenog pristupa i korištenja, [4].

Jedna od bitnih zadaća SUBP-a, koju valja izdvojiti, je provođenje i administriranje korisničkih prava. S obzirom da su baze podataka višekorisnički sustavi, nužno je osigurati dodjelu ovlasti pristupu, izmjeni i brisanju podataka te isto tako poštivati dodijeljena prava. Postoje razlike u nekim SUBP-ovima po načinu upravljanja i provođenja pravima za korištenje podataka. Implementacija SUBP-a se većinom provodi po već prijašnjim dodijeljenim ulogama i pravima. U posljednje vrijeme sve rjeđe se koristi hijerarhijski model prava, i pristupa se modernijem modelu s kojim vlasnik podataka dodijeli prava drugom korisniku bez direktnog upletanja administratora same baze podataka.

Arhitektura SUBP-a je podijeljena na vanjsku razinu, pojmovnu razinu i unutarnju razinu. Razina koja je najbliža korisniku je vanjska razina. Ona predstavlja aplikaciju koja služi za pristup podacima i rad s njima. Vanjska razina se formira sukladno potrebama i ograničenjima korisnika u uporabi baze podataka. Ova razina često se naziva razinom pregleda jer korisnicima prikazuje samo relevantne informacije iz baze podataka, dok sakriva ostale detalje koji nisu potrebni za određeni korisnički profil, [4].

Pojmovna razina (logički opis) odnosi se na način na koji se baza podataka oblikuje prema pravilima projektiranja. Na ovoj razini definira se logička struktura baze, uključujući vrste podataka, organizaciju tablica, odnose među njima (primarni i strani ključevi) i ograničenja koja osiguravaju točnost i dosljednost podataka. Unutarnja razina (fizički opis) bavi se načinom na koji se podaci fizički pohranjuju i organiziraju na disku. Ona opisuje kako se elementi s logičke razine prevode u konkretne lokacije na disku. Najmanja logička jedinica kojom sustav za upravljanje bazom podataka upravlja je stranica. Ova razina također uključuje upravljanje radnom memorijom, [4].



Slika 1. Razine arhitekture SUBP-a, [3]

Da bi se sustav upravljanja bazom podataka smatrao relacijskim, mora ispunjavati određeni skup dvanaest pravila koje je 1985. godine postavio tvorac relacijskog modela, dr. Edgar F. Codd. Osnovno pravilo zahtijeva da sustav ima sve potrebne funkcionalnosti za potpuno upravljanje podacima u skladu s relacijskim modelom. SUBP se smatra relacijskim ako ispunjava ovo osnovno pravilo i barem šest od preostalih pravila, a pravila su, [5]:

1. **Pravilo prikaza informacija** (eng. „The Information Rule“) – svi podaci u relacijskoj bazi predstavljeni su isključivo na logičkoj razini, kroz vrijednosti unutar tablica.
2. **Pravilo zajamčenog pristupa** (eng. „Guaranteed Access Rule“) – pristup svakom pojedinačnom podatku u relacijskoj bazi mora biti omogućen koristeći kombinaciju naziva tablice, primarnog ključa i naziva stupca.
3. **Sistematično postupanje s nepoznatim vrijednostima** (eng. „Systematic Treatment of Null Values“) – relacijski sustav mora podržavati nepoznate vrijednosti (koje nisu prazni nizovi ili nule) na dosljedan način, bez obzira na tip podatka, kako bi se prikazao sadržaj koji nedostaje ili nije primjenjiv.
4. **Dinamički online katalog zasnovan na relacijskom modelu** (eng. „Dynamic Online Catalog Based on the Relational Model“) – struktura baze podataka treba biti predstavljena na isti način kao i podaci, omogućavajući primjenu istih relacijskih pravila na upite nad shemom baze kao i na upite nad podacima.

5. **Pravilo sveobuhvatnog jezika za rad s podacima** (eng. „Comprehensive Data Sublanguage Rule“) – iako sustav može podržavati više jezika, mora postojati barem jedan jezik sa sintaksom koji omogućava definiranje podataka, kreiranje pogleda, manipulaciju podacima, te postavljanje transakcijskih i integritetskih ograničenja.
6. **Pravilo ažuriranja pogleda** (eng. „View Updating Rule“) – svi pogledi koji mogu biti ažurirani u teoriji, moraju biti ažurirani i putem SUBP-a, omogućujući korisnicima da mijenjaju podatke kroz sve dostupne poglede.
7. **Pravilo visokog nivoa rukovanja podacima** (eng. „High-level Insert, Update, and Delete“) – SUBP mora omogućiti manipulaciju relacijama kao jedinstvenim entitetima, omogućujući operacije poput dohvaćanja, umetanja, izmjene i brisanja podataka u jednoj naredbi.
8. **Neovisnost fizičkih podataka** (eng. „Physical Data Independence“) – aplikacije i programske implementacije ne smiju biti pogođene promjenama u načinu pohrane ili pristupa podacima na fizičkoj razini.
9. **Neovisnost logičkih podataka** (eng. „Logical Data Independence“) – aplikacije i programske implementacije ne smiju biti pogođene promjenama u strukturi baze podataka ili njenim tablicama.
10. **Neovisnost integriteta** (eng. „Integrity Independence“) – pravila za očuvanje integriteta sistema moraju se definirati pomoću relacijskog jezika i pohraniti u katalog baze podataka, a ne u aplikacije ili druge programske dijelove.
11. **Neovisnost distribucije (**eng. „Distribution Independence“) – sustav za upravljanje bazom podataka mora omogućiti distribuciju podataka preko više sistema, bez potrebe da korisnik bude svjestan te distribucije.
12. **Pravilo protiv zaobilaženja (**eng. „Non-subversion Rule“) – ako sustav koristi i jezik niže razine, taj jezik ne smije omogućiti zaobilaženje ograničenja integriteta koja su definirana relacijskim jezikom višeg nivoa.

2.2. Relacijski podatkovni model

Relacijska baza podataka je temeljena na relacijskom modelu podatka. Ona se sastoji od skupa povezanih tablica, odnosno relacija u obliku dvodimenzionalnih tablica. Osnovni objekt relacijske baze podatka je tablica i u njoj su pohranjeni podaci. Prilikom kreiranja baze,

u oblikovanju konceptualne sheme prije svega mora se otkriti entitete, veze i atribute. Za definiranje podataka i veza između podatka koristi se modeliranje entiteta i veza (eng. Entity-Relationship Model - ER Model). S obzirom da je u stvarnom svijetu teško pogoditi relacijsku shemu, upotrebljava se pomoćna fraza zvana ER-model koji se dalje pretvara u relacijski model. Za bolje shvaćanje ER modela potrebni su nam sljedeći pojmovi:

- Entiteti: događaji ili objekti koji su od interesa,
- Veze: spona među entitetima koji su od interesa,
- Atributi: obilježje entiteta i veza koja su od interesa [6].

Entiteti su osnovni elementi za koje se prikupljaju informacije i kojima se mogu odrediti različite karakteristike. Entitet je nešto što može postojati ili ne postojati i može se prepoznati. To može biti fizički predmet (kao auto ili zaposlenik), događaj (poput kupovine auta ili nogometne utakmice). Nakon što se odluči koji entiteti će se pratiti, treba se odrediti karakteristike koje će ih opisivati. Entiteti se opisuju atributima na primjer, atributi za auto mogu biti registracija, broj šasije, model, boja, i slično. Ako neki atribut može imati više vrijednosti i/ili zahtijeva dodatne atribute, onda se smatra novim entitetom.

Tip entiteta definiran je njegovim imenom i pripadajućim atributima. U jednoj konceptualnoj shemi ne smije biti dva entiteta s istim imenom, a svi atributi unutar istog entiteta moraju imati različita imena. Međutim, različiti entiteti mogu imati atribute s istim imenom (npr. entiteti „poslodavac“ i „zaposlenik“ mogu imati atribut „prezime“). Kandidat za ključ je atribut ili skup atributa koji jedinstveno identificira primjerak entiteta. Na primjer, za entitet „auto“, atribut „registracija“ može biti kandidat za ključ, jer svako vozilo ima jedinstvenu registraciju. Ako entitet ima više od jednog kandidata za ključ, odabire se jedan koji će se koristiti kao primarni ključ, [6].

Veze predstavljaju odnose između entiteta. Ove veze uspostavljaju odnose između dva ili više tipova entiteta (na primjer, veza „zaposlen“ između entiteta „zaposlenik“ i „tvrtka“). Veze služe da bi izrazili kako su entiteti povezani. Veze se mogu klasificirati prema različitim kriterijima, [7]:

- Kardinalnost: odnosi se na broj entiteta koji mogu biti povezani.
  - 1:1 (jedan na jedan) – jedan entitet iz prvog skupa povezan je s jednim entitetom iz drugog skupa.
  - 1:N (jedan na više) – jedan entitet iz prvog skupa povezan je s više entiteta iz drugog skupa.

    - N:M (više na više) – više entiteta iz prvog skupa povezano je s više entiteta iz drugog skupa.
- Obaveznost članstva: odnosi se na to da li je veza obavezna ili opcionalna.
    - Obavezno – entitet mora biti povezan s nekim entitetom iz drugog skupa.
    - Opcionalno – entitet može biti povezan, ali ne mora nužno biti.
- Brojnost: odnosi se na broj entiteta koji sudjeluju u vezi.
    - Unarne – veza između entiteta istog tipa.
    - Binarne – veza između dva različita tipa entiteta.
    - Ternarne – veza između tri različita tipa entiteta.

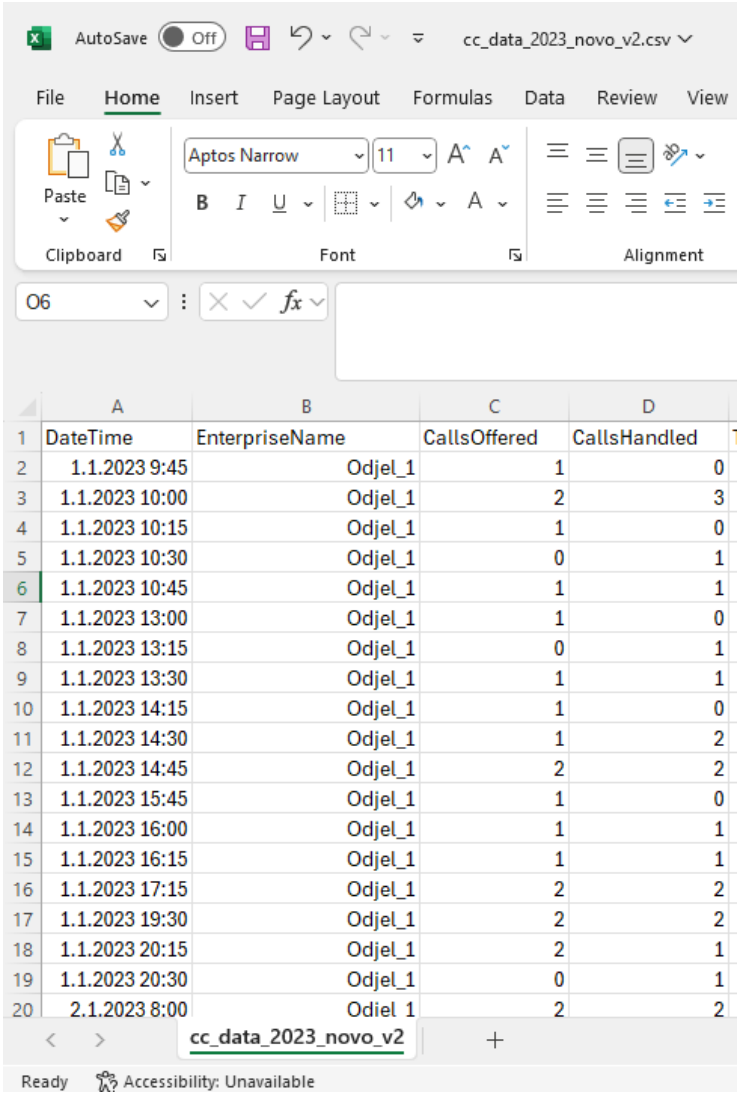
Postoji više vrsta relacijskih baza podataka koje uključuju različite tehnologije i pristupe za upravljanje i pohranu podataka, a to su, [7]:

- Klasične relacijske baze podataka su tradicionalni sustavi poput MySQL-a, PostgreSQL-a i Microsoft SQL Server-a. Ove baze koriste SQL za izvođenje upita i upravljanje podacima. Podaci su organizirani u tablice koje se sastoje od redaka i stupaca, a sheme i ključevi pomažu u održavanju integriteta podataka.
- Objektno-relacijske baze podataka kombiniraju značajke relacijskih modela s objektno-orijentiranim pristupom. Ove baze omogućuju pohranu i upravljanje složenijim strukturama podataka, poput objekata, nasljeđivanja i metoda. Oracle Database je primjer ove vrste baze.
- In-memory relacijske baze podataka pohranjuju podatke u radnoj memoriji umjesto na diskovima, što omogućuje izuzetno brze upite. Primjeri takvih baza uključuju SAP HANA i Oracle TimesTen.
- Distribuirane relacijske baze podataka omogućuju pohranu podataka na više fizičkih lokacija. Ova funkcionalnost pomaže u postizanju skalabilnosti i visoke dostupnosti. Apache Cassandra je jedan od primjera distribuiranih relacijskih baza podataka.
- Cloud relacijske baze podataka koje se nalaze na cloud platformama, omogućujući skaliranje i upravljanje putem interneta. Ove baze nude fleksibilnost i mogućnosti plaćanja prema korištenju. Primjeri ove vrste baze podataka su Amazon RDS, Google Cloud SQL, Microsoft Azure SQL Database.

Svaka od ovih vrsta relacijskih baza podataka ima specifične karakteristike koje ih čine prikladnima za različite potrebe sustava ili aplikacija, u pogledu performansi, skalabilnosti i funkcionalnosti, [7].

3. Sustav za spremanje i prikaz podataka iz pozivnog centra

Podaci koji se obrađuju u ovom radu proizlaze iz pozivnog centra telekomunikacijske firme. Podaci su prikupljeni kroz cijelu 2023. godinu, te su spremljeni u Excel tablicu csv formata. Potrebni podaci za ovaj rad su bili datum, dolazni pozivi i odgovoreni pozivi. Kao što je vidljivo na slici 2 količina poziva je bila spremljena u intervalima po petnaest minuta unutar dvadeset i četiri sata, te su bili grupirani po odjelima unutar telekomunikacijske firme. Naziv stupca DateTime predstavlja datum i vrijeme, EnterpriseName predstavlja naziv odjela, CallsOffered su dolazni pozivi, a CallsHandled su odgovoreni pozivi. Ukupna količina redaka s podacima je 273665.



	A	B	C	D
1	DateTime	EnterpriseName	CallsOffered	CallsHandled
2	1.1.2023 9:45	Odjel_1	1	0
3	1.1.2023 10:00	Odjel_1	2	3
4	1.1.2023 10:15	Odjel_1	1	0
5	1.1.2023 10:30	Odjel_1	0	1
6	1.1.2023 10:45	Odjel_1	1	1
7	1.1.2023 13:00	Odjel_1	1	0
8	1.1.2023 13:15	Odjel_1	0	1
9	1.1.2023 13:30	Odjel_1	1	1
10	1.1.2023 14:15	Odjel_1	1	0
11	1.1.2023 14:30	Odjel_1	1	2
12	1.1.2023 14:45	Odjel_1	2	2
13	1.1.2023 15:45	Odjel_1	1	0
14	1.1.2023 16:00	Odjel_1	1	1
15	1.1.2023 16:15	Odjel_1	1	1
16	1.1.2023 17:15	Odjel_1	2	2
17	1.1.2023 19:30	Odjel_1	2	2
18	1.1.2023 20:15	Odjel_1	2	1
19	1.1.2023 20:30	Odjel_1	0	1
20	2.1.2023 8:00	Odjel_1	2	2

Slika 2. Prikupljeni podaci

Prikupljeni podaci koji se nalaze u formatu .csv uvesti (eng. import) će se u novo kreiranu bazu podataka. Unutar programa Microsoft SQL Server Management Studio (SSMS)

izvoditi će se upiti nad tablicom, kako bi se došlo do željenih analiza i rezultata. Prikaz rezultata bit će prikazan u novo kreiranoj web-aplikaciji, koja će se kreirati povezivanjem baze podataka unutar Visual Studia, putem MVC okruženja. Detaljniji opis ove procedure biti će opisan u sljedećim po poglavljima.

3.1. SQL Server Management Studio

SQL Server Management Studio (SSMS) integrirano je okruženje za upravljanje bilo kojom SQL infrastrukturom. SSMS se koristi za pristup, konfiguraciju, upravljanje, administraciju i razvoj svih komponenti SQL Servera, Azure SQL baze podataka, Azure SQL upravljane instance, SQL Servera na Azure VM i Azure Synapse Analytics. SSMS pruža jedan sveobuhvatni uslužni program koji kombinira široku grupu grafičkih alata s mnogo uređivača skripti kako bi omogućio pristup SQL Serveru za programere i administratore baza podataka svih razina vještina, [8].

Microsoft je objedinio sve te alate u jedan, s fokusom na pružanje alata koji odgovara potrebama programera i administratora baze podataka. SSMS je alat koji pruža ulaznu točku za gotovo sve funkcionalnosti SQL Servera. SQL Server predstavlja sustav koji definira metode koje se koriste za upravljanje bazama podataka. Upravlja relacijskim bazama podataka i sadrži veliki izbor aplikacija za obradu transakcija, analitiku u poduzeću IT okruženja i poslovnu inteligenciju. SSMS koristi se za upravljanje poslužiteljem i bazama podataka. SSMS služi i akademskim potrebama i aplikacijama na razini industrije koje su nam danas dostupne. Postoji nekoliko različitih verzija: Enterprise, Standard, Web, Developer i Express, [9].

3.2. Strukturirani jezik upita

SQL je kratica za strukturirani jezik upita, najčešće se izgovara „SQL“, a ponekad i „Sie-Quel“. To je standardizirani jezik koji se koristi za upravljanje relacijskim bazama podataka. Korištenje baze podataka kroz SQL nudi mnoge prednosti u odnosu na tradicionalne sustave tekstualnih datoteka, osobito kada je riječ o pohranjivanju i upravljanju podacima. SQL jezik je prvobitno osmišljen kao alat za pretraživanje podataka u bazi podataka. Kada se ovaj jezik koristi unutar aplikacije, to se naziva SQL programiranje. Na primjer, SQL se može koristiti za ažuriranje podataka kroz web aplikaciju. SQL programiranje omogućuje umetanje zapisa u bazu podataka (INSERT), pretraživanje (SELECT), ažuriranje (UPDATE) i brisanje podataka (DELETE), [10].

SQL jezik sastoji se od nekoliko različitih grupa funkcionalnosti, [11]:

- **Jezik za definiranje podatkovne strukture (DDL)**: ova grupa omogućava kreiranje, izmjenu i brisanje definicija tablica i pogleda, te postavljanje ograničenja kako bi se očuvao integritet podataka.
- **Jezik za manipuliranje podacima (DML)**: omogućuje izvršavanje upita za dohvaćanje, upisivanje, izmjenu i brisanje redaka u tablicama.
- **Okidači i napredna ograničenja**: ova funkcionalnost podržava automatsko izvršavanje određenih radnji u bazi podataka, ovisno o događajima ili promjenama koje se dogode, koji su definirani pomoću okidača.
- **Ugrađeni i promjenjivi SQL**: ugrađeni SQL omogućava pokretanje SQL koda unutar drugih programskih jezika poput C-a ili COBOL-a, dok promjenjivi SQL omogućuje kreiranje i izvršavanje upita tijekom izvršavanja programa.
- **Udaljeni pristup bazi i veza klijent-poslužitelj**: ova funkcionalnost omogućava definiranje načina na koji aplikacija komunicira s SQL poslužiteljem i način na koji se pristupa podacima putem mreže.
- **Upravljanje transakcijama**: ova grupa omogućava kontrolu nad načinom na koji se transakcije obrađuju i izvršavaju u sustavu upravljanja bazama podataka.
- **Sigurnost**: sadrži mehanizme za kontrolu korisničkog pristupa podacima, uključujući prava pristupa tablicama i pogledima.
- **Napredne funkcionalnosti**: ove funkcionalnosti uključuju objektno-orijentirane mogućnosti, rekurzivne upite, upite za podršku prilikom odlučivanja, funkcije za rudarenje podataka te alate za upravljanje prostornim, tekstualnim i XML (eng. Extensible Markup Language) podacima, [11].

Prilikom unosa podataka u tablice, stupce unutar tablice je potrebno definirati oblikom, odnosno tipom podataka. Tipovi podataka služe za definiranje vrijednosti koje se čuvaju u danom stupcu. Bilo da se baza podataka projektira, kreira izvještaj, analiziraju podaci i sl. bitno je poznavati koje se vrste podataka mogu čuvati i na koji način se mogu upotrebljavati. U Tablici 1 je prikazano nekoliko osnovnih tipova podataka u SQL-u, [12]:

TIP PODATAKA	OPIS	PRIMJER
CHAR	Niz određene duljine, čuva maksimalno 8000 oznaka, a svaka oznaka zauzima jedan bajt	'TipoviPodataka'
NCHAR	Unicode niz određene duljine, čuva maksimalno 4000 oznaka, a svaka oznaka zauzima dva bajta	N'TipoviPodataka'
VARCHAR	Niz promjenjive duljine, čuva maksimalno 8000 oznaka, a svaka oznaka zauzima jedan bajt	'TipoviPodataka'
NVARCHAR	Unicode niz promjenjive duljine, čuva maksimalno 4000 oznaka, a svaka oznaka zauzima dva bajta	N'TipoviPodataka'
VARCHAR(MAX)	Niz promjenjive duljine, čuva maksimalno 2GB	'TipoviPodataka'
NVARCHAR(MAX)	Unicode niz promjenjive duljine, čuva maksimalno 2GB	N'TipoviPodataka'
INT	Cijeli broj, zauzima 4 bajta	123
SMALLINT	Mali cijeli broj, zauzima 2 bajta	123
DECIMAL	Decimalni broj sa fiksnom točkom	123.45
NUMERIC	Decimalni broj sa fiksnom točkom	123.45
FLOAT	Decimalni broj sa plutajućim zarezom	123.45
REAL	Realni broje, zauzima 4 bajta	123.45
DATE	Datum	'2024-09-07'
TIME	Vrijeme	'15:30:00'
DATETIME	Kombinacija datuma i vremena	'2024-09-07 15:30:00'
BIT	Binarna vrijednost (0 ili1)	1

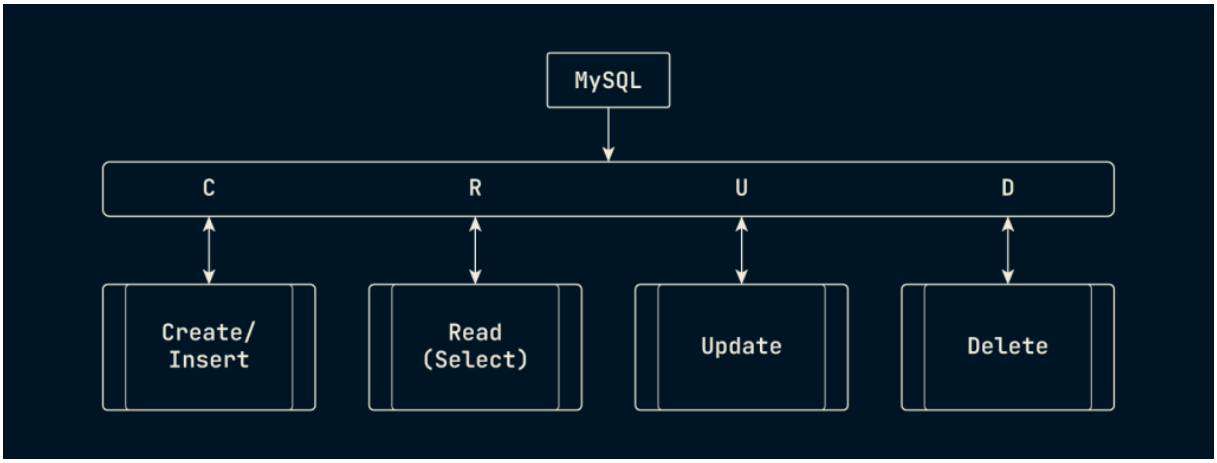
Tablica 1. Tipovi podataka

Kratica CRUD je često korištena u razvoju softvera i aplikacija. Kratica opisuje četiri osnovne operacije koje se mogu izvesti na bazi podataka: stvaranje ili kreiranje (eng. Create), čitanje (eng. Read), ažuriranje (eng. Update) i brisanje (eng. Delete). Programeri ili administratori baze podataka, često imaju zadatak kreirati korisničke uloge koje nose različite CRUD dozvole. Dodjeljivanje i upravljanje CRUD dozvolama vrlo često je zadatak u

programiranju. CRUD opisuje četiri osnovne operacije koje korisnik može izvesti u bazi podataka, [13].

Korisnici mogu:

- stvoriti ili umetnuti nove zapise u tablicu,
- iščitati ili odabrati postojeće zapise iz tablice,
- ažurirati ili izmijeniti postojeće zapise pohranjene unutar tablice,
- brisati ili uklanjati postojeće zapise iz tablice, [13].



Slika 3. CRUD, [13]

3.3. Izrada baze podataka s podacima iz pozivnog centra

U ovom poglavlju opisuje se proces kreiranje baze podataka, zatim umetanje podataka u tablicu i opisuju se upiti kojima se dobivaju određeni rezultati, kako bi pregled i analiza podataka bili pregledniji. Za kreiranje baze podataka koristi se operacija „create“. Na slici 4 se vidi naredba za kreiranje baze podataka pod nazivom „CallCentar“.

```
CREATE DATABASE CallCentar
GO
USE CallCentar
GO
```

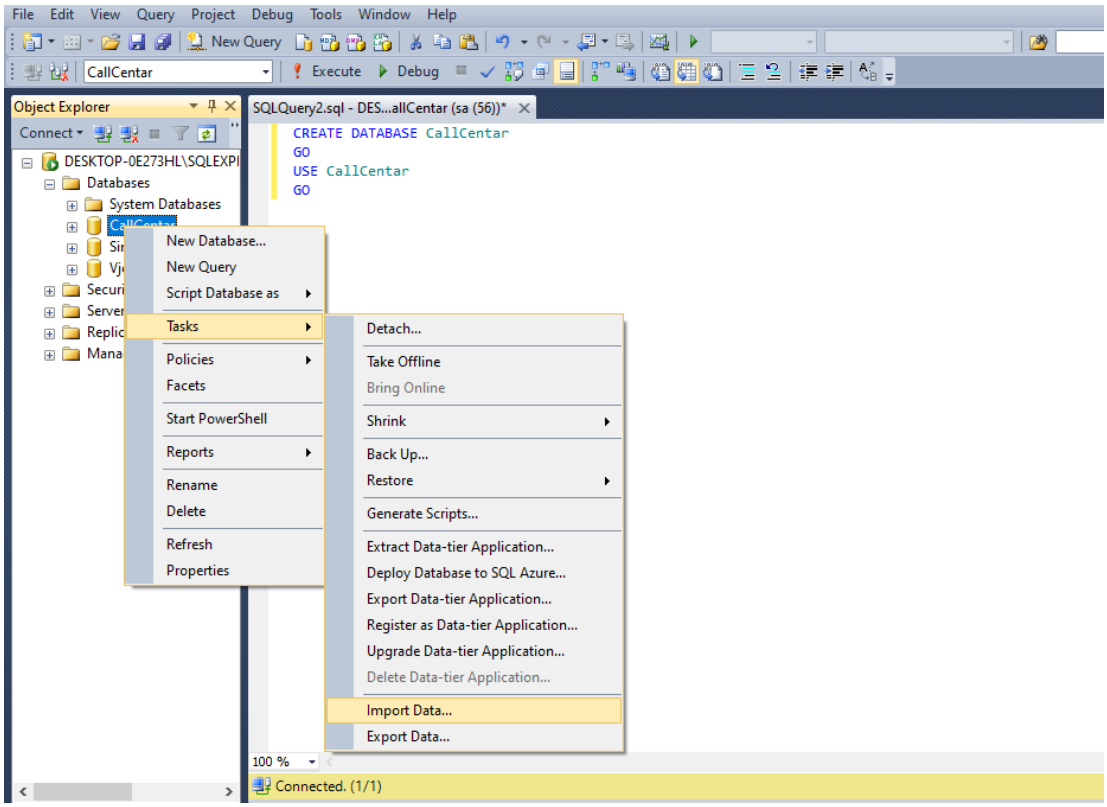
Slika 4. Kreiranje baze podataka

Nakon što je baza podataka kreirana, potrebno je umetnuti tablicu sa podacima iz pozivnog centra. Podaci su spremljeni u datoteci koja je u obliku .csv datoteke. CSV (eng. Comma Separated Values) format najčešći je format za uvoz i izvoz proračunskih tablica i baza podataka. CSV format koristio se mnogo godina prije pokušaja da se format opiše na standardiziran način RFC 4180. Nedostatak dobro definiranog standarda znači da često postoje

suptilne razlike u podacima koje proizvode i koriste različite aplikacije. Te razlike mogu učiniti neugodnim procesiranje CSV datoteka iz više izvora. Ipak, dok se graničnici i znakovi navodnika razlikuju, ukupni je format dovoljno sličan da je moguće napisati jedan modul koji može učinkovito manipulirati takvim podacima, skrivajući detalje čitanja i pisanja podataka od programera, [14].

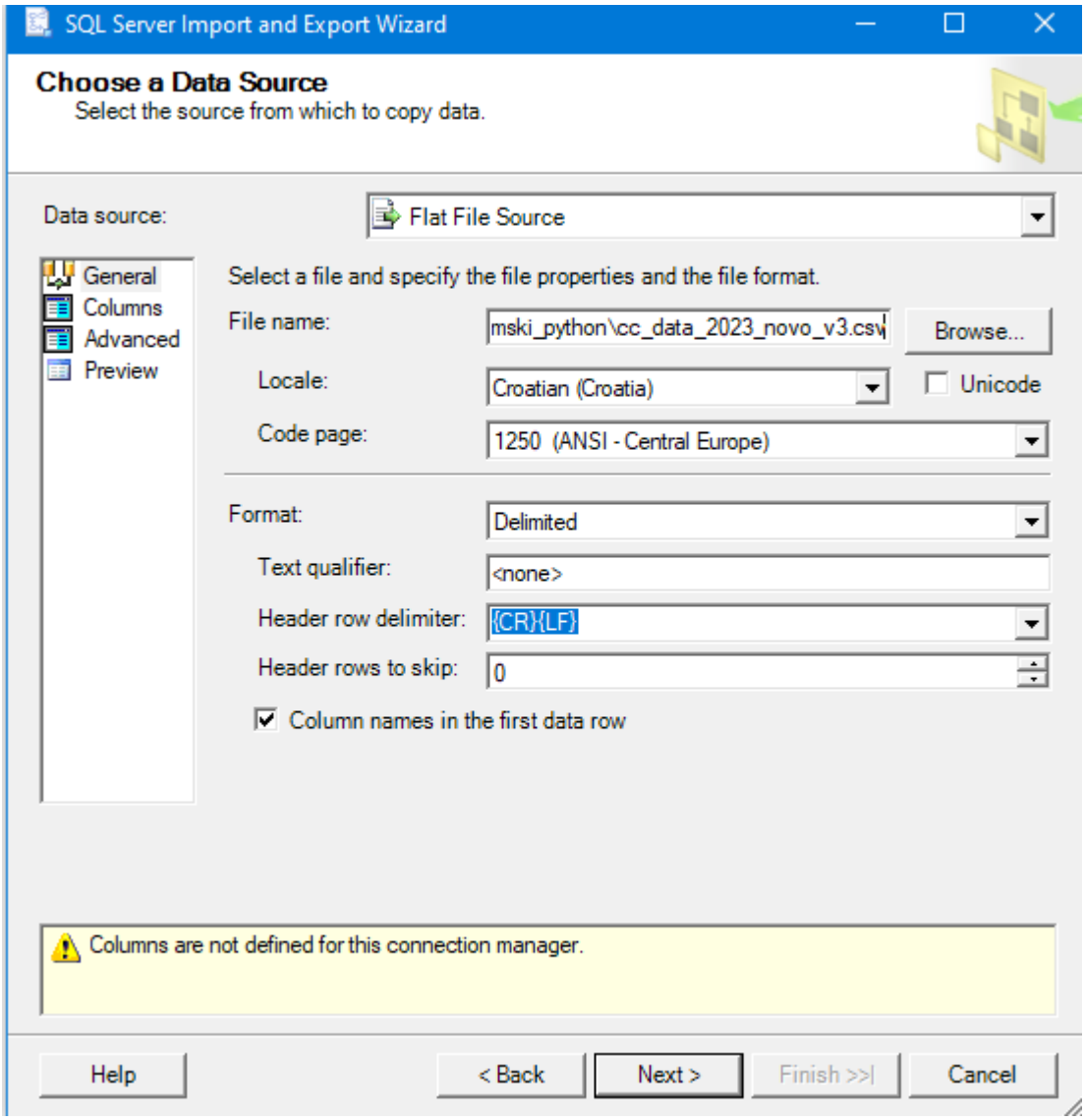
Modul csv implementira klase za čitanje i pisanje tabličnih podataka u CSV formatu. Omogućuje programerima da kažu "zapišite ove podatke u formatu koji preferira Excel" ili "čitajte podatke iz ove datoteke koju je generirao Excel", a da ne znaju točne detalje CSV formata koji koristi Excel. Programeri također mogu opisati CSV formate koje druge aplikacije razumiju ili definirati vlastite CSV formate posebne namjene. Moduli i csv objekti čitaju i pišu nizove. Programeri također mogu čitati i pisati podatke u obliku rječnika koristeći i klase, [14].

Postoji više načina na koji se podaci mogu umetnuti u tablice baze podataka. Ukoliko se radi o manjoj količini podataka koju je potrebno unijeti u tablicu, za to se koristi naredba „insert“. S obzirom da se ovdje radi o dosta većoj količini podataka koristio se drugačiji način od ručnog upisa. Nakon što je baza podataka kreirana, ona se prikaže u izborniku pod nazivom „Object Explorer“ koji služi za pregled i upravljanje objektima u svakoj instanci SQL Servera. Pozicionira se na kreiranu bazu podataka i desnim klikom se otvara padajući izbornik, u kojem se odabire stavka „Tasks“ te se nakon otvara novi padajući izbornik u kojem se odabere „Import Data…“ .



Slika 5. Import data

Nakon odabira „Import Data…“ otvara se novi prozor koji se naziva „SQL Serever Import and Export Wizard“ što na hrvatskom znači čarobnjak za uvoz i izvoz. Taj čarobnjak pomaže stvoriti jednostavan paket za uvoz i izvoz podataka iz baze podataka. Također tako zvani čarobnjak može stvoriti odredišnu bazu podataka i tablicu u koje je potrebno umetnuti podatke. Unutar čarobnjaka potrebno je izabrati izvor podataka iz padajućeg izbornika, u ovom slučaju odabir je bio „Flate File Source“ te odabrati datoteku sa podacima. Na slici 6 je prikaz prethodno napisanog.



Slika 6. Biranje izvora datoteke

Nakon što je datoteka izabrana, u nastavku postoji opcija „Edit Mapings“ u kojoj je moguće odabrati željene tipove podataka unutar određenih stupaca, kako bi podaci u bazi podataka bili uvezeni u ispravnom formatu. Na kraju pritiskom na tipku „Finish“ pokreće se uvoz podataka. Učitana je tablica pod nazivom „cc_data_2023_novo_v3“, stupci koji se nalaze u tablici su „DatumVrijeme“, „EnterpriseName“ (nazivi odjela), „CallsOffered“ (dolazni pozivi) i „CallsHandled“ (odgovoreni pozivi). Tablica je prikazana na slici 7, i njeni stupci. Stupac „DatumVrijeme“ ima tip podataka datetime koji predstavlja format za datum i vrijeme. Stupac „EnterpriseName“ je nvarchar(50) što označuje niz od 50 znakova, „CallsOffered“ i „CallsHandled“ imaju tip podatka int, a to je cijeli broj.

cc_data_2023_novo_v3

	Column Name	Condensed Type	Nullable
	DatumVrijeme	datetime	Yes
	EnterpriseNa...	nvarchar(50)	Yes
	CallsOffered	int	Yes
	CallsHandled	int	Yes

Slika 7. Tablica u SQL-u

Radi provjere, unesenih podataka u tablicu, koristi se naredba „select“ pomoću koje se pretražuje željeni stupac u tablici, ili ukoliko se stavi znak zvjezdice (*) upit izvlači sve podatke koji se nalaze u tablici. Na slici 8 je prikazano kako se izvlače svi podaci iz tablice pod nazivom „cc_data_2023_novo_v3“ u kojoj su uvezeni svi podaci iz pozivnog centra.

```
SELECT * FROM cc_data_2023_novo_v3
```

Slika 8. Prikaz naredbe "select"

Rezultat upita prikazuje nam da su svi podaci dobro učitani jer je isti broj redaka kao i u Excel tablici u kojoj se nalaze prikupljeni podaci. Vidljivi su svi stupci, te je ukupna količina redaka podataka 273665. Na slici 9 su prikazani rezultati upita, radi zaštite podataka na rezultatima nisu vidljivi nazivi odjela.

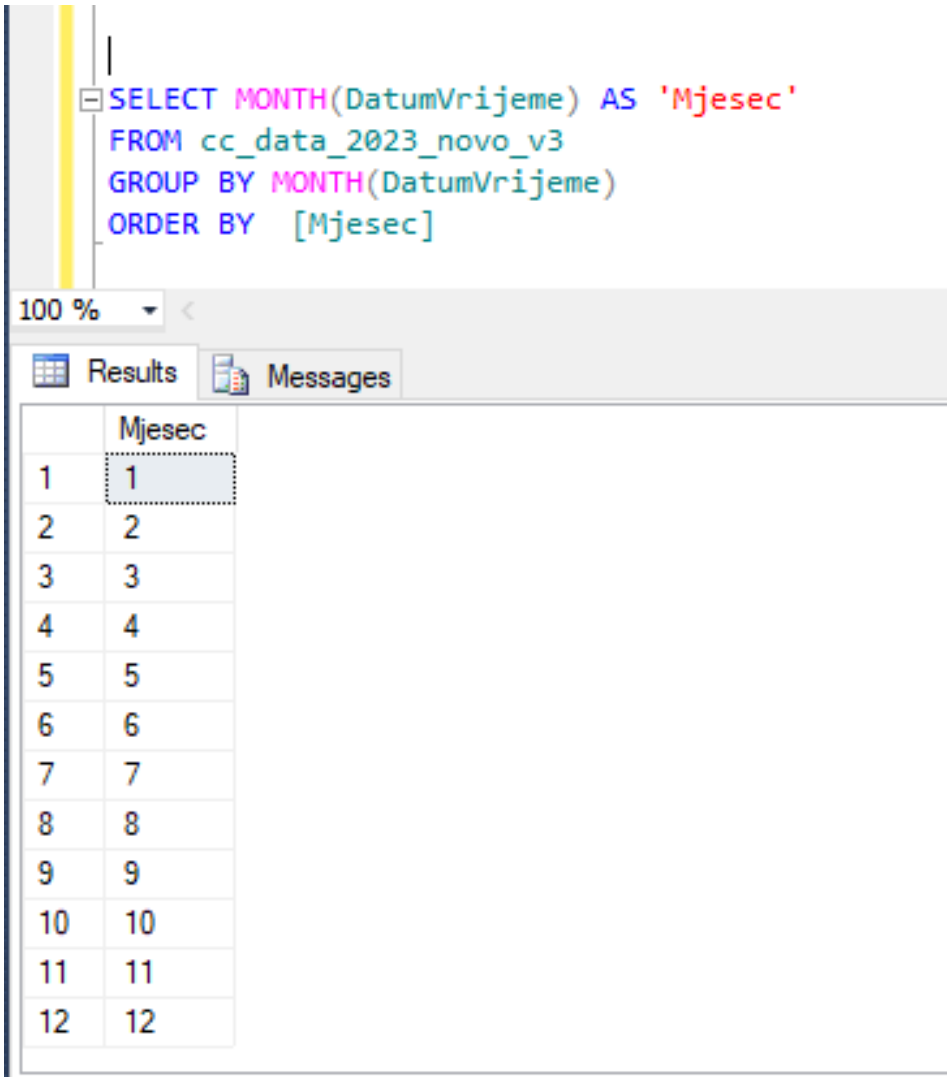
Results | Messages

	DatumVrijeme	CallsOffered	CallsHandled
1	2023-01-01 09:45:00.000	1	0
2	2023-01-01 10:00:00.000	2	3
3	2023-01-01 10:15:00.000	1	0
4	2023-01-01 10:30:00.000	0	1
5	2023-01-01 10:45:00.000	1	1
6	2023-01-01 13:00:00.000	1	0
7	2023-01-01 13:15:00.000	0	1
8	2023-01-01 13:30:00.000	1	1
9	2023-01-01 14:15:00.000	1	0
10	2023-01-01 14:30:00.000	1	2
11	2023-01-01 14:45:00.000	2	2

Query executed successfully. | DESKTOP-0E273HL\SQLEXPRESS ... | sa (52) | CallCentar | 00:00:03 | 273665 rows

Slika 9. Rezultat upita

Dodatno kako bi provjerili ispravnost podataka, bilo je važno da se unutar podataka nalaze zapisi za svaki dan tijekom cijele 2023. godine. Takvu provjeru zapisa je odrađena kroz upite koji su prikazani na slici 10 i 11.



Slika 10. Upit za provjeru mjeseca

Funkcija MONTH vraća cijeli broj koji predstavlja mjesec navedenog datuma, dok funkcija DAY vraća cijeli broj koji predstavlja dan navedenog datuma. GROUP BY u jedan zapis kombinira zapise s identičnim vrijednostima u navedenom popisu polja. ORDER BY naredba vraća raspoređene vrijednosti, ukoliko se ostavi prazno podrazumijeva se funkcija „asc“ koja raspoređuje uzlazno, a ukoliko se stavi „desc“ raspoređuje se silazno. AS naredba dodaje naziv stupca u kojem se nalaze rezultati koji su napisani ispred naredbe AS.

Upitom koji je prikazan na slici 11, odrađena je provjera datuma. Na taj način je vidljivo jesu li podaci ispravni, odnosno postoje li zapisi u tablici za svaki dan u godini. Ono što je vidljivo je da su zapisi bili ispravni, s obzirom da je dohvaćeno 365 različitih redaka.

```
SELECT MONTH(DatumVrijeme) AS 'Mjesec',
DAY(DatumVrijeme) AS 'Dan'
FROM cc_data_2023_novo_v3
GROUP BY DAY(DatumVrijeme) , MONTH(DatumVrijeme)
ORDER BY  [Mjesec], [Dan]
```

100 %

Results | Messages

	Mjesec	Dan
354	12	20
355	12	21
356	12	22
357	12	23
358	12	24
359	12	25
360	12	26
361	12	27
362	12	28
363	12	29
364	12	30
365	12	31

Slika 11. Upiti za provjeru dana

Pogled (view) je objekt u bazi podataka koji predstavlja virtualnu tablicu. Kao i tablica, pogled prikazuje podatke u tabličnom obliku. Pogled se stvara izvršavanjem upita. Podaci koje taj upit vraća su podaci koji će biti dostupni putem pogleda. Stupci pogleda – njihov redoslijed, nazivi i tipovi podataka, kao i retci koje pogled prikazuje, definirani su upitom pomoću kojeg je pogled stvoren. Pogled može prikazivati podatke koji su kombinacija više tablica, podatke koji su dobiveni grupiranjem redaka, odnosno rezultat proizvoljno složenog upita. Pogledi u prvom redu služe da bi se složeni upiti mogli definirati na jednom mjestu te da bi se na taj način podaci iz baze mogli lakše i jednostavnije koristiti. Poslovnu logiku dovoljno je definirati na jednom mjestu – u pogledu, a potom se pogled može pozivati s više mjesta. Na taj je način lakše pisati i razumjeti složene upite, [15].

Za kreiranje pogleda pod nazivom „SumaPoziva“, koji je vidljiv na slici 12 unutar select upita sumiraju se određene vrijednost s agregatnom funkcijom SUM. Na početku select upita nalazi se ključna riječ TOP, jer pogled ne može izvršiti sortiranje podataka bez te ključne riječi.

Pogled „SumaPoziva“ predstavlja sumu na mjesečnoj razini. U rezultatima je vidljivo da je najveći broj dolaznih poziva bio u siječnju čak njih 182140. Najveća količina propuštenih poziva vidljiva je u siječnju, srpnju i kolovozu, što se može pretpostaviti da je u to vrijeme veći broj agenata na godišnjem odmoru te da je nedostajalo radne snage.

```
--Kreiranje pogleda
create view SumaPoziva as
select top 12 MONTH(DatumVrijeme) as 'Mjesec',
sum(callsOffered) as 'BrojDolaznihPoziva',
sum(callsHandled) as 'BrojOdgovorenihPoziva',
(sum(callsOffered)) - (sum(callsHandled)) as 'PropustenBrojPoziva'
from cc_data_2023_novo_v3
group by MONTH(DatumVrijeme)
order by  [Mjesec]


select * from SumaPoziva
```

100 %

Results | Messages

	Mjesec	BrojDolaznihPoziva	BrojOdgovorenihPoziva	PropustenBrojPoziva
1	1	182140	156167	25973
2	2	164273	153089	11184
3	3	171729	160600	11129
4	4	145497	135423	10074
5	5	161113	148639	12474
6	6	154652	141682	12970
7	7	171274	143459	27815
8	8	165634	144855	20779
9	9	139347	130396	8951
10	10	150515	141838	8677
11	11	138388	130660	7728
12	12	133409	124786	8623

Slika 12. Kreiranje pogleda „SumaPoziva“

Za dnevni pregled, kreirani je pogled pod nazivom „SumaPozivaPoDanu“. Select upit zbraja vrijednosti dolaznih poziva i riješenih poziva agregatnom funkcijom SUM. Kako bi se podaci mogli grupirati po danima u mjesecu potrebno je koristiti funkciju DAY. Funkcija CONCAT služi kako bi vrijednosti spojile u jedan stupac, u ovom slučaju radi prikaza datuma u novom stupcu pod nazivom „DanUMjesecu“.

```
--KREIRANJE POGLEDA2
create view SumaPozivaPoDanu as
select  concat(DAY(DatumVrijeme),'/' ,MONTH(DatumVrijeme),'/',YEAR(DatumVrijeme))as 'DanUMjesecu' ,
sum(callsOffered) as 'BrojDolaznihPoziva',
sum(callsHandled) as 'BrojOdgovorenihPoziva',
(sum(callsOffered)) - (sum(callsHandled)) as 'PropustenBrojPoziva'
from cc_data_2023_novo_v3
group by  day(DatumVrijeme),MONTH(DatumVrijeme),YEAR(DatumVrijeme)

select * from SumaPozivaPoDanu
```

Slika 13. Kreiranje pogleda „SumaPozivaPoDanu“

Upitom pogleda „SumaPozivaPoDanu“ dobiva se rezultat u kojem je vidljiva suma, odnosno zbroj dolaznih poziva i zbroj odgovorenih poziva u danu, rezultati vidljivi na slici 14. Ukupna količina podataka to jest redaka je 365 što ujedno i prikazuje da se upit odradio točno jer je toliko dana bilo u 2023. godini. Najveća količina zaprimljenih poziva bila je 01.08.2023. godine, što se može pretpostaviti da je moguće došlo do pada mreže te je veća količina poziva bila upućena prema pozivnom centru.

Results | Messages

	DanUMjesecu	BrojDolaznihPoziva	BrojOdgovorenihPoziva	PropustenBrojPoziva
1	1/1/2023	928	844	84
2	2/1/2023	4501	4264	237
3	3/1/2023	4843	4528	315
4	4/1/2023	5479	5161	318
5	5/1/2023	5431	5129	302
6	6/1/2023	2807	2670	137
7	7/1/2023	2963	2754	209
8	8/1/2023	2207	2033	174
9	9/1/2023	7232	6648	584
10	10/1/2023	7456	6457	999
11	11/1/2023	8454	6558	1896
12	12/1/2023	9004	6005	2999
13	13/1/2023	9166	5717	3449
14	14/1/2023	4182	3251	931
15	15/1/2023	2438	2312	126
16	16/1/2023	9796	4862	4934
17	17/1/2023	8037	7416	621
18	18/1/2023	8225	7618	607
19	19/1/2023	7396	6962	434

Query executed successfully. | DESKTOP-0E273HL\SQLEXPRESS ... | sa (52) | CallCentar | 00:00:00 | 365 rows

Slika 14. Rezultat upita "SumaPozivaPoDanu"

Kako je analiza podataka išla dalje pojavila se potreba za prikazom u postocima. U ovom slučaju koristio se način koje je prikazan na slici 15. Kreiranje novog pogleda pod nazivom „MjesecniPrikaz“ u kojem su se radila dijeljenja kako bi se dobili postoci. Funkcija FORMAT i , 'N2' zaokružuje rješenja na dvije decimale. Na slici 15 su prikazani i rezultati. Najbolji, odnosno najveći postotak odgovorenih poziva bio je u listopadu, što se može pretpostaviti da je tada najveći broj agenata na radnom mjestu. Dosta dobar postotak odgovorenih poziva je bio i u studenom iako je vidljivo da je u studenom bilo dosta manja količina dolaznih poziva. Najveći postotak propuštenih poziva bio je u siječnju, srpnju i

kolovozu kada i najveći broj agenata na godišnjim odmorima, a i bila je veća količina dolaznih poziva.

```
CREATE VIEW MjesecniPrikaz AS
SELECT *, FORMAT(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakPropustenihPoziva' ,
FORMAT(BrojDolaznihPoziva*100.0/1877971,'N2')+'%' AS 'MjesecniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
FROM SumaPoziva


SELECT * FROM MjesecniPrikaz
```

Results | Messages

	Mjesec	BrojDolaznihPoziva	BrojOdgovorenihPoziva	PropustenBrojPoziva	PostotakPropustenihPoziva	MjesecniPostotakDolazPoz	PostotakOdgPoziva
1	1	182140	156167	25973	14.26%	9.70%	85.74%
2	2	164273	153089	11184	6.81%	8.75%	93.19%
3	3	171729	160600	11129	6.48%	9.14%	93.52%
4	4	145497	135423	10074	6.92%	7.75%	93.08%
5	5	161113	148639	12474	7.74%	8.58%	92.26%
6	6	154652	141682	12970	8.39%	8.24%	91.61%
7	7	171274	143459	27815	16.24%	9.12%	83.76%
8	8	165634	144855	20779	12.55%	8.82%	87.45%
9	9	139347	130396	8951	6.42%	7.42%	93.58%
10	10	150515	141838	8677	5.76%	8.01%	94.24%
11	11	138388	130660	7728	5.58%	7.37%	94.42%
12	12	133409	124786	8623	6.46%	7.10%	93.54%

Slika 15. Kreiranje pogleda „MjesecniPrikaz“

Za pregled rezultata u postotcima za dnevni prikaz, koristio se način kreiranja pogleda za svaki mjesec pojedinačno, radi jednostavnijeg pregleda i izgleda. Unutar ovog selecta koristio se pod upit na način da je ponovno upisan select, ali sa zagradama. Na slikama 16.,17.,18. i 19. su prikazani upiti prikaz rezultata po danu.

```
CREATE VIEW PrikazZaSijecanj as
select *, format(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' as 'PostotakPropustenihPoziva',
FORMAT(BrojDolaznihPoziva*100.0/(SELECT SUM(BrojDolaznihPoziva) FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/1/%'),'N2')+'%' AS 'DnevniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
 from SumaPozivaPoDanu
 where DanUMjesecu LIKE '%/1/%'

 CREATE VIEW PrikazZaVeljacu as
select *, format(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' as 'PostotakPropustenihPoziva',
FORMAT(BrojDolaznihPoziva*100.0/(SELECT SUM(BrojDolaznihPoziva) FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/2/%'),'N2')+'%' AS 'DnevniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
 from SumaPozivaPoDanu
 where DanUMjesecu LIKE '%/2/%'

 CREATE VIEW PrikazZaOzujak as
select *, format(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' as 'PostotakPropustenihPoziva',
FORMAT(BrojDolaznihPoziva*100.0/(SELECT SUM(BrojDolaznihPoziva) FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/3/%'),'N2')+'%' AS 'DnevniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
 from SumaPozivaPoDanu
 where DanUMjesecu LIKE '%/3/%'
```

Slika 16. Kreiranje pogleda 1

```
CREATE VIEW PrikazZaTravanj AS
SELECT *, FORMAT(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakPropustenihPoziva',
FORMAT(BrojDolaznihPoziva*100.0/(SELECT SUM(BrojDolaznihPoziva) FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/4/%'),'N2')+'%' AS 'DnevniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/4/%'

SELECT * FROM PrikazZaTravanj

CREATE VIEW PrikazZaSvibanj AS
SELECT *, FORMAT(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakPropustenihPoziva',
FORMAT(BrojDolaznihPoziva*100.0/(SELECT SUM(BrojDolaznihPoziva) FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/5/%'),'N2')+'%' AS 'DnevniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/5/%'

SELECT * FROM PrikazZaSvibanj

CREATE VIEW PrikazZaLipanj AS
SELECT *, FORMAT(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakPropustenihPoziva',
FORMAT(BrojDolaznihPoziva*100.0/(SELECT SUM(BrojDolaznihPoziva) FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/6/%'),'N2')+'%' AS 'DnevniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/6/%'

SELECT * FROM PrikazZaLipanj
```

Slika 17. Kreiranje pogleda 2

```
CREATE VIEW PrikazZaSrpanj AS
SELECT *, format(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakPropustenihPoziva',
FORMAT(BrojDolaznihPoziva*100.0/(SELECT SUM(BrojDolaznihPoziva) FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/7/%'),'N2')+'%' AS 'DnevniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/7/%'

SELECT * FROM PrikazZaSrpanj

CREATE VIEW PrikazZaKolovoz AS
SELECT *, FORMAT(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakPropustenihPoziva',
FORMAT(BrojDolaznihPoziva*100.0/(SELECT SUM(BrojDolaznihPoziva) FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/8/%'),'N2')+'%' AS 'DnevniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/8/%'

SELECT * FROM PrikazZaKolovoz

CREATE VIEW PrikazZaRujan AS
SELECT *, FORMAT(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakPropustenihPoziva',
FORMAT(BrojDolaznihPoziva*100.0/(SELECT SUM(BrojDolaznihPoziva) FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/9/%'),'N2')+'%' AS 'DnevniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/9/%'

SELECT * FROM PrikazZaRujan
```

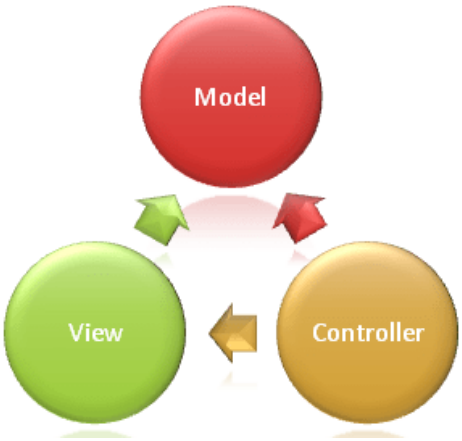
Slika 18. Kreiranje pogleda 3

```
CREATE VIEW PrikazZaListopad AS
SELECT *, FORMAT(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakPropustenihPoziva',
FORMAT(BrojDolaznihPoziva*100.0/(SELECT SUM(BrojDolaznihPoziva) FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/10/%'),'N2')+'%' AS 'DnevniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/10/%'

SELECT * FROM PrikazZaListopad

CREATE VIEW PrikazZaStudeni AS
SELECT *, FORMAT(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakPropustenihPoziva',
FORMAT(BrojDolaznihPoziva*100.0/(SELECT SUM(BrojDolaznihPoziva) FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/11/%'),'N2')+'%' AS 'DnevniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/11/%'

SELECT * FROM PrikazZaStudeni

CREATE VIEW PrikazZaProsinac AS
SELECT *, FORMAT(PropustenBrojPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakPropustenihPoziva',
FORMAT(BrojDolaznihPoziva*100.0/(SELECT SUM(BrojDolaznihPoziva) FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/12/%'),'N2')+'%' AS 'DnevniPostotakDolazPoz',
FORMAT (BrojOdgovorenihPoziva*100.0/BrojDolaznihPoziva,'N2')+'%' AS 'PostotakOdgPoziva'
FROM SumaPozivaPoDanu
WHERE DanUMjesecu LIKE '%/12/%'

SELECT * FROM PrikazZaProsinac
```

Slika 19. Kreiranje pogleda 4

3.4. Spajanje baze podataka i MVC-a

Model-View-Controller (MVC) arhitektura dijeli aplikaciju u tri glavne skupine komponenti: model, pogled i kontroler. Ova arhitektura pomaže u postizanju razdvajanja logike. Korištenjem ove arhitekture, korisnički zahtjevi se usmjeravaju do kontrolera koji je odgovoran za rad s modelom za izvođenje korisničkih radnji i/ili dohvaćanje rezultata upita. Kontroler odabire prikaz koji će se prikazati korisniku i daje mu sve potrebne podatke o modelu. Dijagram koji prikazuje tri glavne komponente i njihove odnose je prikazan na sljedećoj slici, [16]:



Slika 20. Dijagram MVC-a, [28]

Okvir ASP.NET Core MVC je lagani prezentacijski okvir otvorenog koda koji se može vrlo lako testirati te je optimiziran za korištenje s ASP.NET Core. ASP.NET Core MVC pruža način temeljen na obrascima za izradu dinamičkih web stranica te omogućuje čisto odvajanje interesa. Daje potpunu kontrolu nad oznakama, podržava TDD (eng. Test Driven Development) razvoj i koristi najnovije web standarde, [16].

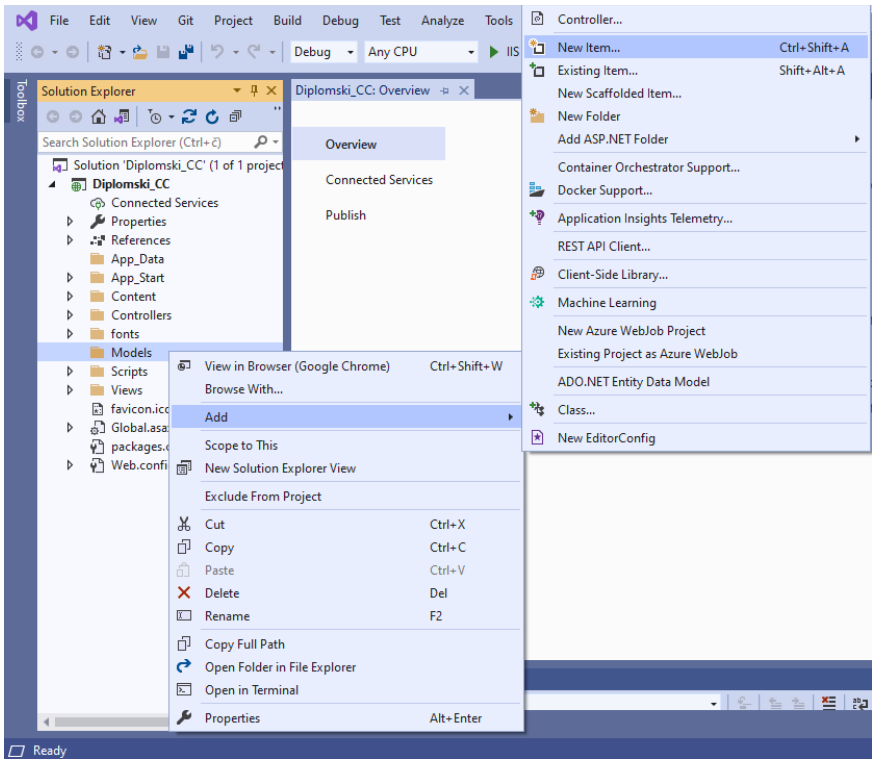
U MVC okruženju, pogled upravlja prezentacijom podataka aplikacije i interakcijom korisnika. Pogled je HTML predložak s ugrađenim oznakama Razor. Razor markup je kod koji je u interakciji s HTML markupom za izradu web stranice koja se šalje klijentu. U ASP.NET Core MVC, nalaze se .cshtml datoteke koje koriste C# programski jezik u Razor označavanju. Obično su datoteke prikaza grupirane u direktorije s nazivom za svaki od kontrolera aplikacije, [16].

Kontroler se koristi za definiranje i grupiranje skupa akcija. Akcija je metoda na kontroleru koja obrađuje zahtjeve. Kontrolori logički grupiraju slične radnje. Ova agregacija radnji omogućuje zajedničku primjenu zajedničkih skupova pravila, kao što su usmjeravanje,

korištenje predmemorije i autorizacija. Zahtjevi se preslikavaju na radnje putem usmjeravanja. Kontroleri se aktiviraju i odlažu na temelju zahtjeva.

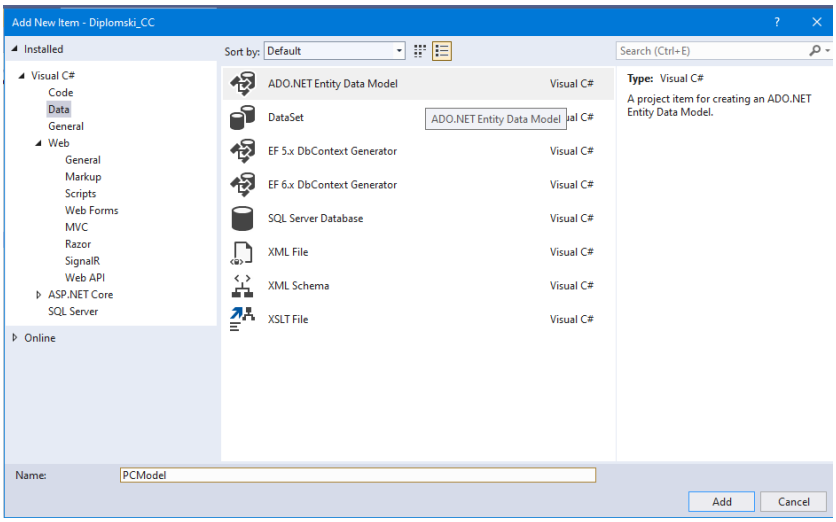
Model u MVC aplikaciji predstavlja stanje aplikacije i bilo koju poslovnu logiku ili operacije koje ona treba izvesti. Poslovna logika treba biti sadržana u modelu, zajedno s bilo kojom logikom implementacije za održavanje stanja aplikacije. Strogo tipizirani prikazi obično koriste tipove ViewModel dizajnirane da sadrže podatke za prikaz. Kontroler stvara i popunjava te instance ViewModela iz modela, [16].

Za spajanje podataka iz baze podataka, koja se nalazi u programu SQL, sa Visual Studiom unutar MVC okruženja odabere se komponenta model.



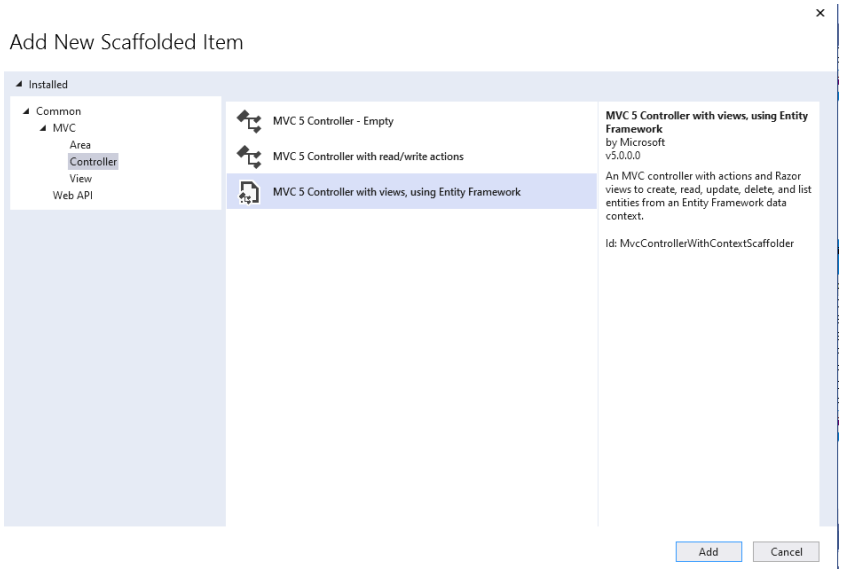
Slika 21. Povezivanje s bazom

ADO.NET Entity Data Model je skup biblioteka koji opisuju strukturu podataka, bez obzira na njihov pohranjeni oblik. Dakle on podatke koji su spremljeni u bazu podataka transformira u klase koje prebacuje u web-aplikaciju. Entity Data Model rješava izazove opisujući strukturu podataka u smislu entiteta i odnosa koji su neovisni o bilo kojoj shemi pohrane. To čini pohranjeni oblik podataka irelevantnim za dizajn i razvoj aplikacije. Budući da entiteti i odnosi opisuju strukturu podataka kako se koriste u aplikaciji (a ne njihov pohranjeni oblik), oni se mogu razvijati kako se razvija aplikacija, [17].



Slika 22. ADO.NET Entity Data Model

Potom se odabire *EF Designer from database* koji stvara model u *EF Designeru* na temelju postojeće baze podataka. Može se odabrati veza s bazom podataka, postavke za model i objekte baze podataka koje će se uključiti u model. Klase s kojima će aplikacija komunicirati generiraju se iz modela. U sljedećem koraku odabire se server SQL za spajanje s bazom pomoću server imena u SQL programu te se odabire željena autentifikacija, u ovom slučaju se koristila SQL Server Authentication sa potrebnim podacima za prijavu, i naziv baze podataka s kojom se spaja u ovom slučaju naziv baze je „*CallCentar*“. Kada je baza spojena potrebno je dodati kontrolere kako bi prikazali podatke u web-aplikaciji, prikazano na slici 23.



Slika 23. Dodavanje kontrolera

MVC 5 Contoller with views, using Entity Framework je opcija koja će generirati kontroler i prikaze za ažuriranje, brisanje, stvaranje i prikaz podataka u modelu. Za svaku tablicu koja je prenijeta iz baze podataka, u ovom slučaju poglede, potrebno je kreirati

kontrolere. Kontroler ispisuje metode i kreira poglede. Za ovaj prikaz podataka koji je vidljiv na slici 26 prikaz je odrađen preko pogleda pod nazivom „*Index*“. Unutar „*Index*“ pogleda ispisan je kod koji je napisan koristeći Razor sintaksu. Ovaj kod prikazuje tabularni prikaz podataka za svaki dan u mjesecu, sa statistikama poziva (dolaznih, odgovorenih, propuštenih, itd.). Podaci u tablici su prikazani dinamički na osnovu modela koji je proslijeđen iz kontrolera u ovaj prikaz. U nastavku detaljnije objašnjeno što svaki dio koda koji na prikazan na slikama 24 i 25, radi:

1. @{ ViewBag.Title = "Index"; }:
    - ViewBag omogućava prenošenje podataka između kontrolera i pogleda i dinamički je objekt. Atribut naslov (eng. Title) se obično koristi u HTML <title> tagu ili kao naslov pogleda.
2. <h2>Prikaz za sijecanj</h2>:
    - ovo je statički HTML element koji prikazuje naslov u okviru stranice. U ovom slučaju, to je naslov druge veličine (H2) sa tekstom "Prikaz za sijecanj".
3. @Html.ActionLink("Create New", "Create") :
    - kreira hyperlink koristeći Html.ActionLink pomoćnika. Ovaj link vodi na akciju Create unutar trenutnog kontrolera i ima tekst "Create New". Link se nalazi unutar paragraf elementa .
4. <table class="table">:
    - započinje HTML tablicu sa klasom "table", što sugerira da se koristi CSS klasa za stiliziranje tablice.
5. Stupci tablice (<th> elementi):
    - stupci koriste @Html.DisplayNameFor(...) kako bi dinamički prikazale nazive stupaca na osnovu modela. model => model.X označava polje iz modela za koje se prikazuje ime. Ovo omogućava generiranje naziva stupaca koji su povezani sa nazivima atributa u modelu.
6. @foreach (var item in Model):
    - petlja *foreach* prolazi kroz sve stavke u modelu i za svaku stavku kreira redak u tablici.

7. Redovi u tablici (<tr> elementi):

    - za svaki objekt u modelu (item), kreiraju se redovi u tablici. Unutar svakog reda, @Html.DisplayFor(...) se koristi za prikaz vrijednosti polja tog objekta, npr. item.DanUMjesecu, item.BrojDolaznihPoziva itd.

8. Akcijski linkovi u posljednjem stupcu:

    - posljednja kolona u svakom redu sadrži tri linka (Edit, Details, Delete). Ovi linkovi koriste @Html.ActionLink da bi se stvorili URL-ovi koji pozivaju odgovarajuće akcije (Edit, Details, Delete) na kontroleru, koristeći item.ID kao parametar.

9. Zatvaranje tablice (</table> element):

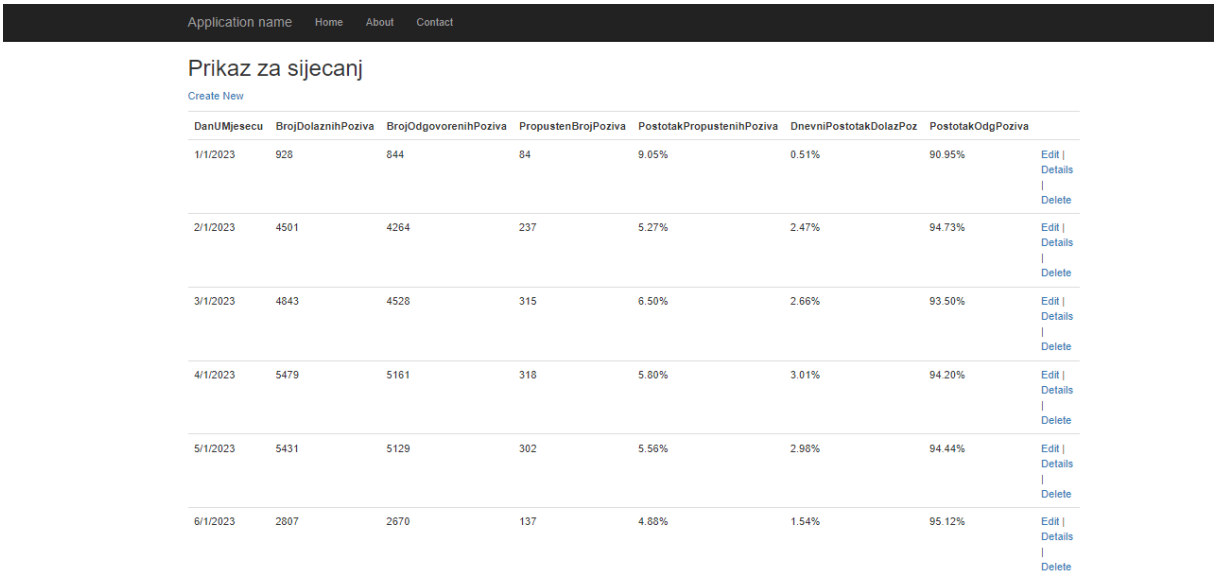
    - zatvara HTML tablicu.

```
<h2>Prikaz za sijecanj</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.DanUMjesecu)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.BrojDolaznihPoziva)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.BrojOdgovorenihPoziva)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PropustenBrojPoziva)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PostotakPropustenihPoziva)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DnevniPostotakDolazPoz)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PostotakOdgPoziva)
        </th>
        <th></th>
    </tr>
```

Slika 24. Prikaz Index koda

```
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.DanUMjesecu)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.BrojDolaznihPoziva)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.BrojOdgovorenihPoziva)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.PropustenBrojPoziva)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.PostotakPropustenihPoziva)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.DnevniPostotakDolazPoz)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.PostotakOdgPoziva)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.ID }) |
            @Html.ActionLink("Details", "Details", new { id=item.ID }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.ID })
        </td>
    </tr>
}
```

Slika 25. Prikaz Index koda



Slika 26. Prikaz podataka kroz web-aplikaciju

4. Primjena strojnog učenja u svrhu predikcije budućeg opterećenja pozivnog centra

4.1. Strojno učenje

Strojno učenje sustavima omogućuje djelovanje na inteligentan način, odnosno daje im mogućnost učenja i unaprjeđenja na temelju prethodnog iskustva bez promjene u arhitekturi i prilagodbe programskog koda, [18]. Strojno učenje se danas koristi u većini domena, kao na primjer financijama, komunikacijama, prijevozu, sigurnosti, znanosti, bankarstvu i osiguranju, [19]. Strojno učenje (eng. machine learning) obuhvaća tehnike u kojima računalo uči rješavati specifične i usko usmjerene zadatke iz podataka – kaže se da se odluke o strojnom učenju temelje na podacima. Tehnike strojnog učenja uključene su u krovno područje umjetne inteligencije (eng. artificial intelligence), koje, međutim, pokriva mnogo šire područje istraživanja. Područje vezano uz strojno učenje je takozvano rudarenje podataka (eng. data mining), gdje uz pomoć različitih tehnika, uključujući i strojno učenje, obrađuju se i proučavaju podaci te se pokušava iz njih razlučiti uzorke i, kao rezultat, nova znanja. Izraz rudarenje podataka koristi se prilikom susreta s problemom korištenja samih metoda strojnog učenja kao alata za rješavanje drugih problema, umjesto da ih se samo implementira, [20].

Strojno učenje može rješavati brojne probleme, kao na primjer: klasifikaciju dokumenata, obrada govora, prepoznavanje govora, obrada prirodnog jezika, identifikacija govornika, optičko prepoznavanje znakova, otkrivanje lica, prepoznavanje i identifikacija objekta, upadi u mrežu, otkrivanje kartičnih prevara, sustav preporuka, medicinska dijagnostika, analiza mreža gena i proteina, predviđanje funkcija proteina i drugi. Zbog velikog raspona problema koje može riješiti strojno učenje, njegova primjena je gotovo u svim područjima, a ponajviše se primjenjuje u područjima bankarstva, financija, prodaje, farmacije, naftne kompanije i transporta, [21].

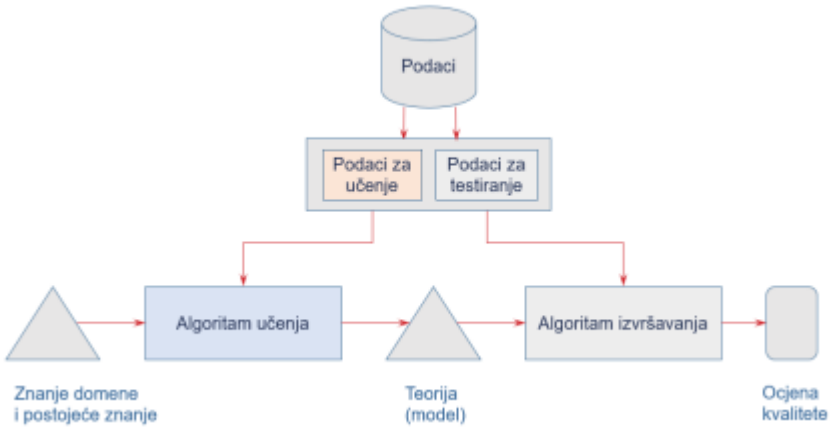
Nakon što se problem definira, moraju se prikupiti podaci za analizu. Prije početka prikupljanja podataka, potrebno je, ukoliko postoji mogućnost, odrediti kakvi podaci trebaju biti kako bi predstavljali reprezentativan skup koji uključuje sve moguće klase rješenja. Kako bi spriječili da samo prikupljanje podatka bude preskupo i preopširno, važno je odrediti značajke koje doprinose rješenju. Poželjno je da skup podataka bude što više opširan, no to često i nije moguće. Podatke je zatim potrebno raspodijeliti na skupove za treniranje i testiranje u određenom omjeru. Ovisno o odabranom algoritmu, potrebno je podatke dodatno pripremiti

npr.: normalizacija numeričkih grešaka, kodiranje kategorijskih varijabli, i drugo. Prilikom odabira željenog algoritma strojnog učenja potrebno je obratiti pažnju na sljedeće, [22]:

- klasifikacija podataka – ukoliko podaci nisu klasificirani, odabire se algoritam nenadziranog učenja, dok se u suprotnom odabire algoritam nadziranog učenja,
- točnost i mogućnost interpretacije rezultata – ukoliko je poželjna mogućnost jednostavne interpretacije rezultata, preporuča se linearna regresija, dok veća točnost rezultata za sobom povlači fleksibilnost modela, što znači da će rezultate biti teže interpretirati (pr. putem dijagrama),
- veličinu skupa podataka za treniranje modela – ukoliko skup podataka ima mali broj opažanja i velik broj značajki, preporuča se odabrati linearnu regresiju ili Bayesovu mrežu,
- brzinu ili vrijeme treniranja – veća točnost modela za sobom povlači dulje vrijeme treniranja, kao i veća količina podataka. Linearna regresija i Bayesove mreže jednostavni su za implementaciju i brzi, dok neuronske mreže i nasumične šume zahtijevaju puno vremena za treniranje,
- linearnost – neki od algoritama rade temeljem pretpostavke da se klase podataka mogu odvojiti ravnom linijom – ukoliko su podaci linearni, takvi algoritmi rade prilično dobro, dok je u suprotnom potrebno odabrati nasumičnu šumu ili neuronsku mrežu, [22].

Nakon što se odabere algoritam, definiraju se potrebni parametri te slijedi izrada modela. Potom se model trenira sa skupom podataka za treniranje čime model uči na kako bi prilikom testiranja mogao dati željene izlazne vrijednosti. Nakon što se izvrši treniranje, model se testira te se provjera točnost predviđanja na podacima koje služe za test. Ukoliko je potrebno evaluacijom se provode promjene nad modelom te se model ponovno trenira. Kada točnost predviđanja bude zadovoljavajuća, model se koristi za predikciju.

Proces učenja samog modela sastoji se od: prikupljene podatke podijeli se u dva skupa, jedan služi za učenje, odnosno treniranje, a drugi za testiranje, potom se kreira algoritam učenja koji korištenjem skupa podataka za učenje omogućava da model uči i kreira teoriju o podacima, nakon toga se korištenjem algoritma izvršavanja vrši testiranje naučene teorije modela i ocjenjuje se kvaliteta naučenog. Na slici 27 je prikazan proces učenja.



Slika 27. Proces učenja, [22]

S obzirom da strojno učenje ima široki raspon primjene, postoji više metoda učenja u kojima se koriste poznati algoritmi strojnog učenja. Pregled metoda i algoritama je u nastavku.

4.2. Metode strojnog učenja

Zbog svoje popularnosti primjene, strojno učenje konstanto razvija nove metode i algoritme kako bi se problemi rješavali što efikasnije i brže te kako bi se suprotstavili raznim problemima koji su vezani uz podatke (manji skup podataka, nepotpuni skupovi, veliki skupovi) nad kojima algoritmi uče. Postoje četiri najčešće prihvaćene metode strojnog učenja, a to su nadzirano učenje (eng. supervised learning), nenadzirano učenje (eng. unsupervised learning), polunadzirano učenje (eng. semisupervised learning) i učenje nagrađivanjem (eng. reinforcement learning).

Algoritmi nadziranog učenja predviđaju na temelju skupa primjera. Na primjer, povijesne prodaje mogu se koristiti za procjenu budućih cijena. U nadziranom učenju postoji ulazna varijabla koja se sastoji od označenih podataka o treningu i željene izlazne varijable. Koristi se algoritam za analizu podataka o učenju kako bi se naučila funkcija koja preslikava ulaz u izlaz. Ova pretpostavljena funkcija preslikava nove, nepoznate primjere generalizacijom iz podataka o obuci kako bi se predvidjeli rezultati u neviđenim situacijama, [23].

- Klasifikacija: kada se podaci koriste za predviđanje kategoričke varijable, nadzirano učenje se također naziva klasifikacija. Ovo je slučaj kada se slici dodjeljuje oznaka ili indikator, bilo psa ili mačke. Kada postoje samo dvije oznake, to se naziva binarna klasifikacija. Kada postoji više od dvije kategorije, problemi se nazivaju višeklasna klasifikacija.

- Regresija: koristi se kada se predviđaju kontinuirane vrijednosti, primjerice kada se predviđaju cijene stanova ili vrijeme potrebno da se izvrši neki proces u proizvodnji.
- Predviđanje: ovo je proces predviđanja budućnosti na temelju prošlih i sadašnjih podataka. Najčešće se koristi za analizu trendova. Uobičajen primjer može biti procjena prodaje za sljedeću godinu na temelju prodaje tekuće godine i prethodnih godina, [23].

Izvođenje učenja bez nadzora, uključuje stvaranje modela koji može izvući uzorke iz neoznačenih podataka. Od njega se traži da otkrije intrinzične obrasce koji su u podlozi podataka, kao što je struktura grupiranja, nisko-dimenzionalna mnogostrukost ili rijetko stablo i graf.

- Grupiranje: grupiranje skupa primjera podataka tako da su primjeri u jednoj skupini (ili jednom klasteru) sličniji (prema nekim kriterijima) od onih u drugim skupinama. To se često koristi za segmentiranje cijelog skupa podataka u nekoliko grupa. Analiza se može provesti u svakoj skupini kako bi se korisnicima pomoglo pronaći unutarnje obrasce.
- Smanjenje dimenzija: smanjenje broja varijabli koje se razmatraju. U mnogim primjenama, neobrađeni podaci imaju vrlo velike značajke, a neke su značajke suvišne ili irelevantne za zadatak. Smanjenje dimenzionalnosti pomaže pronaći pravi, latentni odnos, [23].

Izazov s nadziranim učenjem je taj što označavanje podataka može biti skupo i dugotrajno. Ako su oznake ograničene, mogu se koristiti neoznačeni primjeri kako bi se poboljšalo nadzirano učenje. Budući da stroj u ovom slučaju nije potpuno nadziran, kaže se da je stroj polunadziran. S polu-nadziranim učenjem koriste se neoznačeni primjeri s malom količinom označenih podataka kako bi se poboljšala točnost učenja.

Učenje s nagrađivanjem još je jedna grana strojnog učenja koja se uglavnom koristi za probleme sekvencijalnog donošenja odluka. U ovoj vrsti strojnog učenja, za razliku od nadziranog i nenadziranog učenja, ne moraju se imati nikakvi podaci unaprijed; umjesto toga, agent za učenje stupa u interakciju s okolinom i u hodu uči optimalnu politiku na temelju povratnih informacija koje prima iz te okoline. Konkretno, u svakom vremenskom koraku agent promatra stanje okoline, odabire radnju i promatra povratnu informaciju koju dobiva od

okoline. Povratna informacija o djelovanju agenta ima mnogo važnih komponenti. Jedna komponenta je rezultirajuće stanje okoline nakon što je agent djelovao na nju, [23].

Druga komponenta je nagrada (ili kazna) koju agent dobiva izvođenjem te određene radnje u tom određenom stanju. Nagrada je pažljivo odabrana kako bi bila usklađena s ciljem za koji se obučava agenta. Koristeći stanje i nagradu, agent ažurira svoju politiku donošenja odluka kako bi optimizirao svoju dugoročnu nagradu. S nedavnim napretkom dubokog učenja, učenje s potkrepljenjem privuklo je značajnu pozornost jer je pokazalo zapanjujuće performanse u širokom rasponu aplikacija kao što su igre, robotika i upravljanje procesima, [23].

4.3. Algoritmi strojnog učenja

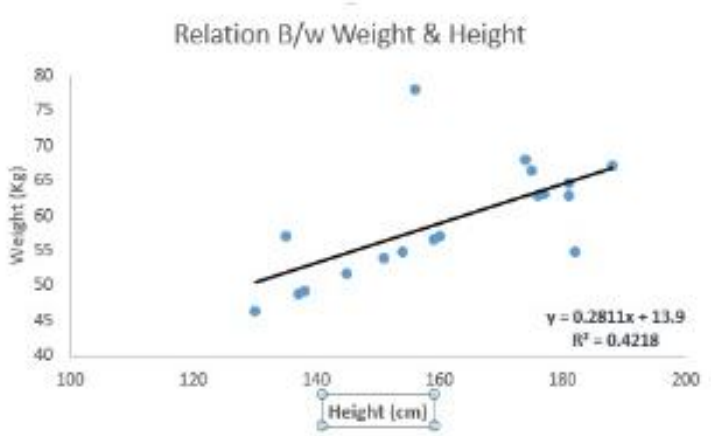
Algoritam strojnog učenja radi treniranje modela, odnosno radi optimizaciju parametara. Oni primjenjuju računalne metode koje „uče“ informacije izravno iz podataka bez pomoći teorijske jednadžbe i modele. Što je veći broj uzoraka koji su dostupni za učenje, algoritam postaje bolji odnosno daje bolje rezultate, [24].

Algoritmi se razlikuju po načinu po kojem uče i po tome što se koriste za rješavanje različitih problema. Najčešće se koriste algoritmi: stablo odlučivanja (eng. Decision Tree), nasumična šuma (eng. Random Forrest), linearna regresija (eng. Linear Regression), logistička regresija (eng. Logistic Regression), Bayesova mreža (eng. Naive Bayes), k-najbliži susjedi (eng. k-Nearest Neighbors, kNN), K-sredine (eng. K-Means), neuronske mreže (eng. Neural Network, NN), potporni vektori (eng. Support Vector Machine, SVM), algoritam smanjenja dimenzionalnosti (eng. Dimensionality Reduction Algorithms) i algoritmi pojačavanja gradijenta (eng. Gradient Boosting algorithms), [25]. U nastavku su opisani pojedini algoritmi.

4.3.1. Linearna regresija

Linearna regresija prikazuje odnos između ulaza x i izlaza y, svaka promjena vrijednosti x, y će se proporcionalno promijeniti. Jednostavna linearna regresija je ukoliko se radi sa samo jednom ulaznom varijablom, a višestruka linearna regresija ukoliko se radi sa više ulaznih varijabli. Ona se koristi za procjenu stvarnih vrijednosti (broj poziva, ukupna prodaja) temeljem kontinuiranih varijabli. Linearna regresija se prikazuje na grafu, a predstavljena je linearnom jednadžbom: $\hat{y} = a \cdot x + b$. Cilj linearne regresije je kreirati liniju koja opisuje odnos zavisne i nezavisne varijable.

Primjer grafa linearne regresije koja odgovara linearnoj jednadžbi y=0,2811x+13,9 je prikazan na slici 28, [25].



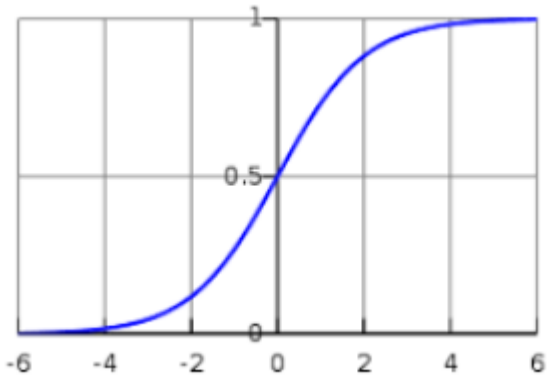
Slika 28. Primjer linearne regresije, [25]

4.3.2. Logistička regresija

Logistička regresija pretvara ulazne podatke u jednu od dvije kategorije. Djeluje kao binomni klasifikator. Logističku regresiju se može zamisliti kao prekidač za uključivanje i isključivanje. Može stajati samostalno ili se neka njegova inačica može koristiti kao matematička komponenta za formiranje sklopki ili vrata koja prenose ili blokiraju protok informacija. Koristi se za procjenu diskretnih vrijednosti, poput binarnih vrijednosti (0 ili 1), da ili ne, točno ili netočno, istina ili neistina. Kao i svaki prekidač, logistička regresija može biti komponenta u većem krugu, [26]. Ovaj algoritam često se koristi za klasteriranje pri nenadziranom učenju, a često se koristi i kao funkcija aktivacije u posljednjem koraku algoritma, [26,27].

Na slici 29 je prikazan graf logističke funkcije gdje se vidi da ima oblik S ili sigmoid spljošten na vrhu i dnu, dok brzo prelazi između dva stanja prije ulaska u jedan od dugih, asimptotskih repova. To znači da se unos može nakupljati dugo dok ga funkcija još uvijek tumači kao "isključen", ali dodavanjem postupnog više signala na pravom mjestu, funkcija se okreće na "uključeno" i ostaje "uključeno" zauvijek. Kao funkcija, logistička regresija jednostavno je krivulja u obliku slova S koja može unijeti bilo koji realni broj i prevesti ga u vrijednost između 0 i 1. Na gornjem grafikonu uzimaju se kontinuirane vrijednosti između -6 i 6 i mapiraju se na vrijednosti između 0 i 1. Ovdje je formula koja izvodi to preslikavanje, [26].

$$1/(1 * e^{-z})$$



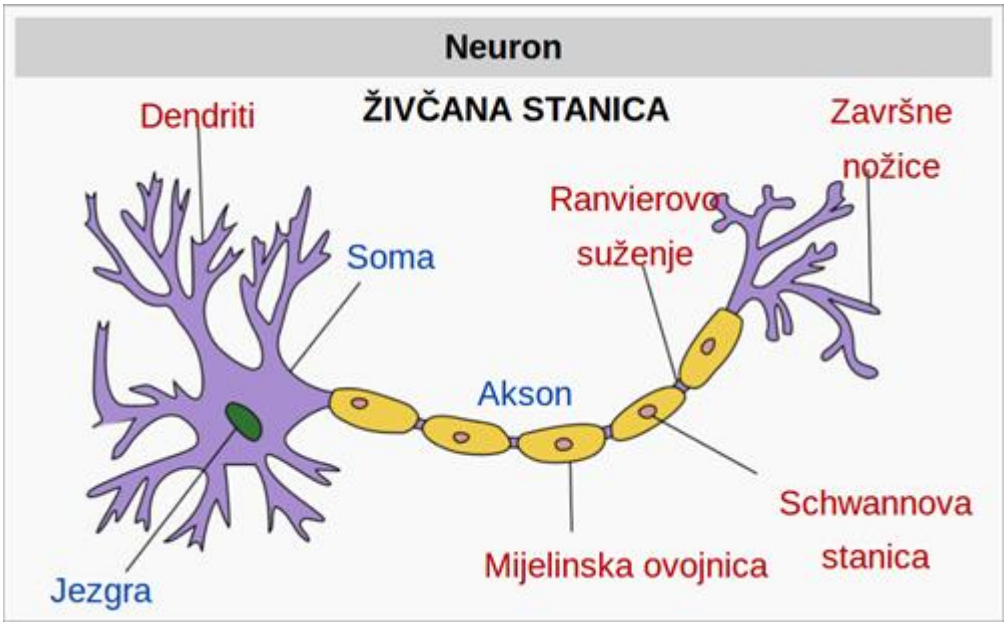
Slika 29. Graf logističke funkcije, [26]

4.3.3. Neuronska mreža

Neuronske mreže su skup algoritama, oblikovanih prema ljudskom mozgu, koji su dizajnirani za prepoznavanje uzoraka. Oni tumače senzorne podatke kroz neku vrstu strojne percepcije, označavanja ili klasteriranja sirovog unosa. Uzorci koje prepoznaju su numerički, sadržani u vektorima, u koje se moraju prevesti svi podaci iz stvarnog svijeta, bilo da se radi o slikama, zvuku, tekstu ili vremenskoj seriji. Neuronske mreže pomažu grupirati i klasificirati podatke. Mogu se zamisliti kao sloj koji grupira i kategorizira podatke koji se pohranjuju i kojima se upravlja. Omogućuju grupiranje neoznačenih podataka na temelju sličnosti među unosima te klasificiraju podatke kada postoji označeni skup podataka za treniranje. Neuronske mreže, također, mogu izdvojiti značajke koje se šalju drugim algoritmima za klasteriranje i klasifikaciju; tako da se duboke neuronske mreže mogu zamisliti kao komponente većih aplikacija strojnog učenja koje uključuju algoritme za učenje s nagrađivanjem, klasifikaciju i regresiju, [28].

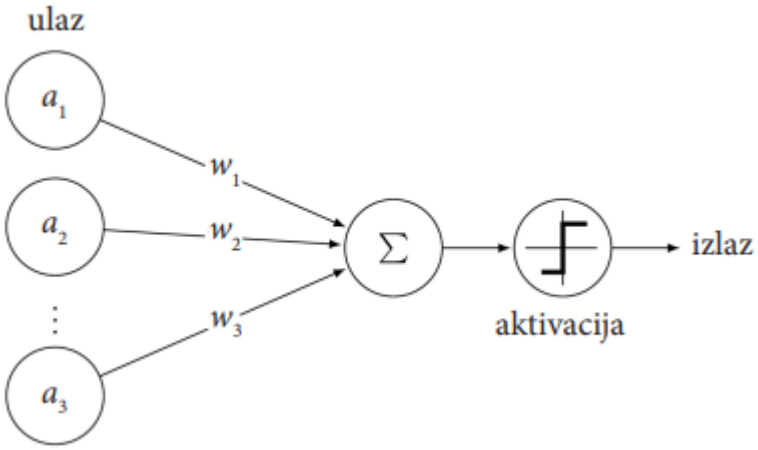
Signal, u biološkom neuronu, prenosi se putem sinaptičkih veza na dendrite te doprinosi smanjenju ili povećanju njegove aproksimacije. Nakon što dostigne određenu razinu aproksimacije, tada neuron odašilje izlazni signal koji se putem aksona prenosi na druge neurone. Budući da se moždana kora sastoji od prosječno 10^{11} neurona, dok se sinaptičke veze između dendrita uspostavljaju se elektrokemijskim putem i ostvarene su između 10^3 – 10^4 susjednih neurona, može se zaključiti da mozak predstavlja složen paralelni sustav obrade informacija. Vrijeme potrebno za prijenos signala sinaptičkom vezom iznosi nekoliko milisekundi. Prepoznavanje lica druge osobe traje sekundu što čini proces u moždanoj kori od nekoliko stotina paralelnih koraka. Sinaptičkom vezom se prenosi mala količina informacija

koja iznosi nekoliko bita, te se može zaključiti da je takav paralelni sustav iznimno učinkovit i optimiziran evolucijskim razvojem, [29]. Na slici 30 je prikaz biološkog neurona.



Slika 30. Biološki neuron, [29]

Model biološkog neurona koji je prikazan na slici 31, može se zapisati jednostavno pomoću skalarnog umnoška vektora (a) ulaznih aktivacija i vektora (w) intenziteta sinaptičkih veza odnosno težine, i sa realnom funkcijom σ koja uz dodatno normiranje izlaznog signala, imitira aktivaciju neurona. U primjeni neuronskih mreža funkcije za aktivaciju odabiru se prema definiciji problema za koji se priprema neuronska mreža te se najčešće primjenjuju: po dijelovima linearna, po dijelovima konstantna, logistička (sigmoidna, hiperbolni tangens) itd. Na vrijednost skalarnog umnoška dodaje se posmak ***b*** (eng. bias), njegova svrha je poboljšanje performansi neuronske mreže. Model jednog neurona može se zapisati pomoću vrijednosti funkcije, [29].



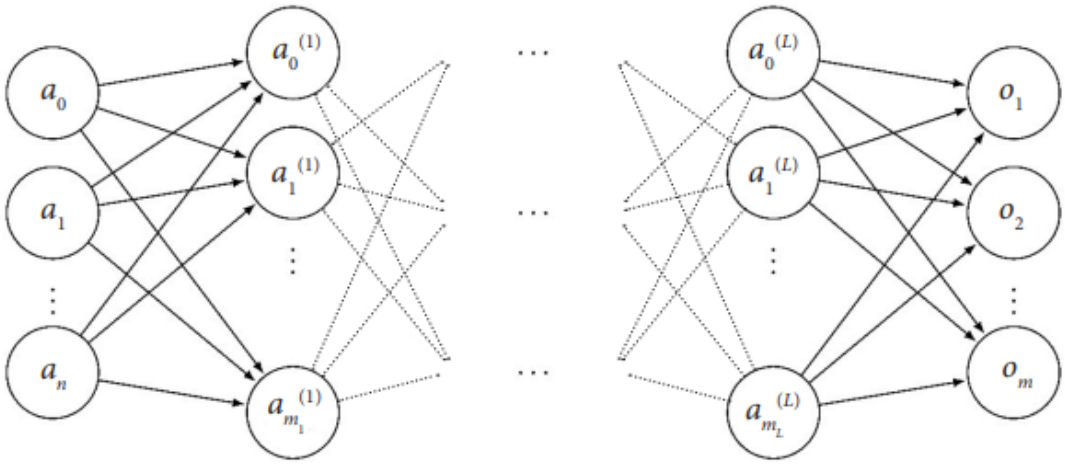
Slika 31. Model biološkog neurona, [29]

$$w^T \cdot a = \sum_{i=1}^{n} w_i \, a_i \tag{1}$$

$$\sigma(w^T \cdot a + b) \tag{2}$$

U modelu jednog neurona može se primijetiti kako je nelinearnost određena aktivacijskom funkcijom, a isti takav model može se dobiti ukoliko se za aktivacijsku funkciju odabere po dijelovima konstantna ili linearna. Složeni sustav može se graditi prijenosom vrijednosti aktivacijske funkcije na druge umjetne neurone. To znači da se mreža umjetnih neurona formira množenjem matrice težina ($\boldsymbol{W}$) s vektorom vrijednosti aktivacijske funkcije za vektor ($\boldsymbol{a}$) ulaznih neurona. Slikovito takav model predstavlja paralelni sustav koji simulira aktivaciju sloja bioloških neurona u višeslojnoj neuronskoj mreži. Pomoću vektora i matrica taj model može se zapisati vrijednostima (realni vektor) funkcije, [29].

$$\sigma: \mathbb{R}^n \rightarrow \mathbb{R}^m \Rightarrow a^{(i+1)} = \sigma(W^{(i)})a^{(i)} + b^{(i)} \tag{3}$$



Slika 32. Neuronska mreža s L međuslojeva, [29]

Na slici 32 je prikazan primjer neuronske mreže koja se sastoji od ulaznog (***a***) sloja, L međuslojeva ($a^{(1)}$ … $a^{(L)}$) i izlaznog (***o***) sloja. Karakteristično je da se na takav način umrežuje više slojeva umjetnih neurona, dok početni sloj neurona (vektor ***a***) predstavlja ulazni podatak, a završni sloj neurona (vektor ***o***) predstavlja izlazni podataka neuronske mreže. Međuslojevi u mreži nazivaju se skrivenim slojevima (eng. hidden layer). Oni se formiraju s različitim brojem neurona ovisno o aktivnosti i tipu koju neuronska mreža imitira. Ne koriste se uvijek isti oblici neuronskih mreža, primjerice, drukčiji oblici neuronskih mreža koriste se za prepoznavanje oblika, a drukčiji oblici se upotrebljavaju za optimizacijske probleme. Topološke razlike u obliku mreže određene su prirodom problema koji se rješava i načinom (učenjem) formiranja matrice težina. Važno je spomenuti kako se tijekom procesa učenja formira matrica težina i vektor posmaka za svaki pojedini sloj neuronske mreže osim za ulazni, [29].

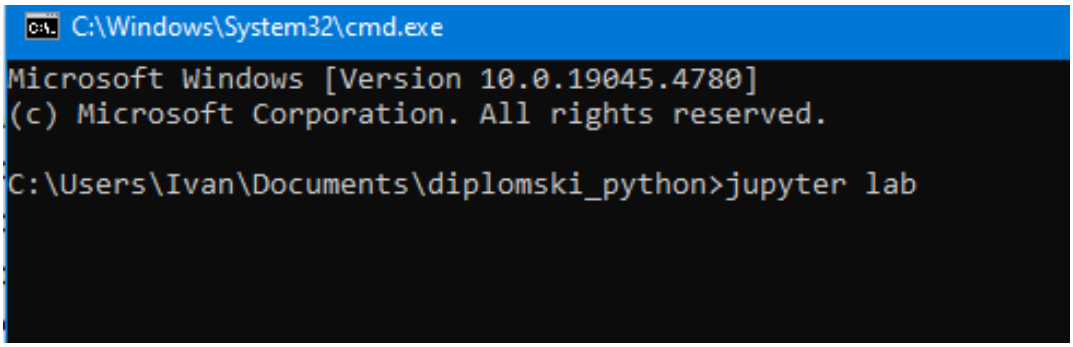
Definicija za grešku neuronske mreže je razlika vrijednosti između očekivane vrijednosti i rezultata, zapravo predviđanja algoritma. Funkcija koja se često koristi za izračun greške je srednja kvadratna pogreška (eng. mean squared error, MSE). MSE se koristi za mjerenje prosječne kvadratne razlike između predviđenih vrijednosti i stvarnih vrijednosti u skupu podataka. Ukoliko je algoritmu proslijeđen skup podataka s više instanci, tada se greška računa kao prosjek svih kvadratnih greški u tom prolazu. Greška služi radi optimizacije, odnosno učenja algoritma, a vizija je da se prilikom tog procesa testiranja greška smanjuje, te da algoritam točnije predviđa rezultate, [30].

Rješenje problema optimizacije modela je treniranje modela. Cilj treniranja je poboljšati sve parametre u modelu, te bi model na taj način, točnije mapirao ulazne vrijednosti s njihovima izlaznim oznakama (klasama). Način poboljšanja parametara u modelu ovisi o odabranom

algoritmu, jer postotci poboljšavaju algoritmom. U većinu slučajeva za rješavanje problema primjenom strojnog učenja, na raspolaganju je samo jedan skup podataka, taj skup podataka je potrebno razdijeliti u tri skupine. Te skupine predstavljaju skup podataka za treniranje, odnosno učenje, za validaciju i za testiranje modela, gdje, naravno, najveći broj podataka treba biti u skupini za treniranje. Skup podataka koji služi za treniranje, koristi se za višekratno slanje kroz mrežu kako bi model učio. Validacijski skup podataka ne sudjeluje direktno u izgradnji modela, nego služi za odabir varijabli i usporedbu modela. On provjerava koliko kvalitetno je model nešto naučio. Neoznačeni podaci nalaze se u skupu ta testiranje, te su oni za model novi podaci jer ih model nije vidio. To omogućava provjeru točnosti modela, jer se on odnosi jednako prema svim ulazima koje obrađuje i na temelju toga predviđa rezultat, [31].

5. Analiza i rezultati predikcije

Programiranje strojnog učenja odrađeno je preko sučelja pod nazivom „Jupyter“. U tom programu koristio se pyhton kao programski jezik. Python je interpretirani, interaktivni, objektno orijentirani programski jezik. Može se koristiti kao jezik proširenja za aplikacije koje trebaju programibilno sučelje. Python podržava module i pakete, što potiče modularnost programa i ponovnu upotrebu koda, [32]. Projekt „Jupyter“ je neprofitan i slobodno dostupan javnosti, nastao iz „I Pyhton“ projekta. Koristi se za interaktivno znanost o podacima i znanstveno računalstvo u svim programskim jezicima. „Jupyter“ je interaktivna radna bilježnica u kojoj se može unositi tekst, jednostavnije pokretati pyhton programi, obrađivati podaci, prikazivati podaci u vidu tabele i dijagrama, [33]. Da bi se aplikacija „Jupyter“ pokrenula potrebno je preko komandnog prozora u Windowsu pozvati akciju pokretanja aplikacije, prikazano na slici 33.



Slika 33. CMD

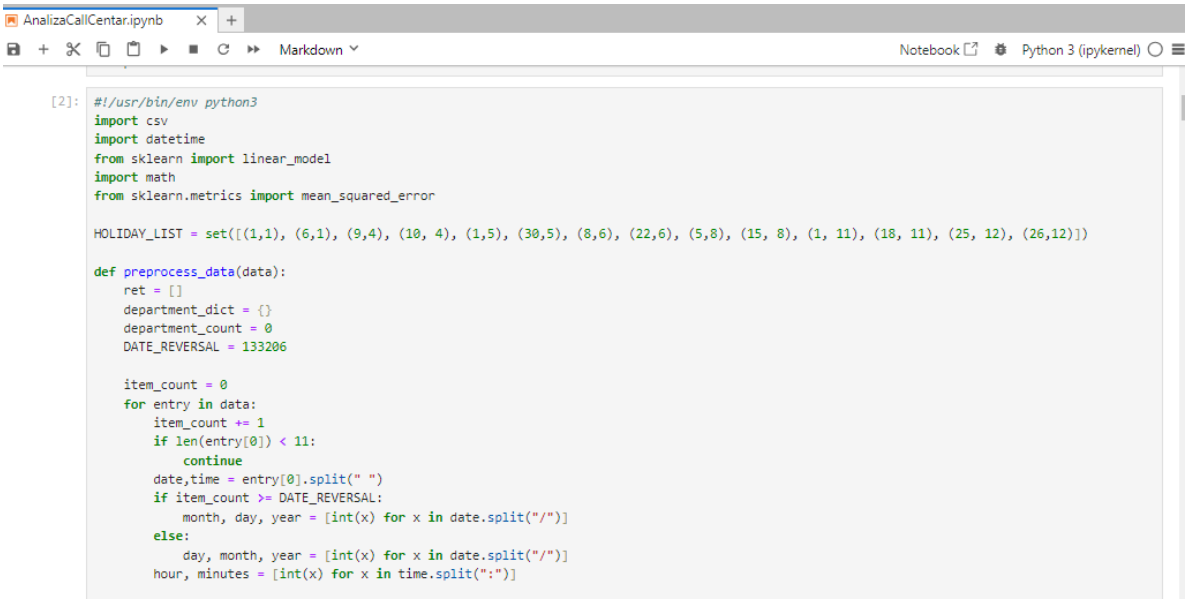
Kada se otvorila aplikacija, za početak je potrebno napisati naredbe za korištenje biblioteke Matplotlib u Pythonu za prikazivanje grafova. Kod prikazan na slici 34 omogućuje jednostavno korištenje Matplotlib biblioteke za crtanje grafova unutar Jupyter Notebooka, tako da se grafovi automatski prikazuju u bilježnici odmah nakon izvršavanja ćelije s kodom.

```
[1]: from matplotlib import pyplot as plt
     %matplotlib inline
```

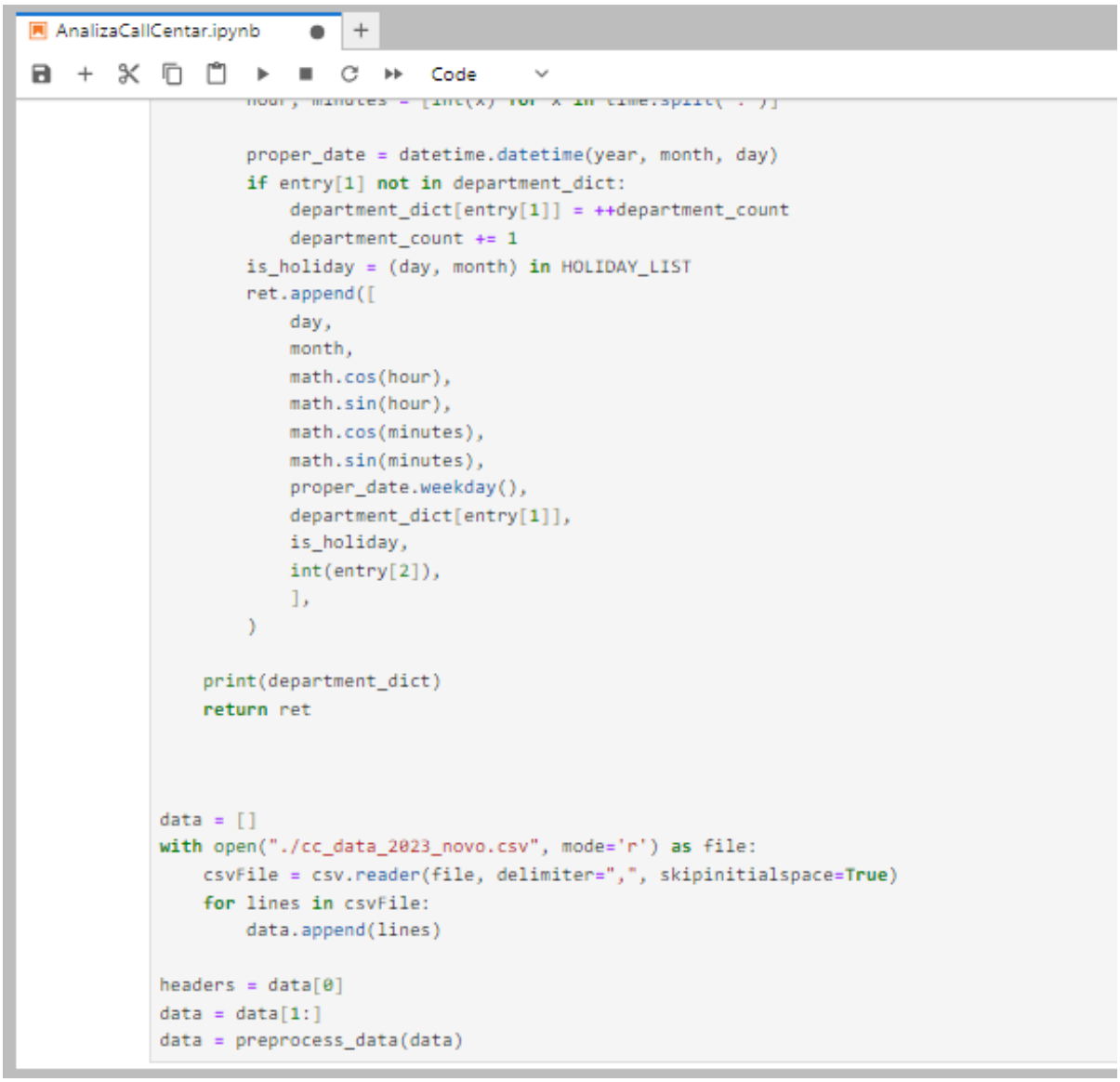
Slika 34. Kod za kreiranje grafova

Na slikama 35 i 36 prikazan je početni dio u kojem se nalazi nekoliko zadataka vezanih uz obradu podataka iz csv datoteke, pripremu podataka za strojno učenje i transformaciju vremenskih podataka. Unutar csv datoteke predani su podaci koji su prikupljeni iz pozivnog centra. Podaci koji se obrađuju obuhvaćaju cijelu 2023. godinu i predstavljaju prometno

opterećenje pozivnog centra. Predani podaci su broj dolaznih poziva te datum. U prvom dijelu zapisani su moduli koji omogućavaju rad sa CSV datotekama, rad s datumima i vremenom, modul za regresijske modele iz biblioteke scikit-learn, modul za matematičke funkcije i funkcija za izračunavanje srednje kvadratne pogreške. Zatim je ispisana lista pod nazivom „HOLIDAY_LIST“ gdje se nalaze svi praznici i blagdani Republike Hrvatske. Nakon toga dolazi funkcija za obradu podataka, zatim obrada svakog unosa, obrada datuma i vremena, obrada odjela i praznika, dodavanje obrađenih podataka u listu, ispis i vraćanje obrađenih podataka, učitavanje podataka iz CSV datoteke i priprema podataka.



Slika 35. Jupyter 1



Slika 36. Jupyter 2

Prikaz postojećih podatka određen je agregacijom po danima po ukupnom broju poziva. Koristio se Python kod koji služi za vizualizaciju podataka putem histograma, gdje se prikazuje broj određenih događaja za svaki dan u mjesecu. Kod stvara 12 histogram grafova, po jedan za svaki mjesec u godini.

Parametri funkcije:

- data: popis podataka koji se koriste za crtanje histograma. Svaki element u data sadrži informacije o danu, mjesecu i nekoj vrijednosti.
- filter_month: mjesec za koji se želi prikazati histogram (npr. 1 za siječanj, 2 za veljaču itd.).
- hist: os na koju se graf dodaje (ovo omogućava crtanje više grafova na istoj slici).

- title: naslov grafa.

Opis funkcije:

1. pls = [0] * 31: stvara listu s 31 nulom, gdje će se brojiti vrijednosti za svaki dan u mjesecu (pretpostavlja se da mjesec može imati najviše 31 dan).
2. xs = []: kreira prazan popis xs, koji će sadržavati brojeve dana u mjesecu (od 1 do 31).
3. for i in range(31): popunjava xs s brojevima od 1 do 31.
4. for d in data: prolazi kroz sve podatke u data.
5. if month == filter_month: provjerava podudara li se mjesec u podatku s mjesecem koji se filtrira (filter_month). Ako se podudara, povećava odgovarajuću vrijednost u pls za taj dan.
6. hist.bar(xs, pls, alpha=0.5): crta histogram gdje su xs dani u mjesecu, a pls odgovarajuće vrijednosti.
7. hist.title.set_text(title): postavlja naslov za graf.

Stvaranje figure i podgrafova:

- plt.subplots(6, 2, ...): stvara mrežu od 12 podgrafova (6 redova i 2 stupca), gdje će svaki podgraf prikazivati histogram za jedan mjesec.
- sharey=True: dijeli istu osu y među svim podgrafovima (olakšava usporedbu).
- tight_layout=True: osigurava da su podgrafovi lijepo raspoređeni bez preklapanja.
- figsize=(16,20): postavlja veličinu cijele slike na 16x20 inča.

Svaki poziv funkcije plot_month_histogram stvara histogram za jedan mjesec. Funkcija se poziva 12 puta, po jednom za svaki mjesec, te se na odgovarajuću poziciju u mreži podgrafova (axs) postavlja odgovarajući histogram. Opisani kodovi vidljivi su na slici 37. Na iscrtanim histogramima vidljivi su rezultati opterećenja pozivnog centra kao količina upućenih poziva prema pozivnom centru. Iz histograma vidljivo je da je značajno manje opterećenje tijekom vikenda uključujući i ne radne dane, odnosno manja količina upućenih poziva. Značajni porast je vidljiv prvog kolovoza (August) što se može pretpostaviti da se dogodio pad u mreži te je bila veća količina poziva zbog prijava problema.

```
[3]: def plot_month_histogram(data, filter_month, hist, title):
         pls = [0] * 31
         xs = []
         for i in range(31):
             xs.append(i + 1)
         for d in data:
             day, month = d[0], d[1]
             if month == filter_month:
                 pls[day-1] += d[-1]
         hist.bar(xs, pls,alpha=0.5)
         hist.title.set_text(title)

     fig, axs = plt.subplots(6, 2, sharey=True, tight_layout=True, figsize=(16,20))
     plot_month_histogram(data, 1, axs[0][0], "January")
     plot_month_histogram(data, 2, axs[0][1], "February")
     plot_month_histogram(data, 3, axs[1][0], "March")
     plot_month_histogram(data, 4, axs[1][1], "April")
     plot_month_histogram(data, 5, axs[2][0], "May")
     plot_month_histogram(data, 6, axs[2][1], "June")
     plot_month_histogram(data, 7, axs[3][0], "July")
     plot_month_histogram(data, 8, axs[3][1], "August")
     plot_month_histogram(data, 9, axs[4][0], "September")
     plot_month_histogram(data, 10, axs[4][1], "October")
     plot_month_histogram(data, 11, axs[5][0], "November")
     plot_month_histogram(data, 12, axs[5][1], "December")
```

Slika 37. Prikaz postojećih podataka

Na slici 38 prikazani su histogrami koji prikazuju količinu dolaznih poziva u danu za 2023. godinu. Može se reći da je po histogramima vidljivo smanjena količina dolaznih poziva tijekom vikenda. Također je vidljivo da je veći broj dolaznih poziva u periodima od 5. do 12. u mjesecima, što je razumljivo jer u tim periodima se obično korisnicima šalju računi. Svakako je vidljivo za pojedine dane znatno veći broj dolaznih poziva, što se može svrstati u grupu izvanrednih situacija u kojima je došlo do poteškoća unutar mreže, te je veći broj korisnika kontaktirao pozivni centar kako bi prijavio poteškoću.

Slika 38. Histogrami s postojećim podacima

Nakon što su postojeći podaci prikazani u histogramima, započeto je s algoritmima strojnog učenja. Prvi algoritam koji se koristio bio je linearna regresija za predikciju budućeg opterećenja. Predani su prikupljeni podaci iz pozivnog centra odnosno broj dolaznih poziva i datum, te se nastoji predvidjeti buduće opterećenje za iduću godinu. U nastavku je pojašnjenje algoritma koji je vidljiv za slici 39, odnosno koda za linearnu regresiju.

1. **Priprema podataka za modeliranje**

- ys: ovdje se stvara lista ys koja sadrži zadnju vrijednost (ciljanu vrijednost) iz svakog zapisa u data.
- xs: stvara se lista xs koja sadrži sve vrijednosti iz zapisa osim zadnje (ulazni podaci za model).

2. **Stvaranje i treniranje modela**

- reg: stvara se objekt linearne regresije iz biblioteke sklearn.
- reg.fit(xs, ys): model se trenira pomoću podataka xs (ulazne varijable) i ys (ciljne varijable).

3. **Testiranje modela**

- reg.predict([xs[0]]): predviđa se vrijednost za prvi ulazni podatak iz xs koristeći trenirani model.
- ys[0]: prikazuje stvarnu, odnosno očekivanu vrijednost za prvi unos, koju model treba predvidjeti.
- **rezultat:** model je predvidio vrijednost [3.74655179] za prvi ulazni podataka, a stvarna vrijednost je 1, to pokazuje određenu pogrešku u predikciji
- mean_squared_error(ys, reg.predict(xs)): izračunava srednju kvadratnu pogrešku (MSE) između stvarnih vrijednosti ys i predviđenih vrijednosti reg.predict(xs), što daje mjeru preciznosti modela. **MSE** je mjera koja pokazuje koliko su predikcije modela udaljene od stvarnih vrijednosti.
- formula za MSE:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i}) \tag{4}$$

- n je broj uzoraka
- y_i su stvarne vrijednosti
- $\widehat{y_i}$ su predviđene vrijednosti modela

- **rezultat:** 43.7934243038463 pokazuje prosječnu kvadratnu razliku između stvarnih i predviđenih vrijednosti za cijeli skup podataka. Što je ova vrijednost manja, to je model precizniji. U ovom slučaju, MSE je poprilično visok, što može značiti da model ima značajne pogreške u predikcijama.

4. Generiranje podataka za određeni mjesec

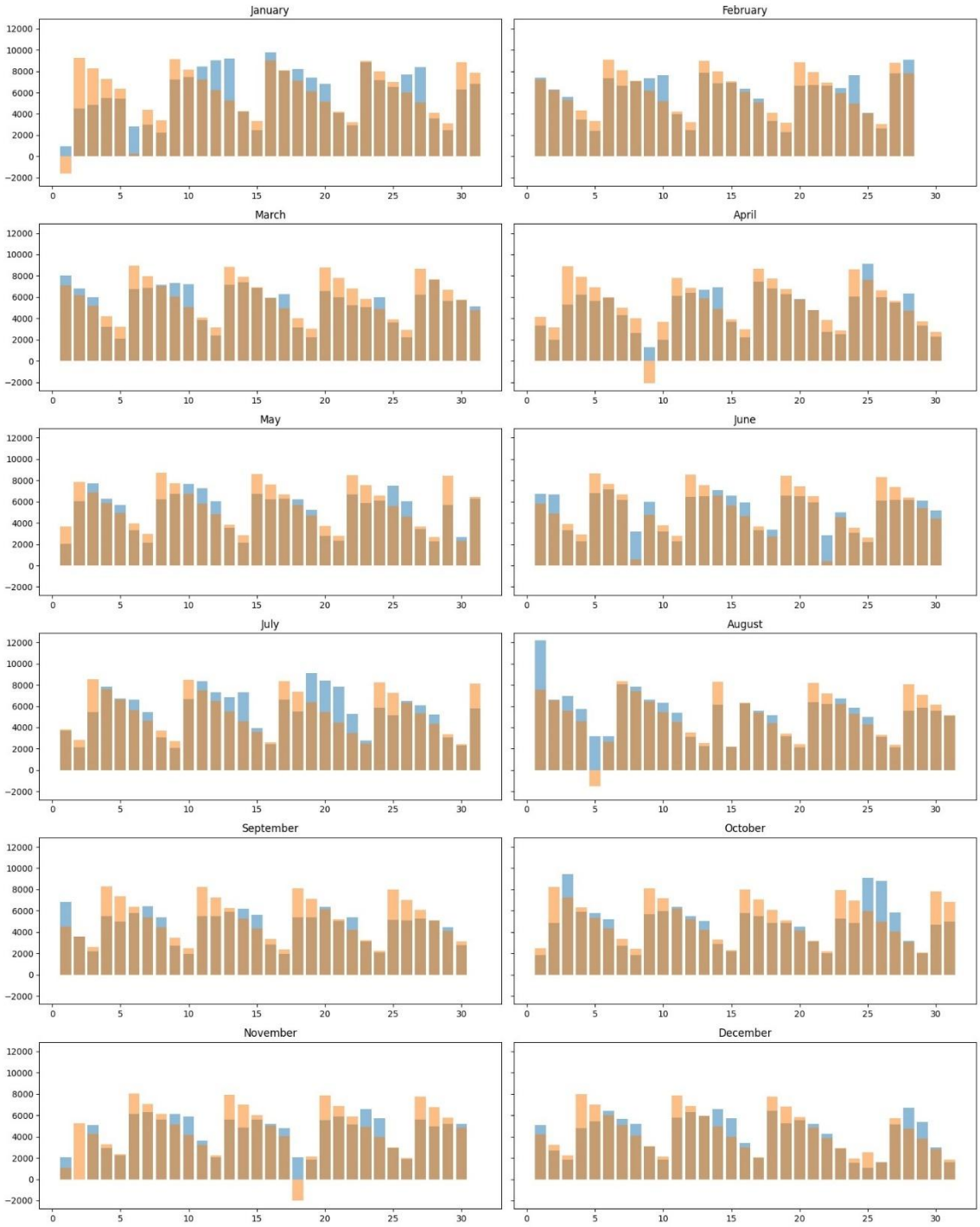
- generate_data_for_month: funkcija koja generira podatke za određeni mjesec, simulirajući različite kombinacije dana, sati, minuta, odjela, itd.
- month_length: rječnik koji sadrži broj dana za svaki mjesec u godini.
- petlje: for day in range(1, month_length[month] + 1): iterira kroz sve dane u mjesecu, for hour in range(0, 12): iterira kroz sate (od 0 do 11), for minutes in range(0, 59, 15): iterira kroz minute u koracima od 15 minuta, for department_index in range(26): iterira kroz 26 različitih odjela.
- generiranje značajki: za svaki dan, sat, minute i odjel, generiraju se značajke poput kosinusa i sinusa za vrijeme (za bolju modelsku performansu), te se bilježi je li taj dan praznik.
- predikcija: ys = model.predict(xs) koristi trenirani model da predvidi ciljne vrijednosti za generirane podatke.
- spajanje podataka: predviđene vrijednosti se dodaju natrag u popis xs kako bi se dobila potpuna simulacija.
- vraćanje podataka: funkcija vraća popis xs s pridodanim predikcijama.

Linearna regresija

[4]:
```
ys = [d[-1] for d in data]
xs = [d[:-1] for d in data]
reg = linear_model.LinearRegression()
reg.fit(xs,ys)
print(reg.predict([xs[0]]), ys[0])
print(mean_squared_error(ys, reg.predict(xs)))

def generate_data_for_month(month, model):
    xs = []
    month_length = {1:31,2:28,3:31,4:30,5:31,6:30,7:31,8:31,9:30,10:31,11:30,12:31}
    for day in range(1,month_length[month]+1):
        proper_date = datetime.datetime(2023, month, day)
        is_holiday = (day, month) in HOLIDAY_LIST
        for hour in range(0,12):
            for minutes in range(0, 59, 15):
                for department_index in range(26):
                    xs.append([
                        day,
                        month,
                        math.cos(hour),
                        math.sin(hour),
                        math.cos(minutes),
                        math.sin(minutes),
                        proper_date.weekday(),
                        department_index,
                        is_holiday,
                        ],
                    )
    ys = model.predict(xs)
    for i in range(len(xs)):
        xs[i].append(ys[i])
    return xs
```

Slika 39. Algoritam linearne regresije

Zatim je pozvana funkcija za kreiranje histograma. Na slici 40 su prikazani histogrami predikcije budućeg opterećenja označeni narančastom bojom, kreirani algoritmom linearne regresije. Na histogramima je vidljivo da je za čak četiri dana rezultat predikcije dolaznih poziva pao ispod nule, zapravo predvidio je broj poziva u minusu što je nerealno i što potvrđuje da model ima znatne greške u predviđanju budućeg opterećenja.



Slika 40. Predikcija linearne regresije

Sljedeći algoritam koji se koristio za predikciju je jednostavna neuronska mreža. Jednostavna je iz razloga što ne ide u više slojeva. U nastavku je opisan kod za taj algoritam, a Python kod algoritma je prikazan na slici 41. Kod implementira model neuronske mreže koristeći MLPRegressor iz biblioteke scikit-learn za regresiju, a zatim evaluira performanse modela koristeći srednju kvadratnu pogrešku (MSE). Evo detaljnog opisa svakog dijela koda:

1. **Import modula i funkcija**

- train_test_split: funkcija za podjelu podataka na trening i test setove.
- MLPRegressor: regresijski model temeljen na višeslojnim perceptronima (neuronskim mrežama).
- mean_squared_error: funkcija za izračun srednje kvadratne pogreške (MSE).

2. **Podjela podataka na trening i test setove**
   - train_test_split: dijeli podatke na dva dijela: X_train i y_train (podaci za treniranje modela) i X_test i y_test (podaci za testiranje modela).
   - random_state=1: osigurava da podjela podataka bude reproducibilna.
3. **Treniranje modela neuronske mreže**
   - MLPRegressor: stvara model neuronske mreže sa sljedećim parametrima: activation="relu": koristi ReLU funkciju aktivacije (Rectified Linear Unit), random_state=1: osigurava točnost rezultata, max_iter=200: maksimalni broj iteracija za treniranje modela, hidden_layer_sizes=(20): definira arhitekturu mreže s jednim skrivenim slojem koji sadrži 20 neurona.
   - .fit(X_train, y_train): treniranje modela na trening podacima.
4. **print(regr.predict(X_test[:2]), y_test[:2]):** model za predikciju koji se temelji na prva dva uzorka iz testnog skupa, i stvarne vrijednosti koje odgovaraju prva dva uzorka
   - **rezultati:** [2.83014019 14.37899774] su vrijednosti koje je model predvidio za prva dva uzorka, dok su stvarne vrijednosti [6, 15], vidljivo je veće odstupanje za prvi uzorak, dok je za drugi uzorak manje odstupanje
5. **print(mean_squared_error(y_train, regr.predict(X_train))):** izračunava se MSE, predviđa se vrijednost za sve uzorke u trenirajućem skupu, te se uspoređuje sa stvarnim vrijednostima
   - **rezultat:** 26.223253147789745 je rezultat MSE-a, što prikazuje da model ima određenu razinu greške u predikciji nad podacima za treniranje
6. **print(mean_squared_error(y_test, regr.predict(X_test))):** izračunava MSE za testni skup podataka
   - **rezultat:** 25.18125532843256 je rezultat MSE za testni skup, vrijednost pogreške je manja na trenirajućem skupu, svakako prikazuje razinu pogreške modela za predikciju
7. **print(y_test[:8]) i print(regr.predict(X_test[:8])):** prvi dio ispisuje osam stvarnih vrijednosti iz testnog skupa podataka, dok drugi dio je model koji predviđa osam uzoraka iz testnog skupa podataka

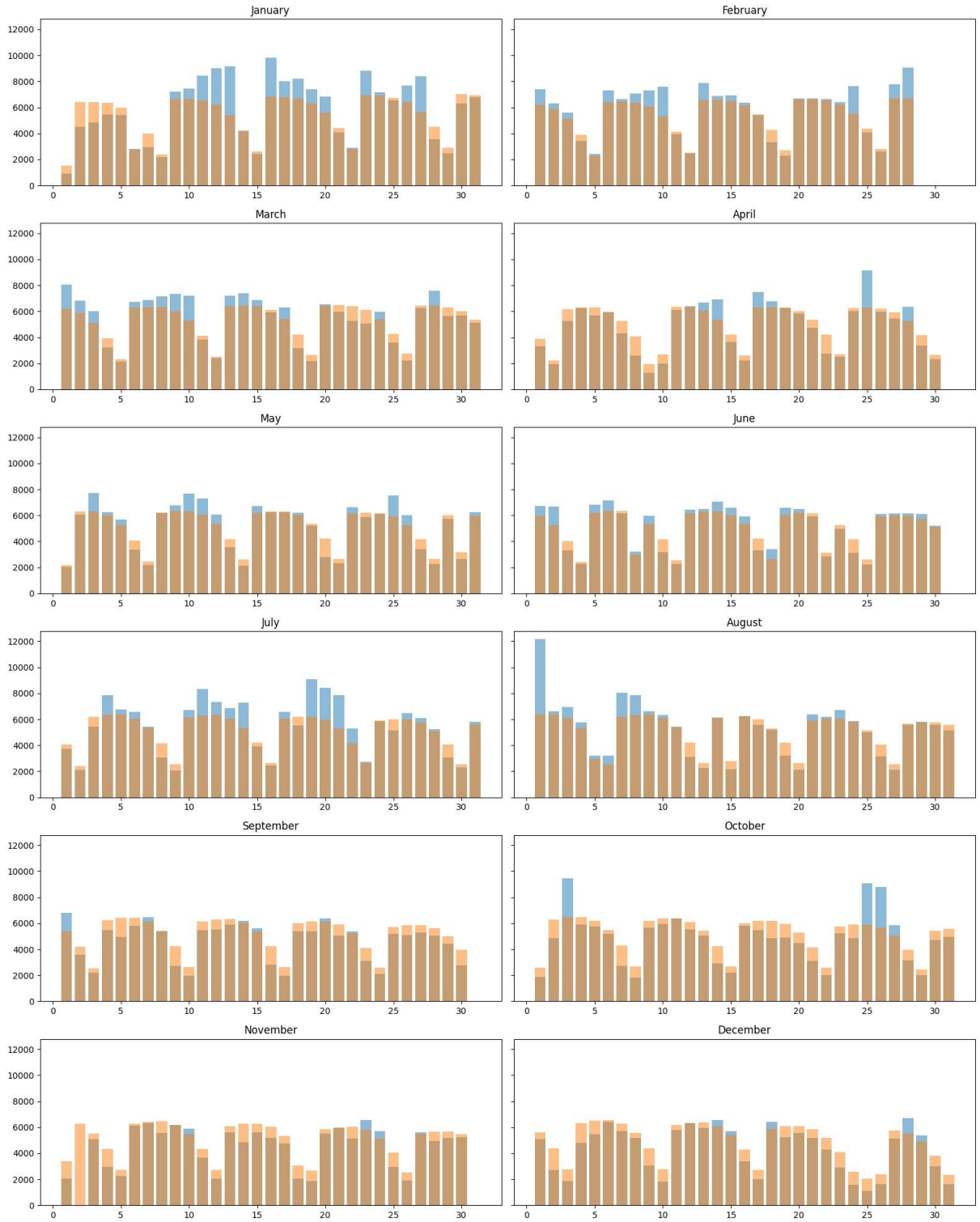
- **rezultati:** osam ispisanih stvarnih vrijednosti je [6, 15, 8, 5, 3, 6, 2, 4], dok su predviđene vrijednosti [4.3302844 12.75069484 14.70346197 6.58527362 4.59491806 3.78296269 0.82651708 3.35347729], vidljivo je da model ponekad griješi za neke uzorke, dok je za neke prilično precizan

Jednostavna neuronska mreza

```
[5]: from sklearn.model_selection import train_test_split
     from sklearn.neural_network import MLPRegressor

     X_train, X_test, y_train, y_test = train_test_split(xs, ys,random_state=1)
     regr = MLPRegressor(activation="relu",random_state=1, max_iter=200, hidden_layer_sizes=(20)).fit(X_train, y_train)
     print(regr.predict(X_test[:2]), y_test[:2])
     print(mean_squared_error(y_train, regr.predict(X_train)))
     print(mean_squared_error(y_test, regr.predict(X_test)))
     print(y_test[:8])
     print(regr.predict(X_test[:8]))
```

Slika 41. Algoritam neuronske mreže

Na slici 42 prikazana je predikcija budućeg opterećenja za 12 mjeseci označeno narančastom bojom, dok je plavo prikazan dosadašnjih podataka. Model jednostavne neuronske mreže u usporedbi s lineranom regresijom, je precizniji u prikazu predikcije budućeg opterećenja. Njegove vrijednosti nisu pale ispod nule. Za vikende je predvidio smanjeno opterećenje, dok je za neke uzorke, dane, predvidio i veće opterećenje nego što je u podacima koji su predani. Svakako je model odradio poprilično dobru predikciju budućeg opterećenja te bi mogao poslužiti kao model s kojim bi se moglo raditi.



Slika 42. Histogrami jednostavne neuronske mreže

I za kraj je korištena kompleksna neuronska mreža. To je neuronska mreže sa 40 neurona na ulazu potom 20 pa 15 do izlaza. Kao model aktivacije neurona se korist relu funkcija. Također dodana je i klasična sigmoida koja pokazuje nešto bolje rezultate. Python kod algoritma je prikazan na slici 43.

Detaljan opis koda:

1. **Uvoz potrebnih modula**

   - train_test_split: funkcija za podjelu podataka na trening i test setove.
   - MLPRegressor: regresijski model temeljen na višeslojnim perceptronima (neuronskim mrežama).
   - mean_squared_error: funkcija za izračunavanje srednje kvadratne pogreške (MSE).

2. **Podjela podataka na trening i test setove**

   - X_train i y_train: podaci koji će se koristiti za treniranje modela.
   - X_test i y_test: podaci koji će se koristiti za testiranje modela.
   - random_state=1: održava konzistentnost rezultata prilikom ponovnog pokretanja koda.

3. **Treniranje prvog modela neuronske mreže s ReLU aktivacijskom funkcijom**

   - activation="relu": koristi ReLU (eng. Rectified Linear Unit) funkciju aktivacije.
   - hidden_layer_sizes=(40,20,15): definira mrežu s tri skrivena sloja koji sadrže 40, 20 i 15 neurona.
   - max_iter=500: maksimalni broj iteracija za treniranje modela.

4. **Treniranje drugog modela neuronske mreže s logističkom aktivacijskom funkcijom**

   - activation="logistic": koristi logističku funkciju aktivacije, koja je sigmoidalna funkcija.

5. **Predikcija i evaluacija za prvi model (ReLU aktivacija)**

   - regr_40_20_15.predict(X_test[:2]): predviđa ciljne vrijednosti za prva dva primjera iz testnog seta koristeći model s ReLU aktivacijom.
   - **rezultati:** [2.64805793 11.62254312] su vrijednosti koje je model predvidio, a stvarne vrijednosti su [6, 15], vidljivo je odstupanje predviđenih vrijednosti od stvarnih vrijednosti
   - mean_squared_error(y_train, regr_40_20_15.predict(X_train)): izračunava MSE na trening setu.

- **rezultati:** 15.31252456578023 je rezultat MSE trenirajućih podatka, te je vrijednost značajno manja u usporedbi s dosadašnjim rezultatima
- mean_squared_error(y_test, regr_40_20_15.predict(X_test)): izračunava MSE na testnom setu.
- **rezultati:** 15.177774039477224 rezultat MSE testnih podataka, u usporedbi s trenirajućim razlika je u decimalama
- y_test[:8] i regr_40_20_15.predict(X_test[:8]): uspoređuje stvarne i predviđene vrijednosti za prvih osam primjera iz testnog seta.
- **rezultati:** stvarne vrijednosti su [6, 15, 8, 5, 3, 6, 2, 4] dok su predviđene vrijednosti [2.64805793 11.62254312 14.41363876 5.91348362 2.7236155 3.4751048 0.88177159 1.42663227] gdje vide odstupanja između stvarnih vrijednosti i predviđenih vrijednosti

6. **Predikcija i evaluacija za drugi model (logistička aktivacija)**

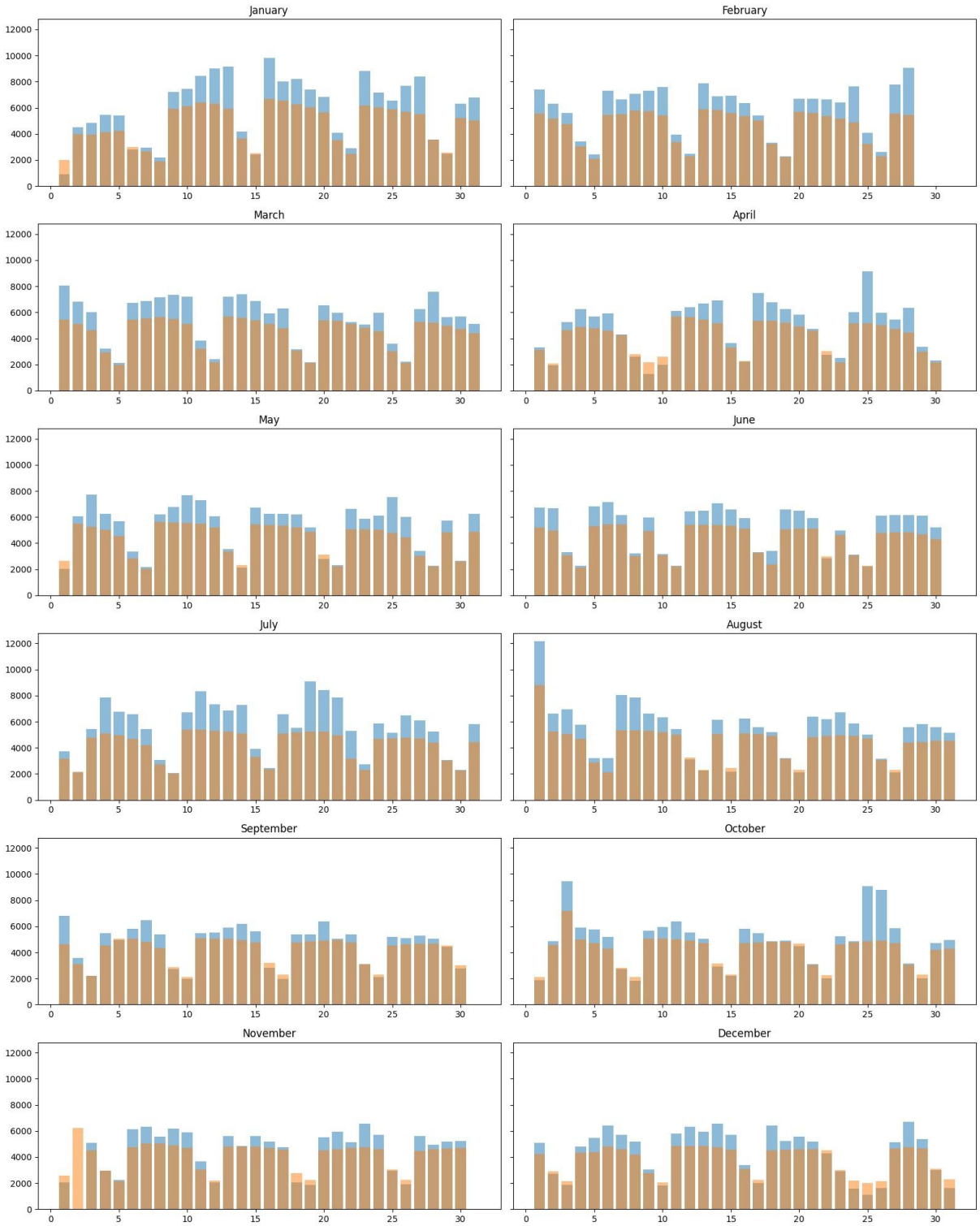
- slične operacije kao u koraku 5, ali ovdje se koriste rezultati dobiveni s modelom koji koristi logističku aktivaciju.
- **rezultati:** [4.3302844 12.75069484] su vrijednosti koje je model predvidio, a stvarne vrijednosti su [6, 15], vidljivo je da je odstupanje predviđenih vrijednosti od stvarnih vrijednosti u usporedbi s ReLU aktivacijom malo manje
- **rezultati:** 13.933677246694508 je rezultat MSE trenirajućih podatka, te je vrijednost dosta manja u usporedbi s dosadašnjim rezultatima
- **rezultati:** 13.923621258027518 rezultat MSE testnih podataka, u usporedbi s trenirajućim razlika je u neprimjetna
- **rezultati:** stvarne vrijednosti su [6, 15, 8, 5, 3, 6, 2, 4] dok su predviđene vrijednosti [4.3302844 12.75069484 14.70346197 6.58527362 4.59491806 3.78296269 0.82651708 3.35347729] odstupanja između stvarnih vrijednosti i predviđenih vrijednosti u usporedbi s ReLU aktivacijom su manja, odnosno rezultati su precizniji

Ovaj kod omogućuje usporedbu dvaju različitih modela neuronske mreže s različitim funkcijama aktivacije (ReLU i logistička) i istom arhitekturom skrivenih slojeva. Izlazi iz mean_squared_error funkcije daju uvid u to koji model bolje generalizira podatke i ima manju pogrešku na trening i test setovima. Usporedba stvarnih i predviđenih vrijednosti pomaže u procjeni točnosti svakog modela.

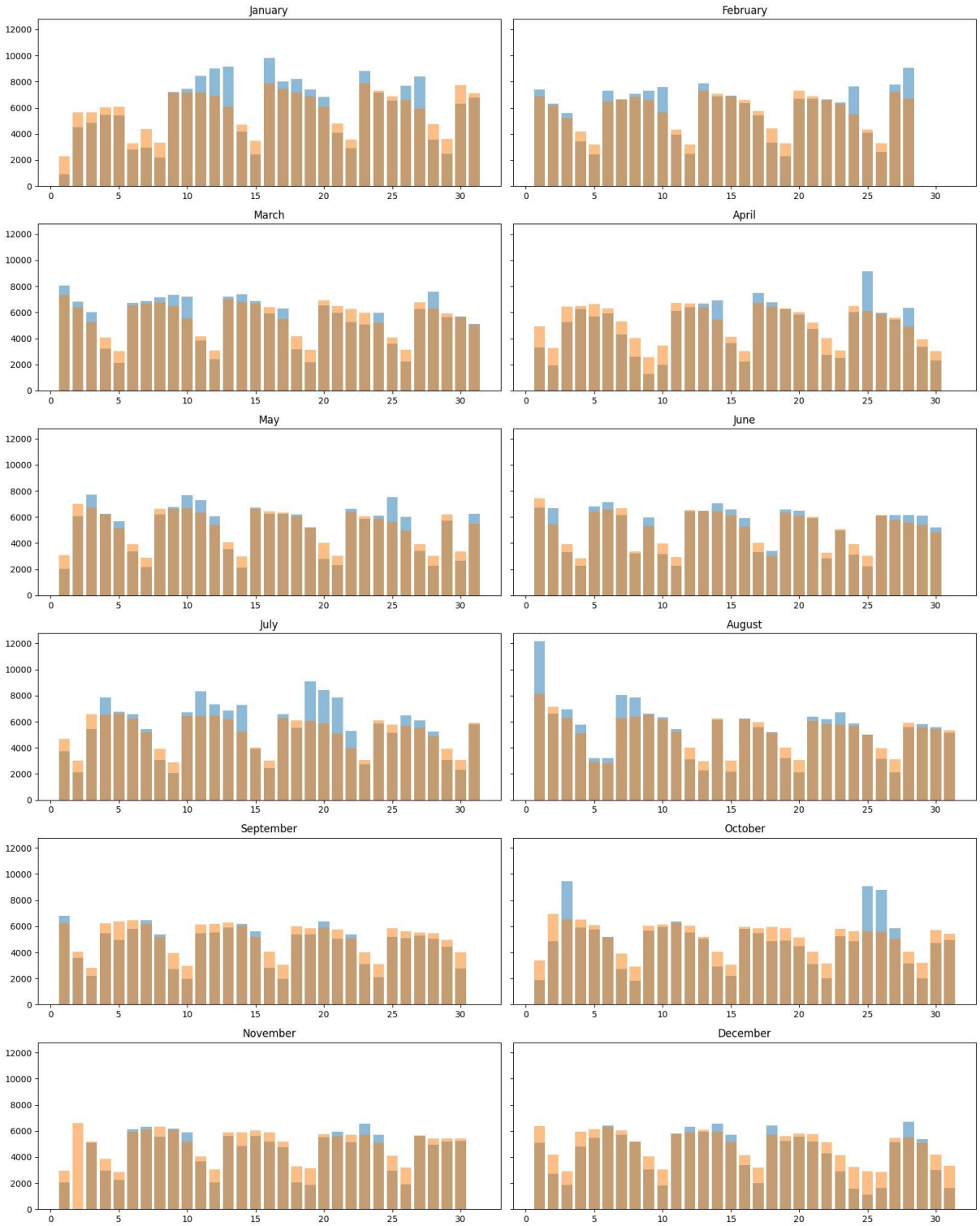
```
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor

X_train, X_test, y_train, y_test = train_test_split(xs, ys,random_state=1)
regr_40_20_15_logistic = MLPRegressor(activation="logistic",random_state=1, max_iter=500, hidden_layer_sizes=(40,20,15)).fit(X_train, y_train)
regr_40_20_15 = MLPRegressor(activation="relu",random_state=1, max_iter=500, hidden_layer_sizes=(40,20,15)).fit(X_train, y_train)
print(regr_40_20_15.predict(X_test[:2]), y_test[:2])
print(mean_squared_error(y_train, regr_40_20_15.predict(X_train)))
print(mean_squared_error(y_test, regr_40_20_15.predict(X_test)))
print(y_test[:8])
print(regr_40_20_15.predict(X_test[:8]))

print(regr_40_20_15_logistic.predict(X_test[:2]), y_test[:2])
print(mean_squared_error(y_train, regr_40_20_15_logistic.predict(X_train)))
print(mean_squared_error(y_test, regr_40_20_15_logistic.predict(X_test)))
print(y_test[:8])
print(regr_40_20_15_logistic.predict(X_test[:8]))
```

Slika 43. Algoritam kompleksne neuronske mreže

Na slikama 44 i 45 vidljivi su rezultati predikcije kompleksne neuronske mreže sa ReLU aktivacijom i logističkom aktivacijom. Između ova dva modela, rezultati nemaju značajnih, oku vidljivih razlika. Svakako se može reći da su ovo najprecizniji modeli od svih testiranih u ovom radu. S obzirom na rezultate pogreške i predikcije vrijednosti precizniji model je s logističkom aktivacijom, te se na taj model predikcije budućeg opterećenja pozivnog centra može osloniti.



Slika 44. Histogrami kompleksne neuronske mreže



Slika 45. Histogrami kompleksne neuronske mreže sa logističkom aktivacijom

6. Zaključak

U radu je prikazan sveobuhvatan pristup analizi i optimizaciji prometnog opterećenja pozivnog centra kroz upotrebu relacijskih baza podataka, web-aplikacija i algoritama strojnog učenja. Kroz svaki korak ovog procesa, od izrade relacijske baze podataka do primjene naprednih algoritama, uspješno su integrirani ključni elementi za postizanje cilja, a to je poboljšanje učinkovitosti rada pozivnog centra i predikcije budućih opterećenja.

Izgradnja relacijske baze podataka, temeljena na podacima pozivnog centra, omogućila je strukturirano pohranjivanje i manipulaciju podacima. Korištenjem SQL Server Management Studija, kreirana je baza koja osigurava integritet i dosljednost podataka, pružajući čvrst temelj za sve daljnje analize. Primjena strukturiranog jezika upita (SQL) omogućila je efikasno dohvaćanje i analizu podataka, što je ključno za razumijevanje obrazaca u pozivima i prepoznavanje vremenskih razdoblja s najvišom razinom opterećenja.

U daljnjem dijelu rada, izrađena je web-aplikacija koja omogućava pristup podacima na intuitivan i pregledan način. Ova aplikacija, izgrađena u MVC okruženju, omogućava korisnicima jednostavan pristup rezultatima analize, vizualizaciju podataka te interaktivno istraživanje informacija. Kroz web-aplikaciju, korisnici mogu jednostavno pregledavati ključne metrike pozivnog centra, čime se olakšava donošenje odgovarajućih odluka o optimizaciji resursa i upravljanju opterećenjem.

Poseban naglasak stavljen je na primjenu strojnog učenja kao alata za predikciju budućeg opterećenja pozivnog centra. Kroz korištenje algoritama poput linearne i logističke regresije, te neuronskih mreža, moguće je identificirati obrazac u povijesnim podacima i primijeniti te spoznaje na predviđanje budućih trendova. Ovi modeli omogućavaju preciznu predikciju vremena najvećeg opterećenja, što je ključno za optimizaciju raspodjele resursa i planiranje radnog vremena agenata.

U konačnici, rezultati analize pokazali su da je moguće unaprijediti učinkovitost rada pozivnog centra kroz integraciju različitih tehnologija i pristupa. Korištenjem relacijskih baza podataka, strukturiranog jezika upita, web-aplikacija i algoritama strojnog učenja, ostvarena je cjelovita platforma koja ne samo da omogućava efikasnije upravljanje trenutnim opterećenjem, već i proaktivno djelovanje na temelju predikcija budućeg opterećenja. Sami rezultati predikcije budućeg opterećenja pozivnog centra i nisu baš zadovoljavajući s obzirom da nije bilo moguće smanjiti rezultate pogreške, točnije poboljšati preciznost i točnost predikcije zbog manjka

podataka. Najbolje rezultate prikazao je algoritam kompleksne neuronske mreže s logističkom aktivacijom.

Ovaj rad pruža solidnu osnovu za daljnji razvoj i implementaciju sličnih sustava u drugim poslovnim okruženjima. Integracija naprednih tehnologija, kao što su strojno učenje i baze podataka, pokazala se kao vrlo korisna u rješavanju problema visokog opterećenja pozivnog centra, te se ista metodologija može primijeniti i u drugim sektorima gdje je potrebno optimizirati radni proces i resurse. Ovaj rad potvrđuje važnost multidisciplinarnog pristupa u rješavanju poslovnih problema i ukazuje na smjerove daljnjeg istraživanja i unapređenja u području optimizacije poslovnih procesa.

Literatura

[1] M. Varga: Baze podatka, Konceptualno, logičko i fizičko modeliranje podataka, Zagreb, 2022.

[2] J. Martin, “Computer Data - Base Organization”, Prentice - Hall, New Jersey, 1977.

[3] T. Teorey, S. Lightstone, T. Nadeau, Database Modeling & Design: Logical Design, 2006.

[4] T. Carić, M. Buntić: Uvod u relacijske baze podataka, Zagreb 2015.

[5] S. E. Dr. S. Sumathi, Fundamentals of Relational Database Management Systems, Berlin; London : Springer-Verlag Berlin Heidelberg , 2007.

[6] R. Manger: Baze podataka, Element, Zagreb, 2014.

[7] A. Silberschatz, H. F. Korth, S. Sudarshan: Database System Concepts, Seventh Edition, New York, 2020.

[8] Download SQL Server Management Studio (SSMS), Preuzeto s: https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16 [Pristupljeno: 17. kolovoza 2024.]

[9] M. Ilić, L. Kopanja, D. Zlatković, M. Trajković, D. Ćurguz: Microsoft SQL server and Oracle: Comparative performance analysis, The 7th International conference, Vrnjačka Banja, 2021.

[10] „Šta je SQL?“, Preuzeto s: https://bs.itpedia.nl/2017/11/28/wat-is-sql/ [Pristupljeno: 17. kolovoza 2024.]

[11] J. G. Raghu Ramakrishnan, Database Management Systems, 2003.

[12] Upoznajte različite tipove podataka u SQL-u, Preuzeto s: https://www.doit.rs/upoznajte-razlicite-tipove-podataka-u-sql-u/ [Pristupljeno: 17. kolovoza 2024.]

[13] CRUD Operations in SQL, Preuzeto s: https://five.co/blog/crud-operations-in-sql/ [Pristupljeno: 17. kolovoza 2024.]

[14] CSV File Reading and Writing, Preuzeto s: https://docs.python.org/3/library/csv.html [Pristupljeno: 17. kolovoza 2024.]

[15] Uvod u SQL Preuzeto s: https://www.srce.unizg.hr/sites/default/files/edu/programiranje/D301_polaznik_20170223.pdf [Pristupljeno: 17. kolovoza 2024.]

[16] Overview of ASP.NET Core MVC Preuzeto s: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0&WT.mc_id=dotnet-35129-website#mvc-pattern [Pristupljeno: 01. rujna 2024.]

[17] Entity Data Model Preuzeto s: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/entity-data-model [Pristupljeno: 01. rujna 2024.]

[18] Sarker, I. H. Machine Learning: Algorithms, Real-World Applications and Research Directions. SN Comput. Sci. 2, 2021.

[19] V. Jurčić, Strojno učenje u uvjetima manje raspoloživosti podataka, Politehnika: Časopis za tehnički odgoj i obrazovanje, Svezak 7, Broj 2, 2023.

[20] S. Karakatič, I. Fister Ml., Strojno učenje, 2022.

[21] M. Mohri, A. Rostamizadeh, A. Talwalkar, Foundations of Machine Learning. London: The MIT Press, 2018.

[22] Y. Kinha, „An easy guide to choose the right Machine Learning algorithm“, 2020, Preuzeto s:. https://www.kdnuggets.com/2020/05/guide-choose-right-machine-learning-algorithm.html [Pristupljeno: 17. kolovoza 2024.]

[23] Which machine learning algorithm should I use?, 2017, Preuzeto s: https://blogs.sas.com/content/subconsciousmusings/2020/12/09/machine-learning-algorithm-use/ [Pristupljeno: 17. kolovoza 2024.]

[24] Strojno učenje, Preuzeto s: https://www.fkit.unizg.hr/_download/repository/MUI_Strojno%20ucenje.pdf [Pristupljeno: 17. kolovoza 2024.]

[25] S. Ray, „Commonly used Machine Learning Algorithms (with Python and R Codes)“, 2017. Preuzeto s: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/ [Pristupljeno: 17. kolovoza 2024.]

[26] „A Beginner's Guide to Logistic Regression For Machine Learning“ Preuzeto s: https://wiki.pathmind.com/logistic-regression [Pristupljeno: 17. kolovoza 2024.]

[27] J. Šnajder, Strojno učenje, 2021. Preuzeto s: https://www.fer.unizg.hr/_download/repository/SU-2020-06-LogistickaRegresija[1].pdf [Pristupljeno: 17. kolovoza 2024.]

[28] „A Beginner's Guide to Neural Networks and Deep Learning“. Preuzeto s: https://wiki.pathmind.com/neural-network [Pristupljeno: 17. kolovoza 2024.]

[29] R. Banov, A. Valent, J. Anušić, Neuronske mreže za početnike Preuzeto s: https://hrcak.srce.hr/file/425148 [Pristupljeno: 17. kolovoza 2024.]

[30] Loss in a Neural Network explained, Preuzeto s: https://deeplizard.com/learn/video/Skc8nqJirJg, [Pristupljeno: 17. kolovoza 2024.]

[31] „Train, Test, & Validation Sets Explained“. Preuzeto s: https://deeplizard.com/learn/video/Zi-0rlM4RDs [Pristupljeno: 17. kolovoza 2024.]

[32] Python. Preuzeto s: https://www.python.org/doc/essays/blurb/?external_link=true [Pristupljeno: 6. rujan 2024.]

[33] Jupyter. Preuzeto s: https://jupyter.org/ [Pristupljeno: 6. rujan 2024.]

Popis slika

Popis tablica

Sveučilište u Zagrebu
Fakultet prometnih znanosti
Vukelićeva 4, 10000 Zagreb

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOSTI

Izjavljujem i svojim potpisom potvrđujem da je diplomski rad
(*vrsta rada*)

isključivo rezultat mojega vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju upotrijebljene bilješke i bibliografija. Izjavljujem da nijedan dio rada nije napisan na nedopušten način, odnosno da je prepisan iz necitiranog rada te da nijedan dio rada ne krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu završnog/diplomskog rada pod naslovom _Obrada i analiza prometnog opterećenja u svrhu povećanja učinkovitosti pozivnog centra_ , u Nacionalni repozitorij završnih i diplomskih radova ZIR.

Student/ica:

U Zagrebu, _11. rujna 2024____ Matea Hrsto, HMatea
(ime i prezime, *potpis*)