Predikcija vremena pomoću osobne meteorološke stanice

Papić, Marko

Sveučilište Jurja Dobrile u Puli
Tehnički Fakultet u Puli

Marko Papić

PREDIKCIJA VREMENA POMOĆU OSOBNE METEOROLOŠKE STANICE

Diplomski rad

Pula, ožujak, 2025. godine

Sveučilište Jurja Dobrile u Puli
Tehnički Fakultet u Puli

Marko Papić

PREDIKCIJA VREMENA POMOĆU OSOBNE METEOROLOŠKE STANICE

Diplomski rad

JMBAG: 0303090551, redovni student
Studijski smjer: Računarstvo
Predmet: Ugradbeni računalni sustavi
Mentor: Doc. dr. sc. Karlo Griparić

Pula, ožujak, 2025. godine

Tehnički fakultet u Puli

Ime i prezime studenta/ice Marko Papić

JMBAG 0303090551

Status: ☒ redoviti ☐ izvanredni

PRIJAVA TEME DIPLOMSKOG RADA

Doc.dr.sc. Karlo Griparić

Ime i prezime mentora

Računarstvo

Studij

Ugradbeni računalni sustavi

Kolegij

Potvrđujem da sam prihvatio/la temu završnog/diplomskog rada pod naslovom:

Predikcija vremena pomoću osobne meteorološke stanice

(na hrvatskom jeziku)

Weather prediction using a personal weather station

(na engleskom jeziku)

Datum: 2.4.2024.

Doc. dr. sc. Karlo Griparić

(Ime i prezime nastavnika)

Ugradbeni računalni sustavi

(Predmet)

Sveučilište Jurja Dobrile **u Puli**

Tehnički fakultet u Puli

ZADATAK TEME DIPLOMSKOG RADA

Pristupniku Marku Papiću **MBS: 0303090551**

Studentu Tehničkog fakulteta u Puli, izdaje se zadatak za diplomski rad – tema rada pod nazivom:

PREDIKCIJA VREMENA POMOĆU OSOBNE METEOROLOŠKE STANICE

Sadržaj zadatka: Diplomski rad usredotočen je na razvoj osobne meteorološke stanice bazirane na Arduino platformi, sposobne za mjerenje različitih meteoroloških parametara kao što su temperatura, vlažnost zraka, tlak zraka i brzina vjetra, pomoću niza senzora. Glavni cilj rada je korištenje prikupljenih podataka za razvoj i treniranje neuronske mreže koja će se koristiti za analizu meteoroloških podataka u svrhu predikcije kratkoročnih vremenskih prilika. Rad će obuhvatiti faze prikupljanja i obrade podataka, implementaciju i optimizaciju algoritama dubokog učenja te evaluaciju modela s ciljem postizanja pouzdanih predikcija.

Rad obraditi sukladno odredbama Pravilnika o završnom radu Sveučilišta u Puli.

Marko Papić

(Ime i prezime studenta):

Redovni

(status izvanredni/redovni)

Diplomski studij Računarstva

(studij) **Datum:** 02.04.2024. **Potpis nastavnika** _____

Sadržaj

1. Uvod 1
2. Pregled problema i ciljevi rada 2
2.1. Ciljevi rada 2
3. Arduino 3
3.1. Arduino Uno Rev3 3
3.2. Funkcionalnosti i primjene 5
3.3. Integrirano razvojno okruženje (IDE) 5
3.3.1. Instalacija Arduino IDE-a 5
4. Senzori i dodaci 8
4.1. Eksperimentalna pločica 8
4.2. 9V baterija, adapter i jumper žice 9
4.2.1. Jumper žice 9
4.3. DHT22 senzor za temperaturu i vlagu 10
4.4. BMP280 senzor za tlak zraka 11
4.5. Anemometar za brzinu vjetra 12
4.6. Strukturno povezivanje i blokovska shema sustava 14
4.7. Postavljanje okoline za rad 15
5. Kod za mjerenje i prikupljanje podataka 16
5.1. Definicija pinova i konstanti za anemometar 16
5.2. Inicijalizacija sustava i očitavanje podataka 17
5.3. Prikupljanje, obrada i pohrana vremenskih podataka 17
5.3.1. Normalizacija i enkodiranje podataka 18
5.3.2. Priprema podataka za analizu 19
5.3.3 Pohrana podataka u CSV format 21

6. Neuronska mreža za predikciju vremenskih uvjeta 23
6.1. Ključni izazovi kod vremenskih podataka 23
6.2. Priprema podataka 24
6.3. Teorijske osnove i ključni koncepti modela 25
6.4. Eksperimentalni modeli i rezultati 32
6.4.1. Model v3 – prvi značajan iskorak u DNN arhitekturi 33
6.4.2. Model v7 – prvi napredni LSTM model 34
6.4.3. Model v13 – završna evaluacija i problemi 35
6.5. Završni LSTM model 38
6.6. Predikcija kratkoročnih vremenskih uvjeta 41
6.7.1. Rezultati i analiza predikcije 43
6.7.2. Problemi i ograničenja modela 44
7. Zaključak 46
8. Popis slika 47
9. Popis tablica 48
10. Prilozi 49
11. Literatura 50
12. Sažetak 52
13. Abstract 53

1. Uvod

Istraživanje vremenskih uvjeta od velike je važnosti za svakodnevni život, osobito tijekom ljetnih mjeseci kada ekstremni uvjeti poput vrućine i vlage imaju značajan utjecaj na ljudske aktivnosti i planiranje. Vremenski uvjeti značajno utječu na svakodnevni život, osobito tijekom ljetnih mjeseci kada visoke temperature i vlažnost mogu otežati boravak na otvorenom. Danas je postalo neizostavno koristiti pametne telefone ili druge digitalne uređaje za provjeru trenutnih i budućih vremenskih prilika. Ljudi prate vremensku prognozu iz različitih razloga, bilo zbog planiranja putovanja, organizacije događaja ili jednostavno prilagodbe dnevnim aktivnostima. Kako bi se osigurali točni i pravovremeni podaci o vremenskim uvjetima, koriste se meteorološke stanice koje kontinuirano prikupljaju i analiziraju atmosferske podatke. Ovaj rad bavi se razvojem osobne meteorološke stanice temeljene na Arduino Uno razvojnoj pločici i raznim senzorima, s ciljem prikupljanja ključnih meteoroloških podataka. Osim toga, u radu se istražuje primjena neuronske mreže dubokog učenja koja koristi prikupljene podatke za predikciju budućih vremenskih uvjeta. Korištenje Arduina i kompatibilnih senzora odabrano je zbog njihove pristupačnosti, jednostavnosti integracije i fleksibilnosti u programiranju. Ovaj sustav omogućuje korisniku prilagodbu različitih parametara bez potrebe za naprednim znanjem elektrotehnike ili programiranja. Cilj rada je izraditi model koji će s visokom preciznošću predviđati vremenske uvjete, pri čemu se naglasak stavlja na točnost predikcije i učinkovitost primijenjene metode.

2. Pregled problema i ciljevi rada

Tradicionalne metode predikcije vremenskih uvjeta većinom se provode putem državnih meteoroloških stanica i prikupljanjem satelitskih podataka, što nije financijski priuštivo za individualce koje se ovime bave. Razvitkom pristupačnih senzora i platformi poput Arduina omogućena je izrada vlastitih meteoroloških stanica koje se mogu prilagoditi specifičnim potrebama korisnika. Ove stanice omogućuju prikupljanje točnih i lokaliziranih podataka. Glavni izazov u ovom procesu predstavlja obrada i analiza prikupljenih podataka, čime bi se omogućila pouzdana vremenska prognoza.

2.1. Ciljevi rada

Cilj ovog rada je izgradnja osobne meteorološke stanice korištenjem Arduino platforme, prikupljanje podataka o temperaturi, relativnoj vlažnosti zraka, tlaku zraka te brzini vjetra i zatim razvoj i treniranje modela neuronske mreže za predikciju kratkoročnih vremenskih uvjeta. Za izradu neuronske mreže potrebno je prikupiti podatke pomoću Arduina, dok će se podaci koji nisu dostupni putem senzora izračunavati formulama i preuzimati iz resursa s OpenWeather-a. Osnovni zadatak je prikupiti čim više podataka kako bi točnost neuronske mreže bila veća. Meteorološka stanica može se koristiti u različite svrhe i može dati preciznije i dosljednije podatke nego velike meteorološke stanice. Isto tako, njena primjena nije ograničena na vremensku prognozu već se može koristiti za poljoprivredu, npr. za staklenike, ali može koristiti i za kućnu upotrebu kako bi se izmjerila količina vlage te pratila temperatura. Također, nabavom dodatnih senzora stanica se može prilagoditi različitim upotrebama kao npr. sprječavanje trovanja ugljikovim monoksidom u za to osjetljivim područjima. Može se zaključiti da se stanica može koristiti na više načina i lako nadograđivati, a uz to sve jeftinija je od dostupnih uređaja za kupovinu.

3. Arduino

Arduino je platforma otvorenog koda koja obuhvaća i hardverske i softverske komponente. Arduino zajednica uključuje korisnike i programere koji razvijaju projekte koristeći razvojne ploče s mikrokontrolerima, ali i one koji sudjeluju u dizajniranju i unapređenju novih modela tiskanih pločica. Ove razvojne ploče, poznate kao Arduino modeli, koriste se za izradu raznih prototipova i projekata otvorenog koda. Mikrokontroleri dolaze u različitim veličinama i specifikacijama, omogućujući prilagodbu različitim primjenama ovisno o zahtjevima projekta. Manji mikrokontroleri, poput onih iz serije ATtiny, koriste se u jednostavnijim projektima s ograničenim resursima i nižim energetskim zahtjevima, dok veći modeli, poput Arduino Mega s većim brojem pinova i većom memorijom, omogućuju složenije operacije i veću procesnu snagu. Osim toga, mikrokontroleri dolaze u različitim kućištima, uključujući DIP (eng. „Dual In-line Package“) i SMD (eng. „Surface-Mount Device“) verzije, što im omogućuje integraciju u raznovrsne prototipove i industrijske aplikacije. (Arduino.cc (2023) Arduino Documentation.)

3.1. Arduino Uno Rev3

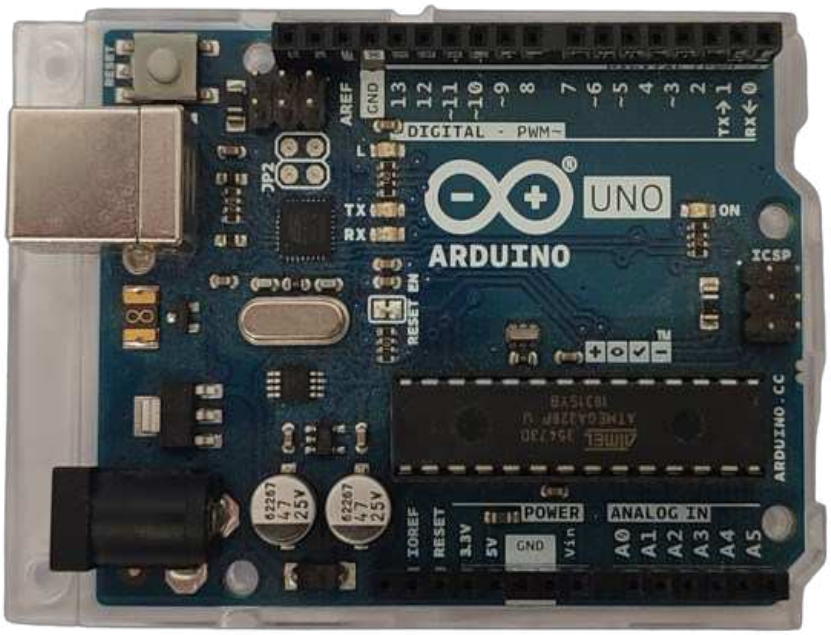
Razvojna ploča koja će se koristiti za izradu osobne meteorološke stanice je Arduino Uno Rev3, prikazana na slici 1. Ova tiskana pločica koristi ATmega328P mikrokontroler, koji omogućuje rad i upravljanje različitim elektroničkim komponentama. Radni napon ploče iznosi 5V, dok preporučeni ulazni napon varira između 7 i 12V, a maksimalni raspon napajanja je od 6 do 20V. Arduino Uno Rev3 ima 14 digitalnih ulazno/izlaznih pinova, od kojih se 6 može koristiti za PWM izlaz. PWM (eng. „Pulse Width Modulation“) omogućuje generiranje analognih signala pomoću digitalnih impulsa, što je korisno za kontrolu jačine svjetla LED dioda ili brzine vrtnje motora. Ploča ima 6 analognih ulaznih pinova, koji omogućuju precizno očitavanje različitih senzorskih vrijednosti. Rev3 koristi 20 mA DC po svakom I/O pinu te 50 mA DC za 3.3V pin, što je dovoljno za većinu manjih elektroničkih projekata. Ploča ima Flash memoriju kapaciteta 32 KB, SRAM od 2 KB i EEPROM od 1 KB, dok je radni takt procesora 16 MHz. Ove specifikacije čine Arduino Uno Rev3 prikladnim za projekte koji ne zahtijevaju visoku procesorsku snagu i veliki

napon. Za povezivanje s računalom koristi se USB konektor, koji služi i za programiranje mikrokontrolera. Ako je potrebno dodatno napajanje, ploča podržava 9V bateriju, koja se može priključiti putem DC ulaznog priključka. Arduino Uno Rev3 podržava tri vrste komunikacijskih protokola: UART, I2C i SPI. (Banzi, M. i Shiloh, M. (2022) Getting Started with Arduino: The Open Source Electronics Prototyping Platform)

UART (eng. „Universal Asynchronous Receiver-Transmitter“) omogućuje serijsku komunikaciju kod koje se podaci prenose asinkrono, pri čemu se mogu konfigurirati brzina prijenosa i format podataka.

I2C (Inter-Integrated Circuit) je sinkroni komunikacijski protokol koji omogućuje povezivanje više uređaja koristeći samo dvije žice – jednu za podatke i jednu za takt. Ovaj protokol razvila je tvrtka Philips Semiconductors

SPI (eng. „Serial Peripheral Interface“) je sinkroni serijski komunikacijski protokol, koji se koristi za brzu razmjenu podataka između integriranih krugova na kratkim udaljenostima. Arduino Uno Rev3 predstavlja kompaktno i pouzdano rješenje za različite elektroničke projekte i sustave automatizacije, uključujući izradu meteoroloških stanica i drugih IoT rješenja. (Valvano, J. (2020) Embedded Systems: Introduction to ARM Cortex-M Microcontrollers. CreateSpace Independent Publishing)



Slika 1 - Arduino Uno Rev3 (izvor: fotografija autora)

3.2. Funkcionalnosti i primjene

Arduino Uno podržava velik broj funkcionalnosti, što ga čini korisnim za širok spektar projekata. U području elektroničkih projekata često se koristi za izradu LED svjetala, brojila ili alarma. Također se primjenjuje u automatizaciji, gdje omogućava upravljanje automatiziranim procesima u kućanstvu i industriji. Osim toga, Arduino Uno ima značajnu primjenu u senzorskim sustavima, gdje omogućava prikupljanje podataka o okolišu putem senzora za mjerenje temperature, vlage i tlaka zraka. U obrazovnim institucijama često se koristi kao alat za učenje elektronike i elektrotehnike, dok je među hobistima popularan za razvoj personaliziranih projekata, uključujući robote, pametne kućne uređaje i meteorološke stanice. Zahvaljujući jednostavnosti i svestranosti, Arduino se nametnuo kao jedna od najpopularnijih razvojnih platformi, omogućujući korisnicima svih razina znanja da realiziraju svoje ideje. U sljedećim poglavljima detaljno će biti opisano kako koristiti Arduino Uno pločicu za izgradnju osobne meteorološke stanice te na koji način integrirati različite senzore za prikupljanje meteoroloških podataka.

3.3. Integrirano razvojno okruženje (IDE)

Za korištenje Arduina potrebno je imati integrirano razvojno okruženje (eng. IDE – „Integrated development environment“) koje je moguće pronaći na Arduino platformi. IDE koristi programski jezik C++ uz neke dodatne promjene koje pomažu za bolji razvoj. Arduino IDE čini nekoliko prozora. Najbitniji su: prozor za programiranje, prozor za izlaz (eng. „Output“) i prozor „Serijski monitor“ (eng. „Serial monitor“). Najvažniji i prvi je prozor gdje se piše kod za program. Zatim output prozor gdje se izvodi program tj. kod. Serijski monitor omogućuje prikaz podataka dobivenih sa senzora, promjenjivost njihovog stanja iliti svako njihovo skeniranje okoline. Nakon instalacije Arduino razvojnog okruženja, moguće ga je pokrenuti po prvi puta. Nakon pokretanja programa, prikazuje se prozor za kompiliranje koda prikazan na slici 2.

3.3.1. Instalacija Arduino IDE-a

Arduino IDE (Integrated Development Environment) razvojno je okruženje namijenjeno programiranju i upravljanju Arduino mikrokontrolerima. Omogućuje pisanje, učitavanje i

upravljanje kodom za različite Arduino projekte te podržava više operativnih sustava, uključujući Windows, macOS i Linux. Za preuzimanje Arduino IDE-a potrebno je posjetiti službenu Arduino web stranicu i odabrati verziju prilagođenu operativnom sustavu. Na Windows operativnim sustavima instalacija se vrši pokretanjem preuzetog instalacijskog paketa i slijedeći upute prikazane na ekranu. Korisnici macOS-a trebaju preuzeti .zip datoteku, raspakirati je i premjestiti aplikaciju u mapu „Applications“. Na Linux sustavima instalacija se obavlja putem terminala pomoću naredbi specifičnih za distribuciju koja se koristi.

Slika 2 - Integrirano razvojno okruženje (izvor: slika autora)

Za Arduino okruženje potreban je Arduino uređaj, tako da je potrebno povezati uređaj na računalo, što je moguće pomoću konektora tipa USB 2.0 Cable Type A/B prikazanog na slici 3. Ovaj konektor glavni je izvor napajanja za Arduino uređaj, dok je sekundarni izvor napajanja moguće povezati na bačvastu DC utičnicu ili na eksperimentalnu pločicu koja će kasnije biti opisana. Ovaj konektor biti će korišten za serijsku komunikaciju između računala i Arduina, što je bitno zbog podataka poput lokacije, točnog vremena, promjene stanja, grešaka i sl.

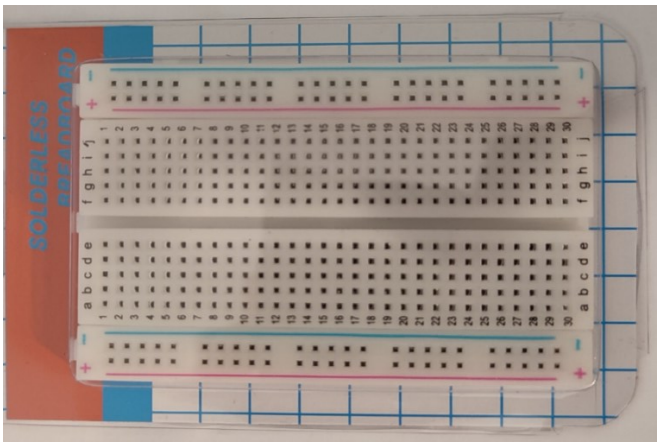
Slika 3 - USB 2.0 Cable Type A/B (izvor: fotografija autora)

4. Senzori i dodaci

Razvoj osobne meteorološke stanice zahtijeva korištenje različitih senzora koji mogu mjeriti ključne meteorološke parametre. U ovom poglavlju biti će opisani senzori koji će biti korišteni, uključujući njihove tehničke specifikacije, način rada i postupak povezivanja s Arduino Uno pločicom.

4.1. Eksperimentalna pločica

Eksperimentalna pločica (eng. „Solderless breadboard) prikazana na slici 4. je pomagalo za spajanje privremenih električkih strujnih krugova. Dolazi u raznim oblicima i dimenzijama, ali svaka pločica služi istoj svrsi. Pločica se sastoji od plastičnog kućišta, a na gornjoj strani se nalaze rupice namijenjene za umetanje različitih komponenti tj. njihovih nožica. Svaka rupica ima standardizirani međusobni razmak od 2.5 mm. Ovaj razmak je standardiziran jer većina komponenti koje se spaja u strujni krug ima ovaj razmak. Rupice na eksperimentalnoj pločici međusobno su povezane i to prema određenim pravilima. Razlikujemo tri osnovna dijela pločice koja su označena različitim bojama. Svaka pločica ima dva seta „tračnica“ koje služe za napajanje, a označene su crvenom i plavom bojom. Crvena boja je označena znakom plus (+), i ona se koristi za napon, dok je plava boja označena znakom minus (-), a ona služi za uzemljenje (GND). Po sredini pločice prolazi vertikalni rascjep koji dijeli dva strujna područja. Ovaj rascjep također je standardiziran te omogućava stvaranje integriranih krugova. Unutar sivog područja nalaze se rupice koje su spojene vodoravno, čineći u jednom redu dva vodoravna niza od po pet rupica. Pločica često može biti označena brojevima kod redova (1-60) i slovima kod stupaca (a-j). Glavni razlog za korištenje eksperimentalne pločice je izbjegavanje lemljenja koje bi uzrokovalo da električna veza više nije privremena nego stalna. Povezivanje komponenti na eksperimentalnu pločicu vrši se umetanjem nožica komponente te spajanjem električnim žicama na Arduino ili na plus i minus, ovisno o vrsti konekcije.



Slika 4 - Eksperimentalna pločica (izvor: fotografija autora)

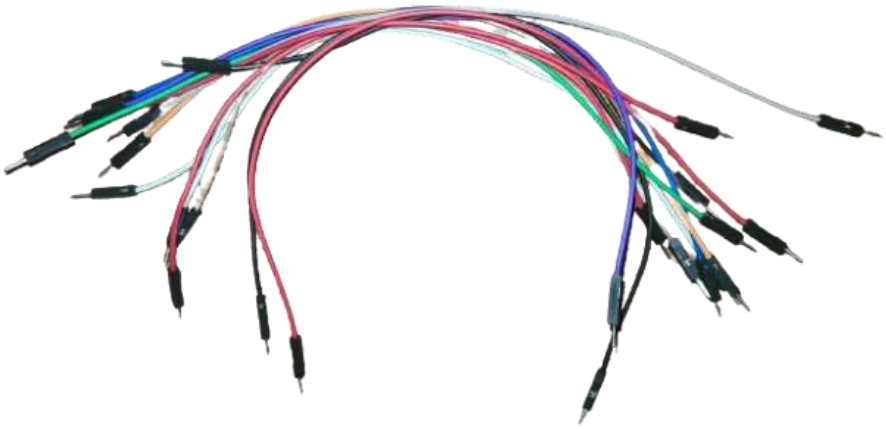
4.2. 9V baterija, adapter i jumper žice

Za ispravan rad meteorološke stanice preko Arduino platforme, potrebno je osigurati dodatno napajanje i odgovarajuće dodatke. U ovom dijelu opisana je svrha i način korištenja 9V baterije, adaptera za napajanje i jumper žica koje omogućuju povezivanje svih dijelova sustava. 9V baterija koristi se kao dodatni izvor napajanja za Arduino Uno razvojnu ploču. Budući da Arduino može raditi na više izvora napajanja (USB, adapter, baterija), 9V baterija u ovom slučaju pruža dodatan izvor napajanja za anemometar jer isti koristi 7-24V DC. Ova baterija spaja se na Arduino putem adaptera s priključkom za bačvasti DC konektor. Adapter za 9V bateriju omogućuje povezivanje 9V baterije s Arduino Uno pločicom. S jedne strane adapter ima priključak koji se spaja na bateriju, dok s druge strane ima standardni bačvasti DC konektor koji se može uključiti u ulaz za napajanje na Arduinu. Crvena žica predstavlja pozitivni (VCC), dok crna žica označava uzemljenje (GND)

4.2.1. Jumper žice

Jumper žice, prikazane na slici 5. koriste se za povezivanje elektroničkih komponenti na eksperimentalnoj pločici te za spajanje senzora s Arduino pločicom. Ove žice omogućuju brzo i jednostavno povezivanje komponenti bez potrebe za lemljenjem, što ih čini iznimno pogodnima za eksperimentalne projekte i razvoj prototipa. Postoji nekoliko vrsta jumper žica, ovisno o tipu konektora na njihovim krajevima. Muško-muško (MM, eng. "Male-Male") žice koriste se za povezivanje između dvaju muških konektora, dok se muško-

žensko (MF, eng. "Male-Female") varijanta koristi kada je potrebno spojiti muški konektor s komponentom koja ima ženski priključak. Žensko-ženske (FF, eng. "Female-Female") žice omogućuju povezivanje između dviju komponenti koje imaju muške konektore. U ovom projektu korištene su isključivo MM (muško-muško) jumper žice, budući da omogućuju povezivanje senzora i ostalih elektroničkih komponenti s Arduino pločicom putem eksperimentalne pločice. Njihova fleksibilnost i jednostavnost korištenja čine ih nezaobilaznim elementom u eksperimentalnim elektroničkim sustavima.



Slika 5 - Jumper žice (izvor: fotografija autora)

4.3. DHT22 senzor za temperaturu i vlagu

DHT22 je senzor za mjerenje temperature i vlage zraka. Sastoji se od tiskane pločice (eng. „Printed circuit board“) i bijelog kućišta koje štiti unutarnje komponente i omogućuje cirkulaciju zraka kroz perforacije. Unutar kućišta nalazi se kapacitivni senzor vlažnosti i termistor za mjerenje temperature. Za spajanje na eksperimentalnu pločicu koriste se tri izlazna pina. Prvi pin je napajanje - VCC (eng. „Voltage common collector“) od 3.3 do 6V. U sredini je signalni pin za komunikaciju s pločicom i posljednje je uzemljenje (GND). Čip unutar senzora prerađuje očitane podatke i šalje ih kao digitalni signal prema mikrokontroleru. Senzor je prikazan na slici 6. Mjerni opseg temperature iznosi od -40°C do +80°C+ uz preciznost temperature od ±0.5°C. Mjerni opseg vlage je 0-100% RH (eng. „Relative humidity) dok je njezina preciznost ±2-5% RH. Za korištenje je potrebno

instalirati odgovarajuću knjižnicu (npr. DHT knjižnicu) za jednostavnu integraciju. (Adafruit (2023) DHT22 Temperature-Humidity Sensor Guide)

Slika 6 - DHT22 senzor (izvor: fotografija autora)

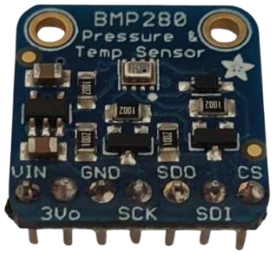
DHT22 senzor povezati će se vodoravno na tri različita retka. Tri jumper žice dovoljne duljine bit će odrezane i pripremljene za spajanje. Prvi kraj prve žice spojit će se u VIN pin redak na eksperimentalnoj pločici, dok će se drugi kraj spojiti na plus (+) stupac, na mjesto gdje je već povezano napajanje od 5V. Nova žica bit će korištena za povezivanje reda GND na eksperimentalnoj pločici s minus (-) stupcem, gdje je prethodno spojeno uzemljenje (GND). Posljednja žica bit će spojena na digitalni ulaz senzora DHT22, dok će se drugi kraj povezati na jedan od digitalnih pinova na Arduino pločici, primjerice D2. (Monk, S., 2018)

4.4. BMP280 senzor za tlak zraka

BMP280 senzor je za mjerenje atmosferskog tlaka i može dodatno pružiti mjerenje temperature. Sastoji se od tiskane pločice na kojoj se nalazi senzor za tlak i termistor, te 6 pinova za spajanje. Senzor je prikazan na slici 7. Mjerni opseg tlaka iznosi od 300 hPa do 1100 hPa, uz preciznost od ±1 hPa. Osim tlaka, senzor može mjeriti temperaturu u rasponu od -40°C do +85°C s preciznošću od ±1°C. BMP280 se napaja naponom između 1.71V i 3.6V, što ga čini kompatibilnim s različitim mikrokontrolerskim platformama. Senzor koristi I2C ili SPI komunikacijski protokol, što omogućuje povezivanje s mikrokontrolerima poput Arduina. Za povezivanje senzora putem I2C protokola, potrebno je spojiti SCK i SDI pinove na odgovarajuće ulaze mikrokontrolera. Pri korištenju SPI

protokola, dodatno se koriste CS i SDO pinovi. Napajanje senzora vrši se spajanjem pina VIN na 3.3V ili 5V, dok se GND povezuje na uzemljenje sustava. SCK pin se može spojiti na analogni ulaz mikrokontrolera, primjerice A5, dok se SDI povezuje na drugi analogni ulaz, poput A4. Za rad s BMP280 senzorom potrebno je koristiti odgovarajuću knjižnicu, poput Adafruit BMP280, koja omogućuje jednostavnu integraciju i očitavanje podataka putem I2C ili SPI protokola. (Lady Ada, Adafruit BMP280 Barometric Pressure + Temperature Sensor Breakout).

Za BMP280 senzor prati se postupak kao i za DHT22 senzor. VIN će se spojiti na (+) redak, GND na redak za uzemljenje (-). Pinove SCK i SDI spojiti će se na jedne od analognih pinova na Arduino pločici (npr. A4 i A5).



Slika 7 - BMP280 senzor (izvor: fotografija autora)

4.5. Anemometar za brzinu vjetra

Anemometar je senzor za mjerenje brzine vjetra koji radi na principu određivanja brzine vrtnje rotora s lopticama. S obzirom na način spajanja, postoji više izvedbi anemometara. Za potrebe mjerenja brzine vjetra osobne meteorološke stanice korišten je anemometar s impulsnim izlazom prikazan na slici 8. Odabrani anemometar zahtjeva dodatno napajanje istosmjernim naponom koji je osiguran iz baterijskog izvora od 9V. Ovo osigurava 7-24V istosmjerni napon (DC) , što je dovoljno za korištenje anemometra. Mjerni opseg anemometra kreće se od 0 do 32 m/s, a izlazni signal predstavlja niz impulsa, pri čemu je broj impulsa proporcionalan brzini vjetra. Senzor generira analogni impulsni signal svaki put kada se rotor okrene, što omogućuje precizno mjerenje brzine vjetra brojanjem impulsa u određenom vremenskom intervalu. Za povezivanje

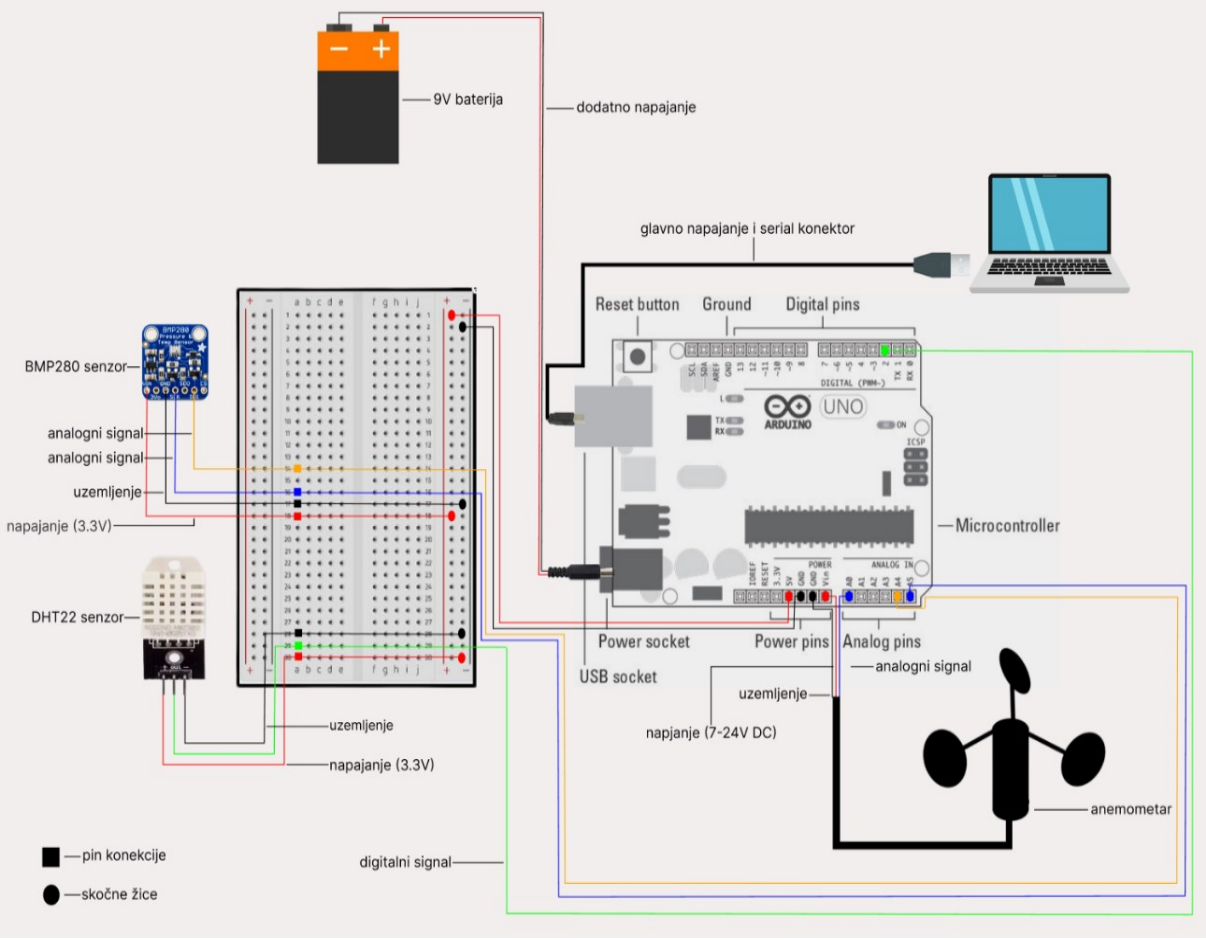
anemometra s Arduino pločicom potrebno je spojiti VCC pin na napon između 7V i 24V DC, pri čemu se koristi VIN pin na Arduino pločici, jer se na njega prenosi dodatno napajanje iz baterije od 9V. Uzemljenje senzora spaja se na odgovarajući GND pin na Arduino pločici. Signalni izlaz anemometra povezuje se na jedan od analognih pinova mikrokontrolera, primjerice A0, koji očitava generirane impulse i pretvara ih u brzinu vjetra. (Adafruit, Anemometer Wind Speed Sensor with Analog Voltage Output)

Slika 8 - Impulsni anemometar (izvor: fotografija autora)

Nakon što su spojeni senzore za temperaturu, vlagu i pritisak potrebno je priključiti anemometar koji je nešto kompliciraniji za spajanje. Postupak započinje povezivanjem žica na Arduino Uno. Anemometar dolazi s tri žice: napon, ulazni (eng. „Input“) i uzemljenje (GND). Najsigurniji način povezivanja ovih žica je povezivanje istih na dodatne jumper žice kako bi se osigurala stabilna konekcija. Za ovaj postupak najbolje je koristiti dodatne žice istih boja kako se ne bi pomiješale. Smeđa žica povezuje se na VIN pin na Arduinu, crna žica spaja se na GND pin radi uzemljenja, dok se plava ulazna žica (Input) povezuje na jedan od analognih ulaza, primjerice A0. Nakon što su žice uspješno povezane s Arduinom, preostaje povezivanje dodatnog izvora napajanja. Za napajanje se koristi 9V baterija, koja se povezuje s Arduinom pomoću odgovarajuće utičnice. Jedan dio priključka spaja se na bateriju, dok se drugi dio povezuje na bačvasti priključak na Arduinu, čime se osigurava dovod napona od 9V na VIN pin ploče. (Adafruit, Anemometer Wind Speed Sensor with Analog Voltage Output)

4.6. Strukturno povezivanje i blokovska shema sustava

Na kraju će se Arduino povezati s eksperimentalnom pločicom. Uzet će se dvije jumper žice te će se prvi kraj prve žice povezati na pin 5V na Arduinu, a drugi kraj žice u plus (+) na eksperimentalnoj pločici. Uzet će se druga žica te će se prvi kraj povezati u uzemljenje (GND) na Arduinu a drugi kraj u minus (-) na eksperimentalnoj pločici. Ovime će se osigurali da je eksperimentalna pločica pod naponom od 5V uz pripadajuće uzemljenje, što je korisno jer sada nije potrebno spajati žice za napon i uzemljenje sa senzora direktno u Arduino nego u plus i minus sekcije eksperimentalne pločice. S ovim postupkom osigurati će se više rupica za povezivanje jer ih je na Arduinu ograničeno. (Monk, S. (2018) Programming Arduino: Getting Started with Sketches. 2. izd. McGrawHill Education). Završni prikaz sheme integriranog sustava prikazan je na slici 9.



Slika 9 - Blokovska shema sustava (izvor: vlastiti rad)

4.7. Postavljanje okoline za rad

Nakon instalacije Arduino IDE-a, potrebno je postaviti okruženje za rad s senzorima i modulima. Za rad sa senzorima DHT22, BMP280 potrebno je instalirati odgovarajuće knjižnice. U Arduino IDE-u potrebno je ići u Skica > Uključi biblioteku > Upravljanje bibliotekama..., pretražiti i instalirati knjižnice „DHT sensor library by Adafruit“ i „Adafruit BMP280 Library“. Za anemometar nisu potrebne biblioteke, nego je potrebno dobro postaviti kod (Horowitz, P. i Hill, W. (2015)). Za DHT22 senzor potrebno je knjižnicu uključiti u kod dodavanjem linije za uključivanje DHT biblioteke na početak programa. Zatim je senzor potrebno inicijalizirati naredbom DHT dht(DHTPIN, DHTTYPE); gdje DHTPIN predstavlja pin na koji je spojen senzor, a DHTTYPE tip senzora (za DHT22 koristimo DHT22). Za BMP280 senzor knjižnica će se uključiti u kod dodavanjem linije za uključivanje BMP280 biblioteke na početak programa. Zatim će se inicijalizirati senzor putem linije Adafruit_BMP280 bmp; i unutar funkcije *za postavljanje (setup())* će se pozvati bmp.begin();

5. Kod za mjerenje i prikupljanje podataka

U ovom poglavlju biti će objašnjena funkcionalnost obje skripte za praćenje i obradu podataka sa senzora koje ćemo koristiti u svrhu izrade neuronske mreže za predikciju vremenskih uvjeta. Cjeloviti kod moguće je pronaći u Prilogu 1. Objasniti će se funkcionalnost koda korištenog za mjerenje i praćenje vremenskih uvjeta koristeći senzore DHT22, BMP280 i anemometar. Arduino koristi C++ programski jezik, te ga se programira kroz Arduino IDE. Kroz ovaj kod mogu biti prikupljeni podaci o temperaturi, vlažnosti, tlaku zraka i brzini vjetra. Kako bi se omogućilo korištenje različitih senzora i komunikacijskih protokola, potrebno je učitati nekoliko ključnih biblioteka. Ove biblioteke koriste se za rad s I2C komunikacijom, dohvaćanje podataka s BMP280 senzora tlaka, rad s DHT senzorom temperature i vlažnosti, te upravljanje ostalim komponentama sustava. Njihovom primjenom osigurava se ispravan rad senzora, obrada podataka i njihova kasnija upotreba u predikciji vremenskih uvjeta. Za pravilno funkcioniranje senzora, definirani su odgovarajući digitalni i analogni pinovi na razvojnoj pločici. DHT senzor temperature i vlažnosti povezan je na određeni digitalni pin, a njegov tip (DHT22) specificiran je kako bi se omogućilo pravilno očitavanje podataka. BMP280 senzor tlaka inicijaliziran je putem I2C komunikacije, koja omogućuje prijenos podataka između senzora i mikrokontrolera. Ova inicijalizacija omogućuje pravilnu komunikaciju mikrokontrolera sa senzorima i dohvaćanje podataka.

5.1. Definicija pinova i konstanti za anemometar

Anemometar se koristi za mjerenje brzine vjetra te radi na principu određivanja brzine vrtnje rotora. Kako bi se podaci s ovog senzora pravilno interpretirali, potrebno je definirati način povezivanja i obrade očitanih vrijednosti. Analogni pin koristi se za povezivanje anemometra s mikrokontrolerom, čime se omogućuje prijenos električnih signala u sustav. Vrijednosti očitanog napona obrađuju se kako bi se omogućila njihova konverzija u brzinu vjetra. Korištenjem odgovarajućih konstanti definiraju se pravila za pretvaranje analognih očitanja u napon te način na koji se taj napon transformira u odgovarajuće vrijednosti brzine vjetra. Minimalne i maksimalne vrijednosti napona povezuju se s

odgovarajućim brzinama vjetra, čime se osigurava skaliranje očitanih podataka. Ove postavke omogućuju pravilnu interpretaciju analognih signala te njihovu konverziju u meteorološke vrijednosti.

5.2. Inicijalizacija sustava i očitavanje podataka

Prilikom pokretanja sustava provodi se inicijalizacija serijske komunikacije i senzora te se provjerava njihova dostupnost. Serijska komunikacija omogućuje ispis podataka na računalo radi praćenja rada sustava. Senzori DHT22 i BMP280 inicijaliziraju se kako bi se omogućilo mjerenje temperature, vlažnosti i tlaka zraka. U slučaju nedostupnosti senzora, sustav prikazuje poruku o grešci kako bi korisnik mogao poduzeti odgovarajuće radnje. Ovaj postupak osigurava ispravan rad sustava i spremnost svih senzora za dohvaćanje podataka. Nakon inicijalizacije, glavni programski tijek kontinuirano dohvaća i obrađuje podatke sa senzora. DHT22 senzor mjeri temperaturu i vlažnost zraka te, u slučaju neuspješnog očitanja, prikazuje poruku o grešci. BMP280 senzor očitava atmosferski tlak i radi pretvorbu u hektopaskale. Svi očitani podaci ispisuju se na serijski monitor, čime se omogućuje njihovo praćenje u stvarnom vremenu. Mjerenje brzine vjetra provodi se pomoću anemometra, pri čemu se analogna vrijednost senzora očitava i pretvara u odgovarajući napon. Ako je izmjereni napon ispod minimalnog praga, brzina vjetra postavlja se na nulu. Na temelju izmjerenog napona, koristeći unaprijed definirane parametre kalibracije, provodi se izračun brzine vjetra. Nakon završetka izračuna, podaci o izmjerenom naponu i brzini vjetra ispisuju se kako bi bili dostupni za analizu. Kako bi se izbjegla prečesta očitanja i osigurala relevantnost podataka, vrijeme čekanja između svake iteracije mjerenja postavljeno je na 1000ms. Sustav čeka određeno vrijeme prije nego što ponovno izvrši očitanja sa svih senzora. Ovaj interval omogućuje prikupljanje stabilnih i reprezentativnih meteoroloških podataka, smanjujući fluktuacije koje mogu nastati uslijed kratkotrajnih promjena u okolini. Tako se osigurava da podaci budu što precizniji i korisniji za daljnju analizu i predikciju vremenskih uvjeta.

5.3. Prikupljanje, obrada i pohrana vremenskih podataka

U ovom poglavlju biti će objašnjen način prikupljanja meteoroloških podataka iz senzora i OpenWeather API-ja (eng. „Application programming interface“) te njihova obrada i

pohrana u standardizirani format. API označava programsko sučelje koje omogućuje komunikaciju između različitih softverskih sustava. Putem API-ja omogućuje se razmjena podataka između aplikacija bez potrebe za direktnim pristupom izvornom kodu ili bazi podataka. Python skripta služi za prikupljanje podataka koji će se koristiti za izradu neuronske mreže za predikciju kratkoročnih vremenskih uvjeta. Sustav prikuplja podatke iz dvaju glavnih izvora. Prvi izvor jesu senzori povezani s Arduinom tj. fizički senzori koji bilježe temperaturu, vlažnost, tlak zraka i brzinu vjetra. Drugi izvor je *OpenWeather API* koji pruža dodatne meteorološke informacije poput postotka oblaka, smjera vjetra i opisa vremenskih uvjeta.

Podaci iz oba izvora pohranjuju se u CSV datoteku u vremenskom intervalu od jedan sat. Prikupljanje podataka odvija se u beskonačnoj petlji, gdje sustav u određenim vremenskim razmacima očitava vrijednosti i ažurira bazu podataka. Kod za prikupljanje podataka moguće je pronaći u Prilogu 2.

5.3.1. Normalizacija i enkodiranje podataka

Kako bi podaci bili pripremljeni za obradu neuronskom mrežom, prolaze kroz proces enkodiranja, normalizacije i transformacije varijabli. Kategorijske varijable, poput vremenskih uvjeta (*"Vedro"*, *"Oblaci"*, *"Kiša"*) i godišnjih doba (*"Proljeće"*, *"Ljeto"*), zamjenjuju se numeričkim vrijednostima, čime se omogućuje njihova obrada u modelu.

Kontinuirane varijable, uključujući temperaturu, tlak i vlažnost zraka, skaliraju se u raspon [0,1] kako bi neuronska mreža mogla učinkovito raditi s podacima. Time se osigurava stabilnost treniranja modela i izbjegavanje dominacije pojedinih varijabli zbog razlika u njihovim apsolutnim vrijednostima. Smjer vjetra transformira se u dvije vektorske komponente, označene kao wind_x i wind_y, umjesto izražavanja u stupnjevima. Ova metoda osigurava kontinuitet podataka te sprječava probleme koji nastaju pri prijelazu između 359° i 0°. Za normalizaciju kontinuiranih podataka koristi se min-max skaliranje, pri čemu se svaka vrijednost transformira prema formuli: $Xnorm = \frac{X - Xmin}{Xmax - Xmin}$. Ova metoda koristi se za skaliranje kontinuiranih varijabli u određeni raspon, najčešće između 0 i 1, kako bi se osigurala ujednačenost podataka i poboljšala stabilnost modela. X

označava izvornu vrijednost podataka kojeg se normalizira. $Xmin$ označava najmanju vrijednost u skupu podataka (minimalnu vrijednost varijable). $Xmax$ je najveća vrijednost u skupu podataka (maksimalna vrijednost varijable), dok je $Xnorm$ normalizirana vrijednost nakon primjene min-max skaliranja. Normalizacija funkcionira tako da se oduzmu minimalne vrijednosti ($X - Xmin$) čime se vrijednosti pomiču tako da minimalna vrijednost u skupu postane nula i zatim se dijeli s rasponom ($Xmax - Xmin$) kako bi se podaci skalirali da maksimalna vrijednost postane 1, a sve ostale vrijednosti se smještaju unutar raspona [0,1].

Za normalizaciju podataka korišteni su sljedeći normalizacijski faktori prikazani na slici 10. Korištene su vrijednosti koje najbolje odgovaraju stvarnim uvjetima. Na primjer, vlažnost se kreće od 23% do 100%. Kada je vlažnost niska znači da je zrak jako suh i da je mogućnost padalina jako niska. Naprotiv, kada je vlažnost jako visoka, velika je mogućnost padalina. Za temperaturu se koriste realne granice za mjesto Vrbnik na otoku Krku. Temperatura rijetko prelazi ispod ništice no -10°C je realna donja granica, dok je +40°C realna gornja granica.

NORMALIZACIJSKI FAKTORI	
vlažnost	(23,100)
temperatura	(-10, 40)
tlak zraka	(989, 1037)
osjet	(-10,40)
minimalna danja temperatura	(0.00, 19.55)
maksimalna danja temperatura	(-10, 40)
prosječna vlažnost	(23,100)
promjena tlaka	(-10,10)

Slika 10 - Normalizacijski faktori (izvor: vlastiti rad)

5.3.2. Priprema podataka za analizu

Unutar skripte automatski se izračunavaju dodatni meteorološki parametri kako bi se omogućila preciznija analiza vremenskih uvjeta. Sezona se određuje na temelju dana u

godini, čime se omogućava kategorizacija podataka prema godišnjim dobima. Duljina dana računa se uzimajući u obzir zemljopisnu širinu lokacije, čime se osigurava točno određivanje trajanja dnevnog svjetla za svaki dan u godini. Osjet temperature izračunava se uzimanjem u obzir stvarne temperature zraka, vlažnosti i brzine vjetra, čime se dobiva subjektivni osjećaj topline. Brzina vjetra konvertira se u odgovarajuću kategoriju prema Beaufortovoj ljestvici, što omogućuje klasifikaciju jakosti vjetra u razumljivijem obliku. Ljestvica je prikazana na slici 11. Svaki izračunati podatak označava se vremenskom oznakom u UTC formatu, čime se osigurava preciznost i dosljednost podataka prilikom kasnije analize.

Stupanj	m/s	km/h	čv	Opis vjetra
0	0-1	0-1	0-1	tišina
1	2	2-5	2-3	lahor
2	3-4	6-11	4-6	povjetarac
3	5-6	12-19	7-10	slab vjetar
4	7-8	20-28	11-15	umjeren vjetar
5	9-11	29-38	16-21	umjereno jak vjetar
6	12-14	39-49	22-27	jak vjetar
7	15-17	50-61	28-33	vrlo jak vjetar
8	18-21	62-74	34-40	olujni vjetar
9	22-24	75-88	41-47	oluja
10	25-28	89-102	48-55	žestoka oluja
11	29-32	103-117	56-63	orkanska oluja
12	>32	>118	>63	orkan

Slika 11 - Beaufortova ljestvica jačine vjetra (izvor: vlastiti rad)

Sustav kontinuirano provjerava postoje li podaci s istom vremenskom oznakom u CSV datoteci kako bi se izbjeglo dupliciranje. Prvo se dohvaća trenutni vremenski pečat. Zatim se provjerava postoji li zapis s istim vremenom u bazi podataka. Ako podatak već postoji,

preskače se unos. Ako podatak ne postoji, dodaje se u CSV datoteku. Ova provjera optimizira proces pohrane i sprječava nepotrebno gomilanje istih vrijednosti.

5.3.3 Pohrana podataka u CSV format

Nakon što su svi podaci prikupljeni, normalizirani i provjereni, pohranjuju se u CSV datoteku, unutar koje se nalaze različiti stupci s relevantnim informacijama. Vremenska oznaka zapisana je u UTC formatu pod oznakom datum, dok se podaci sa senzora pohranjuju u stupcima vlažnost, temp i tlak. Dodatni podaci dohvaćeni iz OpenWeather API-ja uključuju osjet, oblaci, brzinu_vjetra, transformaciju u Beaufortovu ljestvicu i smjer_vjetra. Kategorizirani vremenski uvjeti zabilježeni su u stupcima vrijeme i vrijeme_opis, dok se sezonske varijable, poput godišnjeg doba i duljine dana, spremaju pod oznakama sezona i trajanje_dana Također, izračunava se razlika u tlaku u odnosu na referentnu vrijednost i pohranjuje u stupcu promjena_tlaka.

Podaci se dodaju u CSV datoteku jedino ako nisu zapisani. Ovime se osigurava efikasna organizacija i priprema za daljnju analizu. Skripta omogućuje automatizirano prikupljanje i obradu meteoroloških podataka, pri čemu se normalizirani i obrađeni podaci pohranjuju u CSV formatu. Na taj način osigurava se njihova spremnost za analizu i treniranje neuronske mreže za vremensku prognozu. Ovim sustavom postiže se konzistentnost podataka, smanjenje dupliciranja i optimizacija obrade, čime se omogućuje preciznije i učinkovitije predviđanje budućih vremenskih uvjeta.

Python skripta koristi se za obradu i pohranu podataka u CSV formatu kako bi se osigurala centralizirana organizacija informacija. Uz podatke prikupljene putem Arduina, koriste se i podaci s OpenWeather API-ja, dok dodatne formule omogućuju daljnju analizu i vizualizaciju rezultata. Primjenom pomičnih prosjeka, podaci se dodatno filtriraju kako bi se uklonile kratkotrajne varijacije i omogućilo jasnije praćenje dugoročnih trendova vremenskih uvjeta.

Prikazana je Tablica 1, koja sadrži usporedbu originalnih i normaliziranih podataka. Uzorak iz podataka iskorišten je kako bi se omogućila kratka analiza razlika između sirovih i obrađenih vrijednosti.

Tablica 1 - Usporedba originalnih i normaliziranih podataka (izvor: vlastiti rad)

datum	vlažnost	temp	tlak	osjet	oblaci	brzina_vjetra	vjetar_bf	smjer_vjetra
2024-02-29 23:00:00 +0000 UTC	79	13.01	1009	12.43	89	2.71	2	89
2024-03-01 00:00:00 +0000 UTC	77	13.57	1009	12.99	92	3.45	3	102
2024-03-01 01:00:00 +0000 UTC	80	13.16	1009	12.62	100	2.41	2	81
2024-03-01 02:00:00 +0000 UTC	83	12.48	1008	11.95	100	2.91	2	40
2024-03-01 03:00:00 +0000 UTC	85	12.48	1007	12	100	3.75	3	34
2024-03-01 04:00:00 +0000 UTC	91	11.7	1006	11.3	100	3.18	2	24
2024-03-01 05:00:00 +0000 UTC	94	11.31	1005	10.95	100	3.18	2	30
2024-03-01 06:00:00 +0000 UTC	95	11.01	1006	10.65	100	2.72	2	80
2024-03-01 07:00:00 +0000 UTC	95	11.01	1007	10.65	100	4.96	3	153
2024-03-01 08:00:00 +0000 UTC	95	11.4	1007	11.08	100	2.57	2	108
2024-03-01 09:00:00 +0000 UTC	92	12.46	1007	12.16	100	3.14	2	35
2024-03-01 10:00:00 +0000 UTC	86	13.57	1007	13.23	100	1.59	2	40
2024-03-01 11:00:00 +0000 UTC	80	14.72	1007	14.34	100	0.6	1	7

6. Neuronska mreža za predikciju vremenskih uvjeta

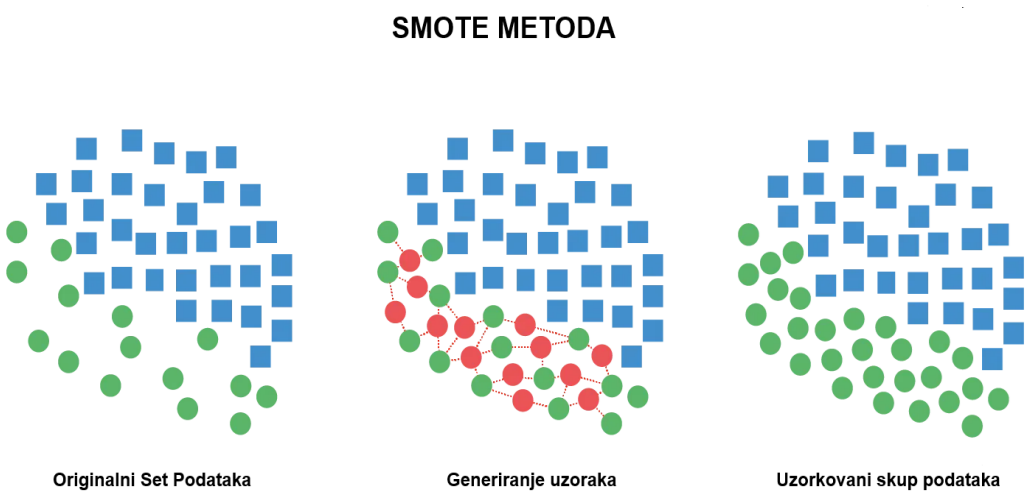
Za kratkoročnu predikciju vremena prvo je potrebno napraviti dobar model pomoću kojega ćemo kasnije testirati neuronsku mrežu na podacima u stvarnom vremenu. Kako bi postigli dobar rezultat s visokom točnosti, potrebno je imati dovoljan broj podataka. Podaci su prikupljani u razdoblju od kraja veljače 2024. godine do sredine siječnja 2025. godine, čime je osigurana dovoljna raznolikost u podacima jer se vrijeme mijenja kroz godišnja doba, no i dalje će postojati nedostaci, budući da je u mjestu Vrbnik uglavnom vedro i velik je broj sunčanih sati, što znači da će određene klase prevladavati. Daljnji fokus bit će usmjeren na definiranje ciljeva neuronske mreže, detaljno objašnjenje korištenih podataka, analizu pokušaja izgradnje modela te konačni model koji omogućuje visoku točnost u predikciji stvarnih podataka.

Razvoj neuronske mreže za predikciju vremenskih uvjeta imao je jasan cilj – izgraditi model sposoban za precizno predviđanje vremenskih uvjeta u rasponu od 6 do 24 sata unaprijed. Kako bi se postigao ovaj cilj, bilo je potrebno osigurati kvalitetne podatke, pripremiti ih za obradu te konstruirati model koji može prepoznati vremenske obrasce i ovisnosti unutar podataka. Cilj je dakle bio naučiti mrežu na podatke iz prošlosti te primijeniti to znanje na buduće predikcije. Umjesto učenja modela na real-time podacima, odlučeno je koristiti višemjesečne povijesne podatke zbog nekoliko razloga. Prvi razlog je stabilnost u treniranju zato što višemjesečni podaci omogućuju bolju generalizaciju modela i jer obuhvaćaju širok spektar vremenskih uvjeta. Slijedeći razlog jesu dugoročni obrasci zbog toga što vremenski uvjeti pokazuju sezonske varijacije koje je potrebno prepoznati i uključiti u model. Posljednji razlog je smanjenje smetnji u podacima kako bi smanjili podložnost kratkotrajnim anomalijama koje otežavaju treniranje modela.

6.1. Ključni izazovi kod vremenskih podataka

Rad s vremenskim podacima nosi nekoliko izazova koji su morali biti riješiti tijekom razvoja modela. Prvi izazov je to što su prikupljeni podaci vremenski zavisni, što je zahtijevalo implementaciju modela koji može razumjeti tu zavisnost. Za to je bilo potrebno koristiti model sekvencijalne prirode. Slijedeći problem je to što se vremenski uvjeti

mijenjaju u ovisnosti o sezoni, dobu dana i drugim faktorima. Stoga je potrebno obraditi podatke tako da model može prepoznati ove obrasce. Ključno je i balansirati podatke jer su određeni vremenski uvjeti znatno učestaliji od drugih (npr. oblačno vrijeme pojavljuje se češće od snijega). Korištene su metode balansiranja podataka poput SMOTE kako bi se osiguralo da model pravilno nauči sve klase. SMOTE (eng. „Synthetic Minority Oversampling Technique“) metoda prikazana na slici 12. je metoda pomoću koje se generiranju sintetički uzorci za manje zastupljene vremenske uvjete kako bi se osiguralo ravnomjernije učenje modela (npr. jako rijetko je klasifikacija vremena bila „grmljavinska oluja s kišom“. (Alex, S.A. (2025) 'Imbalanced data learning using SMOTE and deep learning architecture with optimized features', *Neural Computing & Applications)*



Slika 12 - SMOTE metoda za balansiranje podataka (izvor: https://emilia-orellana44.medium.com/smote-2acd5dd09948)

6.2. Priprema podataka

Priprema podataka prvi je i ključni korak u razvoju neuronske mreže, jer kvalitetno pripremljeni podaci omogućuju modelu da uči po obrascima i generalizira predikcije na stvarne vremenske uvjete. Podaci dolaze iz dva izvora. Najveći dio podataka dolazi putem meteorološke stanice (Arduino senzori) gdje se prikupljaju podatci o temperaturi, tlaku, vlažnosti, brzini i smjeru vjetra u realnom vremenu. Ostali podaci dolaze s

OpenWeather API-a, a to su povijesni vremenski podaci koji sadrže detaljne informacije o atmosferskim uvjetima, prikupljeni istom skriptom kao i za meteorološku stanicu
Kako bi podaci bili u ispravnom formatu za treniranje modela, provodilo se pretprocesiranje. Normalizacijom podataka su svi numerički podaci skalirani korištenjem min-max skaliranja kako bi se osigurala ujednačenost i brža konvergencija modela. Ovi podaci skalirani su po realnim vrijednostima za mjesto Vrbnik, Krk (npr. temperatura od -10°C do +40°C). Provela se transformacija smjera vjetra gdje se smjer vjetra (wind_deg) konvertira u njegove komponente wind_x i wind_y koristeći sinus i kosinus funkcije. Završno su dodani atributi poput dana u godini (eng. „Day of year“), sata u danu (eng. „Hour“) te trajanja dana (eng. „Day length“) kako bi model mogao učiti sezonske obrasce. Budući da su vremenski podaci sekvencijski, potrebno je kreirati vremenske sekvence kao ulazne podatke za model. Koristiti će se ulazne sekvence od 24h, 48h i 72h kako bi modeli bili testirani s različitim duljinama sekvenci te da bi se pronašla optimalna duljina za predikciju budućih uvjeta.

Jedan od ključnih izazova je neravnomjerna distribucija vremenskih uvjeta. Korištena je SMOTE metoda i metoda filtriranja rijetkih klasa gdje će vremenski uvjeti koji su se pojavili manje od određenog broja puta biti grupirani u kategoriju "Ostalo" kako bi se poboljšalo treniranje modela.

6.3. Teorijske osnove i ključni koncepti modela

U ovom poglavlju bit će objašnjene temeljne teorijske osnove i ključni koncepti koji se koriste pri izradi neuronske mreže. Svaki od ovih pojmova predstavlja neophodan faktor u radu modela te utječe na njegovu preciznost i sposobnost generalizacije podataka.

Prekomjerno prilagođavanje

Prekomjerno prilagođavanje (eng. „Overfitting“) pojava je u strojnom učenju pri kojoj model postaje previše složen i ne samo da prepoznaje osnovne obrasce u podacima, već i šum te slučajne fluktuacije. Kao rezultat, model postiže iznimno dobre rezultate na podacima za treniranje, ali pokazuje slabiju sposobnost generalizacije na nove,

prethodno neviđene podatke. Drugim riječima, model se previše prilagođava specifičnim podacima na kojima je treniran, što smanjuje njegovu praktičnu upotrebljivost. Overfitting se najčešće pojavljuje kada je model previše složen u odnosu na količinu i kvalitetu dostupnih podataka. Primjerice, ako neuronska mreža ima previše slojeva ili parametara, može doći do memoriranja podataka umjesto učenja generalnih obrazaca. Ova pojava je češća u slučajevima kada je skup podataka za treniranje malen ili kada podaci sadrže visok udio šuma. (Goodfellow, I., Bengio, Y. i Courville, A., 2016)

Overfitting se prepoznaje po visokoj točnosti na skupu za treniranje, dok su performanse na skupu za validaciju ili testiranje značajno slabije. Ako model postiže gotovo savršene rezultate na podacima za treniranje, ali pokazuje nisku točnost na novim podacima, može se zaključiti da je došlo do prekomjernog prilagođavanja. Dodatni pokazatelj overfittinga je prekomjerna složenost modela – ako neuronska mreža sadrži previše slojeva ili neurona, povećava se vjerojatnost nastanka ovog problema. Kako bi se smanjio rizik od overfittinga, koriste se različite tehnike prilagodbe modela. Regularizacija dodaje penalizaciju velikim vrijednostima težina unutar modela, čime se sprječava njihovo nekontrolirano povećanje. Najčešće korištene metode su L1 i L2 regularizacija. Još jedna učinkovita metoda je nasumično gašenje neurona (eng. „Dropout“), pri čemu se određeni postotak neurona isključuje tijekom treniranja, čime se osigurava da model ne postane previše ovisan o određenim značajkama. Povećanje skupa podataka također doprinosi smanjenju overfittinga, budući da veći broj podataka omogućuje modelu prepoznavanje općenitijih obrazaca umjesto memoriranja pojedinačnih primjera. Smanjenje složenosti modela, odnosno smanjenje broja slojeva ili neurona, često se primjenjuje kako bi se umanjio rizik od prekomjernog prilagođavanja podacima. Konačno, rano zaustavljanje (eng. „Early stopping“) omogućuje prekid treniranja modela u trenutku kada performanse na validacijskom skupu prestanu rasti, čime se sprječava da model nauči irelevantne obrasce prisutne u podacima za trening.

Podprilagođavanje

Podprilagođavanje (eng. „Underfitting“) se pojavljuje u strojnom učenju kada model nije dovoljno složen da bi prepoznao osnovne obrasce u podacima. Kao rezultat, model ne postiže zadovoljavajuće performanse ni na podacima za treniranje niti na novim, nepoznatim podacima. Ova pojava suprotna je od prekomjernog prilagođavanja, pri kojem model postaje previše složen i uči podatke za treniranje umjesto da prepoznaje opće zakonitosti. (Goodfellow, I., Bengio, Y. i Courville, A., 2016)

Podprilagođavanje se najčešće pojavljuje kada je model previše jednostavan u odnosu na složenost podataka. Ova pojava može nastati zbog nekoliko razloga. Ako neuronska mreža sadrži premalo slojeva ili neurona, ne može obraditi složenije obrasce u podacima. Nedostatak dovoljne količine podataka za treniranje također može uzrokovati podprilagođavanje, budući da model nema dovoljno informacija za prepoznavanje relevantnih značajki. Isto tako, ako je regularizacija previše restriktivna, model može postati previše ograničen te neće biti sposoban naučiti potrebne obrasce. Također, ako je odabrana neprikladna arhitektura modela za zadani problem, poput korištenja linearne regresije za obradu nelinearnih podataka, model neće moći ostvariti odgovarajuće performanse.

Podprilagođavanje se prepoznaje po niskoj točnosti na skupu za treniranje, što ukazuje na to da model ne uspijeva naučiti pravilne odnose između ulaznih podataka i izlaza. Osim toga, model pokazuje nisku točnost i na skupu za validaciju ili testiranje, budući da nije uspio naučiti osnovne obrasce iz podataka. Ako je model izuzetno jednostavan, primjerice sadrži samo jedan sloj neurona, povećava se vjerojatnost podprilagođavanja. Kako bi se smanjio rizik od podprilagođavanja, koriste se različite metode poboljšanja modela. Povećanjem složenosti modela, dodavanjem većeg broja slojeva ili neurona, omogućava se bolja obrada složenih obrazaca u podacima. Korištenjem većeg skupa podataka za treniranje model dobiva više informacija, čime se poboljšava sposobnost prepoznavanja relevantnih značajki. Smanjenjem snage regularizacije, poput smanjenja vrijednosti parametra λλ u L2 regularizaciji, omogućuje se veća fleksibilnost modela, čime se poboljšava njegova sposobnost učenja obrazaca. Pravilnim odabirom arhitekture

modela, primjerice korištenjem dubokih neuronskih mreža za složene podatke, osigurava se bolja prilagodba podacima. (Goodfellow, I., Bengio, Y. i Courville, A., 2016)

Ponekad je za poboljšanje modela potrebno dulje treniranje. Produženjem broja epoha omogućuje se bolja prilagodba modela, jer mu se daje dovoljno vremena da prepozna složenije odnose među podacima. Ako bi LSTM model sadržavao samo jedan LSTM sloj s vrlo malim brojem neurona, vjerojatno bi došlo do podprilagođavanja. Ova pojava bila bi vidljiva kao niska točnost i na skupu za treniranje i na skupu za validaciju. Kako bi se to spriječilo, moglo bi se povećati broj slojeva u mreži ili povećati broj neurona unutar postojećih slojeva.

Nasumično gašenje neurona (eng. „Dropout“)
Dropout je tehnika regularizacije koja se koristi u neuronskim mrežama kako bi se spriječilo overfitting. Tijekom treniranja određeni postotak neurona u sloju nasumično se isključuje, odnosno postavlja na nulu. Time se onemogućuje sudjelovanje tih neurona u propagaciji unaprijed (eng. „Forward pass“) i propagaciji unatrag (eng. „Backward pass“) tijekom te iteracije treniranja.

Dropout funkcionira na dva načina. Tijekom svake iteracije treniranja, svaki neuron ima određenu vjerojatnost da bude isključen, primjerice 0.3 ili 30%. Na taj način mreža se prisiljava da ne postane previše ovisna o pojedinačnim neuronima, već da uči robusnije obrasce. Tijekom testiranja svi neuroni ostaju aktivni, ali se njihovi izlazi skaliraju s vjerojatnošću dropout-a kako bi se nadoknadila činjenica da su tijekom treniranja neki neuroni bili isključeni. (Goodfellow, I., Bengio, Y. i Courville, A., 2016)

Primjenom dropout-a postiže se nekoliko važnih prednosti. Overfitting se smanjuje, budući da model ne postaje previše ovisan o pojedinačnim neuronima, čime se smanjuje rizik od prekomjernog prilagođavanja podacima. Generalizacija modela se poboljšava, jer model postaje robusniji i bolje se prilagođava novim podacima. Također, dropout djeluje kao ansambl metoda, jer se može smatrati vrstom metode u kojoj se trenira više manjih mreža koje zajedno daju bolje rezultate.

U LSTM modelu, dropout je primijenjen nakon svakog LSTM sloja s postotkom od 0.3 (30%). Tijekom treniranja, 30% neurona nasumično je isključeno u svakoj iteraciji, čime se smanjio rizik od overfittinga i poboljšala sposobnost modela da generalizira podatke.

Slojevi

Sloj u neuronskoj mreži je skup neurona koji primaju ulazne podatke, obavljaju izračune i prosljeđuju rezultate sljedećem sloju. U potpuno povezanim slojevima, neuroni unutar sloja povezani su s neuronima u prethodnom i sljedećem sloju.

Ulazni sloj prvi je sloj u mreži kojemu se prosljeđuju ulazni podaci, pri čemu broj neurona u ovom sloju obično odgovara broju značajki u podacima. Skriveni slojevi smješteni su između ulaznog i izlaznog sloja te obavljaju većinu izračuna u mreži. Njihov broj, kao i broj neurona u svakom sloju, određuje složenost modela. Izlazni sloj predstavlja posljednji sloj u mreži, čiji broj neurona ovisi o vrsti zadatka, primjerice klasifikaciji ili regresiji. Slojevi imaju ključnu ulogu u određivanju složenosti modela, budući da veći broj slojeva i neurona omogućuje modelu učenje složenijih obrazaca, ali istovremeno povećava rizik od overfittinga. Također, neuronska mreža omogućuje hijerarhijsko učenje, pri čemu se u početnim slojevima uče jednostavniji obrasci, poput rubova, dok se u dubljim slojevima identificiraju složeniji obrasci, poput objekata u konvolucijskim mrežama. (Goodfellow, I., Bengio, Y. i Courville, A. (2016))

L2 regularizacija i Batch normalizacija

L2 regularizacija (također poznata kao Ridge regularizacija) je tehnika koja se koristi za sprječavanje prenaučavanja dodavanjem penalizacije za velike vrijednosti težina u modelu. Ovom metodom kvadratne vrijednosti težina dodaju se u funkciju gubitka, čime se osigurava da težine ostanu što je moguće manje. (Goodfellow, I., Bengio, Y. i Courville, A. (2016))

L2 regularizacija dodaje dodatni član u funkciju gubitka prema izrazu: $Gubitak = Originalni\ gubitak + \lambda \sum_i w_i^2$ gdje λ predstavlja hiperparametar koji kontrolira snagu regularizacije, dok w_i označava težine modela.

Primjenom ove metode smanjuje se overfitting, jer penalizacija velikih težina sprječava prekomjernu složenost modela. Ovime model postaje robusniji i bolje generalizira na nove podatke. Također, težine modela postaju manje osjetljive na šum prisutan u podacima, čime se postiže stabilnost modela.

Batch normalizacija (eng. „BatchNormalization“) tehnika je koja se koristi u dubokom učenju radi stabilizacije i ubrzavanja treniranja neuronskih mreža. Ovom metodom normaliziraju se ulazne vrijednosti svakog sloja tako da njihova srednja vrijednost bude blizu 0, a standardna devijacija blizu 1. Normalizacija se primjenjuje na svaku mini-seriju (batch) podataka tijekom treniranja, što je i razlog naziva metode.
U procesu normalizacije za svaki mini-batch izračunava se srednja vrijednost i varijanca, nakon čega se podaci prilagođavaju tako da imaju srednju vrijednost 0 i standardnu devijaciju 1. Nakon toga, podaci se skaliraju i pomiču pomoću dvaju naučenih parametara, gamma i beta, kako bi se očuvala sposobnost modela da uči složene funkcije. Korištenjem batch normalizacije ubrzava se treniranje, budući da normalizacija podataka u svakom sloju sprječava probleme s gradijentima, što dovodi do stabilnijeg procesa učenja. Također, smanjuje se ovisnost modela o početnim vrijednostima težina i hiperparametrima, čime se poboljšava stabilnost modela. Osim toga, batch normalizacija može smanjiti overfitting, iako to nije njezina primarna svrha, dok model postaje otporniji na veće stope učenja, omogućujući brže postizanje optimalnih rezultata. Batch normalizacija se obično primjenjuje između slojeva neuronske mreže, osobito u dubokim konvolucijskim mrežama (CNN) i LSTM mrežama. Uobičajeno je dodati normalizaciju nakon Dense sloja i prije aktivacijske funkcije. U LSTM modelu, batch normalizacija korištena je nakon Dense sloja sa 16 neurona i prije konačnog izlaznog sloja, što je omogućilo stabilizaciju i ubrzanje treniranja, osobito u radu s podacima koji imaju velike varijacije u skalama. (Goodfellow, I., Bengio, Y. i Courville, A. (2016))

Adam optimizer

Adam (Adaptive Moment Estimation) predstavlja optimizacijski algoritam koji kombinira prednosti metoda AdaGrad i RMSProp. Ovim algoritmom omogućuje se prilagodba stope učenja za svaki parametar modela, što ga čini vrlo učinkovitim za širok raspon problema. Adam koristi eksponencijalno ponderirane prosjeke gradijenata (prvi moment) i kvadrat gradijenata (drugi moment) kako bi prilagodio stopu učenja, što omogućuje bržu konvergenciju i stabilnije treniranje.

Ovaj algoritam koristan je iz nekoliko razloga. Prilagodljiva stopa učenja omogućuje automatsko podešavanje vrijednosti za svaki parametar, smanjujući potrebu za ručnim podešavanjem hiperparametara. Osim toga, brža konvergencija omogućuje da Adam postiže optimalne rezultate brže od tradicionalnih metoda poput Stochastic Gradient Descent (SGD). Također, robusnost algoritma osigurava da Adam dobro funkcionira čak i u prisutnosti šuma u podacima, što ga čini pogodnim za širok raspon problema. (Goodfellow, I., Bengio, Y. i Courville, A. (2016))

Matrica konfuzije

Matrica konfuzije je tablica koja se koristi za evaluaciju performansi klasifikacijskog modela. Ona prikazuje odnos između stvarnih i predviđenih klasa. Svaki redak matrice predstavlja stvarnu klasu, a svaki stupac predstavlja predviđenu klasu. Matrica konfuzije omogućuje izračun metrika poput preciznosti, odziva (eng. „Recall“). Preciznost pokazuje koliko je predviđenih instanci klase točno klasificirano, dok odziv prikazuje koliko stvarnih instanci klase model ispravno prepoznaje. F1-rezultat predstavlja harmonijsku sredinu preciznosti i odziva, čime se osigurava balans između tih dviju mjera. U LSTM modelu, matrica konfuzije prikazuje koliko je instanci svake vremenske klase točno predviđeno. To pomaže u identificiranju klasa koje model ima problema s predviđanjem.

Trening, test i validacija

Podaci su podijeljeni u tri skupa: trening skup, validacijski skup i testni skup, čime se osigurava nepristrana evaluacija modela.

Trening skup koristi se za učenje modela, pri čemu model prepoznaje obrasce iz podataka. Validacijski skup koristi se za praćenje performansi modela tijekom treniranja te za podešavanje hiperparametara, čime se otkriva prisutnost overfittinga. Testni skup koristi se isključivo za završnu evaluaciju modela nakon treniranja i ne smije biti korišten tijekom procesa učenja.

Klase

Klase su kategorije ili grupe u kojima model pokušava klasificirati podatke. U ovom modelu, klase predstavljaju različite vremenske uvjete. Cilj modela je da predvidi kojoj klasi pripada određeni skup podataka. Metrikama poput preciznosti, odziva i F1-score izračunavaju se za svaku klasu kako bi se evaluirale performanse modela.

6.4. Eksperimentalni modeli i rezultati

U ovom poglavlju analizirati će se evolucija modela od početnih verzija temeljenih na dubokim neuronskim mrežama (DNN) do konačnog LSTM modela optimiziranog za predikciju vremenskih uvjeta. Ključna poboljšanja obuhvaćena su optimizacijom arhitekture, prilagodbom hiperparametara, balansiranjem podataka i sekvencijalnim pristupom obradi podataka.

U prvim verzijama modela koristi se duboka neuronska mreža (DNN), koja je sastavljena od potpuno povezanih slojeva neurona. Ovi modeli primarno su dizajnirani za probleme klasifikacije i regresije no imaju jednu ključnu manu - kod vremenskih podataka ne mogu pamtiti sekvencijske informacije. Svaki vremenski trenutak se analizira kao zasebna instanca, bez mogućnosti prepoznavanja dugoročnih obrazaca. Zato je napušteno korištenje DNN-a i prijelaz je napravljen na LSTM model. Long Short-Term Memory (LSTM) mreže su dizajnirane da pamte i koriste vremenske ovisnosti. One koriste memorijske ćelije koje omogućuju modelu da prepoznaje dugoročne ovisnosti u vremenskim nizovima. Upravo zbog tog razloga, u kasnijim verzijama implementirani su LSTM modeli, koji su se pokazali mnogo efikasnijima u predikciji budućih vremenskih uvjeta. (Brownlee, J. (2018))

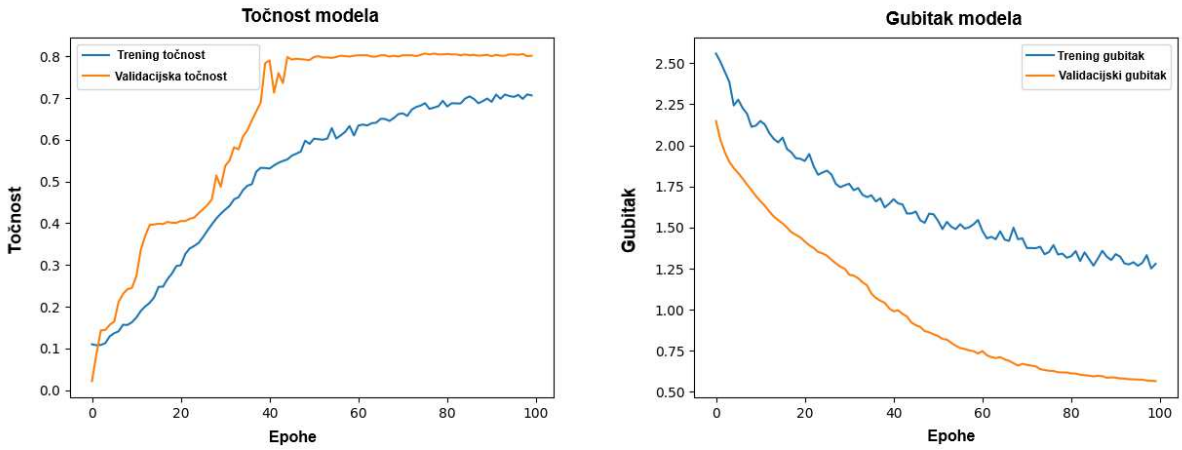
6.4.1. Model v3 – prvi značajan iskorak u DNN arhitekturi

U ranijim modelima korištene su klasične DNN arhitekture, koje su imale nekoliko ključnih problema. Vremenski uzorci nisu mogli biti prepoznati, jer se svaki podatkovni unos tretirao neovisno o ostalima. Također, došlo je do preučenja (overfittinga) pri čemu se model previše prilagodio treniranju, ali nije dobro generalizirao na nepoznate podatke. Uz to, primijećena je neuravnoteženost podataka, budući da su neki vremenski uvjeti, poput oblačnog vremena, bili znatno zastupljeniji od drugih, što je dovelo do loše klasifikacije rjeđih klasa.

Kako bi se poboljšale performanse, u verziji v3 modela uvedene su određene promjene. Dodavanjem BatchNormalization slojeva stabilizirano je treniranje modela te je ubrzana konvergencija. Dropout s vrijednošću do 0.5 smanjio je overfitting i poboljšao generalizaciju. Također, smanjenjem stope učenja (learning rate) na 0.00005, omogućeno je preciznije ažuriranje težina te je poboljšana stabilnost optimizacije.

Rezultati verzije v3 modela pokazali su nekoliko poboljšanja. Stabilnost modela povećana je, što je rezultiralo manjim oscilacijama tijekom procesa treniranja. Nadalje, ostvarena je veća točnost na testnom skupu u usporedbi s prethodnim verzijama modela. Međutim, i dalje je prisutan ključni problem – model nije mogao prepoznati vremenske obrasce jer nije koristio sekvencijske podatke.

Na slici 13 prikazan je graf točnosti i gubitka tijekom treniranja i validacije modela. Točnost i gubitak na skupu za treniranje prikazani su plavom bojom, dok su vrijednosti na skupu za validaciju prikazane narančastom bojom. Postignuta točnost na skupu za treniranje iznosila je 0.7, dok je točnost na skupu za validaciju bila 0.8. Gubitak na skupu za treniranje iznosio je 1.27, dok je gubitak na skupu za validaciju bio 0.66. Konačna točnost modela na testnom skupu, koji je sadržavao neviđene podatke, iznosila je 0.76, dok je testni gubitak bio 66%.



Slika 13 - Rezultati 3.verzije DNN modela (izvor: vlastiti rad)

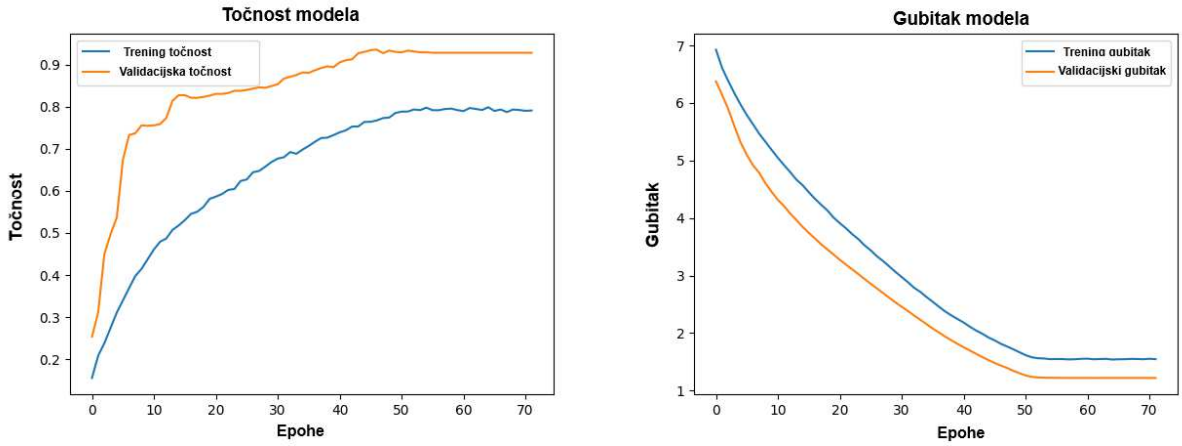
6.4.2. Model v7 – prvi napredni LSTM model

Nakon što su uočena ograničenja DNN-a, odlučeno je da se prijeđe na LSTM neuronske mreže, koje su dizajnirane za obradu vremenskih nizova. Model v7 bio je prvi model koji je koristio sekvencijske podatke, čime je omogućena bolja interpretacija promjena u vremenskim uvjetima.

U modelu v7 uvedene su ključne promjene koje su poboljšale njegovu sposobnost učenja vremenskih obrazaca. Prijelaz s obične tablične reprezentacije na vremenske sekvence omogućio je da svaki podatkovni niz sadrži povijesni kontekst. SMOTE balansiranje primijenjeno je kako bi se riješila neravnomjerna distribucija klasa generiranjem sintetičkih uzoraka. Također, dodavanjem vremenskih značajki, model je proširen tako da uzima u obzir doba dana, sezonu i druge faktore koji utječu na vremenske uvjete.

Rezultati modela v7 pokazali su nekoliko značajnih poboljšanja. To je bio prvi model koji je uspješno učio vremenske obrasce, čime je ostvarena znatno bolja točnost u usporedbi s DNN modelima. Predikcija je poboljšana u odnosu na ranije verzije, jer je model bolje kategorizirao rjeđe vremenske uvjete. Međutim, primijećeno je da validacijska točnost

ostaje nestabilna, što je ukazivalo na potrebu za daljnjom optimizacijom modela. Rezultati modela prikazani su na slici 14.



Slika 14 - Rezultati modela v7 (izvor: vlastiti rad)

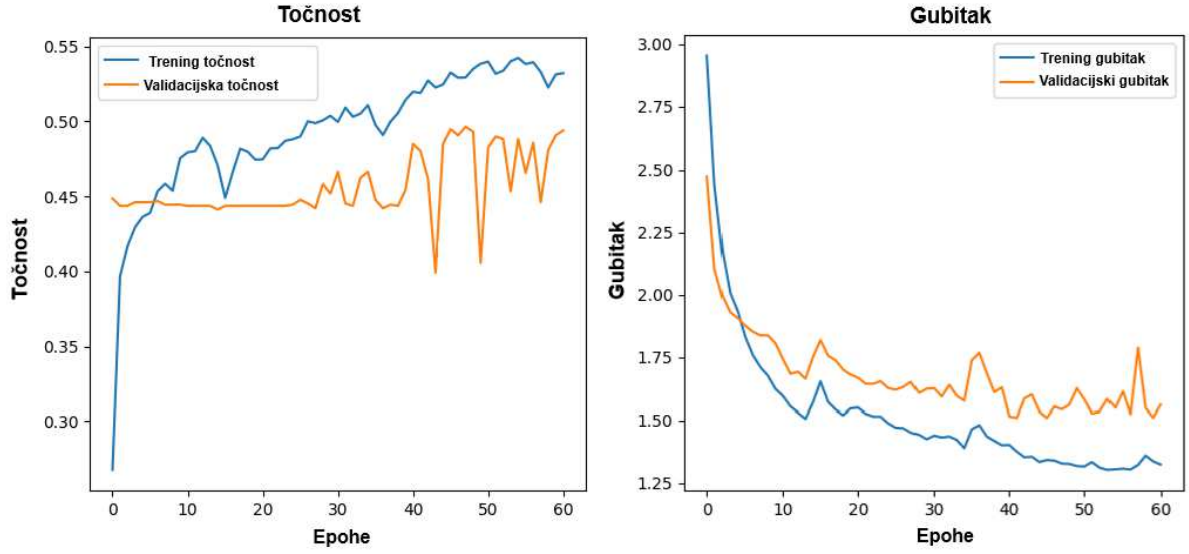
6.4.3. Model v13 – završna evaluacija i problemi

Model v13 trebao je biti završna verzija LSTM modela optimiziranog za predikciju vremenskih uvjeta šest sati unaprijed. Ovaj model razvijen je s ciljem kombiniranja najboljih značajki prethodnih verzija kako bi se postigla najveća točnost i stabilnost. Međutim, evaluacija je pokazala da model nije ostvario očekivane rezultate.

Model v13 implementiran je s nekoliko ključnih postavki. LSTM sloj s 64 neurona i ReLU aktivacijom omogućuje analizu vremenskih obrazaca. Dropout (0.3) i L2 regularizacija (0.01) primijenjeni su radi smanjenja overfittinga, dok je BatchNormalization korišten za stabilizaciju učenja i poboljšanje generalizacije modela. Kao optimizacijski algoritam korišten je Adam optimizer sa stopom učenja od 0.001, čime se osigurava učinkovita optimizacija parametara modela. Ulazne sekvence definirane su u trajanju od 24 sata, što znači da model koristi podatke iz protekla 24 sata za predikciju vremenskih uvjeta u narednih šest sati.

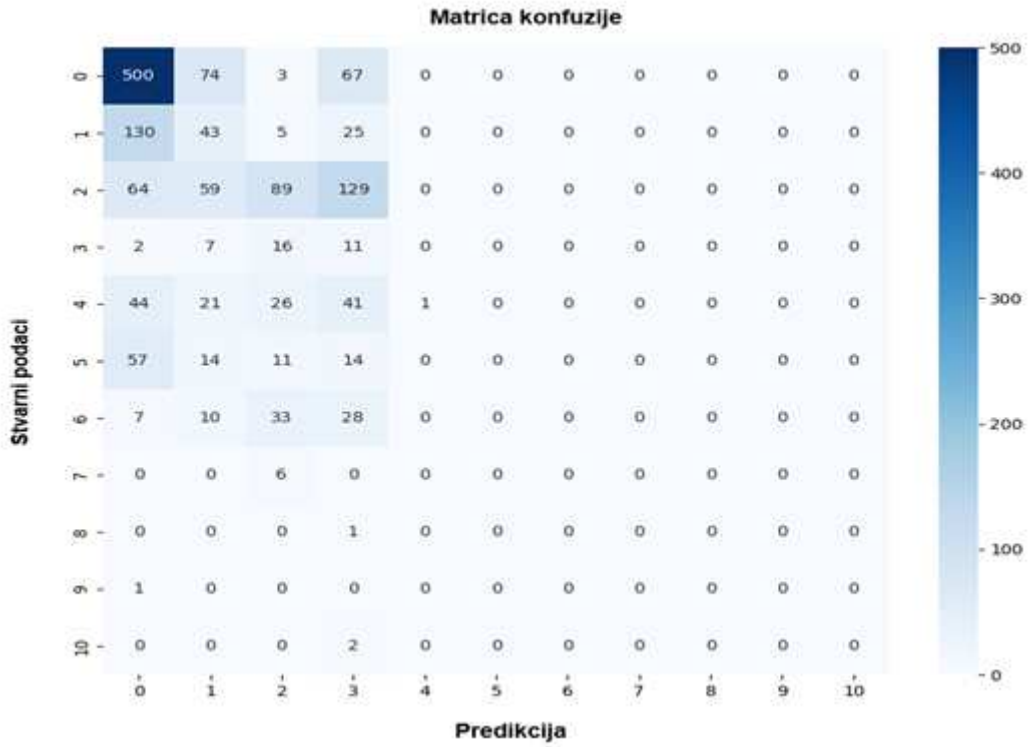
Evaluacija modela v13, prikazana na slici 15, ukazala je na značajne probleme u njegovoj učinkovitosti. Testna točnost modela iznosila je samo 41.79%, što sugerira da model nije uspio generalizirati na nepoznate podatke. S obzirom na to da testna točnost nije prelazila 50%, što je prikazano na slici 17, može se zaključiti da predikcije modela nisu dovoljno

pouzdane. Problemi s validacijskom točnošću također su uočeni tijekom analize. Krivulja gubitka pokazala je oscilacije i nestabilnost tijekom treniranja, dok je velika razlika između točnosti na trening skupu i validacijskom skupu sugerirala da model možda pati od overfittinga.



Slika 15 - Rezultati 13.modela (izvor: vlastiti rad)

Dodatni problemi primijećeni su analizom matrice konfuzije, prikazane na slici 16. Rezultati pokazuju da model nije ispravno klasificirao vremenske uvjete, pri čemu je najveći broj točnih predikcija pripadao dominantnim klasama, dok su rjeđe klase gotovo u potpunosti ignorirane. Ovo ukazuje na problem neuravnoteženosti podataka ili na nedovoljnu sposobnost modela za prepoznavanje svih vremenskih uvjeta.



Slika 16 - Matrica konfuzije za model v13 (izvor: vlastiti rad)

Jedan od značajnih problema uočen je u vrijednosti testnog gubitka, pri čemu se pojavila greška "Test Loss: NaN". Ova pojava može upućivati na lošu numeričku stabilnost modela, uzrokovanu nepravilnostima u podacima, poput prisutnosti NaN vrijednosti, ili na moguće pogreške u implementaciji modela..

```
Epoch 61/150
77/77 [==============================] - 2s 24ms/step - loss: 1.3243 - accuracy: 0.5321 - val_loss: 1.5616 - val_accuracy: 0.4939
49/49 [==============================] - 0s 9ms/step - loss: nan - accuracy: 0.4179
Test Loss: nan
Test Accuracy: 0.41791045665740967
49/49 [==============================] - 1s 7ms/step
PS X:\Weather>
```

Slika 17 - Rezultati testa za Model v13 (izvor: vlastiti rad)

Na temelju provedene analize, identificirano je nekoliko mogućih uzroka slabih performansi modela v13. Regularizacija možda djeluje preagresivno, pri čemu L2 penalizacija (0.01) previše smanjuje težine modela, ograničavajući njegovu sposobnost učenja. Dodatno, kapacitet mreže mogao bi biti nedovoljan, budući da model koristi samo

jedan LSTM sloj s 64 neurona, što može biti neadekvatno za prepoznavanje složenih vremenskih obrazaca. Još jedan potencijalni problem leži u duljini ulaznih sekvenci. Budući da model koristi podatke samo za 24 sata, moguće je da ne dobiva dovoljno informacija za preciznu predikciju vremenskih uvjeta. Također, problem neuravnoteženosti podataka mogao je utjecati na performanse modela, budući da model možda favorizira dominantne klase i zanemaruje rjeđe vremenske uvjete.

Zaključak

Model v13 implementiran je s ciljem da postane najpreciznija verzija LSTM modela ali je evaluacija pokazala da model ne postiže zadovoljavajuće rezultate. Glavni problemi uključuju nisku točnost, nestabilne validacijske performanse, lošu klasifikaciju rjeđih vremenskih uvjeta te probleme s numeričkom stabilnošću. Na temelju ovih rezultata, potrebno je provesti daljnje istraživanje kako bi se pronašla poboljšanja postojeće arhitekture ili kako bi se razvio novi model koji bi osigurao precizniju vremensku predikciju.

6.5. Završni LSTM model

U ovom poglavlju opisuje se proizvoljni model za predikciju vremenskih uvjeta koji koristi LSTM neuronsku mrežu. Cilj modela je predvidjeti vremenske uvjete 24 sata unaprijed na temelju povijesnih podataka o vremenu. Kod koji definira LSTM model nalazi se u Prilogu 3.

Podaci korišteni za treniranje modela učitani su iz prethodno pripremljenog skupa normaliziranih podataka. Ciljna varijabla označena je kao „vrijeme“, što predstavlja glavne vremenske uvjete. Kako bi se podaci pripremili za treniranje modela, provedeni su sljedeći koraci pretprocesiranja. Kategorijska ciljna varijabla vrijeme enkodirana je u numerički oblik, čime je omogućena njezina obrada unutar modela. Vremenski pomak primijenjen je tako da su podaci pomaknuti 24 sata unaprijed, omogućujući predikciju vremenskih uvjeta 24 sata unaprijed. Rijetke klase, definirane kao one s manje od 20 primjera, spojene su u jednu zajedničku klasu pod nazivom „Ostalo“. Konačno, podaci su organizirani u sekvence od 72 sata, čime je omogućeno modelu da prepoznaje vremenske ovisnosti unutar podataka. Kako bi se riješio problem neuravnoteženih klasa,

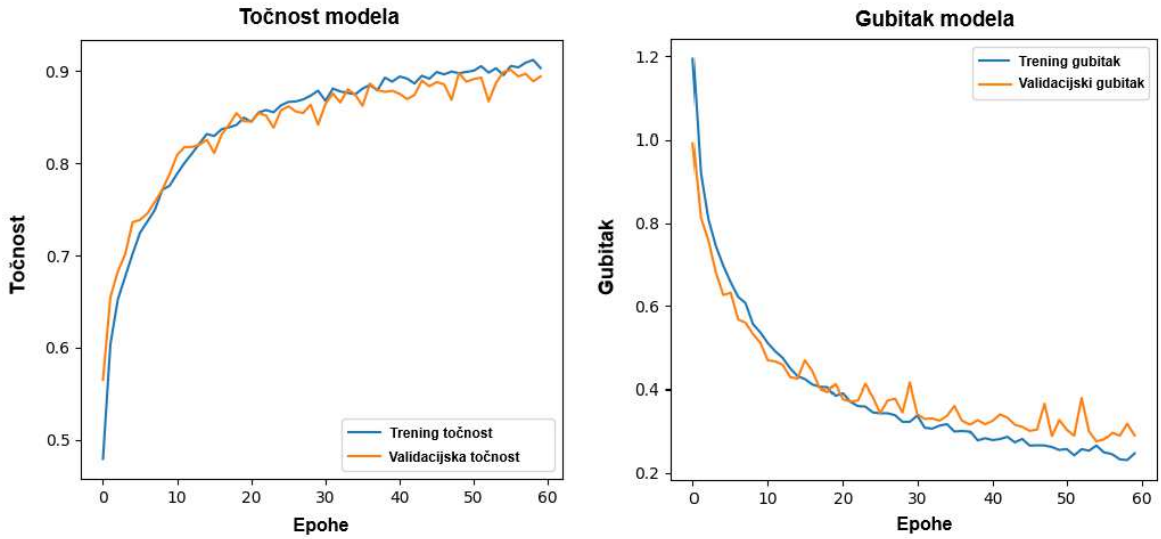
primijenjena je metoda SMOTE. Nakon što je SMOTE korišten, klasna distribucija postala je uravnoteženija, što se pokazalo ključnim za poboljšanje rezultata modela. Model se sastoji od nekoliko slojeva koji omogućuju obradu vremenskih sekvenci i donošenje preciznih predikcija. Prvi sloj je LSTM sloj sa 64 neurona i tanh aktivacijskom funkcijom, pri čemu se vraćaju sekvence kako bi se omogućilo daljnje procesiranje u sljedećem LSTM sloju. Kako bi se smanjio rizik od prenaučavanja, dodan je dropout sloj (0.3), pri čemu se nasumično isključuje 30 posto neurona tijekom treniranja. Drugi LSTM sloj sastoji se od 32 neurona, također s tanh aktivacijskom funkcijom, ali ne vraća sekvence. Ponovno je dodan dropout sloj (0.3) kako bi se dodatno smanjilo prenaučavanje modela. Nakon LSTM slojeva, dodan je Dense sloj s 16 neurona, koji koristi ReLU aktivacijsku funkciju. BatchNormalization sloj normalizira izlaze prethodnog sloja kako bi se ubrzala konvergencija i poboljšala stabilnost modela. Izlaz modela definiran je Dense slojem koji sadrži broj neurona jednak broju jedinstvenih klasa u ciljnoj varijabli. Ovaj sloj koristi softmax aktivacijsku funkciju kako bi se generirale vjerojatnosti za svaku klasu.
Model je kompiliran korištenjem Adam optimizatora sa stopom učenja 0.001 i sparse_categorical_crossentropy kao funkcijom gubitka. Kao metrika evaluacije korištena je točnost (eng. „Accuracy“).

Treniranje modela

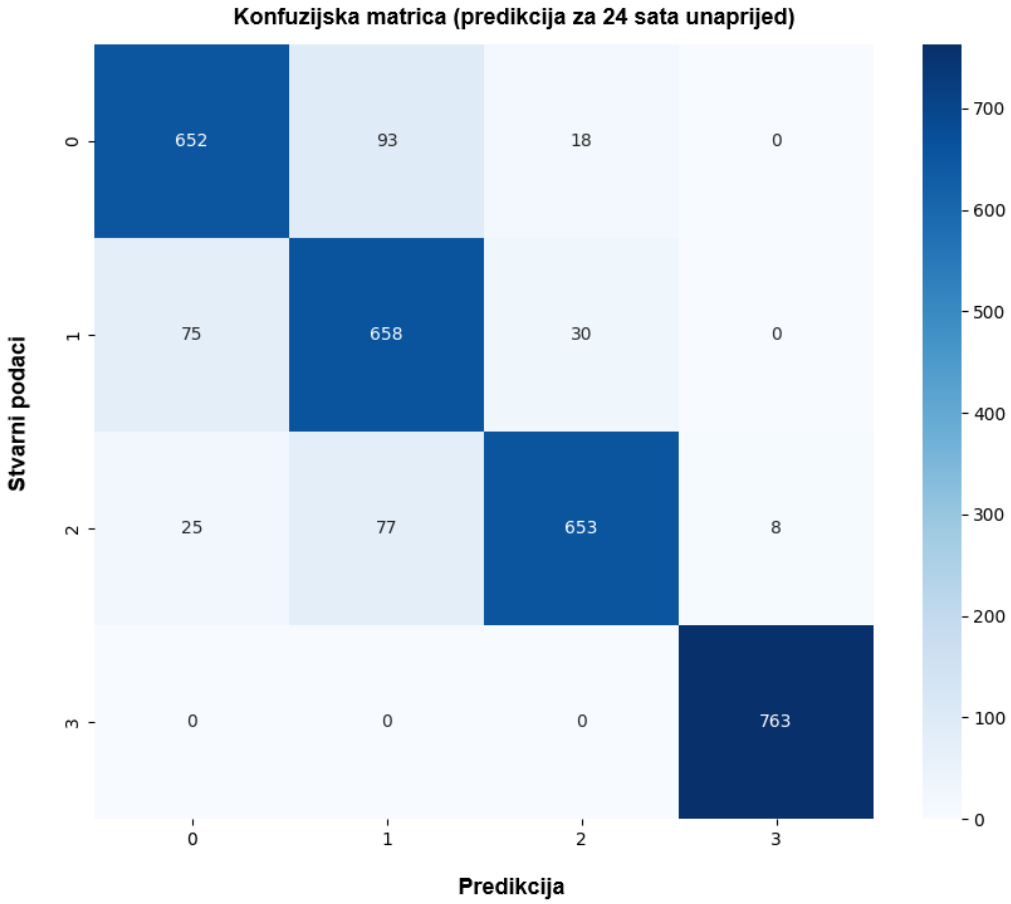
Model je treniran kroz 60 epoha, pri čemu je korištena veličina serije (batch size) od 64 primjera. Kako bi se dodatno riješio problem neuravnoteženih klasa, korištene su težine klasa koje su obrnuto proporcionalne broju primjera u svakoj klasi. Tijekom treniranja, 20 posto podataka izdvojeno je za validaciju, čime je omogućeno praćenje performansi modela i sprječavanje overfittinga. Točnost treninga i validacije prikazana je na slici 18.

Rezultati

Nakon treniranja, model je evaluiran na testnom skupu podataka. Matrica zabune prikazana na slici 19. daje prikaz odnosa između stvarnih i predviđenih klasa, dok klasifikacijski izvještaj daje detaljne metrike za svaku klasu, uključujući preciznost, odziv (recall) i F1-score.



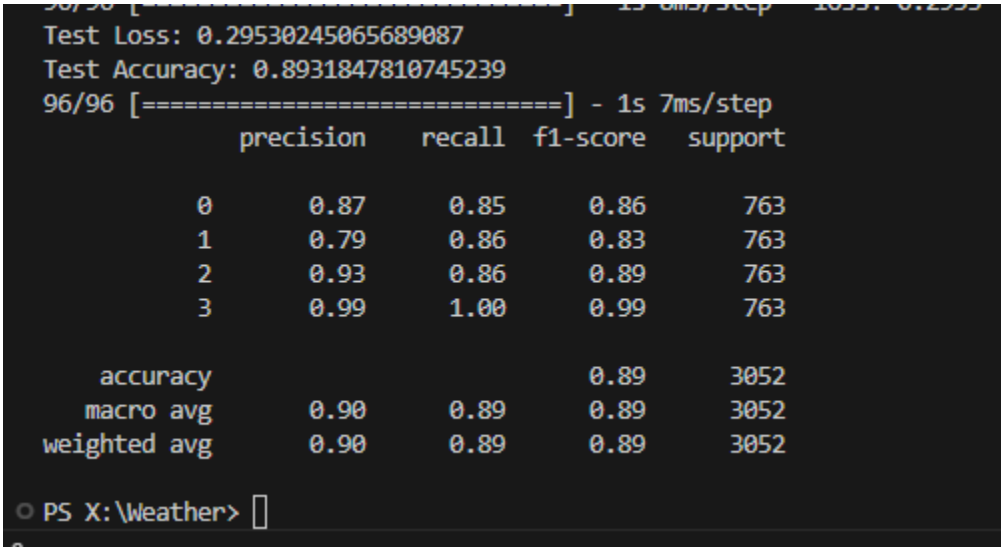
Slika 18 - Rezultati testa za Model v13 (izvor: vlastiti rad)



Slika 19 - Matrica konfuzije proizvoljnog modela (izvor: vlastiti rad)

Zaključak

Predloženi LSTM model pokazuje obećavajuće rezultate u predviđanju vremenskih uvjeta 24 sata unaprijed. Primjena SMOTE metode za balansiranje podataka i korištenje težina klasa za kompenzaciju neuravnoteženih klasa pokazala se učinkovitim u poboljšanju performansi modela. Budući rad mogao bi se usmjeriti na daljnje optimizacije modela, uključujući fine-tuning hiperparametara i korištenje dodatnih značajki za poboljšanje točnosti predviđanja. Završna točnost modela prikazana na slici 20. iznosi 89%, što je zadovoljavajuća predikciju za 24 satno razdoblje.



Slika 20 - Rezultat testiranja proizvoljnog modela (izvor: vlastiti rad)

6.6. Predikcija kratkoročnih vremenskih uvjeta

Nakon testiranja i evaluacije proizvoljnog modela, pri čemu je postignuta testna točnost od 89 posto, vrijeme je za testiranje modela na vremenskim podacima prikupljenima u sadašnjosti. Podaci za proteklih nekoliko dana prikupljeni su kako bi se omogućilo korištenje 72-satnog perioda za predikciju vremenskih uvjeta.

Tablica 2 - "Real time" podaci (izvor: vlastiti rad)

datum	vlažnost	temp	tlak	osjet	oblaci	brzina_vjetra	vjetar_bf
2025-02-11 09:00:00 +0000 UTC	0.96	0.41	0.68	0.41	82	0.12	2
2025-02-11 10:00:00 +0000 UTC	0.93	0.42	0.68	0.42	86	0.11	2
2025-02-11 11:00:00 +0000 UTC	0.91	0.43	0.68	0.43	89	0.12	2
2025-02-11 12:00:00 +0000 UTC	0.84	0.43	0.67	0.43	86	0.09	2
2025-02-11 13:00:00 +0000 UTC	0.9	0.44	0.66	0.44	100	0.09	2
2025-02-11 14:00:00 +0000 UTC	0.84	0.43	0.65	0.43	100	0.09	2
2025-02-11 15:00:00 +0000 UTC	0.79	0.43	0.65	0.43	100	0.1	2
2025-02-11 16:00:00 +0000 UTC	0.9	0.43	0.65	0.43	100	0.13	2
2025-02-11 17:00:00 +0000 UTC	0.98	0.42	0.65	0.42	100	0.15	2

6.7. Predikcija kratkoročnih vremenskih uvjeta

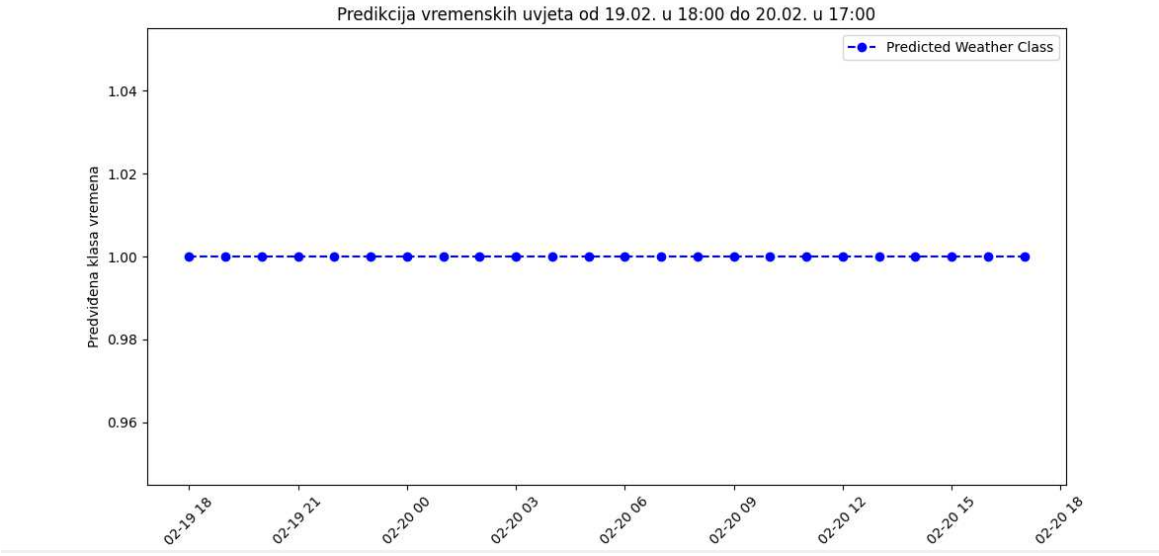
Nakon testiranja i evaluacije proizvoljnog modela, pri čemu je postignuta testna točnost od 89 posto, model se testira na vremenskim podacima prikupljenima u sadašnjosti. Kako bi se omogućila precizna predikcija, korišteni su meteorološki podaci iz vremenskog razdoblja od 11. veljače do 19. veljače, koji su zatim obrađeni kako bi bili u istom formatu kao podaci korišteni pri treniranju modela. Za testiranje stvarnih podataka razvijena je nova skripta, dostupna u Prilogu 4.

Obrada podataka uključuje nekoliko ključnih koraka. Prvo, kontinuirane vrijednosti, poput temperature, tlaka i vlažnosti zraka, normalizirane su kako bi bile usklađene s podacima korištenima tijekom treniranja modela. Smjer vjetra transformiran je u vektorske komponente vjetar_x i vjetar_y, čime se osigurava pravilna reprezentacija vremenskih podataka. Kategorijske varijable, uključujući vremenske uvjete poput "vedro", "oblaci" i "kiša", enkodirane su u numerički oblik. Nakon toga, podaci su organizirani u 72-satne vremenske sekvence, koje model koristi za generiranje predikcija. Nakon provedene obrade, stvoren je ulazni format koji omogućuje generiranje predikcija. Model koristi posljednjih 72 sata podataka kako bi predvidio vremenske uvjete za sljedećih 24 sata. Proces predikcije odvija se iterativno, pri čemu model analizira uzorak podataka iz posljednjih 72 sata i na temelju prepoznatih vremenskih obrazaca generira prvu

vremensku predikciju. Nakon toga, predikcija se dodaje na kraj ulazne sekvence, dok se najstariji podatak uklanja. Ovaj postupak ponavlja se 24 puta, omogućujući predikciju vremenskih uvjeta za narednih 24 sata. Rezultati predikcije uključuju očekivanu klasu vremena za svaki sat unutar vremenskog intervala od 19. veljače u 18:00 do 20. veljače u 17:00.

6.7.1. Rezultati i analiza predikcije

Nakon izvođenja modela, rezultati su pohranjeni u CSV formatu, omogućujući njihovu analizu i usporedbu s realnim vremenskim uvjetima. Na slici 21 prikazani su rezultati predikcije za 24 sata unaprijed. Dobivena predikcija pokazuje dominantnu klasu „oblačno“ tijekom cijelog predviđenog razdoblja. Mogući razlozi za ovakve rezultate uključuju nekoliko čimbenika. Jedan od mogućih razloga je dominacija oblaka u treniranom skupu podataka, zbog čega model favorizira ovu klasu. Također, ulazni podaci korišteni za predikciju mogu ukazivati na oblačno vrijeme, što sugerira da model pravilno prepoznaje trendove u podacima. Osim toga, moguća je potreba za boljim balansiranjem klasa tijekom treniranja modela, čime bi model postao osjetljiviji na manje zastupljene vremenske uvjete. Vizualizacija predikcija pokazuje stabilnu vremensku prognozu, što može ukazivati na stabilne atmosferske uvjete u stvarnim podacima korištenima za testiranje modela.



Slika 21 - Rezultat kratkoročne predikcije (izvor: vlastiti rad)

6.7.2. Problemi i ograničenja modela

Iako model pokazuje solidne rezultate u prepoznavanju vremenskih obrazaca, identificirano je nekoliko problema i ograničenja. Model ima tendenciju favorizirati najčešće klase iz skupa podataka, što može rezultirati smanjenom preciznošću u predikciji manje zastupljenih vremenskih uvjeta, poput kiše ili magle. Ograničenja su primijećena i u pogledu raznolikosti podataka korištenih za treniranje, budući da je model treniran na podacima iz ograničenog vremenskog razdoblja. Nedostatak primjera ekstremnih vremenskih uvjeta, poput oluja, može dovesti do smanjene sposobnosti modela za njihovu prepoznavanje. Dodatno, model je osjetljiv na kvalitetu ulaznih podataka. Ako real-time podaci sadrže šum ili netočnosti, to može negativno utjecati na predikcije. Preciznost modela mogla bi se poboljšati kalibracijom senzora i primjenom filtriranja podataka prije generiranja predikcija.

Još jedno ograničenje odnosi se na prepoznavanje kratkoročnih vremenskih promjena. Model koristi podatke iz proteklih 72 sata za predviđanje budućih vremenskih uvjeta, no iznenadne promjene vremena možda neće biti pravovremeno prepoznate. Poboljšanja modela mogla bi uključivati dodavanje dodatnih atmosferskih varijabli koje značajno utječu na vremenske uvjete.

Kako bi se poboljšala točnost modela i njegova sposobnost generalizacije na različite vremenske uvjete, predloženo je nekoliko nadogradnji.

Raznolikost skupa podataka mogla bi se poboljšati uključivanjem podataka iz različitih sezona, čime bi model naučio prepoznavati sezonske varijacije u vremenskim uvjetima. Također, proširenjem skupa podataka s više ekstremnih vremenskih uvjeta, poput oluja, snježnih padalina i magle, povećala bi se sposobnost modela za predikciju manje čestih vremenskih pojava.

Kako bi se poboljšala ravnoteža klasa tijekom treniranja, mogu se koristiti tehnike poput SMOTE-a, čime bi se povećao broj podataka iz manje zastupljenih vremenskih uvjeta. Osim toga, korištenjem ponderiranih gubitaka unutar modela može se naglasiti važnost rjeđih vremenskih stanja, čime bi model postao osjetljiviji na manje česte pojave.

Dodavanje novih značajki moglo bi poboljšati preciznost modela. Uključivanjem geoprostornih podataka, poput nadmorske visine, može se dodatno poboljšati sposobnost modela za prepoznavanje lokalnih vremenskih obrazaca. Također, analiza promjena tlaka zraka i njihova povezanost s vremenskim uvjetima mogla bi pridonijeti većoj preciznosti predikcija.

Povećanje kapaciteta modela moglo bi se postići testiranjem naprednijih arhitektura neuronskih mreža, poput transformera ili kombinacije CNN-LSTM modela, čime bi se omogućila bolja analiza vremenskih uzoraka. Dodatno, povećanjem broja vremenskih slojeva u LSTM mreži, model bi mogao dublje obraditi podatke i prepoznati složenije obrasce u vremenskim uvjetima.

Konačno, poboljšanja u vizualizaciji i interpretaciji predikcija mogla bi dodatno olakšati analizu rezultata. Uvođenjem mapa topline koje prikazuju razlike između stvarnih i predviđenih vremenskih uvjeta te izradom interaktivnih grafikona, omogućila bi se dublja analiza vremenskih trendova i učinkovitosti modela.

Predikcija real-time podataka pokazala je da model uspješno generalizira vremenske uvjete u skladu s ulaznim podacima. Ipak, identificirani su određeni problemi i ograničenja, prvenstveno vezani uz balansiranost klasa i raznolikost podataka.

Predložena poboljšanja mogu povećati preciznost modela i poboljšati njegovu sposobnost prepoznavanja manje zastupljenih vremenskih uvjeta. Dobiveni rezultati predstavljaju važan korak u evaluaciji modela, omogućujući dodatna poboljšanja u točnosti vremenske prognoze i sposobnosti neuronske mreže za generalizaciju.

7. Zaključak

Ovaj rad istražio je primjenu osobne meteorološke stanice i dubokog učenja za predikciju vremenskih uvjeta. Razvijen je automatizirani sustav koji prikuplja meteorološke podatke putem Arduino senzora i OpenWeather API-ja, uz naglasak na kontinuirano prikupljanje atmosferskih uvjeta. Prikupljeni podaci uključuju temperaturu, tlak, vlažnost, brzinu i smjer vjetra te oblačnost. Nakon prikupljanja, podaci su normalizirani, enkodirani i pripremljeni za analizu pomoću LSTM neuronske mreže, posebno dizajnirane za rad s vremenskim nizovima.

Model je treniran na temelju 72-satnih vremenskih sekvenci i testiran za predikciju vremenskih uvjeta 24 sata unaprijed. Tijekom evaluacije model je postigao testnu točnost od 89%, što pokazuje njegovu sposobnost prepoznavanja vremenskih obrazaca. Primjenom modela na real-time podatke, dobiveni su rezultati koji pokazuju da model uspješno generalizira dominantne vremenske uvjete, ali ima sklonost favoriziranju češćih klasa poput oblačnog vremena.

Glavni izazovi uključuju bolje balansiranje podataka, proširenje skupa podataka kako bi obuhvatio različite vremenske uvjete i optimizaciju arhitekture modela. Moguća poboljšanja uključuju testiranje naprednijih modela poput CNN-LSTM, poboljšanje načina obrade ulaznih podataka te učenje na većem vremenskom rasponu kako bi model bio robusniji. Također, uvođenje dodatnih meteoroloških varijabli, poput geoprostornih podataka i atmosferskog pritiska kroz dulji vremenski period, moglo bi poboljšati predikcije.

Unatoč izazovima, rad potvrđuje da se kombinacijom senzorskih podataka i neuronskih mreža može ostvariti učinkovita lokalna vremenska prognoza. Razvijeni sustav može se primijeniti u poljoprivredi, pametnim kućama, industriji i istraživanju, pridonoseći boljem razumijevanju i predviđanju mikroklimatskih uvjeta.

8. Popis slika

Slika 1 - Arduino Uno Rev3 (izvor: fotografija autora) 4
Slika 2 - Integrirano razvojno okruženje (izvor: slika autora) 6
Slika 3 - USB 2.0 Cable Type A/B (izvor: fotografija autora) 7
Slika 4 - Eksperimentalna pločica (izvor: fotografija autora) 9
Slika 5 - Jumper žice (izvor: fotografija autora) 10
Slika 6 - DHT22 senzor (izvor: fotografija autora) 11
Slika 7 - BMP280 senzor (izvor: fotografija autora) 12
Slika 8 - Impulsni anemometar (izvor: fotografija autora) 13
Slika 9 - Blokovska shema sustava (izvor: vlastiti rad) 14
Slika 10 - Normalizacijski faktori (izvor: vlastiti rad) 19
Slika 11 - Beaufortova ljestvica jačine vjetra (izvor: vlastiti rad) 20
Slika 12 - SMOTE metoda za balansiranje podataka (izvor: https://emilia-orellana44.medium.com/smote-2acd5dd09948) 24
Slika 13 - Rezultati 3.verzije DNN modela (izvor: vlastiti rad) 34
Slika 14 - Rezultati modela v7 (izvor: vlastiti rad) 35
Slika 15 - Rezultati 13.modela (izvor: vlastiti rad) 36
Slika 16 - Matrica konfuzije za model v13 (izvor: vlastiti rad) 37
Slika 17 - Rezultati testa za Model v13 (izvor: vlastiti rad) 37
Slika 18 - Rezultati testa za Model v13 (izvor: vlastiti rad) 40
Slika 19 - Matrica konfuzije proizvoljnog modela (izvor: vlastiti rad) 40
Slika 20 - Rezultat testiranja proizvoljnog modela (izvor: vlastiti rad) 41
Slika 21 - Rezultat kratkoročne predikcije (izvor: vlastiti rad) 43

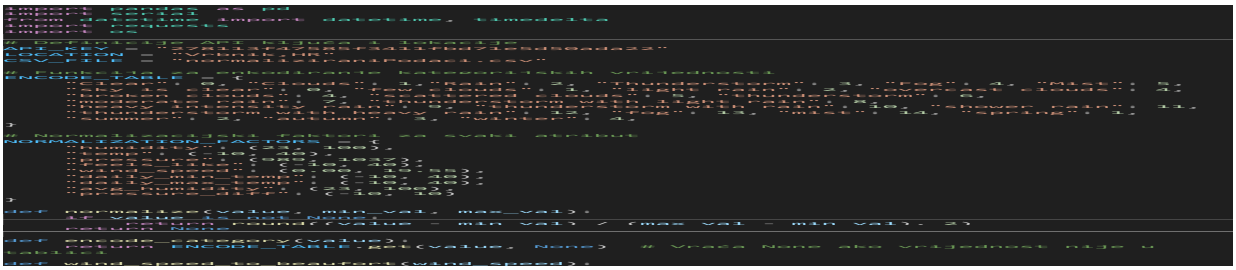
9. Popis tablica

Tablica 1 - Usporedba originalnih i normaliziranih podataka (izvor: vlastiti rad) 22
Tablica 2 - "Real time" podaci (izvor: vlastiti rad) 42

10. Prilozi

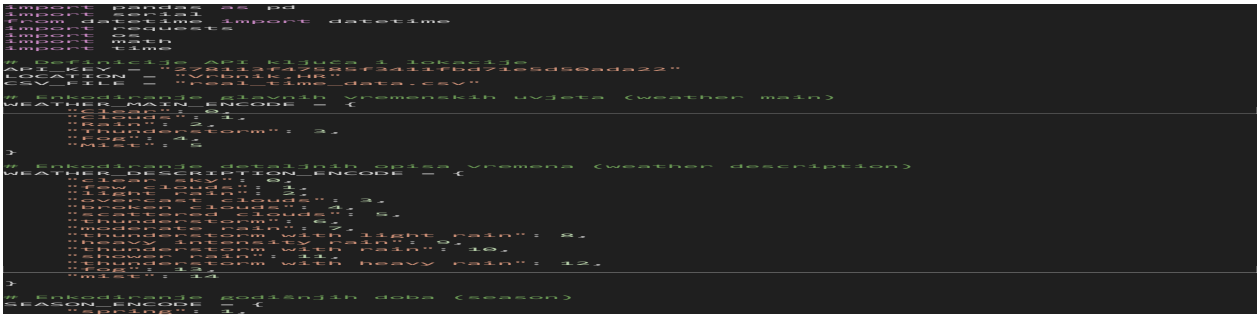
1. **Prilog 1. – Arduino kod**

2. **Prilog 2. – Kod za prikupljanje podataka**



3. **Prilog 3. – LSTM.py**

4. **Prilog 4. - realtime.py**



11. Literatura

Arduino.cc (2023) Arduino Documentation. Dostupno na: https://www.arduino.cc (Pristupljeno: 17. rujna 2024.).

Banzi, M. i Shiloh, M. (2022) Getting Started with Arduino: The Open Source Electronics Prototyping Platform. 4. izd. O'Reilly Media.

Valvano, J. (2020) Embedded Systems: Introduction to ARM Cortex-M Microcontrollers. CreateSpace Independent Publishing.

Monk, S. (2018) Programming Arduino: Getting Started with Sketches. 2. izd. McGraw-Hill Education.

Adafruit (2023) DHT22 Temperature-Humidity Sensor Guide. Dostupno na: https://learn.adafruit.com/dht (Pristupljeno: 12. listopada 2024.).

Lady Ada Adafruit BMP280 Barometric Pressure + Temperature Sensor Breakout. Dostupno na: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-bmp280-barometric-pressure-plus-temperature-sensor-breakout.pdf (Pristupljeno: 12. listopada 2024.).

Adafruit, Anemometer Wind Speed Sensor with Analog Voltage Output. Dostupno na: https://www.adafruit.com/product/1733 (Pristupljeno: 30. studeni 2024.).

Horowitz, P. i Hill, W. (2015) The Art of Electronics. 3. izd. Cambridge University Press.
McRoberts, M. (2011) Beginning Arduino. Apress.

Alex, S.A. (2025) 'Imbalanced data learning using SMOTE and deep learning architecture with optimized features', *Neural Computing & Applications*, 37, str. 967–984. DOI: 10.1007/s00521-024-10481-y.

Goodfellow, I., Bengio, Y. i Courville, A. (2016) *Deep learning*. Cambridge, MA: MIT Press

Brownlee, J. (2018) *Deep learning for time series forecasting: Predict the future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery.

12. Sažetak

Predikcija vremenskih uvjeta važna je u mnogim područjima, od svakodnevnog planiranja do profesionalne meteorologije. Ovaj rad istražuje mogućnost primjene osobne meteorološke stanice s dubokim učenjem, kako bi se omogućila preciznija lokalna vremenska prognoza. Razvijen je sustav koji automatizirano prikuplja, obrađuje i analizira meteorološke podatke, a zatim koristi neuronske mreže za predikciju budućih vremenskih uvjeta. Podaci su prikupljeni pomoću Arduino senzora i OpenWeather API-ja, obuhvaćajući temperaturu, tlak, vlažnost, brzinu i smjer vjetra te oblačnost. Prije analize, podaci su očišćeni, interpolirani i normalizirani, kako bi se osigurala njihova konzistentnost. Implementirana je i konverzija smjera vjetra u vektorske komponente, što omogućuje točniju analizu vremenskih obrazaca. Za vremensku prognozu korištena je LSTM neuronska mreža, trenirana na podacima organiziranim u sekvence duljine 72 sata, pri čemu model uči iz prošlih vremenskih uvjeta kako bi predvidio 24 sata unaprijed. Evaluacija modela provedena je pomoću MAE, MSE i R^2 metrika, a rezultati pokazuju da model postiže visoku točnost u predikciji temperature i tlaka, dok su predikcije brzine i smjera vjetra bile manje precizne zbog veće oscilacije podataka. Testiranje na real-time podacima potvrdilo je sposobnost modela da generalizira i prepozna vremenske obrasce, no nije prošlo bez izazova, poput balansiranja podataka i optimizacije arhitekture modela. Daljnja poboljšanja mogla bi uključivati testiranje naprednijih neuronskih mreža (CNN-LSTM), povećanje raznolikosti skupa podataka, uključivanje dodatnih atmosferskih varijabli te optimizaciju vremenskih sekvenci. Zaključno, ovaj rad pokazuje kako se kombinacijom mikrokontrolera i dubokog učenja može postići učinkovita lokalna vremenska prognoza. Razvijeni sustav pruža temelje za daljnji razvoj, s mogućnostima primjene u područjima poput poljoprivrede, energetike, pametnih gradova i istraživanja klime.

Ključne riječi: Predikcija vremenskih uvjeta, osobna meteorološka stanica, duboko učenje, LSTM neuronska mreža, Arduino senzori, OpenWeather API, obrada podataka, brzina i smjer vjetra, analiza klime, atmosferski podaci, optimizacija neuronskih mreža, predikcija vremena u stvarnom vremenu

13. Abstract

Weather prediction is crucial in many fields, ranging from daily planning to professional meteorology. This study explores the integration of a personal meteorological station with deep learning to enable more accurate local weather forecasting. A system has been developed that automatically collects, processes, and analyzes meteorological data and then employs neural networks to predict future weather conditions. The data was collected using Arduino sensors and the OpenWeather API, encompassing temperature, pressure, humidity, wind speed and direction, and cloud cover. Before analysis, the data underwent cleaning, interpolation, and normalization to ensure consistency. Additionally, wind direction was converted into vector components, improving the accuracy of weather pattern analysis. For weather forecasting, an LSTM neural network was trained on data organized into 72-hour sequences, allowing the model to learn from past weather conditions and predict the next 24 hours. The model's performance was evaluated using MAE, MSE, and R^2 metrics, demonstrating high accuracy in predicting temperature and pressure, while predictions of wind speed and direction were less precise due to greater data variability. Testing on real-time data confirmed the model’s ability to generalize and recognize weather patterns, although challenges remained, such as data balancing and model architecture optimization. Future improvements could include testing more advanced neural networks (CNN-LSTM), increasing dataset diversity, incorporating additional atmospheric variables, and optimizing time-series sequences. In conclusion, this study demonstrates that by combining microcontrollers and deep learning, an effective local weather prediction system can be achieved. The developed system provides a foundation for further advancements, with potential applications in agriculture, energy management, smart cities, and climate research.

Keywords: Weather prediction, Personal meteorological station, Deep learning, LSTM neural network, Arduino sensors, OpenWeather API, Data preprocessing, Wind speed and direction, Climate analysis, Atmospheric data, Neural network optimization, Real-time weather forecasting