Predviđanje aktivnosti peptida uporabom neuronskih mreža temeljenih na grafovima

Cvetko, Jakov

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Prijediplomski studij računarstva

Završni rad

Predviđanje aktivnosti peptida uporabom neuronskih mreža temeljenih na grafovima

Rijeka, rujan 2023.

Jakov Cvetko
0069089563

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Prijediplomski studij računarstva

Završni rad

Predviđanje aktivnosti peptida uporabom neuronskih mreža temeljenih na grafovima

Mentor: doc.dr.sc. Goran Mauša

Rijeka, rujan 2023.

Jakov Cvetko
0069089563

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2023.

_____

Ime Prezime

Zahvala

Hvala doc. dr. sc. Goranu Mauši i asistentu Eriku Otoviću na mentorstvu i savjetima tijekom izrade završnog rada. Također, zahvaljujem svojoj obitelji i kolegama na podršci tijekom studija.

Sadržaj

Popis slika **vii**

Popis tablica **ix**

1 Uvod **1**

2 Metodologija **3**

2.1 Korišteni alati 3

2.1.1 Python 3

2.1.2 PyMOL 5

2.1.3 RDKit 7

2.1.4 StellarGraph 9

2.1.5 Pandas 11

2.1.6 Tensorflow 13

2.1.7 Scikit-learn 15

2.2 Skup podataka 17

3 Analiza modela strojnog učenja **18**

3.1 Pristup, obrada i stvaranje grafova temeljenih na molekulama 18

3.1.1 Normiranje značajki 22

3.1.2 Stvaranje grafa 23

3.2 Arhitektura modela neuronske mreže 24
3.2.1 Priprema generatora grafova 24
3.2.2 Neuronska mreža za predviđanje aktivnosti peptida 25
3.3 Treniranje modela strojnog učenja 30
3.3.1 Kompilacija modela 30
3.3.2 Treniranje i vrednovanje modela 31
3.4 Metrike za vrednovanje modela 33

4 Rezultati **37**
4.1 Model 1 39
4.2 Model 2 40
4.3 Model 3 41
4.4 Model 4 42
4.5 Model 5 43
4.6 Model 6 44
4.7 Model 7 45
4.8 Usporedba vrijednosti metrika modela strojnog učenja s referentnim vrijednostima 47

5 Zaključak **51**

Bibliografija **53**

Sažetak **56**

Popis slika

1.1 Prikaz jednog od mnogih antiviralnih peptida korištenih za treniranje modela 2

2.1 Zastupljenost uporabe programskih jezika za strojno učenje 4
2.2 PyMol alat za prikaz kemijskih struktura u prostoru 6
2.3 Primjena RDKit knjižnice u programskog kodu 8
2.4 Primjena StellarGraph knjižnice u programskog kodu 10
2.5 Primjena Pandas knjižnice u programskog kodu 12
2.6 Primjena Tensorflow knjižnice u programskog kodu 14
2.7 Primjena Scikit-learn knjižnice u programskog kodu 16

3.1 Slika koda za obradu molekula 19
3.2 Primjer zapisa formalnog naboja atoma na molekuli nitrata 20
3.3 Opis geometrije i rasporeda orbitala atoma kroz hibridizaciju 21
3.4 Aromatični prsten molekule benzena 21
3.5 Slika koda za normiranje značajki 23
3.6 Slika koda za kreiranje grafa 24
3.7 Implementacija neuronske mreže u programskog kodu 26
3.8 Vizualna reprezentacija arhitekture neuronske mreže 29
3.9 Programski kod zadužen za kompilaciju modela 31
3.10 Podjela skupa podataka na grupe 31

3.11 Podjela na skup za validaciju, testiranje i treniranje 32
3.12 Odnos skupova TP, FP, FN, TN 33
3.13 Izračun osjetljivosti, preciznosti i točnosti 34
3.14 ROC krivulja i AOC površina 36

4.1 Referentne vrijednosti 47
4.2 Grafički prikaz rezultata modela s najboljim performansama 48
4.3 Grafički prikaz usporedbe vrijednosti 50

Popis tablica

2.1 Analiza skupa podataka antiviralnih peptida AVPdb 17

4.1 Specifikacije modela na kojima su vršena istraživanja 38
4.2 Rasponi vrijednosti za korištene metrike. 39
4.3 Vrijednosti metrika modela 1 treniranog s one-hot pristupom. 39
4.4 Vrijednosti metrika modela 1 treniranog s kombiniranim pristupom. 40
4.5 Vrijednosti metrika modela 2 treniranog s one-hot pristupom. 40
4.6 Vrijednosti metrika modela 2 treniranog s kombiniranim pristupom. 41
4.7 Vrijednosti metrika modela 3 treniranog s one-hot pristupom. 41
4.8 Vrijednosti metrika modela 3 treniranog s kombiniranim pristupom. 42
4.9 Vrijednosti metrika modela 4 treniranog s one-hot pristupom. 42
4.10 Vrijednosti metrika modela 4 treniranog s kombiniranim pristupom. 43
4.11 Vrijednosti metrika modela 5 treniranog s one-hot pristupom 43
4.12 Vrijednosti metrika modela 5 treniranog s kombiniranim pristupom. 44
4.13 Vrijednosti metrika modela 6 treniranog s one-hot pristupom 45
4.14 Vrijednosti metrika modela 6 treniranog s kombiniranim pristupom . 45
4.15 Vrijednosti metrika modela 7 treniranog s one-hot pristupom 46
4.16 Vrijednosti metrika modela 7 treniranog s kombiniranim pristupom . 46

Poglavlje 1

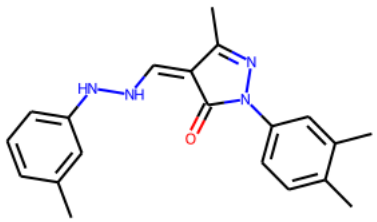
Uvod

Strojno učenje je grana umjetne inteligencije koja se odnosi na skup tehnika i algoritama koji omogućuju računalima učenje iz podataka i donošenje zaključka ili predviđanja temeljem tih podataka bez da su eksplicitno programirani za određeni zadatak [1]. U posljednjem desetljeću, strojno učenje je postalo neizostavan alat u različitim područjima znanosti i tehnologije. Jedno od istaknutih područja unutar strojnog učenja obuhvaća neuronske mreže, koje se inspiriraju strukturom i funkcionalnošću ljudskog mozga.

Neuronske mreže su skup povezanih jedinica, nazvanih neuroni, koji zajedno rade na obradi informacija [2]. Oni su sposobni za duboko učenje, što znači da mogu naučiti složene obrasce i odnose iz podataka. Kroz višeslojnu arhitekturu, podaci se prenose kroz neuronsku mrežu, gdje svaki sloj ima svoju ulogu u ekstrakciji značajki i donošenju konačnog rezultata. Prilikom izrade završnog rada korištena je nova generacija neuronskih mreža, pod nazivom neuronske mreže temeljene na grafovima. Pojam graf neuronske mreže (eng. Graph Neural Networks, GNN) prvi su skovali Scarselli, Gori i suradnici u njihovom radu iz 2009. godine [3]. Grafovi su strukture koje se sastoje od čvorova i bridova koji služe za reprezentaciju odnosa između elemenata. Umjesto dosadašnjeg pristupa korištenja linearnih nizova ili matrica za prikazivanje podataka, grafovi omogućuju modeliranje složenih veza i interakcija među elementima. Neuronske mreže temeljene na grafovima koriste ovu grafove kako bi obavile složene zadatke poput klasifikacije, predviđanja i segmentacije, pri čemu su grafovi ključni za modeliranje veza i susjedstva između elemenata.

Ovaj završni rad, izrađen u okviru projekta *Dizajn katalitički aktivnih peptida i peptidnih nanostruktura* s oznakom UIP-2019-04-7999, istražuje se primjena neuronskih mreža temeljenih na grafovima za predviđanje aktivnosti peptida. Ovim istraživanjem otvara se mogućnost otkrivanja do sada nepoznatih antiviralnih peptida putem dubokog analiziranja njihove grafovske strukture i biološke aktivnosti. Identifikacija novih peptida može predstavljati ključni korak prema razvoju antiviralnih terapija koje ciljaju različite viruse. Korištenje neuronskih mreža temeljenih na grafovima može pružiti dublji uvid u strukturu peptida i njihovu aktivnost te kroz grafovske reprezentacije, moguće je modelirati složene interakcije između aminokiselina.

Antiviralni peptidi su kratki lančani proteini koji se sastoje od aminokiselina koje imaju sposobnost suzbijanja virusnih infekcija te je jedan od takvih peptida vidljiv na slici 1.1. Oni mogu djelovati na različite načine kako bi inhibirali replikaciju virusa i spriječili širenje infekcije [4]. Njihova aktivnost, poput veze s ciljnim proteinima ili uloge u signalnim putevima, ključna je za razumijevanje bioloških mehanizama i razvoj novih lijekova [5]. Za potrebe izrade završnog rada, korišten je skup podataka koji uključuje više od tisuću antiviralnih peptida. Svaki peptid je praćen informacijom o njegovoj aktivnosti, koja je označena kao nula (neaktivnost) ili jedinica (aktivnost). Ova velika i dobro anotirana baza podataka omogućava precizno treniranje neuronskih mreža temeljenih na grafovima.



Slika 1.1 Prikaz jednog od mnogih antiviralnih peptida korištenih za treniranje modela

Poglavlje 2

Metodologija

2.1 Korišteni alati

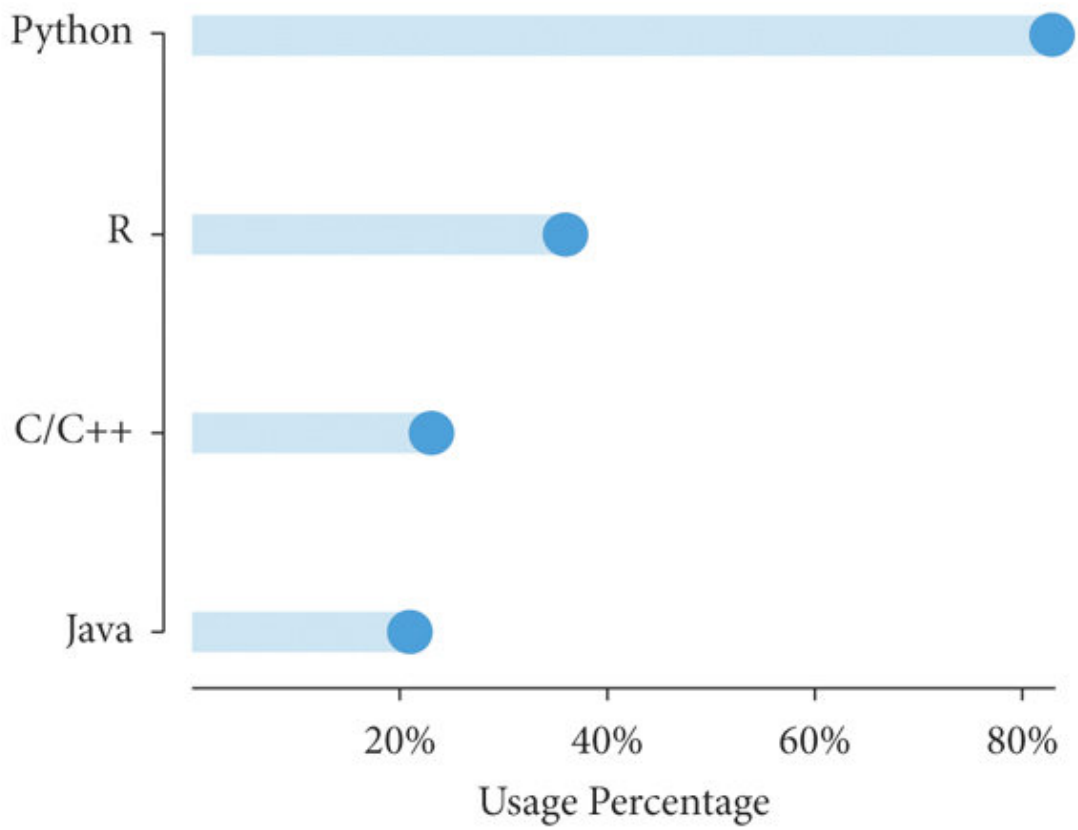
U ovom poglavlju opisat će se alati koji su korišteni u istraživanju aktivnosti molekula peptida. Osim alata spomenut će se skup podataka na kojem treniran model strojnog učenja te i same programske knjižnice i programski jezik bez kojih obrada podataka i stvaranje modela ne bi bila moguća.

2.1.1 Python

Python je programski jezik koji se ističe svojom jednostavnošću, čitljivošću i fleksibilnošću [6]. Jedan od glavnih razloga za popularnost Pythona je njegova jasna i intuitivna sintaksa koja omogućuje programerima da brzo pišu i čitaju kod. Osim toga, Python podržava razne paradigme programiranja poput proceduralnog, objektno orijentiranog i funkcionalnog programiranja, što olakšava razvoj različitih vrsta aplikacija.

Velika prednost Pythona su njegove brojne knjižnice. Python ekosustav nudi bogat izbor specijaliziranih knjižnica za različite domene kao što su web razvoj, baze podataka, znanstveno računanje, automatizacija, igre i mnoge druge. Ove knjižnice omogućuju programerima da brzo i jednostavno dodaju funkcionalnosti svojim projektima, čime se štedi vrijeme i olakšava razvojni proces.

Na slici 2.1 prikazana je zastupljenost uporabe programskih jezika u strojnom učenju i umjetnoj inteligenciji te se iz slike može vidjeti kako je Python najkorišteniji programski jezik. Svojom velikom zajednicom i obiljem knjižnica, kao što su NumPy, pandas, scikit-learn i TensorFlow, Python pruža snažne alate za obradu i analizu podataka, izgradnju modela strojnog učenja te duboko učenje. Ove knjižnice nude mnoge napredne funkcionalnosti koje olakšavaju implementaciju složenih algoritama i istraživanje velikih skupova podataka. Uz njih, programeri mogu koristiti velik broj gotovih funkcija i metoda za manipulaciju podacima, stvaranje vizualizacija i vrednovanje modela.



Slika 2.1 Zastupljenost uporabe programskih jezika za strojno učenje
Izvor: Usage of programming languages for machine learning and data science - ResearchGate [7]

2.1.2 PyMOL

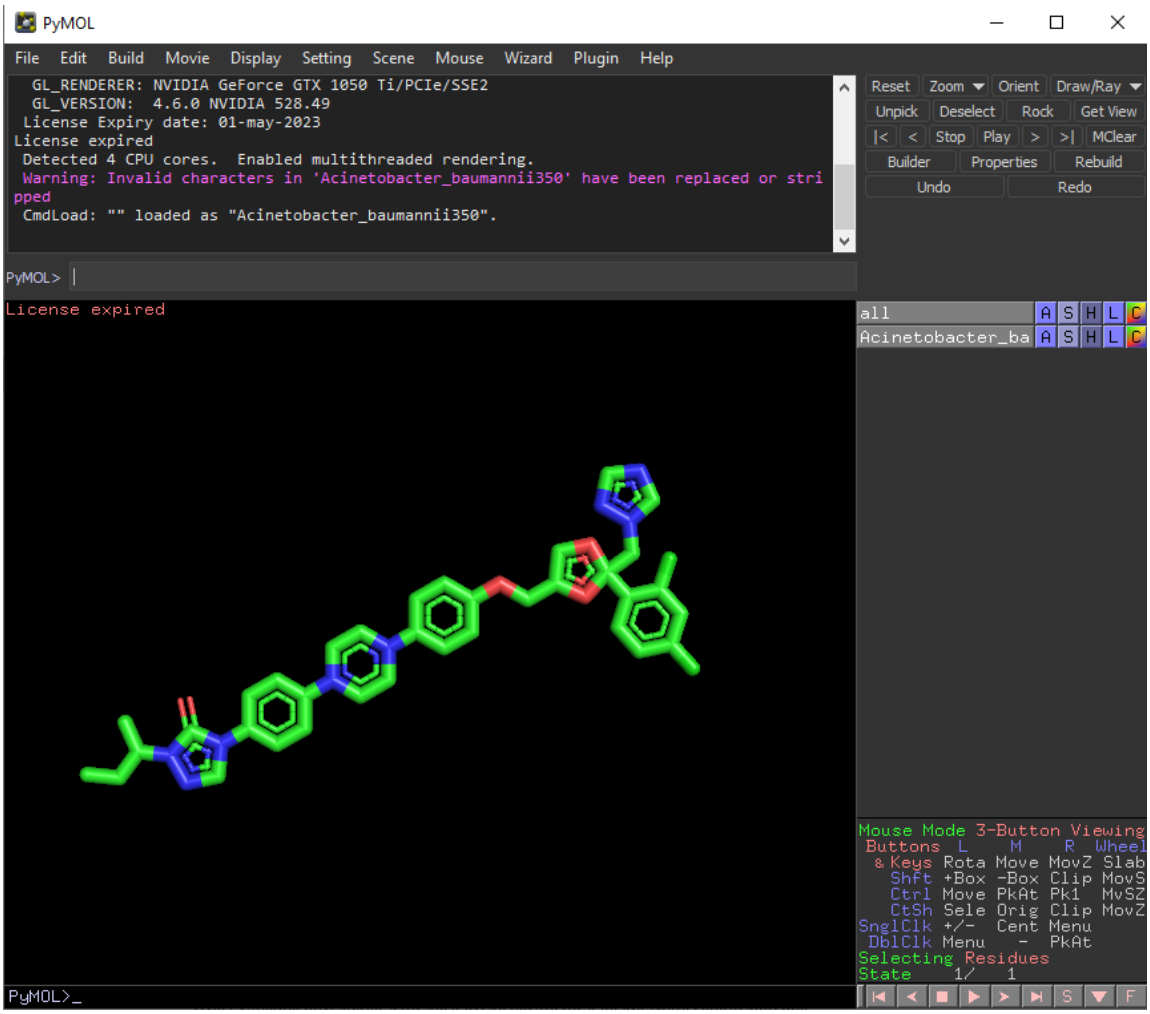
PyMOL je popularni alat za vizualizaciju molekula koji se često koristi u područjima kao što su biokemija, strukturna biologija i računalna kemija [8]. Na slici 2.2 prikazano je PyMOL grafičko sučelje za prikazivanje proteinskih i kemijskih struktura u 3D prostoru unutar kojeg se korisniku pružaju alati za njihovu manipulaciju i analizu.

Jedna od ključnih funkcionalnosti PyMOL-a je mogućnost prikaza molekula na interaktivan način. Korisnici mogu rotirati, zumirati i pomicati molekule kako bi istražili njihovu strukturu i svojstva. Također je moguće prikazati različite vizualne stilove, uključujući bojanje molekula prema atomima, lancima ili funkcionalnim skupinama. Ova fleksibilnost omogućava istraživačima da vizualno istaknu određene aspekte molekula i dobiju dublje razumijevanje njihove strukture.

PyMOL također pruža mogućnosti označavanja molekula, što omogućava dodavanje tekstualnih oznaka i anotacija koje pomažu u identifikaciji ključnih dijelova molekula. Ovo je posebno korisno u analizi proteinskih struktura gdje se može označiti aktivno mjesto ili specifične aminokiseline.

Dodatno, PyMOL podržava animaciju molekula, što omogućava prikazivanje promjena u strukturi tijekom vremena. To je korisno za proučavanje konformacijskih promjena proteina, interakcija molekula ili dinamike reakcija.

Još jedna važna značajka PyMOL-a je mogućnost snimanja vizualizacija u različitim formatima, poput slika, animacija ili čak interaktivnih sesija. Ovo olakšava dokumentiranje i dijeljenje rezultata vizualne analize s drugim znanstvenicima.



Slika 2.2 PyMol alat za prikaz kemijskih struktura u prostoru

2.1.3 RDKit

RDKit je knjižnica otvorenog koda za kemoinformatiku koja omogućava znanstvenicima rad s molekulama na programski način [9]. Ova moćna knjižnica pruža širok spektar alata i funkcionalnosti za generiranje, manipulaciju i analizu kemijskih struktura.

Jedna od ključnih značajki RDKit-a je njegova sposobnost izrade molekula i manipulacije njihovom strukturom. Korisnici mogu stvarati molekule iz raznih formata poput pojednostavljenog sustav za unos molekularnih linija (eng. Simplified Molecular Input Line Entry System, SMILES) ili drugih, kao i mijenjati postojeće molekule dodavanjem, uklanjanjem ili mijenjanjem atoma, veza i funkcionalnih grupa. Također je moguće izvoditi kemijske reakcije nad molekulama kako bi se istraživali kemijski procesi ili stvarale nove spojeve. Na slici 2.3 prikazan je dio programskog koda koji koristi RDKit knjižnicu i njezine funkcije u konverziji SMILES zapisa u molekule iz kojih se dobivaju atomi i njihove značajke koje će se koristiti prilikom stvaranja ulaza u model strojnog učenja.

RDKit također pruža funkcionalnosti za računanje molekulskih svojstava, poput molekulske mase, logP vrijednosti (lipofilnosti), toplinskih i elektronskih svojstava. Ove informacije su ključne za razumijevanje i predviđanje biološke aktivnosti molekula, što je važno u područjima kao što su istraživanje novih lijekova i virtualno probijanje.

Knjižnica RDKit također omogućava pretraživanje kemijskog prostora, što je korisno za pronalaženje sličnih molekula ili generiranje novih molekula s određenim svojstvima. Može se koristiti za razvoj kemijskih modela, kvantitativnu strukturu-aktivnost vezu (QSAR) analizu i virtualno probijanje.

RDKit podržava izvoz molekularnih struktura u različitim formatima, kao što su SMILES, datoteka strukturnih podataka (eng. Structure-Data File, SDF), proteinska baza podataka (eng. Protein Data Bank, PDB) i drugi. Ovo omogućava znanstvenicima da dijele i prenose molekularne podatke među različitim alatima i platformama.

```python
import rdkit.Chem
from rdkit import Chem

# Konvertiranje SMILES nizova u RDKit molekule
mol = Chem.MolFromSmiles(smileString)

# Dohvaćanje atomskih svojstava iz molekule
atoms = mol.GetAtoms()
for atom in atoms:
    degree = atom.GetDegree()
    formal_charge = atom.GetFormalCharge()
    num_radical_electrons = atom.GetNumRadicalElectrons()
    hybridization = atom.GetHybridization().real
    aromatic = atom.GetIsAromatic()
```

Slika 2.3 Primjena RDKit knjižnice u programskog kodu

2.1.4 StellarGraph

StellarGraph je knjižnica za grafove koja omogućava analizu i učenje na grafovima [10]. Ova knjižnica podržava različite vrste grafova, uključujući socijalne mreže, biološke mreže, znanstvene mreže i druge vrste grafova. StellarGraph pruža alate za izgradnju, pretraživanje i analizu grafova, kao i primjenu različitih algoritama strojnog učenja na grafove.

Jedna od ključnih značajki StellarGraph-a je mogućnost izgradnje grafova iz različitih izvora podataka. Grafovi se mogu konstruirati iz strukturiranih podataka poput DataFrames, izravno izvan mrežnih izvora poput datoteka koje sadrže vrijednosti odvojene zarezom (eng. Comma Separated Values, CSV) ili baza podataka. Ovo omogućava fleksibilno i intuitivno modeliranje različitih grafičkih domena. Na slici 2.4 prikazan je isječak programskog koda koji prikazuje stvaranje grafa koristeći DataFrame objekt koji sadrži informacije o povezanosti atoma i listu značajki za svaki od tih atoma. Kreirani grafovi se koriste kao ulaz u model strojnog učenja.

StellarGraph pruža alate za pretraživanje grafova i analizu njihove strukture. Možete pristupiti čvorovima, bridovima i njihovim svojstvima te izvoditi razne operacije nad grafom. Ovo je korisno za izvlačenje važnih informacija, otkrivanje zajednica, pronalaženje centralnih čvorova i druge analitičke zadatke.

Osim toga, StellarGraph omogućava primjenu različitih algoritama strojnog učenja na grafove. To uključuje algoritme za predviđanje veza između čvorova, klasteriranje čvorova, klasifikaciju čvorova i preporuku temeljenu na grafovima. Možete primijeniti ove algoritme na različite probleme, poput otkrivanja povezanih čvorova u socijalnim mrežama, identifikacije bioloških putova u biološkim mrežama ili personalizirane preporuke u sustavima preporuka.

StellarGraph također pruža integraciju s drugim popularnim knjižnicama strojnog učenja, poput TensorFlow-a i scikit-learn-a, omogućavajući nam da kombinirate snagu analize grafova s ostalim alatima strojnog učenja.

```python
from stellargraph.mapper import PaddedGraphGenerator
from stellargraph.layer import DeepGraphCNN
from stellargraph import StellarGraph

# Kreiranje StellarGraph objekta iz grafa
G = StellarGraph(nodes=node_features, edges=edges_df)

# Generator grafova za obradu podataka
generator = PaddedGraphGenerator(graphs=graphs)
gen = generator.flow(X_train, targets=y_train, batch_size=32, symmetric_normalization=False)
```

Slika 2.4 Primjena StellarGraph knjižnice u programskog kodu

2.1.5 Pandas

Pandas je knjižnica za analizu podataka u programskom jeziku Python [11]. Na slici 2.5 prikazan je uporaba Pandas knjižnica koja pruža visoko izražajne i efikasne strukture podataka, posebno DataFrame, koje omogućavaju manipulaciju, transformaciju i analizu podataka na jednostavan način. Pandas je ključni alat za čišćenje, pripremu i obradu podataka u mnogim područjima.

Jedna od ključnih značajki Pandas-a je DataFrame, tablična struktura podataka s fleksibilnim mogućnostima indeksiranja i označavanja. DataFrame omogućava učinkovitu manipulaciju podacima poput čišćenja, filtriranja, transformacije i agregacije. Možete lako izvoditi različite operacije na podacima, kao što su izračuni statističkih mjera, spajanje i grupiranje podataka, te generiranje novih varijabli na temelju postojećih podataka.

Pandas također podržava čitanje i pisanje podataka iz različitih formata, uključujući CSV, Excel, SQL baze podataka i mnoge druge. Možete jednostavno učitavati podatke iz vanjskih izvora, kao i izvoziti podatke u različitim formatima. Ovo je vrlo korisno prilikom radnje s različitim izvorima podataka i integracije s drugim alatima.

Pandas se često koristi u istraživanju podataka, analizi, pripremi podataka za strojno učenje i vizualizaciji. Omogućava brz pristup i obradu velikih skupova podataka, kao i generiranje statističkih izvještaja i vizualizacija. Pandas je također integriran s drugim popularnim knjižnicama za analizu podataka i strojno učenje, što omogućava kombiniranje njegovih mogućnosti s drugim alatima.

```python
import pandas as pd

# Čitanje podataka iz Excel datoteke
data_file = pd.read_excel(filepath_raw, header=0, usecols=["smiles", "label"])

# Kreiranje DataFrame objekta iz liste torki
data_file.reset_index()
listOfTuples = []
for index, row in data_file.iterrows():
    smiles = row['smiles']
    label = row["label"]
    molecule = (row["smiles"], row["label"])
    listOfTuples.append(molecule)

# Pretvorba DataFrame objekta u Seriju
graph_labels = pd.Series(labels)

# Grupiranje i brojanje vrijednosti u Seriji
print(graph_labels.value_counts().to_frame())
```

Slika 2.5 Primjena Pandas knjižnice u programskog kodu

2.1.6 Tensorflow

TensorFlow je popularna knjižnica otvorenog koda za strojno učenje koju je razvio Google [12]. Ona pruža moćne alate za izgradnju, treniranje i vrednovanje različitih modela strojnog učenja, uključujući neuronske mreže i duboko učenje. TensorFlow je vrlo fleksibilan i podržava različite složene operacije i algoritme strojnog učenja, omogućavajući korisnicima eksperimentiranje s modelima i rješavanje različitih probleme u područjima kao što su prepoznavanje uzoraka, obrada prirodnog jezika, računalni vid i mnogi drugi. Na slici 2.6 prikazana je primjena Tensorflow knjižnice prilikom stvaranje modela, kompilacije te definiranja funkcije za treniranja samog modela.

Jedna od ključnih značajki TensorFlow-a je njegova mogućnost definiranja računskih grafova. Računski grafovi omogućavaju nam izražavanje složenih operacija i modela strojnog učenja kao grafova čvorova i bridova, gdje čvorovi predstavljaju matematičke operacije, a bridovi prijenose podatke između čvorova. Ova fleksibilnost omogućava nam izgradnju raznovrsnih modela s više slojeva i složenim arhitekturama.

TensorFlow pruža bogat skup ugrađenih alata i slojeva za izgradnju modela strojnog učenja. Možete koristiti ugrađene slojeve za konstrukciju neuronskih mreža, kao i alate za optimizaciju, regularizaciju i vrednovanje modela. TensorFlow također podržava mogućnost treniranja modela na više procesora ili grafičkih procesora (GPU) radi ubrzanja izračuna.

Osim toga, TensorFlow ima veliku zajednicu korisnika i razvojnih resursa. Postoji mnogo dostupnih pred-treniranih modela koje možete koristiti ili prilagoditi svojim potrebama. TensorFlow također pruža alate za vizualizaciju modela, praćenje napretka treniranja i vrednovanje performansi modela.

```python
import tensorflow as tf
from tensorflow.keras import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dense, Conv1D, MaxPool1D, Dropout, Flatten
from tensorflow.keras.losses import binary_crossentropy

# Definiranje modela
model = Model(inputs=x_inp, outputs=predictions)

# Kompiliranje modela
model.compile(
    optimizer=Adam(lr=0.0001),
    loss=binary_crossentropy,
    metrics=["acc"]
)

# Treniranje modela
history = model.fit(
    train_gen, epochs=epochs, verbose=1, validation_data=val_gen, shuffle=True, callbacks=[callback]
)

# Evaluiranje modela na testnom skupu
test_metrics = model.evaluate(test_gen)
print("\nTest Set Metrics:")
for name, val in zip(model.metrics_names, test_metrics):
    print("\t{}: {:0.4f}".format(name, val))
```

Slika 2.6 Primjena Tensorflow knjižnice u programskog kodu

2.1.7 Scikit-learn

Scikit-learn je popularna knjižnica za strojno učenje u Pythonu koja pruža širok spektar alata i algoritama za različite zadatke strojnog učenja [13]. Ova knjižnica je dizajnirana kako bi bila jednostavna za korištenje, ali istovremeno moćna i fleksibilna.

Scikit-learn podržava razne vrste zadatka strojnog učenja, uključujući klasifikaciju, regresiju, grupiranje, smanjenje dimenzionalnosti i još mnogo toga. Nudi veliki broj ugrađenih algoritama, kao što su slučajne šume, potporni vektori, neuronske mreže, regresijski modeli, algoritmi za grupiranje i još mnogo toga. Ovi algoritmi su implementirani na efikasan način i nude različite konfiguracijske opcije i parametre za prilagodbu modela prema potrebama.

Jedna od glavnih prednosti Scikit-learn-a je njegova funkcionalnost za vrednovanje modela. Na slici 2.7 prikazana je primjena knjižnice i njenih alata za izračunavanje različitih metrika performansi modela, kao što su točnost, preciznost, odziv, F1 mjera, itd. Također nudi mogućnosti za validaciju modela pomoću unakrsne validacije i optimizaciju parametara modela kako bi se postigla bolja performansa.

Scikit-learn također sadrži funkcionalnosti za pretprocesiranje podataka i odabir značajki. Možete koristiti različite tehnike za normalizaciju, standardizaciju ili skaliranje podataka, kao i za manipulaciju nedostajućim vrijednostima ili izdvajanje važnih značajki. Ove funkcionalnosti su ključne za pripremu podataka prije izgradnje modela.

```python
from sklearn import model_selection
from sklearn.metrics import confusion_matrix, precision_score, recall_score, roc_auc_score, roc_curve,
    f1_score

# Podjela podataka na skupove za treniranje i testiranje
train_index, test_index = model_selection.train_test_split(range(len(graphs)), test_size=0.2, stratify
    =graph_labels, random_state=42)
X_train, X_test = graphs[train_index], graphs[test_index]
y_train, y_test = graph_labels.iloc[train_index], graph_labels.iloc[test_index]

# Treniranje modela i evaluacija na testnom skupu
model.fit(train_gen, epochs=epochs, verbose=1, validation_data=val_gen, shuffle=True, callbacks
    =[callback])

# Evaluacija modela na testnom skupu
y_pred = model.predict(test_gen)
y_pred = [0 if prob < 0.5 else 1 for prob in y_pred]

# Izračunavanje različitih metrika performansi
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
fpr = fp / (fp + tn)
tpr = tp / (tp + fn)
tnr = tn / (tn + fp)
gm = math.sqrt(tpr * tnr)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)
```

Slika 2.7 Primjena Scikit-learn knjižnice u programskog kodu

2.2 Skup podataka

Skup podataka (eng. dataset) koji je korišten za ovaj rad je prilagođeni skup AVPdb unutar kojeg su molekule zapisane u formatu SMILES te je svakoj molekuli pridružena oznaka koja pruža informaciju o njegovoj aktivnosti u obliku nule (neaktivnost) ili jedinice (aktivnost) [14]. Prije početka izrade modela strojnog učenja bilo je nužno napraviti dobru analizu podataka unutar skupa kako bi se osiguralo da skup ima ravnomjernu zastupljenost molekula s pozitivnim i negativnim aktivnostima, čime bi se izbjegao problem nepoželjne pristranosti prema jednoj od klasa. U tablici 2.1 prikazana je distribucija pozitivne i negativne klase antiviralnih peptida iz koje se može uočiti približno jednolika raspodjela, što je povoljan čimbenik za treniranje modela strojnog učenja.

Tablica 2.1 Analiza skupa podataka antiviralnih peptida AVPdb

Oznaka klase	negativna (0)	pozitivna (1)
Ukupan broj peptida	444	598
Minimalni broj atoma u molekulama	42	82
Maksimalni broj atoma u molekulama	380	772
Srednja vrijednost broja atoma u molekulama	145	198
Srednja vrijednost broja veza između atoma	295	402

Poglavlje 3

Analiza modela strojnog učenja

3.1 Pristup, obrada i stvaranje grafova temeljenih na molekulama

U ovom dijelu implementacije provodi se detaljna obrada početnih SMILES stringova peptida. Na slici 3.1 prikazan je postupak kojemu je cilj izvući važne informacije o molekulama i stvoriti potrebne značajke za konstrukciju StellarGraph grafova. Prilikom iteracije kroz svaki SMILES string, vrši se analiza svojstava i značajki za svaki atom unutar molekule. Ova obrada uključuje određivanje stupnja atoma, koji predstavlja broj veza koje atom uspostavlja s drugim atomima u molekuli. Također se bilježe formalni naboj atoma, što ukazuje na električni naboj koji atom posjeduje. Broj radikalnih elektrona se također računa kako bi se utvrdio broj elektrona koji sudjeluje u neuparenim spinovima atoma.

Dodatno, vrši se analiza hibridizacije atoma, koja opisuje geometriju i raspored orbitala oko atoma. Hibridizacija može biti sp, sp2 ili sp3, pružajući važne informacije o kemijskim svojstvima atoma. Također se provjerava je li atom aromatičan, tj. pripada li aromatičnom prstenu, što može imati poseban utjecaj na reaktivnost i svojstva molekule. Također, bilježe se svi jedinstveni elementi koji se pojavljuju unutar molekula. To omogućuje izgradnju rječnika koji preslikava svaki element u jedinstveni indeks, što je korisno za daljnju obradu i reprezentaciju elemenata u obliku vektora.

```
for index, row in data_file.iterrows():
    smiles = row['smiles']
    label = row["label"]
    molecule = (row["smiles"], row["label"])
    listOfTuples.append(molecule)


for molecule in listOfTuples:
    mol = Chem.MolFromSmiles(molecule[0])
    atoms = mol.GetAtoms()
    for atom in atoms:
        lst_degree.append(atom.GetDegree())
        lst_formal_charge.append(atom.GetFormalCharge())
        lst_radical_electrons.append(atom.GetNumRadicalElectrons())
        lst_hybridization.append(atom.GetHybridization().real)
        lst_aromatic.append(atom.GetIsAromatic())

        element = atom.GetSymbol()
        if element not in all_elements:
            all_elements.append(element)
```

Slika 3.1 Slika koda za obradu molekula

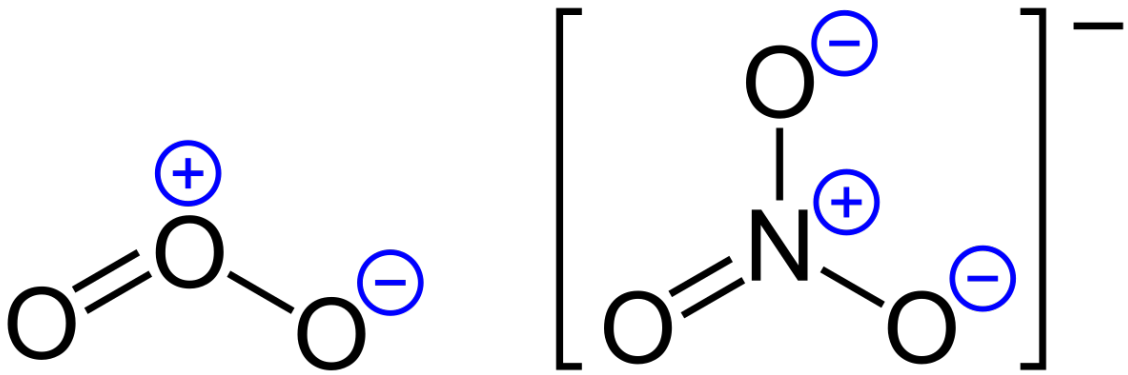
Stupanj atoma

Stupanj atoma je značajka koja se određuje tijekom obrade molekula [15]. On predstavlja broj veza koje atom uspostavlja s drugim atomima u molekuli. Stupanj atoma pruža informaciju o povezanosti i interakcijama atoma unutar molekularne strukture. Identificiranje stupnja atoma omogućava bolje razumijevanje topologije i reaktivnosti molekule.

Formalni naboj atoma

Formalni naboj atoma je još jedna značajka koja se dobiva tijekom obrade molekula [16]. On predstavlja električni naboj koji atom posjeduje. Formalni naboj atoma može biti pozitivan, negativan ili neutralan, ovisno o broju elektrona koje atom prima ili donira tijekom kemijskih reakcija. Na slici 3.2 prikazan je primjer zapisa

formalnog naboja koji pruža uvid u elektrostatičke interakcije unutar molekule i može biti ključna za predviđanje kemijskih svojstava i reaktivnosti.



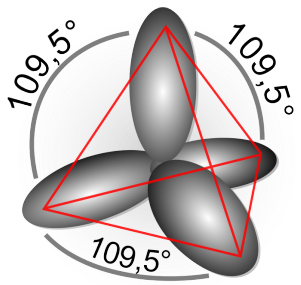
Slika 3.2 Primjer zapisa formalnog naboja atoma na molekuli nitrata

Broj radikalnih elektrona

Broj radikalnih elektrona je značajka koja se također analizira prilikom obrade molekula [17]. On predstavlja broj elektrona koji sudjeluju u neuparenim spinovima atoma. Neupareni elektroni igraju važnu ulogu u reaktivnosti molekula, jer mogu sudjelovati u kemijskim reakcijama i interakcijama s drugim atomima. Identifikacija broja radikalnih elektrona omogućuje bolje razumijevanje stabilnosti i reaktivnosti molekula.

Hibridizacija atoma

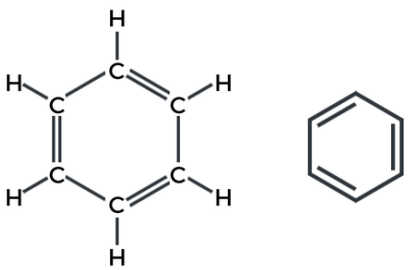
Hibridizacija atoma je značajka koja opisuje geometriju i raspored orbitala oko atoma [18]. Ona može biti sp, sp2 ili sp3, ovisno o načinu kombiniranja orbitala unutar atomskog sustava. Hibridizacija pruža informacije o kemijskim svojstvima atoma, kao što su geometrija veza, kutovi veza, koji su prikazani na slici 3.3, i reaktivnost. Analiza hibridizacije atoma doprinosi boljem razumijevanju strukture i svojstava molekula.



Slika 3.3 Opis geometrije i rasporeda orbitala atoma kroz hibridizaciju

Aromatičnost atoma

Aromatičnost atoma je još jedna važna značajka koja se bilježi tijekom obrade molekula [19]. Aromatičnost se odnosi na prisutnost aromatičnih prstena u molekuli te je jedan takav prsten prikazan na slici 3.4. Aromatični prstenovi imaju posebnu stabilnost i svojstva koja ih razlikuju od ostalih dijelova molekule. Identificiranje aromatičnih atoma omogućuje bolje razumijevanje elektronske strukture, konjugacije i reaktivnosti molekule.



Slika 3.4 Aromatični prsten molekule benzena

3.1.1 Normiranje značajki

U drugom dijelu implementacije, provodi se određivanje minimalnih i maksimalnih vrijednosti za svako svojstvo/značajku kako bi se omogućilo skaliranje tih vrijednosti. Za svaki SMILES string prisutan u datoteci, SMILES se konvertira u molekularni graf koristeći RDKit knjižnica. Iz dobivenog grafa se izvlače informacije o atomima i vezama, a zatim se te informacije pretvaraju u značajke čvorova koje će biti korištene u konstrukciji StellarGraph objekta.

Određivanje minimalnih i maksimalnih vrijednosti za svojstva i značajke omogućuje nam stvaranje raspona za normalizaciju tih vrijednosti. Na slici 3.5 prikazan je korak normiranja kojim se osigurava da sve vrijednosti budu na istoj skali, što olakšava daljnju analizu i obradu podataka. Skaliranje podataka omogućuje modelu da učinkovito koristi sve dostupne informacije bez da bude preosjetljiv na pojedinačna svojstva s većim rasponima vrijednosti.

Nakon dobivanja molekularnog grafa iz SMILES stringa, vrši se ekstrakcija atoma i veza. Ti podaci se zatim pretvaraju u značajke čvorova, koje predstavljaju relevantne informacije o atomima. Primjerice, značajke čvorova mogu uključivati stupanj atoma, hibridizaciju, aromatičnost i druge karakteristike koje su bitne za analizu molekula. Pretvaranje podataka u značajke čvorova omogućuje modelu da razumije strukturu molekula i koristi tu informaciju za daljnju obradu i učenje.

```
for molecule in listOfTuples:
    mol = Chem.MolFromSmiles(molecule[0])
    atoms = mol.GetAtoms()
    for atom in atoms:
        lst_degree.append(atom.GetDegree())
        lst_formal_charge.append(atom.GetFormalCharge())
        lst_radical_electrons.append(atom.GetNumRadicalElectrons())
        lst_hybridization.append(atom.GetHybridization().real)
        lst_aromatic.append(atom.GetIsAromatic())

        element = atom.GetSymbol()
        if element not in all_elements:
            all_elements.append(element)

min_degree, max_degree = min(lst_degree), max(lst_degree)
min_formal_charge, max_formal_charge = min(lst_formal_charge), max(lst_formal_charge)
min_radical_electrons, max_radical_electrons = min(lst_radical_electrons), max(lst_radical_electrons)
min_hybridization, max_hybridization = min(lst_hybridization), max(lst_hybridization)
min_aromatic, max_aromatic = min(lst_aromatic), max(lst_aromatic)
```

Slika 3.5 Slika koda za normiranje značajki

3.1.2 Stvaranje grafa

U posljednjem koraku implementacije, izvedene veze iz molekularnog grafa transformiraju se u oblik prikladan za stvaranje StellarGraph objekta. Koristeći pandas knjižnicu, veze se organiziraju u DataFrame strukturu koja čuva informacije o izvoru i odredištu svake veze. Ovaj format omogućuje daljnju manipulaciju i analizu veza unutar grafa.

StellarGraph objekt koji se stvara u ovom koraku predstavlja cjelokupni molekularni graf. Sastoji se od čvorova koji sadrže relevantne značajke, poput stupnja atoma, hibridizacije, aromatičnosti i drugih svojstava. Također, objekt sadrži informacije o vezama između čvorova, definirajući njihovu prostornu i strukturnu organizaciju. Ova reprezentacija omogućuje dublju analizu molekularnih podataka i primjenu različitih algoritama strojnog učenja na grafove.

Na slici 3.6 prikazan je isječak programskog koda zadužen za stvaranje StellarGraph objekta iz DataFrame objekta unutar kojeg su zapisane veze između atoma i liste značajki atoma. Stvaranje StellarGraph objekta predstavlja završni korak obrade podataka i pripreme za daljnju analizu. Dobiveni objekt pruža snažan te-

melj za razumijevanje strukture molekula i izvođenje složenih analiza na grafovima. Ovisno o potrebama istraživanja, moguće je primijeniti različite algoritme strojnog učenja na objektu, kao što su klasifikacija, regresija, grupiranje ili preporuka temeljena na grafovima.

```
mol = Chem.MolFromSmiles(smileString)
atoms = mol.GetAtoms()
edges = []
for bond in mol.GetBonds():
    edges.append((bond.GetBeginAtomIdx(), bond.GetEndAtomIdx()))
    edges.append((bond.GetEndAtomIdx(), bond.GetBeginAtomIdx()))

edges_df = pd.DataFrame(edges, columns=["source", "target"])

G = StellarGraph(nodes=node_features, edges=edges_df)
```

Slika 3.6 Slika koda za kreiranje grafa

3.2 Arhitektura modela neuronske mreže

U ovom poglavlju detaljno će biti opisani koraci za postavljanje modela, uključujući definiranje slojeva i parametara čijim se izmjenama utječe na performanse modela te stvaranje ulaza koji će se koristiti za treniranje modela. Kako bismo mogli koristiti obrađene i transformirane podatke kao ulaz u model strojnog učenja, ključno je stvoriti generator podataka.

3.2.1 Priprema generatora grafova

U kontekstu nadziranog klasificiranja grafova, koristimo instancu klase PaddedGraphGenerator iz StellarGraph knjižnice. Ovaj generator podataka omogućuje pripremu nizova značajki i matrica susjedstva za klasifikaciju grafova u malim grupama. Kako grafovi mogu imati različit broj čvorova, problem se rješava dodavanjem nadopune za svaku grupu značajki i matrica susjedstva. Također, koristi se logička maska koja

označava koje vrijednosti su valjane, a koje su nadopuna. Tako se osigurava dosljednost podataka i ispravno rukovanje različitim veličinama grafova tijekom treniranja modela za klasifikaciju grafova.

3.2.2 Neuronska mreža za predviđanje aktivnosti peptida

Za izradu modela za klasifikaciju grafova koristimo klasu DeepGraphCNN iz StellarGraph knjižnice zajedno sa standardnim slojevima poput Conv1D, MaxPool1D, Dropout i Dense.

Na slici 3.7 prikazan je isječak programskog koda unutar kojeg se prvo definira broj redaka za izlazni tenzor i veličine slojeva. Zatim se stvara DeepGraphCNN model, koristeći zadane veličine slojeva, aktivacije i ostale parametre. Ulazni i izlazni tenzori se dobivaju iz modela.

Nakon toga slijedi niz slojeva koji transformiraju izlazni tenzor. Prvi je Conv1D sloj koji primjenjuje konvoluciju na tenzoru s određenim brojem filtera i veličinom jezgre. Zatim se primjenjuje MaxPool1D sloj koji vrši maksimalno agregiranje podataka. Sljedeći je Conv1D sloj koji ponovno primjenjuje konvoluciju s drugačijim filterima i veličinom jezgre.

Nakon konvolucijskih slojeva slijedi sloj Flatten koji transformira podatke kako bi se pripremili za sljedeće slojeve. Zatim slijedi Dense sloj koji primjenjuje potpuno povezane neurone s određenim brojem jedinica i aktivacijom ReLU. Nakon toga slijedi Dropout sloj koji služi za regularizaciju mreže, s parametrom koji određuje koji postotak neurona će biti isključen tijekom treniranja.

Konačno, stvara se izlazni sloj Dense s jedinicom za binarnu klasifikaciju i aktivacijom sigmoid. Model se stvara koristeći ulazne i izlazne tenzore te je shema modela prikazana na slici 3.8.

```python
# Definiranje brojeva redova izlaznog tenzora i veličina slojeva
k = 15
layer_sizes = [10, 10, 10, 1]

# Kreiranje DeepGraphCNN model
dgcnn_model = DeepGraphCNN(
    layer_sizes=layer_sizes,
    activations=["tanh", "tanh", "tanh", "tanh"],
    k=k,
    bias=False,
    generator=generator,
)
x_inp, x_out = dgcnn_model.in_out_tensors()

x_out = Conv1D(filters=8, kernel_size=sum(layer_sizes), strides=sum(layer_sizes))(x_out)
x_out = MaxPool1D(pool_size=2)(x_out)

x_out = Conv1D(filters=16, kernel_size=5, strides=1)(x_out)

x_out = Flatten()(x_out)

x_out = Dense(units=64, activation="relu")(x_out)
x_out = Dropout(rate=0.2)(x_out)

predictions = Dense(units=1, activation="sigmoid")(x_out)

model = Model(inputs=x_inp, outputs=predictions)
```

Slika 3.7 Implementacija neuronske mreže u programskog kodu

Konvolucijski slojevi

Nakon stvaranja modela, slijedi niz slojeva za konvoluciju kojima se primjenjuje konvoluciju na ulaznom tenzoru. Broj filtera je parametar kojim određujemo koliko filtera će se primijeniti na podatke. Veličina jezgre označava veličinu koja se koristi za konvoluciju. Korak određuje razmak između primjene jezgre na podatke. Konvolucijski slojevi omogućuju modelu da nauči karakteristike ulaznih podataka.

Max pooling sloj

Nakon konvolucijskog sloja slijedi sloj maksimalnog agregiranja (eng. max pooling), koji smanjuje dimenzionalnost izlaza konvolucijskog sloja zadržavajući samo najvažnije informacije. Parametar za veličinu agregiranja određuje koliko podataka će biti uzorkovano za agregiranje.

Dodatni konvolucijski sloj

Nakon sloja maksimalnog agregiranja, dodajemo još jedan sloj konvolucije, koji nazivamo Conv1D, kako bismo omogućili modelu da nauči dodatne složene značajke iz prethodno obrađenih podataka. U ovom sloju koristimo parametre poput broja filtera, veličina jezgre i korak kako bismo odredili kako će se konvolucija primijeniti na podatke.

Flatten sloj

Nakon konvolucijskih slojeva, koristimo Flatten sloj koji transformira podatke iz višedimenzionalnog oblika u jednodimenzionalan vektor. Ovaj korak je potreban kako bismo pripremili podatke za sljedeće slojeve potpuno povezane neuronske mreže.

Potpuno povezani slojevi

Nakon Flatten sloja slijedi niz potpuno povezanih (Dense) slojeva. Potpuno povezani slojevi imaju veze između svih neurona iz prethodnog sloja i trenutnog sloja. Svaki neuron u ovim slojevima računa svoj izlaz na temelju težina veza s prethodnim slojem i aktivacijske funkcije. Koristimo Dense slojeve s definiranim brojem jedinica (neurona) i aktivacijskom funkcijom relu koja pomaže modelu da nauči složene značajke i donese odgovarajuće odluke.

Dropout sloj

Nakon prvog Dense sloja dodajemo Dropout sloj koji služi za regularizaciju mreže. Dropout slučajno isključuje određeni postotak neurona tijekom treniranja, čime se

smanjuje prenaučenost modela. Parametar "rate" određuje postotak neurona koji će biti isključeni.

Izlazni sloj

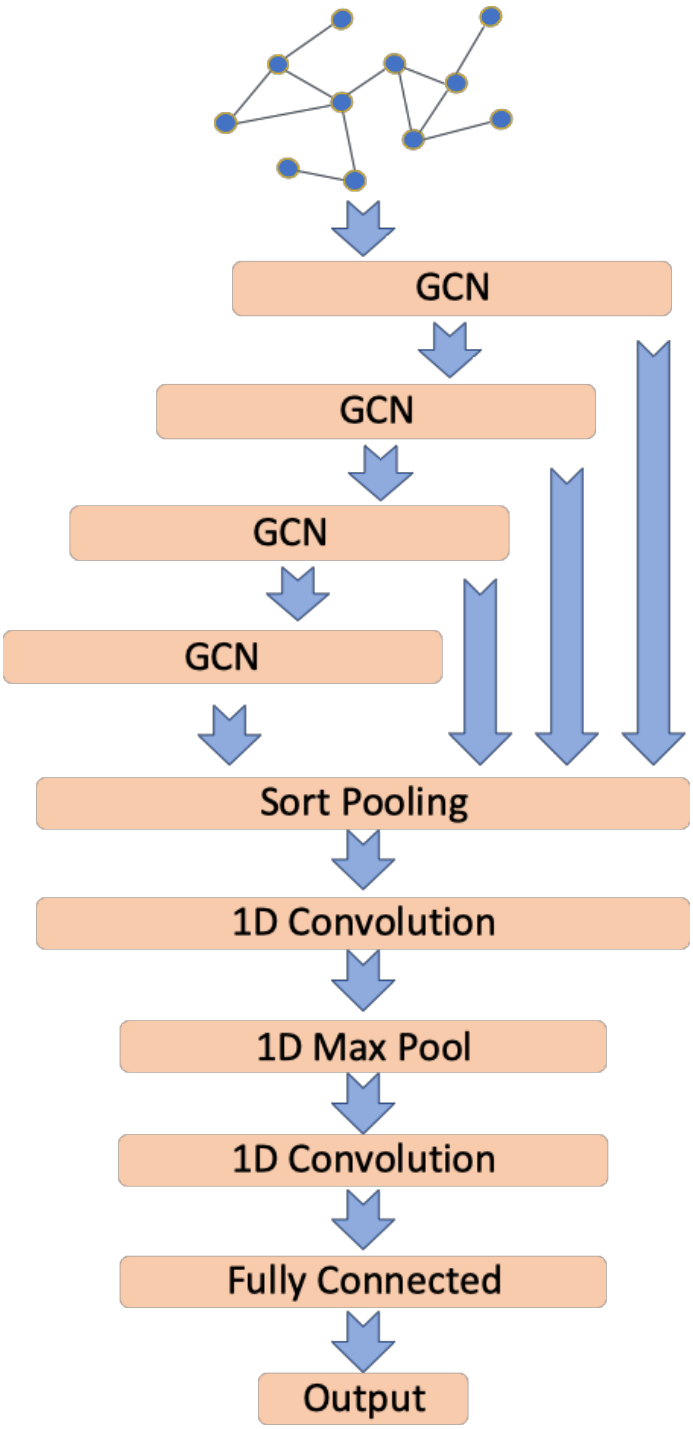
Zadnji sloj je Dense sloj s jedinicom za binarnu klasifikaciju i aktivacijskom funkcijom sigmoid. Ovaj sloj daje predviđanja za klasifikaciju. Sigmoidna funkcija pretvara izlazne vrijednosti u raspon [0, 1], što omogućuje interpretaciju rezultata kao vjerojatnosti pripadnosti različitim klasama.

Parametri modela

Pri definiranju modela za predviđanje aktivnosti peptida, koristimo dva važna parametra - "k" i "layer sizes".

Parametar "k" određuje broj redaka za izlazni tenzor. Izlazni tenzor će imati "k" redaka, dok će broj stupaca biti određen dimenzijama ulaznih podataka. Ovaj parametar omogućuje prilagodbu dimenzija izlaza modela prema specifičnim zahtjevima zadatka.

"Layer sizes" je lista koja određuje veličinu svakog sloja u modelu. Svaki element u listi predstavlja broj jedinica (neurona) u odgovarajućem sloju. Na primjer, ako imamo "layer sizes" definirane kao [10, 10, 10, 1], to znači da prva tri sloja imaju 10 jedinica, dok zadnji sloj ima samo 1 jedinicu. Ove veličine slojeva utječu na složenost modela i njegovu sposobnost izvlačenja značajki iz ulaznih podataka.



Slika 3.8 Vizualna reprezentacija arhitekture neuronske mreže

3.3 Treniranje modela strojnog učenja

U ovom poglavlju temeljito ćemo istražiti fazu treniranja koja je jedna od ključnih faza modela strojnog učenja. Treniranje predstavlja vitalni korak u kojem naš model stječe sposobnost prepoznavanja uzoraka u podacima. U nastavku, obradit ćemo temeljne aspekte nužnih za procesa treniranja modela, te ćemo detaljno proučiti tehnike koje primjenjujemo s ciljem postizanja poboljšanja modela.

3.3.1 Kompilacija modela

Na slici 3.9 prikazan je isječak iz koda zadužen za postupak kompilacije u kojem se modelu dodjeljuju specifični parametri i funkcije koje će se koristiti tijekom treniranja. U navedenom kodu koristimo Adam optimizator, koji je popularan optimizator u strojnom učenju. Adam optimizator je algoritam optimizacije koji se često koristi u strojnom učenju i dubokom učenju. To je proširenje algoritma stohastičkog gradijentnog spusta i kombinira ideje adaptivnih gradijentnih algoritama i metoda za moment. Glavna ideja iza Adam optimizatora je računanje prilagodljivih stopa učenja za svaki parametar na temelju procjena prvog i drugog momenta gradijenata. Drugim riječima, prilagođava stopu učenja za svaki parametar pojedinačno na temelju njihovih povijesnih gradijenata.

Algoritam prati dva eksponencijalna pomična prosjeka gradijenata: prvi moment (srednja vrijednost) i drugi moment (necentrirana varijanca). Ovi prosjeci se ažuriraju pri svakoj iteraciji postupka optimizacije. Adam optimizator također uključuje mehanizam ispravljanja odstupanja kako bi kompenzirao činjenicu da su pomični prosjeci inicijalizirani kao pristrane procjene. Također, definiramo stopu učenja kojom kontroliramo brzinu prilagodbe parametara modela tijekom učenja.

Kao funkciju gubitka koristimo binarnu križanu entropiju. Binarna križana entropija je često korištena funkcija gubitka u binarnoj klasifikaciji. Ona mjeri razliku između stvarnih i predviđenih vjerojatnosti za svaki primjer težinskog skupa podataka. Cilj je minimizirati vrijednost ove funkcije kako bismo postigli što bolje rezultate klasifikacije.

Kao metriku koristimo točnost koja je jedna od najjednostavnijih i najčešće kori-

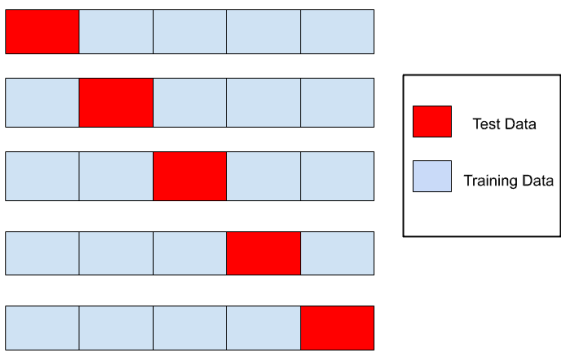
štenih metrika za procjenu performansi klasifikacijskog modela. Ona mjeri udio točno klasificiranih primjera u ukupnom broju primjera. Cilj je maksimizirati točnost kako bismo postigli što veću preciznost u predviđanju klasa.

```
model.compile(
    optimizer=Adam(lr=0.0001),
    loss=binary_crossentropy,
    metrics=["acc"]
)
```

Slika 3.9 Programski kod zadužen za kompilaciju modela

3.3.2 Treniranje i vrednovanje modela

Nakon kompilacije modela, slijedi postupak treniranja i vrednovanja. U ovom koraku koristimo unakrsnu validaciju kako bismo dobili stabilniju procjenu performansi modela. Na slici 3.10 prikazana je vizualna reprezentacija unakrsne validacije u k-preklopa koja podijeli skup podataka na k grupa, pri čemu se jedna grupa koristi kao testni skup dok se preostale grupe koriste kao skupovi za treniranje modela te se taj postupak ponavlja dok sve grupe nisu točno jednom bile upotrebljene kao testni skup. U kodu koristimo StratifiedKFold, koji održava ravnotežu klasa prilikom podjele podataka na grupe. Za svaku grupu, izdvajamo indekse trening i test podataka pomoću metode "split". Zatim koristimo ove indekse za izdvajanje odgovarajućih grafova i oznaka iz skupa podataka.



Slika 3.10 Podjela skupa podataka na grupe

Na slici 3.11 prikazana je isječak programskog koda unutar kojeg se koristi funkcija trainTestSplit za dodatno dijeljenje na skup za treniranje i skup za validaciju. Ovim osiguravamo da imamo odvojeni skup za praćenje performansi i sprečavanje prenaučenosti modela.

Generiramo instance trainGen, valGen i testGen pomoću PaddedGraphGenerator. Ove instance generiraju podatke u obliku prilagođenom za treniranje i vrednovanje modela. Primjenjujemo odgovarajuće veličine grupa i simetričnu normalizaciju prema potrebama modela.

```
14      gen = PaddedGraphGenerator(graphs=graphs)
15
16      trainGen = gen.flow(
17          XTrain,
18          targets=yTrain,
19          batchSize=32,
20          symmetricNormalization=False,
21      )
22
23      valGen = gen.flow(
24          XVal,
25          targets=yVal,
26          batchSize=50,
27          symmetricNormalization=False,
28      )
29
30      testGen = gen.flow(
31          XTest,
32          targets=yTest,
33          batchSize=50,
34          symmetricNormalization=False,
35      )
```

Slika 3.11 Podjela na skup za validaciju, testiranje i treniranje

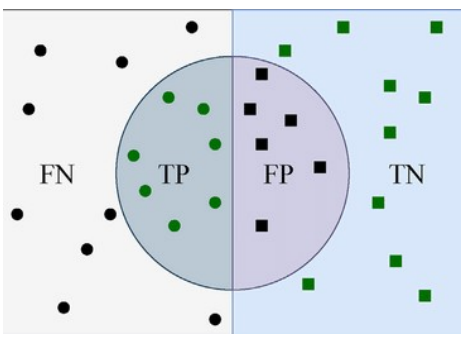
Nakon pripreme skupova podataka, koristimo metodu fit za treniranje modela, pri čemu model uči na trening skupu i prilagođava svoje parametre. Definiramo broj epoha koji određuje koliko puta će model proći kroz cijeli trening skup. Tijekom treninga, koristimo validacijski skup za praćenje performansi modela na podacima koje nije vidio tijekom treniranja. Ovo nam pomaže u detekciji prenaučenosti i optimizaciji modela. Također, koristimo callback funkciju prijevremenog zaustavljanja

koja zaustavlja treniranje ako se gubitak na validacijskom skupu ne poboljša tijekom određenog broja epoha. Nakon svake iteracije cross-validacije, prikupljamo povijest treniranog modela koja će biti korištena za daljnju analizu performansi. Nakon treninga modela, vrednujemo njegove performanse na test skupu. Koristimo model za predviđanje rezultata na test skupu pomoću metode predict. Dobivena predviđanja uspoređujemo sa stvarnim oznakama kako bismo izračunali metrike vrednovanja. U kodu se koriste različite metrike vrednovanja, kao što su površina ispod ROC krivulje, točnost, osjetljivost, specifičnost i Matthewsov korelacijski koeficijent (MCC). Ove metrike pružaju detaljniji uvid u performanse modela u različitim aspektima.

3.4 Metrike za vrednovanje modela

U ovom poglavlju opisat će se metrike modela koje se koriste za vrednovanje performansi modela. Metrike pružaju objektivnu procjenu uspješnosti modela u klasifikaciji aktivnosti peptida te će iste metrike biti upotrijebljene za optimizaciju performansi modela tijekom procesa treninga.

Stopa lažno pozitivnih (eng. False Positive Rate, FPR) je metrika koja se koristi za mjerenje omjera lažno pozitivnih rezultata u odnosu na ukupan broj stvarno negativnih rezultata te se izračunava kao omjer broja lažno pozitivnih i zbroja lažno pozitivnih i stvarno negativnih [20]. FPR je važan u vrednovanju performansi klasifikacijskih modela, gdje može pomoći u određivanju koliko često model pogrešno klasificira negativne primjere kao pozitivne.



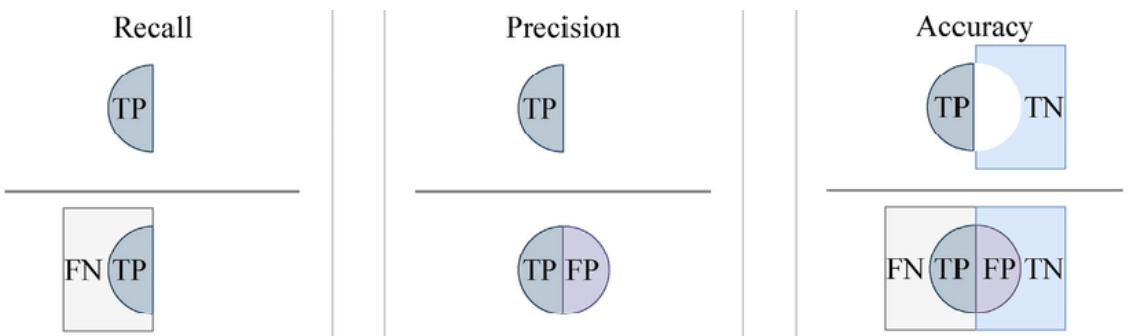
Slika 3.12 Odnos skupova TP, FP, FN, TN
Preuzeto iz [20]

Stopa stvarno pozitivnih (eng. True Positive Rate, TPR) je metrika koja mjeri omjer ispravno predviđenih pozitivnih primjera u odnosu na ukupan broj stvarno pozitivnih primjera [21]. Visoka vrijednost TPR-a ukazuje na to da model dobro pronalazi pozitivne primjere u skupu podataka.

Stopa stvarno negativnih (eng. True Negative Rate, TNR) je metrika koja mjeri omjer ispravno predviđenih negativnih primjera u odnosu na ukupan broj stvarno negativnih primjera [22]. TNR je također poznat kao specifičnost. Visoka vrijednost TNR-a sugerira da model dobro identificira negativne primjere. Na slici 3.12 prikazani su skupovi koji se koristiti za izračunavanje TNR, FPR i TPR metrika modela.

Geometrijska sredina (eng. Geometric Mean, GM) je metrika koja se koristi za kombiniranje TPR i TNR metrika u jedan broj. GM se izračunava kao korijen iz umnoška TPR i TNR. GM je posebno koristan kada su TPR i TNR neujednačeni te omogućuje dobivanje holističke ocjene modela koji odražava oba aspekta performansi.

Preciznost je metrika koja se koristi za mjerenje omjera točno pozitivnih rezultata u odnosu na ukupan broj pozitivnih predviđanja modela [23]. Vizualizacija izračuna prikazana je na slici 3.13 te se računa kao omjer broja točno pozitivnih i zbroja točno pozitivnih i lažno pozitivnih rezultata. Visoka preciznost ukazuje na to da model ima malo lažno pozitivnih predviđanja.



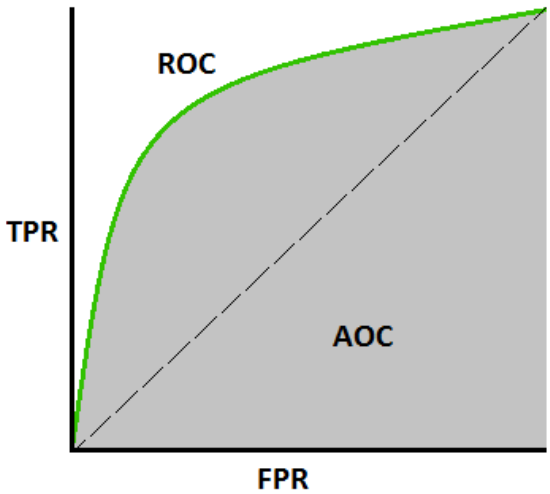
Slika 3.13 Izračun osjetljivosti, preciznosti i točnosti
Preuzeto iz [23]

Osjetljivost je metrika koja mjeri omjer točno pozitivnih rezultata u odnosu na ukupan broj stvarno pozitivnih primjera [24]. Vizualizacija izračuna prikazana je na slici 3.13 te se računa kao omjer broja točno pozitivnih i zbroja točno pozitivnih i lažno negativnih rezultata. Visoka osjetljivost ukazuje na sposobnost modela da identificira što više pozitivnih primjera.

F1 mjera je harmonička sredina preciznosti i osjetljivosti [25]. Izračunava se kao omjer dvostruke vrijednosti umnoška preciznosti i osjetljivosti i njihove zbroja. F1 mjera je korisna kada su preciznost i osjetljivost jednako važni, jer se uzima u obzir oba aspekta performansi modela.

Točnost je jednostavna i često korištena metrika za vrednovanje performansi klasifikacijskih modela [26]. Točnost mjeri omjer točno predviđenih primjera u odnosu na ukupan broj primjera. Vizualizacija izračuna prikazana je na slici 3.13 te se računa kao omjer zbroja točno pozitivnih i točno negativnih rezultata te ukupnog broja primjera. Točnost može biti korisna kada su klase uravnotežene, ali može dati neprecizne rezultate ako postoje velike razlike u broju primjera između klasa.

Na slici 3.14 prikazana je metrika pod nazivom površina ispod ROC krivulje koja se koristi za vrednovanje performansi binarnih klasifikatora [27]. ROC krivulja je graf koji prikazuje odnos između TPR i FPR na različitim pragovima klasifikacije. Vrijednosti površine ispod krivulje (eng. Area Under the Curve, AUC) kreću se od 0 do 1, gdje veći broj ukazuje na bolje performanse klasifikatora. Ako je AUC-ROC jednak 0.5, to ukazuje na slučajno klasificiranje, dok je 1 savršeno klasificiranje.



Slika 3.14 ROC krivulja i AOC površina
Preuzeto iz [27]

Matthewsov korelacijski koeficijent je metrika koja mjeri korelaciju između stvarnih i predviđenih klasifikacija [28]. Izračunava se kao razlika između proizvoda točno pozitivnih i točno negativnih primjera, te proizvoda lažno pozitivnih i lažno negativnih primjera, podijeljena s korijenom umnoška svih vrijednosti. MCC vrijednosti variraju od -1 do +1, gdje -1 predstavlja savršeno suprotnu klasifikaciju, +1 predstavlja savršeno usklađenu klasifikaciju, a 0 je indikacija slučajnog klasificiranja.

Poglavlje 4

Rezultati

Hiperparametri, poput veličine slojeva, broja filtera i broja redaka izlaznog tenzora, imaju ključan utjecaj na rezultate treniranja modela strojnog učenja. Pravilan odabir ovih hiperparametara može dovesti do dobre generalizacije modela i visokih performansi na novim podacima, dok nepravilan odabir može dovesti do prenaučenosti, podnaučenosti ili loših performansi. Prilikom treniranja modela korištene su mnogobrojne kombinacije hiperparametara kako bi se prikupilo što više rezultat iz kojih bi se moglo odrediti kako hiperparametri utječu na model i s kojom kombinacijom parametara dobivamo optimalne rezultate.

U tablici 4.1 prikazane su kombinacije za treniranje modela strojnog učenja, koje su počinjale od niskih vrijednosti parametara k kojim su označeni brojevi redaka izlaznog tenzora te su one počinjale od 15 redova, a najveća vrijednost parametra k je bila postavljena na 35 redova. Jednako tako najmanje veličine slojeva, odnosno najmanji broj neurona u sloju bio je 10, a najviše neurona što je moglo biti u jednom sloju je 32. Unutar svakog sloja nalaze se filtri, čiji se broj mogao kretati od 8 filtra pa sve do 128 filtra u jednom sloju.

Ime modela	**Broj redaka izlaznog tenzora (k)**	**Broj neurona po sloju**	**Broj filtera u prva tri sloja**
Model 1	35	[32, 32, 32, 1]	32 - 64 - 128
Model 2	35	[32, 32, 32, 1]	16 - 32 - 128
Model 3	30	[30, 30, 30, 1]	16 - 32 - 128
Model 4	25	[25, 25, 25, 1]	16 - 32 - 128
Model 5	25	[25, 25, 25, 1]	8 - 16 - 64
Model 6	20	[15, 15, 15, 1]	8 - 16 - 64
Model 7	15	[10, 10, 10, 1]	8 - 16 - 64

Tablica 4.1 Specifikacije modela na kojima su vršena istraživanja

Važno je razlikovati dva pristupa treniranju modela, jedan je koristio samo oznake aktivnosti peptida odnosno one-hot, a drugi je koristio kombinaciju one-hot kodiranja s dodatnim značajkama poput broja radikalnih elektrona, formalnog naboja atoma, hibridizacije i aromatičnosti atoma.

U tablici 4.2 prikazani su rasponi vrijednosti za različite metrike koje se koriste za vrednovanje performansi. Ovi rasponi pomažu u interpretaciji rezultata metrika te svaka metrika ima tri kategorije vrijednosti: loše, osrednje i odlične. Na primjer, za koeficijent Matthewsove korelacije, vrijednosti manje od 0.3 se smatraju lošim, vrijednosti između 0.3 i 0.55 se smatraju osrednjim, dok su vrijednosti veće od 0.55 označene kao odlične. Slično, definirani su rasponi za ostale metrike poput geometrijske sredine, preciznosti, osjetljivosti, F1 mjere i površine ispod ROC krivulje (ROC AUC). Ovi rasponi pomažu u kvantificiranju performansi modela i donošenju informiranih odluka o njegovoj uspješnosti te će se koristiti u nastavku za uspoređivanje rezultata pojedinih modela.

Metrika	Loše vrijednosti	Osrednje vrijednosti	Odlične vrijednosti
MCC	<0.3	0.3 - 0.55	>0.55
GM	<0.65	0.65 - 0.8	>0.8
Precision	<0.6	0.6 - 0.8	>0.8
Recall	<0.6	0.6 - 0.8	>0.8
F1	<0.6	0.6 - 0.8	>0.8
ROC AUC	<0.6	0.6 - 0.8	>0.8

Tablica 4.2 Rasponi vrijednosti za korištene metrike.

4.1 Model 1

Uspoređujući rezultate one-hot modela iz tablice 4.3 s tablicom 4.2 možemo zaključiti da metrike MCC, GM, preciznost i AUC imaju osrednje vrijednosti te metrike F1 i osjetljivost imaju odlične vrijednosti što znači da trenutne vrijednosti hiperparametara daju solidne rezultate, ali još uvijek ima prostora za poboljšanje.

Metric	Average	Maximum	Minimum
MCC	0.523	0.708	0.348
GM	0.746	0.857	0.653
Precision	0.775	0.895	0.696
Recall	0.846	0.932	0.783
F1	0.808	0.872	0.744
ROC AUC	0.755	0.857	0.667
FPR	0.337	0.467	0.136
TPR	0.846	0.932	0.783

Tablica 4.3 Vrijednosti metrika modela 1 treniranog s one-hot pristupom.

Uspoređujući rezultate kombiniranog modela iz tablice 4.4 s tablicom 4.2 možemo zaključiti da metrike MCC, GM, preciznost, AUC, F1 i osjetljivost imaju osrednje vrijednosti što znači da trenutne vrijednosti hiperparametara daju osrednje rezultate, te je potrebno još dodatno optimizirati hiperparametre za postizanje optimalnih rezultata.

Metric	Average	Maximum	Minimum
MCC	0.419	0.585	0.229
GM	0.704	0.791	0.599
Precision	0.747	0.820	0.657
Recall	0.769	0.833	0.717
F1	0.758	0.826	0.693
ROC AUC	0.708	0.792	0.611
FPR	0.353	0.511	0.250
TPR	0.769	0.833	0.717

Tablica 4.4 Vrijednosti metrika modela 1 treniranog s kombiniranim pristupom.

4.2 Model 2

Uspoređujući rezultate one-hot modela iz tablice 4.5 s tablicom 4.2 možemo zaključiti da metrike GM i AUC imaju osrednje vrijednosti te metrike MCC, preciznost, F1 i osjetljivost imaju odlične vrijednosti što znači da trenutne vrijednosti hiperparametara daju zadovoljavajuće rezultate.

Metric	Average	Maximum	Minimum
MCC	0.583	0.701	0.375
GM	0.787	0.852	0.683
Precision	0.820	0.922	0.726
Recall	0.831	0.883	0.750
F1	0.824	0.869	0.738
ROC AUC	0.790	0.854	0.686
FPR	0.252	0.378	0.089
TPR	0.831	0.883	0.750

Tablica 4.5 Vrijednosti metrika modela 2 treniranog s one-hot pristupom.

Uspoređujući rezultate kombiniranog modela iz tablice 4.6 s tablicom 4.2 možemo zaključiti da metrike MCC, GM, preciznost i AUC imaju osrednje vrijednosti te F1 i osjetljivost imaju odlične vrijednosti što znači da trenutne vrijednosti hiperparametara daju solidne rezultate, ali još uvijek ima prostora za poboljšanje.

Metric	Average	Maximum	Minimum
MCC	0.532	0.626	0.429
GM	0.750	0.798	0.687
Precision	0.774	0.823	0.718
Recall	0.856	0.917	0.800
F1	0.812	0.853	0.779
ROC AUC	0.758	0.800	0.703
FPR	0.340	0.444	0.250
TPR	0.856	0.917	0.800

Tablica 4.6 Vrijednosti metrika modela 2 treniranog s kombiniranim pristupom.

4.3 Model 3

Uspoređujući rezultate one-hot modela iz tablice 4.7 s tablicom 4.2 možemo zaključiti da metrike preciznost, GM i AUC imaju osrednje vrijednosti te metrike MCC, F1 i osjetljivost imaju odlične vrijednosti što znači da trenutne vrijednosti hiperparametara daju zadovoljavajuće rezultate.

Metric	Average	Maximum	Minimum
MCC	0.584	0.651	0.411
GM	0.777	0.805	0.698
Precision	0.798	0.882	0.734
Recall	0.866	0.933	0.750
F1	0.829	0.859	0.758
ROC AUC	0.783	0.811	0.703
FPR	0.299	0.409	0.136
TPR	0.866	0.933	0.750

Tablica 4.7 Vrijednosti metrika modela 3 treniranog s one-hot pristupom.

Uspoređujući rezultate kombiniranog modela iz tablice 4.8 s tablicom 4.2 možemo zaključiti da metrike MCC, F1, osjetljivost, preciznost, GM i AUC imaju osrednje vrijednosti što znači da trenutne vrijednosti hiperparametara daju osrednje rezultate te je potrebno još dodatno optimizirati hiperparametre za postizanje optimalnih rezultata.

Metric	Average	Maximum	Minimum
MCC	0.392	0.496	0.264
GM	0.687	0.748	0.605
Precision	0.738	0.837	0.662
Recall	0.759	0.817	0.683
F1	0.746	0.772	0.718
ROC AUC	0.694	0.751	0.625
FPR	0.371	0.533	0.182
TPR	0.759	0.817	0.683

Tablica 4.8 Vrijednosti metrika modela 3 treniranog s kombiniranim pristupom.

4.4 Model 4

Uspoređujući rezultate one-hot modela iz tablice 4.9 s tablicom 4.2 možemo zaključiti da metrike MCC, F1, osjetljivost, preciznost, GM i AUC imaju osrednje vrijednosti što znači da trenutne vrijednosti hiperparametara daju osrednje rezultate te postoji potreba za dodatnom izmjenom hiperparametara kako bi se postigli bolji rezultati.

Metric	Average	Maximum	Minimum
MCC	0.487	0.625	0.335
GM	0.738	0.810	0.663
Precision	0.774	0.852	0.710
Recall	0.799	0.864	0.733
F1	0.786	0.843	0.721
ROC AUC	0.741	0.811	0.667
FPR	0.317	0.409	0.182
TPR	0.799	0.864	0.733

Tablica 4.9 Vrijednosti metrika modela 4 treniranog s one-hot pristupom.

Uspoređujući rezultate kombiniranog modela iz tablice 4.10 s tablicom 4.2 možemo zaključiti da metrike MCC, F1, osjetljivost, preciznost, GM i AUC imaju odlične vrijednosti što znači da trenutne vrijednosti hiperparametara daju odlične vrijednosti metrika. Jednako tako, ovim se ukazuje na važnost dodatnih značajki u poboljšanju rezultata u odnosu na model s istim parametrima koji koristi samo značajku aktivnosti peptida.

Metric	Average	Maximum	Minimum
MCC	0.630	0.783	0.488
GM	0.807	0.890	0.738
Precision	0.833	0.943	0.766
Recall	0.860	0.966	0.783
F1	0.844	0.909	0.783
ROC AUC	0.811	0.890	0.742
FPR	0.238	0.333	0.068
TPR	0.860	0.966	0.783

Tablica 4.10 Vrijednosti metrika modela 4 treniranog s kombiniranim pristupom.

4.5 Model 5

Uspoređujući rezultate one-hot modela iz tablice 4.11 s tablicom 4.2 možemo zaključiti da metrike MCC, F1, osjetljivost, preciznost, GM i AUC imaju osrednje vrijednosti što znači da trenutne vrijednosti hiperparametara daju osrednje rezultate te postoji potreba za dodatnom izmjenom hiperparametara kako bi se postigli bolji rezultati. Još jedno od opažanja je i postepeno opadanje vrijednosti metrika prema donjim granicama osrednjih vrijednosti koje je povezano sa smanjenjem broja redova u izlaznom tenzoru, smanjenjem broja neurona u slojevima i smanjenjem broja filtera u slojevima.

Metric	Average	Maximum	Minimum
MCC	0.452	0.563	0.167
GM	0.717	0.774	0.564
Precision	0.755	0.807	0.632
Recall	0.796	0.881	0.717
F1	0.774	0.823	0.672
ROC AUC	0.723	0.777	0.581
FPR	0.351	0.556	0.250
TPR	0.796	0.881	0.717

Tablica 4.11 Vrijednosti metrika modela 5 treniranog s one-hot pristupom

Uspoređujući rezultate kombiniranog modela iz tablice 4.12 s tablicom 4.2 možemo zaključiti da metrike MCC, F1, osjetljivost, preciznost, GM i AUC imaju osred-

nje vrijednosti što znači da trenutne vrijednosti hiperparametara daju osrednje vrijednosti te postoji potreba za daljnjim izmjenama s ciljem postizanja optimalnih vrijednosti metrika. Jednako tako kao i u modelu koji koristi one-hot, dolazi do postepenog opadanja vrijednosti metrika prema donjim granicama osrednjih vrijednosti koje je povezano sa smanjenjem broja redova u izlaznom tenzoru, smanjenjem broja neurona u slojevima i smanjenjem broja filtera u slojevima.

Metric	**Average**	**Maximum**	**Minimum**
MCC	0.310	0.466	0.150
GM	0.629	0.713	0.534
Precision	0.683	0.735	0.627
Recall	0.793	0.881	0.717
F1	0.733	0.787	0.677
ROC AUC	0.647	0.724	0.571
FPR	0.498	0.636	0.386
TPR	0.793	0.881	0.717

Tablica 4.12 Vrijednosti metrika modela 5 treniranog s kombiniranim pristupom.

4.6 Model 6

Uspoređujući rezultate one-hot modela iz tablice 4.13 s tablicom 4.2 možemo zaključiti da metrike MCC, F1, osjetljivost, preciznost, GM i AUC imaju osrednje vrijednosti što znači da trenutne vrijednosti hiperparametara daju osrednje rezultate te je potrebno još dodatno optimizirati hiperparametre za postizanje optimalnih rezultata. Trenutne vrijednosti metrika su na samoj granici s lošim vrijednostima i daljnjim snižavanjem vrijednosti parametara možemo samo očekivati pogoršanje vrijednosti metrika.

Metric	Average	Maximum	Minimum
MCC	0.390	0.543	0.074
GM	0.680	0.768	0.447
Precision	0.732	0.807	0.590
Recall	0.778	0.883	0.700
F1	0.752	0.822	0.685
ROC AUC	0.692	0.768	0.531
FPR	0.393	0.756	0.244
TPR	0.778	0.883	0.700

Tablica 4.13 Vrijednosti metrika modela 6 treniranog s one-hot pristupom

Uspoređujući rezultate kombiniranog modela iz tablice 4.14 s tablicom 4.2 možemo zaključiti da metrike MCC, F1, preciznost, GM i AUC imaju osrednje vrijednosti te metrika osjetljivost ima odlične vrijednosti. Osim vrijednosti osjetljivosti ostale metrike polako teže prema lošim vrijednostima te se daljnjim snižavanjem vrijednosti hiperparametra mogu očekivati dodatna opadanja.

Metric	Average	Maximum	Minimum
MCC	0.386	0.514	0.201
GM	0.660	0.743	0.554
Precision	0.707	0.783	0.651
Recall	0.828	0.933	0.667
F1	0.761	0.818	0.661
ROC AUC	0.681	0.746	0.600
FPR	0.466	0.659	0.295
TPR	0.828	0.933	0.667

Tablica 4.14 Vrijednosti metrika modela 6 treniranog s kombiniranim pristupom

4.7 Model 7

Uspoređujući rezultate one-hot modela iz tablice 4.15 s tablicom 4.2 možemo zaključiti da metrike F1, preciznosti i AUC imaju osrednje vrijednosti te metrika osjetljivost ima odlične vrijednosti. Jednako tako, metrike MCC i GM imaju loše vrijednosti te

se stoga ova kombinacija hiperparametara uzela kao donja granica do koje se treniрao model jer daljnjim treniranjem s manjim vrijednostima parametara još bi se više odstupalo od optimalnih vrijednosti metrika.

Metric	**Average**	**Maximum**	**Minimum**
MCC	0.240	0.489	0.000
GM	0.566	0.692	0.442
Precision	0.646	0.724	0.571
Recall	0.819	0.933	0.733
F1	0.722	0.809	0.642
ROC AUC	0.607	0.720	0.500
FPR	0.606	0.733	0.477
TPR	0.819	0.933	0.733

Tablica 4.15 Vrijednosti metrika modela 7 treniranog s one-hot pristupom

Uspoređujući rezultate kombiniranog modela iz tablice 4.16 s tablicom 4.2 možemo zaključiti da metrike F1, preciznosti, MCC, GM i AUC imaju osrednje vrijednosti te metrika osjetljivost ima odlične vrijednosti, ali zbog razlog navedenih u one-hot modelu s ovom kombinacijom postavila se granica do koje se model trenirao

Metric	**Average**	**Maximum**	**Minimum**
MCC	0.406	0.531	0.253
GM	0.681	0.741	0.597
Precision	0.722	0.789	0.662
Recall	0.820	0.883	0.750
F1	0.766	0.815	0.718
ROC AUC	0.695	0.753	0.619
FPR	0.430	0.545	0.273
TPR	0.820	0.883	0.750

Tablica 4.16 Vrijednosti metrika modela 7 treniranog s kombiniranim pristupom

4.8 Usporedba vrijednosti metrika modela strojnog učenja s referentnim vrijednostima

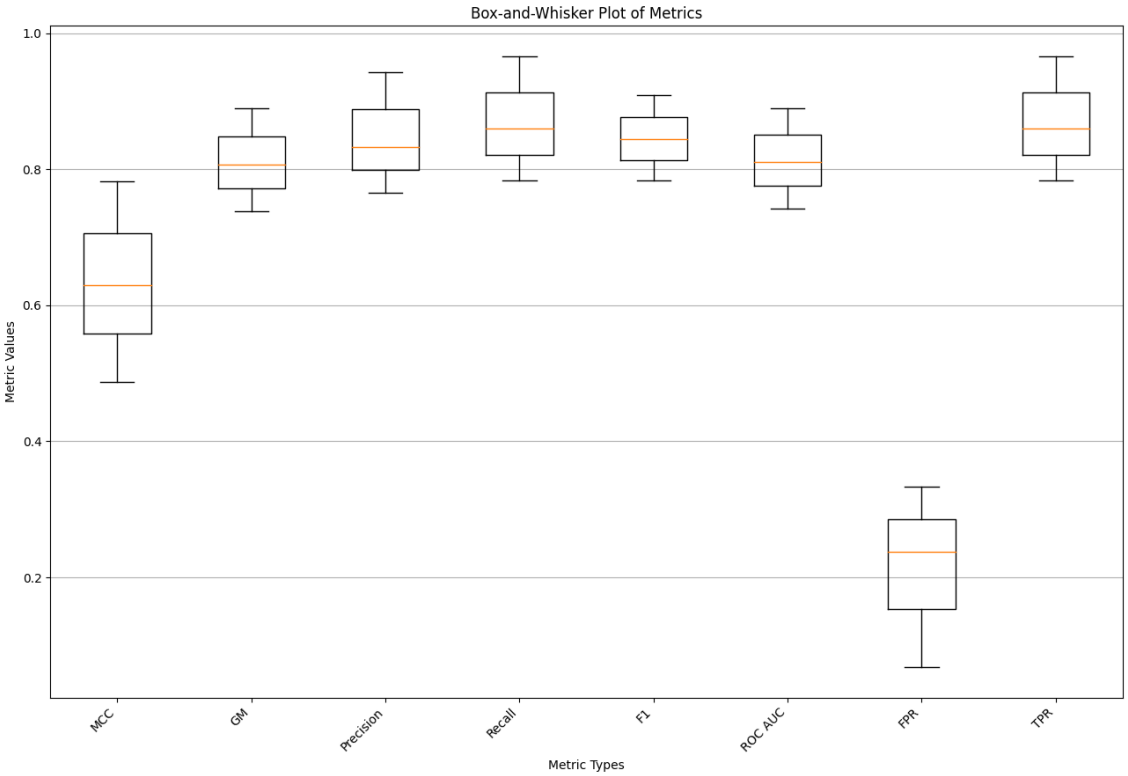
Usporedba vrijednosti metrika modela strojnog učenja s referentnim vrijednostima igra ključnu ulogu u procjeni uspješnosti i performansi modela. Kada razvijamo modele za rješavanje određenog zadatka, često imamo referentne vrijednosti koje predstavljaju točne ili očekivane rezultate za taj zadatak. Te referentne vrijednosti obično proizlaze iz stvarnih mjerenja, eksperimenata ili su predstavljene od strane stručnjaka na temelju njihovog domenskog znanja. Za potrebe ovog rada korištene su referentne vrijednosti sa slike 4.1 koje dolaze iz znanstvenog rada koji se bavi otkrivanjem terapeutskih peptida pomoću tehnike virtualnog prebira podržanog prediktivnim modelima temeljenim na strojnom učenju [29]. Ovaj rad je važan jer predlaže inovativnu hibridnu shemu za virtualni pregled terapeutskih peptida, kombinirajući prednosti različitih pristupa te postiže nadmoćne rezultate u predviđanju antimikrobnih i antivirusnih aktivnosti peptida u usporedbi s postojećim metodama. Autori koriste neuronske mreže za modeliranje sekvenci peptida, pružajući uvid u sinergijski učinak predviđanja na temelju svojstava i poretku aminokiselina. U izradi rada korištena je isti skup podataka AVPdb.

	F1 score	MCC	GM	Recall	Precision	AUC	Dataset
Peptide properties	0.833	0.588	0.782	**0.870***	0.800	0.867*	**AVPdb**
One-hot	0.829	0.610	0.805	0.814	0.845*	0.869*	
Embedding	0.806	0.547	0.772	0.804	0.808	0.837	
Sequential properties	**0.858***	**0.671***	**0.834***	0.855*	**0.863***	**0.882***	

Slika 4.1 Referentne vrijednosti
Preuzeto iz [29]

Nakon treniranja modela i dobivanja njegovih predviđanja, uspoređujemo ta predviđanja s referentnim vrijednostima kako bismo procijenili koliko dobro model funkcionira. To činimo putem različitih metrika vrednovanja, kao što su GM, osjetljivost, preciznost, F1 mjera, ROC AUC i MCC. Svaka metrika pruža drugačiji uvid u performanse modela, ovisno o prirodi zadatka. Na slici 4.2 su prikazane metrike kombiniranog modela s izlaznim tenzorom s 25 redova, brojem neurona u slojevima jednakim [25,25,25,1] i brojem filtera u prva tri sloja: 16 - 32 - 128. Prikazani model je odabran

za usporedbu jer je on imao najbolje vrijednosti od svih korištenih modela.



Slika 4.2 Grafički prikaz rezultata modela s najboljim performansama

Na slici 4.3 prikazana je usporedba vrijednosti referentnih metrika s vrijednostima metrika najboljeg modela. Prilikom izrade usporedbe, za svaku metriku je korištena minimalna i maksimalna vrijednost koju je ta metrika postigla te su one na slici 4.3 prikazane crnom točkom, dok žuti kvadrat predstavlja srednju vrijednost metrike koja je dobivena kao omjer zbroja svih vrijednosti koja je ta metrika postigla i broj preklopa k.

Vrijednost F1 mjere ovog modela je nešto veća od referentne vrijednosti. F1 mjera kombinira preciznost i osjetljivost, te je koristi kao mjera harmonijske sredine između njih. Vrijednosti F1 metrike ukazuju na dobru ravnotežu između točnosti i potpunosti modela, što ga čini izuzetno pouzdanim u različitim situacijama.

GM ovog modela također pokazuje nešto veće vrijednosti od referentnih vrijednosti. GM kombinira osjetljivost i specifičnost te pruža cjelovitu sliku o perfor-

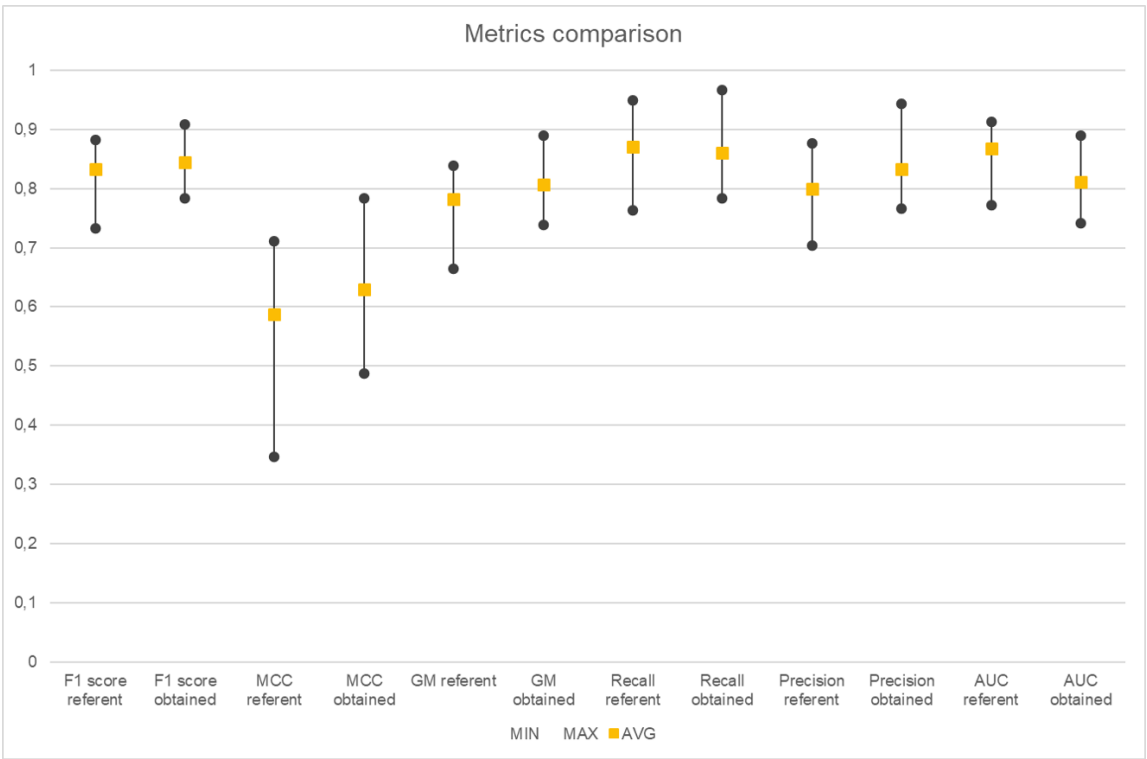
mansama modela. Vrijednosti GM metrike ukazuju na uravnoteženost modela u identificiranju i pozitivnih i negativnih primjera.

Vrijednost Matthewsovog korelacijskog koeficijenta ovog modela također imaju nešto veće vrijednosti s obzirom na referentne vrijednosti. MCC mjeri korelaciju između stvarnih i predviđenih oznaka i često se koristi za procjenu performansi modela u nebalansiranim skupovima podataka. Vrijednosti MCC metrike ukazuju na visoku kvalitetu klasifikacije i točno odražava sposobnost modela da se nosi s raznolikim slučajevima.

Vrijednost preciznosti ovog modela je nešto veća od referentne vrijednosti. Ovakav rezultat ukazuje na točnost modela u identificiranju pozitivnih primjera između svih pozitivno klasificiranih instanci. Vrijednost metrike preciznosti ukazuje da model izuzetno dobro razlikuje relevantne rezultate od lažno pozitivnih.

Model je ostvario impresivno bliske vrijednosti osjetljivosti u usporedbi s referentnom vrijednosti. To znači da model uspijeva prepoznati veći broj stvarno pozitivnih primjera među svim pozitivnim instancama u podacima. Ovakav rezultat omogućuje modelu da maksimalno iskoristi sve relevantne informacije u podacima, čineći ga vrlo pouzdanim u otkrivanju pravih pozitivnih slučajeva.

Vrijednosti AUC metrike za ovaj model je veoma bliska referentnim vrijednostima. Ovo sugerira da model ima izuzetnu sposobnost razlikovanja između pozitivnih i negativnih primjera, što ga čini izvrsnim alatom za klasifikaciju u različitim scenarijima.



Slika 4.3 Grafički prikaz usporedbe vrijednosti

Poglavlje 5

Zaključak

U ovom istraživanju, uspješno je razvijen i vrednovan model strojnog učenja za predviđanje antiviralne aktivnosti peptida zasnivan na GNN, što je rijetko korišteni model u domeni kemije peptida za terapeutske namjene. Konačni model pokazuje iznimno dobre performanse u razlikovanju viralno aktivnih od neaktivnih peptida, što ukazuje na njegovu veliku vrijednost u biološkim istraživanjima.

Osmišljavanje konačnog modela uključivalo je ugađanje hiperparametara kako bi se postigli optimalni rezultati. Kroz iterativan pristup i temeljitu analizu performansi, uspješno su konfigurirani arhitektura modela, kao i parametri učenja, što je dovelo do znatnog poboljšanja njegove točnosti. Istovremeno, u obzir su uzete relevantne biološke informacije, uključujući broj radikalnih elektrona, formalni naboj atoma, hibridizaciju i aromatičnost atoma, čime je obogaćen skup ulaznih podataka i poboljšane su predviđajuće performanse modela.

Rezultati validacije modela ukazuju na njegovu sposobnost da učinkovito razlikuje peptidne sekvence s biološkom aktivnosti od onih bez aktivnosti, što je ključno za identifikaciju potencijalnih kandidata za daljnja eksperimentalna ispitivanja. Korištenjem različitih metrika za procjenu performansi, dublje smo analizirali kako naš model uspješno obavlja tu razliku. Matthewsov koeficijent korelacije, koji iznosi 0.630, pruža nam mjeru kvalitete našeg modela, uzimajući u obzir ravnotežu između točnosti i osjetljivosti. Osim toga, GM iznosi 0.807, što potvrđuje stabilnost i kvalitetu našeg modela. Preciznost modela iznosi 0.833, što znači da je model vrlo dobar

u sprječavanju lažno pozitivnih rezultata. Osjetljivost iznosi 0.860, što ukazuje na visoku sposobnost modela da pravilno identificira stvarno pozitivne primjere. F1 mjera, koja iznosi 0.844, također sugerira ravnotežu između preciznosti i osjetljivosti. Površina ispod ROC krivulje iznosi 0.811, što ukazuje na dobru sposobnost razlikovanja između klasa.

Bibliografija

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," in *IEEE Transactions on Neural Networks*, vol. 20, no. 1, 2009, pp. 61–80.

[4] J. Smith and M. Johnson, "Peptide structure and function," *Journal of Peptide Science*, vol. 30, no. 5, p. e12345, 2020.

[5] S. Jones and M. Smith, "Therapeutic applications of peptides in medicine," *Journal of Medicinal Chemistry*, vol. 40, no. 8, pp. 1234–1256, 2018.

[6] Python Software Foundation, "Python programming language," https://www.python.org/, 2001.

[7] J. Ott, M. Pritchard, N. Best, E. Linstead, M. Curcic, and P. Baldi, "A fortran-keras deep learning bridge for scientific computing," *Scientific Programming*, vol. 2020, pp. 1–13, 08 2020.

[8] Schrödinger, LLC, "The PyMOL molecular graphics system, version 1.8," November 2015.

[9] G. Landrum, "Rdkit: Open-source cheminformatics software," 2016. , s Interneta, https://github.com/rdkit/rdkit/releases/tag/Release_2016_09_4

[10] C. Data61, "Stellargraph machine learning library," https://github.com/stellargraph/stellargraph, 2018.

[11] W. McKinney *et al.*, "Data structures for statistical computing in Python," *Proceedings of the 9th Python in Science Conference*, vol. 445, pp. 51–56, 2010.

[12] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "TensorFlow: A system for large-scale machine learning," 2016.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[14] A. Qureshi, N. Thakur, H. Tandon, and M. Kumar, "AVPdb: A database of experimentally validated antiviral peptides targeting medically important viruses," *Nucleic Acids Research*, vol. 42, no. Database issue, pp. D1147–D1153, 2014.

[15] F. A. Cotton, G. Wilkinson, C. A. Murillo, and M. Bochmann, *Advanced Inorganic Chemistry.* John Wiley Sons, 1999.

[16] P. Atkins and L. Jones, *Chemical Principles: The Quest for Insight.* W. H. Freeman, 2016.

[17] C. K. Ingold, *Structure and Mechanism in Organic Chemistry.* Cornell University Press, 1953.

[18] L. Pauling, *The Nature of the Chemical Bond.* Cornell University Press, 1960.

[19] J. Clayden, N. Greeves, S. Warren, and P. Wothers, *Organic Chemistry.* Oxford University Press, 2012.

[20] F. Provost and T. Fawcett, "Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions," *Knowledge and Information Systems*, vol. 3, no. 4, pp. 387–408, 2001.

[21] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," *Proceedings of the 23rd International Conference on Machine Learning*, pp. 233–240, 2006.

[22] ——, "The relationship between precision-recall and roc curves," *Proceedings of the 23rd International Conference on Machine Learning*, pp. 233–240, 2006.

[23] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to roc, informedness, markedness correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[24] ——, "Evaluation: From precision, recall and f-measure to roc, informedness, markedness correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[25] Y. Sasaki, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PLOS ONE*, vol. 13, no. 3, p. e0194052, 2018.

[26] F. Provost and T. Fawcett, "Robust classification for imprecise environments," *Machine Learning*, vol. 42, no. 3, pp. 203–231, 2001.

[27] T. Fawcett, "Roc graphs: Notes and practical considerations for data mining researchers," *Machine Learning*, vol. 31, no. 1, pp. 1–38, 2004.

[28] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[29] E. Otovic, M. Njirjak, D. Kalafatovic, and G. Mausa, "Sequential properties representation scheme for recurrent neural network-based prediction of therapeutic peptides," *Journal of Chemical Information and Modeling*, vol. 62, no. 12, pp. 2961–2972, 2022.

Sažetak

Ovaj završni rad istražuje predviđanje aktivnosti antiviralnih peptida putem neuronskih mreža temeljenih na grafovima. Rad opisuje sve faze procesa, počevši od konverzije SMILES zapisa peptidnih molekula u grafove pa sve do obuke modela za predviđanje njihovih aktivnosti pomoću neuronskih mreža temeljenih na grafovima. Ovaj rad pruža duboki uvid u primjenu neuronskih mreža temeljenih na grafovima u biokemijskoj analizi te naglašava važnost precizne analize performansi za postizanje pouzdanih rezultata.

Ključne riječi **— antiviralni peptidi, neuronske mreže temeljene na grafovima, SMILES zapisi peptidnih molekula, grafovi**

Abstract

This thesis investigates peptide activity antiviral prediction via graph-based neural networks. The paper describes all stages of the process, starting from the conversion of SMILES records of peptide molecules into graphs and up to the training of models to predict their activities using graph-based neural networks. This paper provides deep insight into the application of graph-based neural networks in biochemical analysis and highlights the importance of precise performance analysis to achieve reliable results.

Keywords **— antiviral peptides, graph-based neural networks, SMILES records of peptide molecules, graphs**