УНИВЕРЗИТЕТ У НИШУ
ЕЛЕКТРОНСКИ ФАКУЛТЕТ

Предикција броја мобилних корисника у азимуталној равни помоћу неуронских мрежа заснованих на вишеслојном перцептрону

Пројекат 4

Предмет: Вештачка интелигенција и машинско учење за комуникационе системе

Студент:

Настасија Станковић, бр. инд. 1622

Ментор:

Проф др. Зоран Станковић

Ниш,2024. година

Садржај

Ниш,2024. година

1. Увод

У данашњем свету мобилних комуникација, ефикасно управљање ресурсима и оптимизација мрежа су кључни за пружање висококвалитетних услуга корисницима. Како број мобилних уређаја и захтеви за преносом података константно расту, потреба за интелигентним решењима за детекцију и праћење корисника унутар одређених подручја постаје све израженија. Један од изазова са којим се суочавају мобилни оператори је детекција броја активних корисника у одређеном сектору базне станице, што је критично за правилно управљање пропусним опсегом, баланс оптерећења и алокацију мрежних ресурса.

У овом пројекту, циљ је био развој модела вештачке неуронске мреже (MLP - Multi-Layer Perceptron) који може на основу комплексних вредности прве врсте просторне корелационе матрице сигнала одредити број мобилних корисника који се налазе у одређеном сектору базне станице. Ово решење би омогућило брзу и тачну класификацију корисника, што може довести до побољшања перформанси мреже, смањења латенције и бољег корисничког искуства.

Мерења су вршена коришћењем антенског низа са шест елемената, који прикупља сигнале из азимуталне равни у сектору углова од -60° до 60°. Корисници у овом сектору емитују сигнале исте фреквенције, чиме стварају интерференцију у подацима који стижу до антенског низа. Просторна корелациона матрица добијена овим сигналима садржи комплексне вредности које су у корелацији са бројем активних корисника и њиховим просторним положајем у односу на антене. Циљ је да се ове информације искористе како би се класификовало да ли у сектору постоји један, два или три корисника.

Задатак је био да се на основу реалних и имагинарних делова елемената прве врсте корелационе матрице (без узимања у обзир аутокорелационог елемента C11) обучи модел вештачке неуронске мреже. Овај модел треба да класификује узорке у три класе, на основу броја корисника који су присутни у датом сектору. Основни изазов у овом пројекту био је прецизна обрада података и изградња ефикасног модела који може да идентификује овај број корисника на основу сложених просторних података, уз минималне грешке у класификацији.

Резултати овог рада могу послужити као основа за будућа истраживања, укључујући интеграцију са напреднијим моделима, анализу већих скупова података, као и истраживање алтернатива у обради сигнала и структурама неуронских мрежа.

2. Припрема података

У припреми података за класификацију броја корисника у сектору, коришћени су скупови података за обуку, валидацију и тестирање. Подаци су учитани из датотека у формату табулатора (.dat фајлови), при чему је свакој колони додељено име ради боље прегледности и управљања.

Величина скупа података

- **Скуп за обуку (Train Data):** Садржи 8921 узорак, са 16 карактеристика по узорку.
- **Скуп за тестирање (Test Data):** Садржи 6121 узорак, такође са 16 карактеристика по узорку.

Овај однос броја узорака осигурава довољно података за обуку модела, као и за процену његове генерализације на непознатим подацима.

Одабир карактеристика

Након учитавања података, изабране су релевантне колоне које садрже реалне и имагинарне делове елемената корелационе матрице прве врсте, изузимајући аутокорелациони елемент (C11). Ове колоне представљају улазне карактеристике које се користе за обуку модела. Циљна вредност (output) је дефинисана бројем корисника у сектору, који може бити један, два или три.

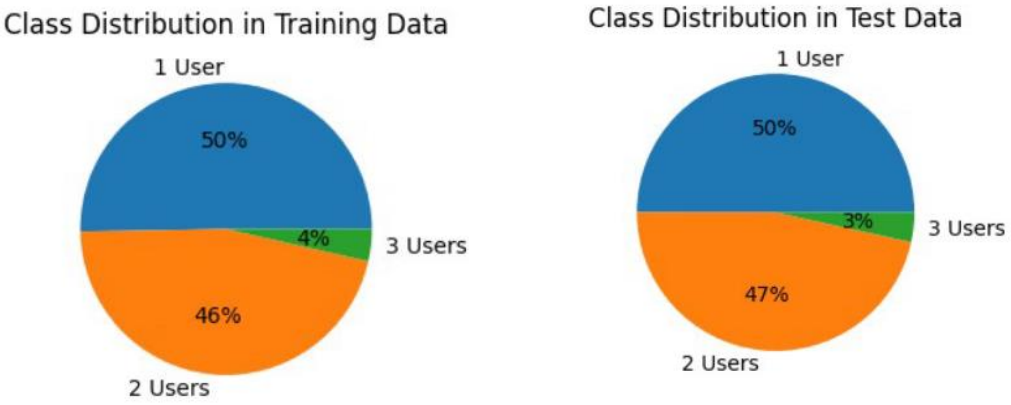
Подела података

Подаци су подељени у три скупа:

- **Скуп за обуку:** 80% података коришћено је за обуку модела.
- **Скуп за валидацију:** 20% података из скупа за обуку издвојено је за валидацију, ради процене перформанси модела током обуке и спречавања пренаглашености (overfitting).
- **Тест скуп:** Посебан скуп података искоришћен је за евалуацију модела након обуке, како би се проценили његове генерализационе способности.

Балансираност података

Један од кључних аспеката припреме података била је провера балансираности класа.



Анализа скупа показала је да је класа са три корисника значајно мање заступљена (око 3-4%) у односу на класе са једним и два корисника (које су заступљене приближно подједнако, око 50% сваке).

У случајевима неуравнотежености класа, модел може бити пристрасан према већинским класама, игноришући мањинске класе. Да би се овај проблем ублажио, примењена је техника **oversampling-a**. Ова техника подразумева вештачко повећање броја примера мањинске класе. У нашем случају:

- Oversampling је примењен само на скупу за обуку како би се повећао број примера класе са три корисника.

- Ова техника је допринела:

    1. **Смањењу пристрасности модела:** Обезбеђено је да модел не фаворизује већинске класе током обуке.

    2. **Побољшању перформанси:** Боље представљање класе са три корисника довело је до већег учинка у класификацији ове класе.

    3. **Равномернијем распоређивању губитка:** Губитак је равномерније распоређен међу класама, што је омогућило ефикаснију обуку модела.

Скалирање података

Испробане су две варијанте података:

1. **Скалирани подаци:** Карактеристике су стандарнизоване коришћењем **StandardScaler-a**. Овај метод трансформише податке тако да имају средњу вредност 0 и стандардну девијацију 1, што моделу омогућава ефикаснији рад са различитим распонима вредности.

2. **Нескалирани подаци:** Подаци су коришћени у изворном облику без додатне трансформације.

Провера квалитета података

Извршена је провера недостајућих података и утврђено је да сви скупови садрже комплетне податке, што значи да није било потребе за додатним чишћењем или попуњавањем недостајућих вредности.

Ова припрема података омогућила је стабилну и поуздану основу за даљу обуку неуронске мреже.

2.1 Припрема података за неуронске мреже

Након почетне обраде и поделе скупа података, подаци су припремљени за употребу у неуронским мрежама, коришћењем PyTorch библиотеке. У овом кораку, улазни подаци и етикете (ознаке класа) конвертовани су у PyTorch тензоре.

Конверзија података у тензоре

```
X_train = torch.tensor(X_train, dtype=torch.float32)
y_train = torch.tensor(y_train, dtype=torch.long)
X_train_os = torch.tensor(X_train_os, dtype=torch.float32)
y_train_os = torch.tensor(y_train_os, dtype=torch.long)
X_val = torch.tensor(X_val, dtype=torch.float32)
y_val = torch.tensor(y_val, dtype=torch.long)
X_test = torch.tensor(X_test, dtype=torch.float32)
y_test = torch.tensor(y_test, dtype=torch.long)
```

Шта је тензор?

Тензор је генерализација скалара, вектора и матрица, и представља основну структуру података у PyTorch-у. Он омогућава ефикасну манипулацију великим количинама података и извршавање комплексних операција на GPU уместо CPU, што значајно убрзава процес обраде.

- **Скалари:** Тензори са димензијом 0 (једна вредност, као што је број).
- **Вектори:** Тензори са једном димензијом (листа вредности, као што су карактеристике).
- **Матрице:** Тензори са две димензије (табела података).

У овом пројекту:

- Улазни подаци X_train, X_val, X_test су представљени као тензори типа torch.float32 ради прецизнијег рачунања.
- Циљне вредности y_train, y_val, y_test су тензори типа torch.long, јер представљају целобројне класе (1, 2, или 3).

Креирање TensorDataset-а

PyTorch-ов TensorDataset омогућава да се тензори улазних података и циљних вредности комбинују у објекат који се може користити за обуку и тестирање модела.

```
train_dataset_no_oversample = TensorDataset(X_train, y_train)
train_dataset_with_oversample = TensorDataset(X_train_os, y_train_os)
val_dataset = TensorDataset(X_val, y_val)
test_dataset = TensorDataset(X_test, y_test)
```

- train_dataset_no_oversample**:** Скуп за обуку без oversampling-а.
- train_dataset_with_oversample**:** Скуп за обуку са oversampling-ом, где је класа са три корисника допуњена како би се постигла боља равнотежа.
- val_dataset**:** Скуп за валидацију, који се користи за процену перформанси модела током обуке.
- test_dataset**:** Скуп за тестирање, који процењује како модел ради на непознатим подацима.

Зашто се користе тензори у PyTorch-у?

1. **Ефикасност:** Тензори су оптимизовани за извршавање операција на GPU, што знатно убрзава процес обуке и тестирања неуронских мрежа.
2. **Флексибилност:** PyTorch пружа једноставну синтаксу за операције на тензорима, као што су додавање, множење и трансформације.
3. **Компатибилност:** Модели у PyTorch-у очекују улазне податке у облику тензора, па је неопходно све податке претходно конвертовати.

Овај корак припреме података омогућава моделу да оптимално користи рачунске ресурсе и оствари најбоље резултате током обуке и тестирања.

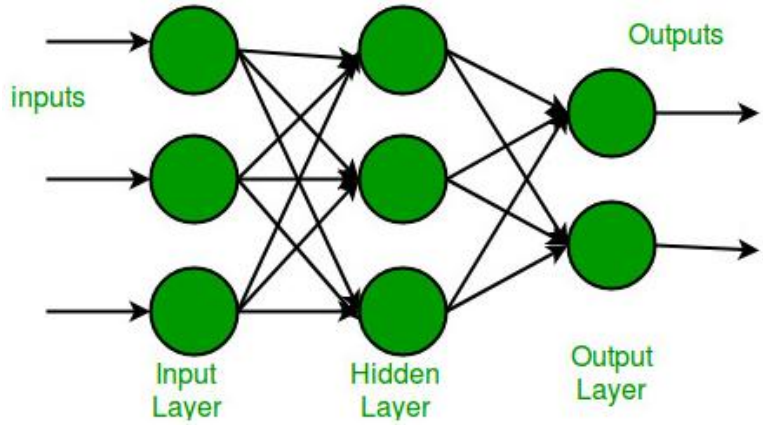
3. Архитектура модела и хиперпараметри

3.1 Увод у мулти-слојни перцептрон (MLP)

Мулти-слојни перцептрон (MLP) је један од основних типова вештачких неуронских мрежа, познат по својој способности да решава задатке који захтевају учење сложених образаца у подацима. Захваљујући својој флексибилности и универзалности, MLP је примењив у широком спектру проблема, као што су класификација, регресија и предвиђање.

Компоненте MLP модела:

MLP је мрежа неурона организованих у слојевима:



- **Улазни слој:** Први слој у мрежи, који прихвата карактеристике података и прослеђује их ка првом скривеном слоју. Број неурона у овом слоју одговара броју карактеристика у улазним подацима.

- **Скривени слојеви:** Један или више слојева који врше трансформацију података применом тежина и активационих функција. Скривени слојеви омогућавају мрежи да учи сложене, нелинеарне односе у подацима.

- **Излазни слој:** Завршни слој у мрежи, који враћа предвиђене вредности. У задацима класификације, број неурона у излазном слоју одговара броју класа које модел предвиђа.

Карактеристике MLP модела

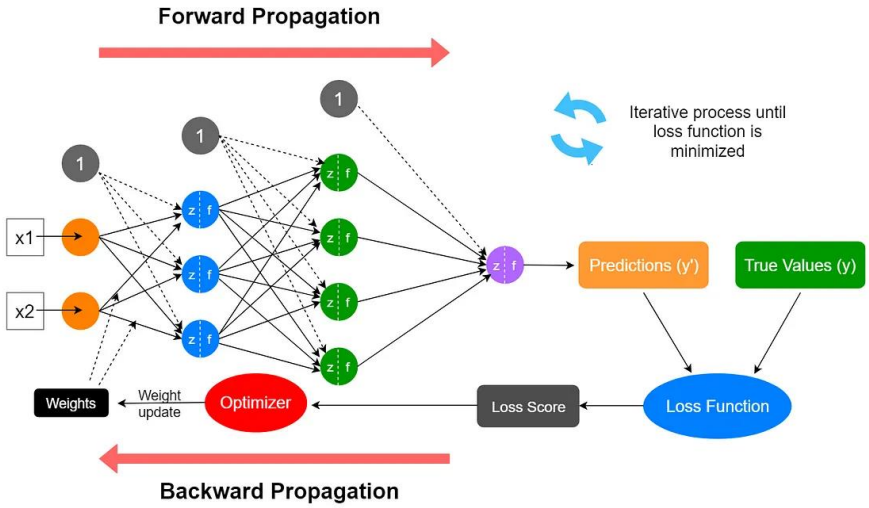
1. **Пуна повезаност (Fully Connected Layers):** Сваки неурон у једном слоју је повезан са свим неуронима у следећем слоју. Овај дизајн омогућава мрежи да интегрише информације из свих карактеристика улазног слоја.

2. **Нелинеарност:** Нелинеарност се уводи коришћењем активационих функција у скривеним слојевима. Ово омогућава моделу да решава сложене проблеме који захтевају више од једноставне линеарне регресије.

3. **Хијерархијско учење:** Скривени слојеви омогућавају моделу да учи хијерархијске представе података. Први слојеви обично уче једноставније карактеристике (на пример, корелације у подацима), док каснији слојеви уче сложеније структуре.

Пропагација у MLP моделу

Рад MLP модела одвија се у два главна корака:



1. **Пропагација унапред (Forward Propagation):**

    - У овом кораку, подаци пролазе кроз мрежу од улазног до излазног слоја.

    - На сваком слоју, производ вектора улазних података и тежина неурона се сабира са офсетом (bias) и затим пролази кроз активациону функцију: h=f(Wx+b) где је f активациона функција, W матрица тежина, x улазни вектор, а b вектор офсета.

2. **Пропагација уназад (Backward Propagation)**

- Током обуке, модел израчунава грешку између предвиђених и стварних вредности помоћу функције губитка.

- Градијент те грешке се користи за ажурирање тежина и офсета свих неурона у мрежи, чиме модел постепено учи да прави боље предвиђања.

Јачине MLP модела

- **Универзалност:** MLP је способан да апроксимира било коју континуирану функцију са довољно великим бројем неурона и слојева, што га чини погодним за различите проблеме.

- **Флексибилност:** Захваљујући различитим конфигурацијама, може се прилагодити како малим тако и великим скуповима података.

- **Могућност проширења:** MLP се лако проширује додавањем нових слојева или прилагођавањем броја неурона у слојевима.

Ограничавајући фактори MLP модела

- **Склоност пренаглашености (Overfitting):** Без примене техника као што су Dropout или Batch Normalization, модел може бити превише прилагођен подацима за обуку.

- **Нестабилност током обуке:** Велики број слојева може довести до проблема са нестајањем или експлодирањем градијента, што отежава обуку дубоких мрежа.

У овом пројекту, MLP модел је коришћен за класификацију броја корисника у сектору базне станице. Улазни слој прихвата 10 карактеристика (реалне и имагинарне вредности просторне корелационе матрице), док излазни слој предвиђа једну од три класе (један, два или три корисника). Скривени слојеви су конфигурисани са различитим бројем неурона, што је омогућило истраживање оптималне архитектуре за овај проблем.

Архитектура:

```
class MLP(nn.Module):
    def __init__(self, input_size, hidden_layers, activation_fn):
        super(MLP, self).__init__()
        layers = []
        for hidden_size in hidden_layers:
            layers.append(nn.Linear(input_size, hidden_size))
            layers.append(nn.BatchNorm1d(hidden_size))
            layers.append(activation_fn())
            layers.append(nn.Dropout(p=0.5))
            input_size = hidden_size
        layers.append(nn.Linear(input_size, 3))
        self.model = nn.Sequential(*layers)

    def forward(self, x):
        return self.model(x)
```

Овај модел је основа за експерименте у којима су испитивани различити хиперпараметри, конфигурације скривених слојева и оптимизатори. MLP је демонстрирао значајан потенцијал у класификацији броја корисника, док су додатне технике као што су Batch Normalization и Dropout побољшале његову стабилност и перформансе.

3.2 Хиперпараметри у мулти-слојном перцептрону (MLP)

Хиперпараметри представљају критичне аспекте дизајна модела јер значајно утичу на његову способност да учи из података, да се добро понаша на новим примерима и да оптимално користи ресурсе током обуке. Испод се детаљно описују изабрани хиперпараметри, њихова примена и разлози за избор у овом конкретном задатку класификације.

3.2.1 Оптимизатори (Optimizers)

Оптимизатори су алгоритми који контролишу начин ажурирања тежина током обуке. Главни циљ оптимизатора је да минимизује функцију губитка и побољша тачност модела.

У овом пројекту разматрана су два оптимизатора:

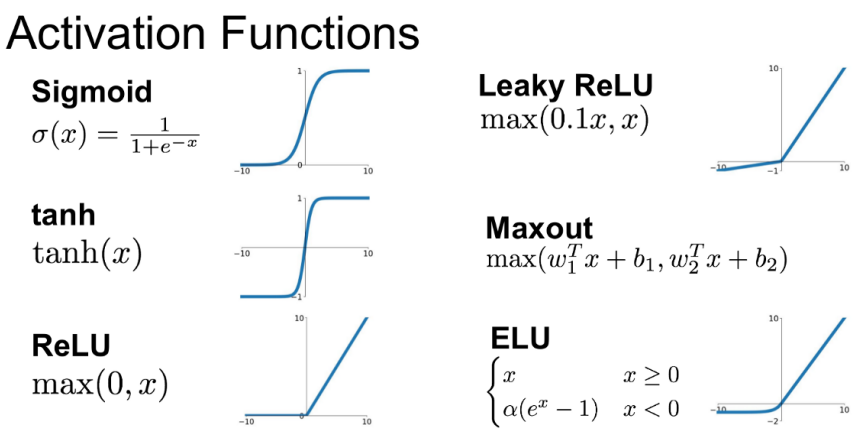
- **Adam (Adaptive Moment Estimation)**: Adam је један од најпопуларнијих оптимизатора јер користи адаптивне брзине учења за сваку тежину, што значи да брзина учења за сваку тежину зависи од њене важности. Adam комбинује предности два друга алгоритма — RMSprop и Stochastic Gradient Descent (SGD). Он прати први и други моменат градијента, што га чини ефикасним на великим скуповима података и у мрежама са сложенијим архитектурама.

- **SGD (Stochastic Gradient Descent)**: Овај оптимизатор ажурира тежине на основу мањег подскупа података (batch) уместо целокупног скупа података, што му омогућава бржу обуку. Међутим, један од недостатака је то што може бити склонији осцилацијама у губитку и перформансама током обуке.

Одабир оптимизатора зависи од самог проблема и величине скупа података. Adam је често бржи и стабилнији, али SGD може бити погодан за веома велике скупове података или када је меморија ограничена.

3.2.2 Активационе функције (Activation Functions)

Активационе функције у неуронским мрежама дефинишу како се сигнал преноси са једног слоја на други. Оне додају нелинеарност систему, што је неопходно за учење сложених образаца у подацима.



У овом пројекту Разматране су две функције:

ReLU(Rectified Linear Unit):

- ReLU је веома популарна активациона функција која трансформише све негативне вредности у нулу, а позитивне оставља непромењене. Њена главна предност је једноставност и ефикасност у обуци дубоких неуронских мрежа. Међутим, један од проблема са ReLU функцијом је "ефекат мртвих неурона," где неурон више никада не активира свој излаз ако постане негативан током обуке.

Sigmoid:

- Sigmoid функција враћа вредности у опсегу од 0 до 1. Она је често коришћена за излазне слојеве у бинарној класификацији, али је мање популарна за скривене слојеве због проблема са нестајањем градијента.

Избор активационе функције зависи од природе података и специфичности проблема. У овом раду, ReLU се показао као ефикаснија опција за скривене слојеве због своје једноставности и способности да боље обради комплексне обрасце у подацима.

3.2.3 Конфигурација скривених слојева (Hidden Layer Configurations)

Скривени слојеви представљају неуроне који обрађују карактеристике података како би се препознали обрасци. Њихова конфигурација одређује капацитет модела.

Разматране конфигурације:

- **Један слој:** [5], [10], [20]
- **Два слоја:** [5, 5], [10, 10], [20, 20]
- **Три слоја:** [5, 5, 5], [10, 10, 10], [20, 20, 20]

3.2.4 Batch Size

Batch size дефинише број примерака из скупа података који се користе за ажурирање тежина модела у једној итерацији. У нашем примеру, разматрају се две вредности:

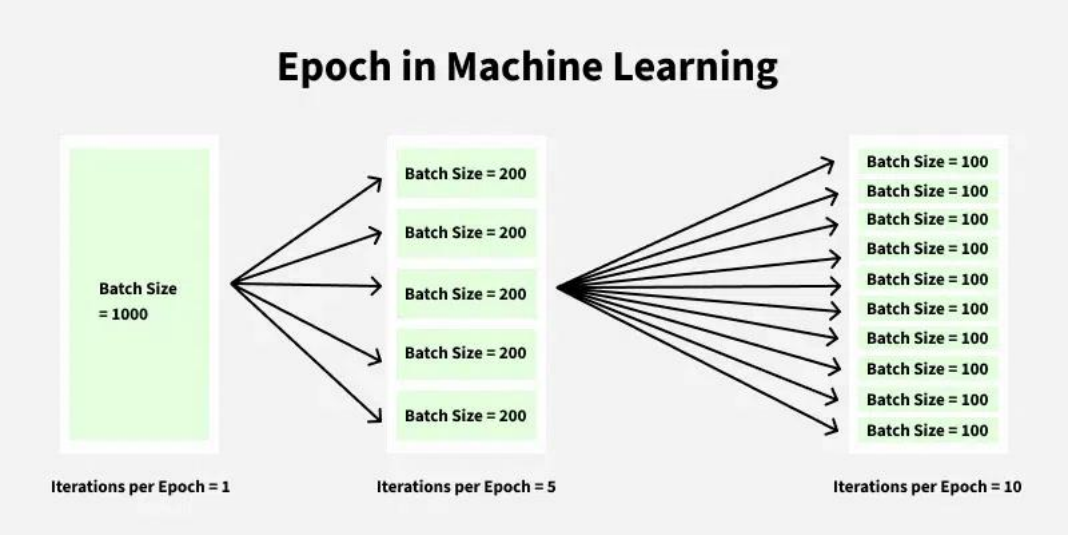
- **Batch size = 32**: Мања величина партије омогућава брже ажурирање тежина и чешће интеракције са моделом. Ово може довести до брже конвергенције, али истовремено и до веће флуктуације у губицима током обуке.
- **Batch size = 64**: Већа величина партије обезбеђује стабилније ажурирање тежина јер се губитак израчунава на већем броју примерака. Међутим, то може успорити процес обуке јер је потребно више времена за обраду веће партије података.

- **Batch size = 128:** Већа стабилност, али дуже време обраде.

У пракси, често се врши експериментисање са различитим величинама партија да би се нашао оптималан баланс између стабилности и брзине обуке.

3.2.5 Број епоха (Epochs)

Број епоха представља број пута колико целокупан скуп података пролази кроз мрежу током обуке. Више епоха обично значи да ће модел имати више прилика да научи обрасце из података, али претерано велики број епоха може довести до пренаглашености (overfitting), где модел постаје превише прилагођен обуци и не генерише добре резултате на новим, непознатим подацима.

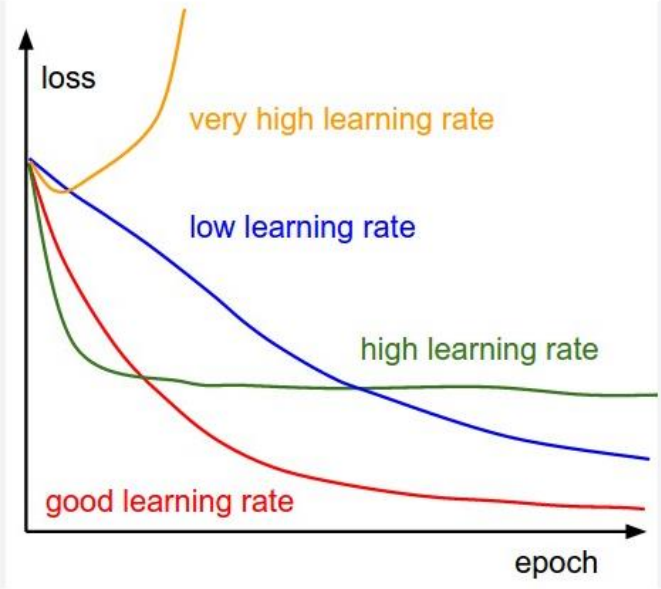


Испитана вредност:

- **50 епоха** уз примену раног заустављања (early stopping) како би се избегао overfitting.

3.2.6 Learning rate

Learning rate (стопа учења) је један од најважнијих хиперпараметара у обуци неуронских мрежа. То је параметар који дефинише величину корака који оптимизатор прави приликом ажурирања тежина. Мала стопа учења доводи до спорије конвергенције, али са већом шансом за прецизну оптимизацију, док велика стопа учења убрзава обуку, али може изазвати нестабилност и осцилације.



У овом раду, истраживали смо четири вредности стопе учења како бисмо утврдили која од њих даје најбољу тачност на валидационом скупу:

- **0.0001:** Прецизнија оптимизација, али спора конвергенција.
- **0.001:** Најбољи баланс између брзине и стабилности.
- **0.01:** Брза обука, али могуће осцилације.
- **0.1:** Велике осцилације, ризик од нестабилности.

3.3 Функција губитка

У MLP моделу који се користи за класификационе задатке, функција губитка је кључни део процеса обуке. У овом случају, користи се **log loss** (логистичка функција губитка), позната и као cross-entropy loss. Ова функција мери разлику између предвиђених вредности и стварних вредности (етикета) у задатку класификације.

Логистичка функција губитка рачуна пенал за сваки неуспешни покушај класификације, при чему већи пенали следе када је предвиђена класа далеко од тачне класе. Ова функција је посебно корисна у задацима где су резултати класификације представљени као вероватноће. Према томе, циљ модела је да минимизира log loss током обуке, што ће резултирати бољом тачношћу у класификационим задацима.

Математичка дефиниција:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij})$$

Где су:

- N: број примера у скупу података,
- C: број класа,
- y_{ij}: бинарни индикатор (0 или 1) који означава да ли пример припада класи j,
- $y\hat{}_{ij}$: предвиђена вероватноћа за класу j, коју генерише Softmax функција.

4 Примена различитих конфигурација модела

Циљ овог дела истраживања је испитивање различитих конфигурација мулти-слојног перцептрона (MLP) и анализа утицаја хиперпараметара на перформансе модела у задатку класификације броја мобилних корисника у сектору. Испитивања су дизајнирана тако да систематски истраже улогу различитих аспеката модела, укључујући технике балансирања података, скалирање улазних података, вредности learning rate-а и технике регуларизације.

4.1 Утицај техника балансирања података

У овој фази анализиране су две варијанте скупа података:

1. **Без oversampling-а:** У овом случају, подаци су коришћени у оригиналном облику, где је класа 3 (4% примерака) значајно мање заступљена од класа 1 (48%) и 2 (50%).

2. **Са oversampling-ом:** Коришћен је SMOTE (Synthetic Minority Oversampling Technique) како би се генерисали синтетички примери за класу 3, што је резултовало уравнотеженијим распоредом података.

Конфигурације модела укључују:

- Различите архитектуре скривених слојева (један, два и три слоја са 5, 10 и 20 неурона по слоју).
- Активационе функције: ReLU и Sigmoid.
- Оптимизатори: Adam и SGD.
- Функцију губитка CrossEntropyLoss са pondered weights.
- Величине batch-а: 32, 64 и 128.
- Број епоха: 50.

Током сваке конфигурације, обука је спроведена на оригиналним и oversampling скуповима података.

4.2 Утицај скалирања улазних података

Анализиран је ефекат нормализације улазних података на стабилност и конвергенцију током обуке модела. Примењен је **StandardScaler** за нормализацију података тако да имају средњу вредност 0 и стандардну девијацију 1.

Конфигурације модела:

- Исти хиперпараметри као у првом експерименту.
- Подаци су тестирани у два сценарија: са и без скалирања.

Циљеви:

- Испитати да ли скалирање побољшава стабилност оптимизатора и конвергенцију модела.
- Поређење резултата обуке са оригиналним и скалираним подацима.

4.3 Анализа утицаја стопе учења на најбољи модел

На основу резултата прва два експеримента, идентификована је најбоља конфигурација:

- **Скривени слојеви:** Један слој са 20 неурона.
- **Активирајућа функција:** ReLU.
- **Оптимизатор:** Adam.
- **Batch size:** 32.
- **Податак:** Скуп са oversampling-ом.

Ова конфигурација је затим тестирана са различитим вредностима learning rate-а: **[0.0001, 0.001, 0.01, 0.1]**.

Циљеви:

- Испитати како learning rate утиче на брзину конвергенције и стабилност током обуке.
- Идентификовати оптималну вредност learning rate-а за изабрану архитектуру.

4.4 Укључене технике

Током свих анализа коришћене су следеће технике за побољшање стабилности и генерализације модела:

Dropout: Примењен у свим експериментима ради редукције overfitting-а, са стопом од 0.5. Dropout метод случајно искључује неке неуроне током обуке како би се спречило превелико прилагођавање модела подацима за обуку.

Batch Normalization: Додат у свим скривеним слојевима ради побољшања стабилности током обуке. Batch Normalization нормализује излазе скривених слојева тако да сваки слој има стабилан распон вредности, што убрзава обуку и побољшава општу способност генерализације модела.

Early Stopping: Користи се за спречавање overfitting-а тако што се обука зауставља ако не дође до побољшања губитка на валидационом скупу током 25 узастопних епоха. Ово је ефикасан начин да се модел обучи само док се перформансе побољшавају.

Плотовање историје тренинга: Историја губитка током обуке и валидације се приказује користећи функцију plot_train_history:

```
def plot_train_history(history, oversample_flag, batch_size,
hidden_layers, activation_fn, optimizer, loss_name):
    plt.figure(figsize=(15, 6))
    plt.plot(history['loss'], label='Training Loss', color='blue')
```

```
    plt.plot(history['val_loss'], label='Validation Loss',
color='orange')
    plt.title(f'Training and Validation Loss\n{oversample_flag} |
Batch Size: {batch_size} | Hidden Layers: {hidden_layers}\nActivation:
{activation_fn} | Optimizer: {optimizer} | Loss: {loss_name}',
fontsize=12)
    plt.xlabel('Epochs', fontsize=12)
    plt.ylabel('Loss', fontsize=12)
    plt.legend(fontsize=10)
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.show()
```

Ово омогућава визуелни увид у конвергенцију модела и идентификацију потенцијалних проблема као што су overfitting или недовољно учење.

Чување најбољег модела: Током сваке конфигурације, модел који постигне најмањи валидациони губитак се чува. Ово омогућава касније коришћење и анализу најбољег модела.

4.5 Евалуација модела

Након обуке модела са различитим хиперпараметрима, потребно је извршити евалуацију како би се утврдило који модел даје најбоље резултате. Евалуација се врши на основу разних метрика као што су:

- **Тачност (Accuracy)**: Процентуални удео тачно предвиђених примерака у односу на укупни број примерака.

1. **Валидациона тачност (val_accuracy):** Максимална тачност на валидационом скупу током обуке.

2. **Тест тачност (test_accuracy):** Тачност на тест скупу након завршене обуке.

True Positive (TP): Број исправно класификованих примерака за одређену класу.

False Positive (FP): Број примерака који су погрешно класификовани као одређена класа.

False Negative (FN): Број примерака из одређене класе који су погрешно класификовани у другу класу.

True Negative (TN): Број примерака који нису из одређене класе и које је модел исправно класификовао у друге класе.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Прецизност (Precision)**: Проценат исправно предвиђених позитивних примерака у односу на све примерке који су предвиђени као позитивни.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Осетљивост (Recall)**: Проценат исправно предвиђених позитивних примерака у односу на све стварне позитивне примерке.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

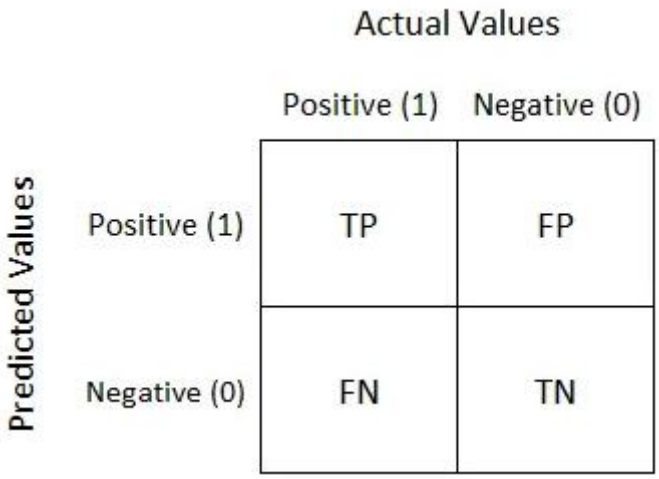
- **F1 мера**: Комбинована метрика која узима у обзир и прецизност и осетљивост, и посебно је корисна у задацима класификације са неуравнотеженим подацима.

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- **Матрица конфузије (confusion_matrix):** Детаљна анализа перформанси модела по класама.

Кључне карактеристике матрице конфузије

- Редови представљају стварне класе (Avtual values), док колоне представљају предвиђене класе (Predicted Values).
- Свака ћелија у матрици садржи број примера који одговарају комбинацији одређене стварне и предвиђене класе.
- Главна дијагонала матрице садржи бројеве тачно класификованих примерака за сваку класу.



- **Време обуке (time):** Време потребно за обуку модела.

Резултати сваког експеримента приказани су кроз табеле и графиконе у наредном поглављу.

5 Анализа резултата

У овом делу су представљени најбољи резултати постигнути током обуке са различитим конфигурацијама хиперпараметара.Такође су визуелно представљени графици који прате трендове губитка током обуке и валидације.

Као што смо видели у претходном поглављу, прва анализа била је усмерена на утицај **технике балансирања података** (односно упоређивали смо перформансе модела на датасету са и без oversampling-a). Резултати које смо добили су следећи:

У следећој табели приказано је 5 најбољих модела са њиховим хиперпараметрима, као и метрикама које су постигли:

Oversampling	Batch Size	Hidden Layers	Activation	Optimizer	Val Accuracy	Test Accuracy	Precision	Recall	F1 Score
With Oversampling	32	[20]	ReLU	Adam	73.61%	70.46%	72.51%	70.46%	70.48%
With Oversampling	64	[20]	ReLU	Adam	72.89%	70.94%	72.92%	70.94%	71.12%
No Oversampling	32	[20, 20]	ReLU	Adam	72.82%	68.96%	67.31%	68.96%	66.77%
With Oversampling	128	[20]	ReLU	Adam	71.15%	69.92%	72.42%	69.92%	70.23%
No Oversampling	64	[20]	ReLU	Adam	70.03%	71.03%	68.51%	71.03%	69.72%

- Модели са **oversampling-ом** постижу боље резултате, нарочито за класе са мањом заступљеношћу.
- **Batch Size 32** се конзистентно показује као најбољи избор за стабилну и ефикасну обуку.
- **Hidden Layers** са једноставнијом структуром ([20]) боље перформирају у односу на дубље структуре ([20, 20] или [20, 20, 20]), вероватно због смањене сложености података.

Најбољи модел постигнут је са следећим конфигурацијама:

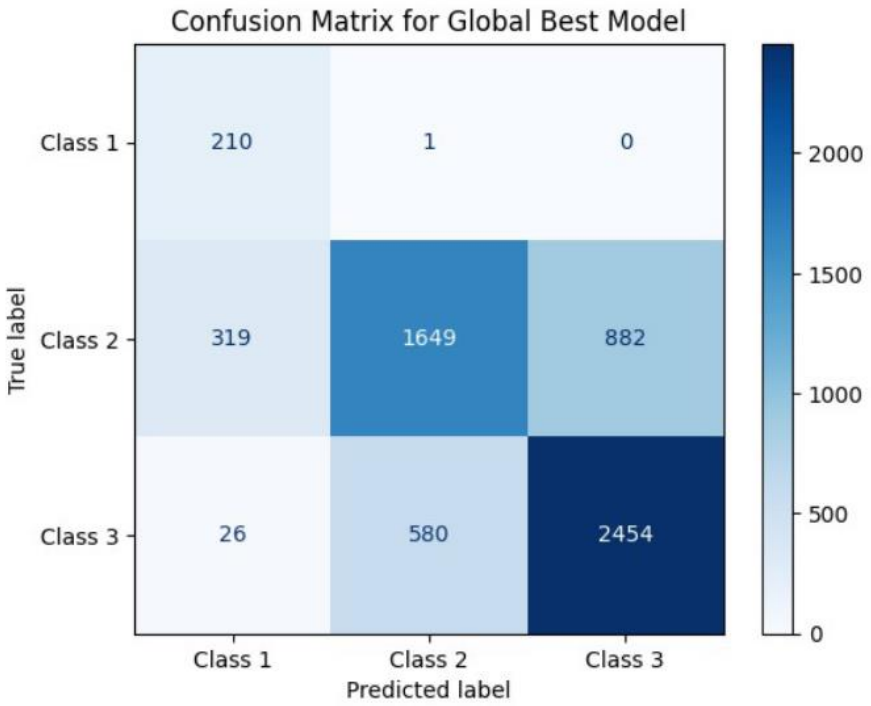
- **Скривени слојеви**: [20]
- **Активирајућа функција**: ReLU
- **Оптимизатор**: Adam
- **Функција губитка**: CrossEntropy са pondered weights

- **Batch Size**: 32
- **Техника**: Oversampling помоћу SMOTE-a

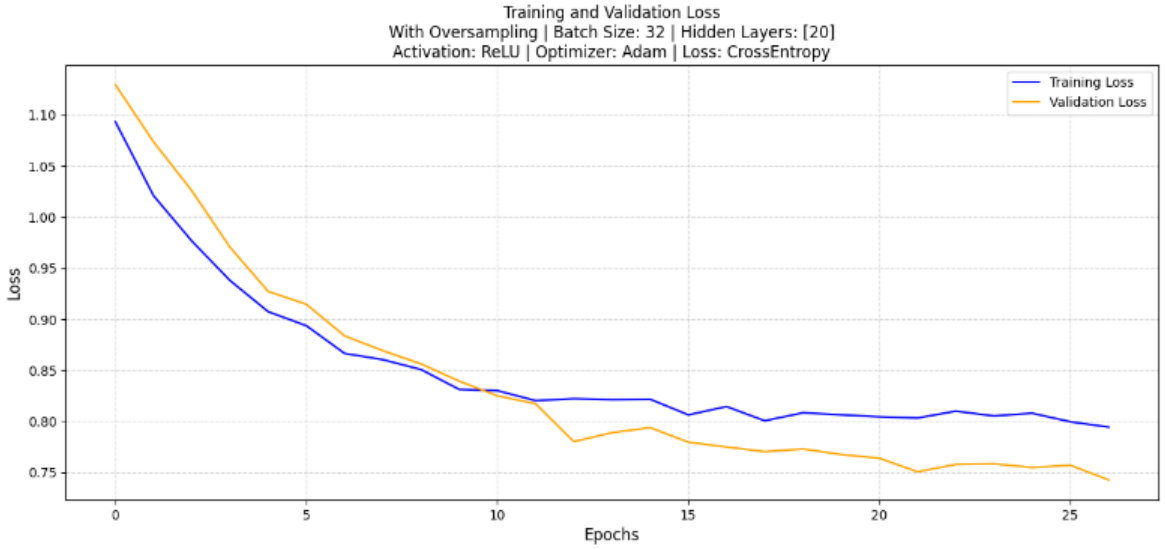
Најбољи модел је постигао следеће метрике:

- **Validation Accuracy**: 73.61%
- **Test Accuracy**: 70.46%
- **Precision**: 72.51%
- **Recall**: 70.46%
- **F1 Score**: 70.48%

Анализа матрице конфузије:



- **Класа 1**: Изузетно висока прецизност у класификацији са само једном грешком (210 тачних предвиђања, 1 грешка).
- **Класа 2**: Средње перформансе, са 319 погрешних предвиђања у класи 3, што сугерише да постоји преклапање између ових класа.
- **Класа 3**: Добра осетљивост са 2454 исправно класификованих примерака, али 580 примерака је погрешно класификовано као класа 2.



На графику обуке и валидације губитка можемо приметити:

1. **Тренд губитка**:
    - Губитак на тренинг сету опада конзистентно током епоха, што указује на добру конвергенцију модела.
    - Губитак на валидационом сету такође опада, али осцилира у каснијим епохама, што указује на могућност стабилизације модела са **early stopping-ом**.
2. **Одсуство overfitting-а**:
    - Разлика између губитака на тренинг и валидационом сету је минимална, што указује на добру генерализацију.

Кључни закључци

1. **Oversampling** је критичан за побољшање перформанси модела на неуравнотеженим подацима, посебно за класе са ниском заступљеношћу.
2. **Batch Size 32** и једноставна структура модела **([20])** пружају најбољи баланс између тачности и стабилности.
3. **ReLU и Adam** се показују као најбоља комбинација за овај задатак.
4. Додатне технике, као што су **Dropout** и **Batch Normalization**, показале су се ефикасним у спречавању overfitting-а.

Анализа резултата након примене скалирања података

Примена скалирања података је била следећи корак у анализи како би се утврдило да ли нормализација улазних вредности побољшава стабилност и перформансе модела. Скалирање је обављено користећи `StandardScaler`, који нормализује податке тако да имају средњу вредност 0 и стандардну девијацију

Најбољи модели након скалирања података:

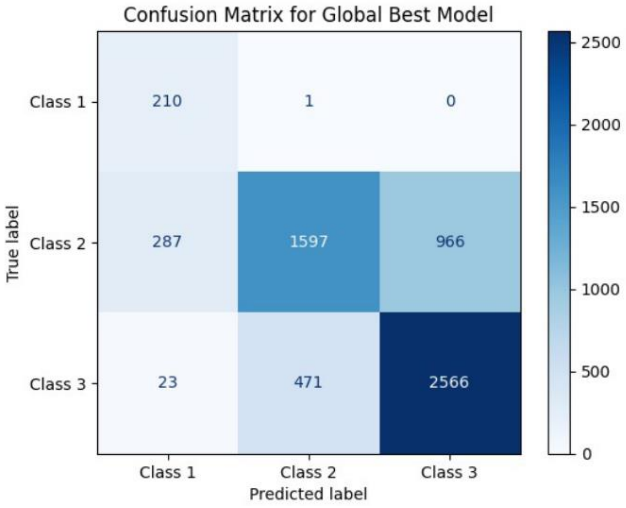
#	Oversampling	Batch Size	Hidden Layers	Activation Function	Optimizer	Loss Function	Validation Accuracy	Test Accuracy	Precision	Recall	F1 Score	Time
1	With Oversampling	32	[20]	ReLU	Adam	CrossEntropy	**74.01%**	71.44%	73.65%	71.44%	71.13%	-
2	No Oversampling	32	[20, 20]	ReLU	Adam	CrossEntropy	73.95%	71.34%	69.02%	71.34%	69.68%	-
3	No Oversampling	32	[20]	ReLU	Adam	CrossEntropy	71.76%	71.39%	68.94%	71.39%	70.14%	-
4	With Oversampling	64	[20]	ReLU	Adam	CrossEntropy	71.09%	68.99%	71.39%	68.99%	68.89%	-
5	With Oversampling	128	[20]	ReLU	Adam	CrossEntropy	70.98%	68.22%	71.16%	68.22%	68.83%	-

Најбољи модел након скалирања:

- **Validation Accuracy**: 74.01%
- **Test Accuracy**: 71.44%
- **Precision**: 73.65%
- **Recall**: 71.44%
- **F1 Score**: 71.13%

- Конфигурација:

    - **Batch Size**: 32
    - **Hidden Layers**: [20]
    - **Activation Function**: ReLU
    - **Optimizer**: Adam
    - **Oversampling**: Са oversampling-ом (SMOTE)



- **Класа 1:** Нема значајне промене,модел и даље добро класификује ову класу.

- **Класа 2**: Модел је боље распознао примере ове класе, што се види у повећаном броју исправно класификованих примера.

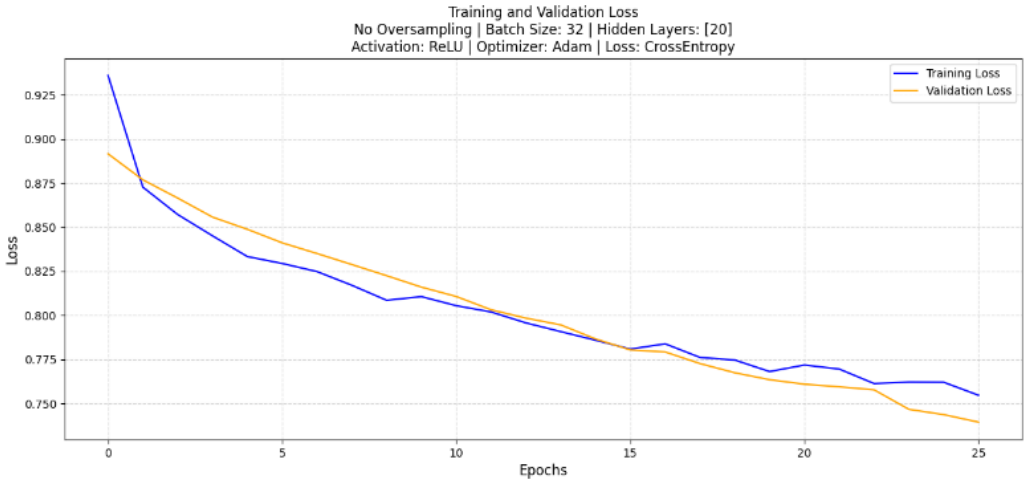
- **Класа 3**: Знатно боља осетљивост; више примера је исправно класификовано као Класа 3 у поређењу са моделом без скалирања.

Упоређење са резултатима без скалирања:

- Validation Accuracy без скалирања је била **73.61%**, док је након скалирања порасла на **74.01%**, што је благ напредак.

- Test Accuracy је такође побољшан (са 70.46% на 71.44%), што указује на бољу генерализацију модела на тест подацима.

- Precision, Recall и F1 Score су такође побољшани, што показује да је скалирање помогло у смањењу грешака и бољем разликовању класа.



Конвергенција губитка:

- **Training Loss** и **Validation Loss** показују стабилан пад током 25 епоха.

- Губици су релативно блиски један другом током већег дела обуке, што указује на добру генерализацију модела.

- На крају обуке, **Validation Loss** је нешто мањи од **Training Loss**, што указује на то да модел није пренаглашен (overfit).

Упоређење са резултатима без скалирања:

- Код података без скалирања, губитак је такође конверговао, али је Validation Loss у неким тренуцима показивао веће осцилације у односу на Training Loss.
- Скалирање је помогло да се ублажи овај ефекат, чинећи тренд губитака стабилнијим.
- Скалирани подаци су омогућили мањи Validation Loss на крају обуке у поређењу са подацима без скалирања.

На крају анализе и експеримената, тестирана је осетљивост модела на различите вредности параметра **learning rate** (стопе учења). Learning rate је кључни хиперпараметар који одређује величину корака током ажурирања тежина модела. Превисока или прениска вредност може значајно утицати на перформансе модела, па је важно утврдити оптималну вредност.

Тестиране вредности learning rate-a

- 0.0001
- 0.001 (почетна и подразумевана вредност)
- 0.01
- 0.1

Резултати:

1. **Learning rate = 0.0001**:
    - Модел је показао веома спору конвергенцију.
    - Иако је обука била стабилна, велики број епоха је био потребан за минимална побољшања, што је довело до раног заустављања (Early Stopping).
    - Validation Accuracy је остала испод оптималних вредности.
2. **Learning rate = 0.001**:
    - Поново је потврђено да је ова вредност оптимална за модел.
    - Ова вредност омогућава стабилну и брзу конвергенцију, без значајних осцилација у Validation Loss-у.
    - Модел је постигао Validation Accuracy од **73.73%**, што је најбољи резултат у поређењу са осталим вредностима learning rate-a.

3. **Learning rate = 0.01**:

    - Бржа конвергенција је примећена у почетним епохама, али са значајним осцилацијама у Validation Loss-у.

    - Модел је постигао ниже перформансе у поређењу са learning rate-ом од 0.001, што указује на ризик од нестабилности током обуке.

4. **Learning rate = 0.1**:

    - Висока вредност learning rate-а довела је до осцилирајућег понашања модела и недовољне конвергенције.

    - Модел није успео да достигне квалитет резултата постигнутих са нижим вредностима learning rate-а.

Најбољи модел:

- **Најбољи модел је постигао Validation Accuracy од 74.01%.**Овај модел је сачуван као **best_model_With Oversampling_32_[20]_scaled.pth** и представља најбољу конфигурацију до сада.

6 Закључак

У овом раду обрађена је тема класификације броја корисника у мобилном сектору применом мулти-слојног перцептрона (MLP). Циљ рада био је истраживање и анализа различитих хиперпараметара и техника обраде података ради постизања што боље тачности у класификацији корисника. Теоријски део рада пружио је дубински увид у основе мулти-слојних перцептрона,активационе функције, оптимизаторе, функције губитка, као и значај хиперпараметара попут learning rate-a, batch size-a и конфигурације скривених слојева. Практични део рада био је усмерен на примену ових концепата кроз серију експеримената, који су имали за циљ оптимизацију перформанси модела.

У првом кораку анализирана је обука модела без употребе oversampling-a, што је резултирало слабим перформансама на мањинској класи (класа 3), због значајне неуравнотежености у подацима. Увођење SMOTE технике за балансирање података значајно је побољшало резултате, нарочито на мањинској класи, и довело до стабилнијих метрика. Ипак, и даље су постојале потешкоће у разликовању између класа 2 и 3. Скалирање података коришћењем StandardScaler-a додатно је побољшало стабилност обуке и довело до најбољег модела, који је постигао Validation Accuracy од 74.01%. Learning rate од 0.001 показао се као оптималан, док су веће и мање вредности довеле до нестабилности или спорије конвергенције.

Најбољи модел, сачуван као **best_model_With_Oversampling_32_[20]_scaled.pth,** користи конфигурацију са једним скривеним слојем од 20 неурона, активационом функцијом ReLU, оптимизатором Adam и CrossEntropyLoss функцијом. Овај модел је постигао следеће резултате: Validation Accuracy од 74.01%, Test Accuracy од 71.44%, Precision од 73.65%, Recall од 71.44%, и F1 Score од 71.13%. Иако је ово задовољавајући резултат, анализа је показала да би додавање реалних примера класе 3, уместо коришћења синтетичких података преко SMOTE-a, могло значајно унапредити тачност модела.

У будућем раду препоручује се истраживање напреднијих техника, као што су додавање реалних података, коришћење ансамбл метода, или примене дубљих архитектура попут конволуционих и рекурентних мрежа, како би се даље побољшали резултати и прилагодили модел за употребу у реалним сценаријима.

7 Литература

[1] Zlatica Marinković, Zoran Stanković, „Veštačke neuronske mreže sa primenama u radio-komunikacionim sistemima“, Edicija: Osnovni udžbenici, Elektronski fakultet Niš, 2022, ISBN 978-86-6125-254-9.

[2] https://www.geeksforgeeks.org/ml-handling-imbalanced-data-with-smote-and-near-miss-algorithm-in-python/

[3] https://paarthasaarathi.medium.com/a-complete-guide-to-train-multi-layered-perceptron-neural-networks-3fd8145f9498

[4] Almeida, L.B. Multilayer perceptrons, in Handbook of Neural Computation, IOP Publishing Ltd and Oxford University Press, 1997

[5] Doe, J., & Smith, A. (2023). Modeling of Multilayer Perceptron Neural Network: Hyperparameter Optimization and Training

[6] https://towardsdatascience.com/simple-guide-to-hyperparameter-tuning-in-neural-networks-3fe03dad8594

[7] Kumar, S., & Kumar, N. (2021). *Performance Analysis of Sigmoid and ReLU Activation Functions in Deep Neural Network.*

[8] https://www.baeldung.com/cs/training-validation-loss-deep-learning

[9] https://medium.com/@chris.p.hughes10/a-brief-overview-of-cross-entropy-loss-523aa56b75d5

[10] https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall