



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
КАТЕДРА ЗА ЕЛЕКТРИЧНА МЕРЕЊА



РАЗВОЈ МЕРНО-ИНФОРМАЦИОНОГ EEG СИСТЕМА ЗА
ИСТРАЖИВАЊЕ УТИЦАЈА КОГНИТИВНИХ СТАЊА НА
ПРОСТОРНО РАЗМИШЉАЊЕ И ЛОГИЧКО РЕЗОНОВАЊЕ

мастер теза

кандидат

Немања Перуничић, Б1 3/2022

ментор

Проф. др Платон Совиль



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:		
Идентификациони број, ИБР:		
Тип документације, ТД:	Монографска документација	
Тип записа, ТЗ:	Текстуални штампани документ	
Врста рада, VR:	Мастер рад	
Аутор, АУ:	Немања Перуничић	
Ментор, МН:	др Платон Совиљ	
Наслов рада, НР:	Развој мерно-информационог EEG система за истраживање утицаја когнитивних стања на просторно размишљање и логичко резоновање	
Језик публикације, ЈП:	Српски	
Језик извода, ЈИ:	Српски	
Земља публиковања, ЗП:	Република Србија	
Уже географско подручје, УГП:	Аутономна Покрајина Војводина	
Година, ГО:	2023.	
Издавач, ИЗ:	Ауторски репримт	
Место и адреса, МА:	Нови Сад, Трг Доситеја Обрадовића 6	
Физички опис рада, ФО:	(поглавља/страна/цитата/табела/спика/графика/прилога)	
Научна област, НО:	Биомедицинско инжењерство	
Научна дисциплина, НД:	Биомедицинско инжењерство	
Предметна одредница/Кључне речи, ПО:	EEG, анализа сигнала, Emotiv, ML, предикција, евалуација	
УДК		
Чува се, ЧУ:	Библиотека Факултета Техничких Наука	
Важна напомена, ВН:		
Извод, ИЗ:	Прво се посматра значај и употреба мерења EEG сигнала током играња видео игрица. Затим се наводе значења таласних опсега сигнала. Даље се говори о самој игрици, и уређају коришћеном за снимање. Након тога следи теоретски, а одмах после и подужи практични пролаз кроз проблематику. На крају се анализирају добијени резултати.	
Датум прихватања теме, ДП:		
Датум одбране, ДО:		
Чланови комисије, КО:	Председник:	
	Члан:	Потпис ментора
	Члан, ментор:	др Совиљ Платон



KEY WORDS DOCUMENTATION

Accession number, ANO:			
Identification number, INO:			
Document type, DT:	Monographic publication		
Type of record, TR:	Textual printed material		
Contents code, CC:	Master project		
Author, AU:	Nemanja Perunicic		
Mentor, MN:	dr Platon Sovilj		
Title, TI:	Development of a measurement and information EEG system for investigating the influence of cognitive states on spatial thinking and logical reasoning		
Language of text, LT:	Serbian		
Language of abstract, LA:	English		
Country of publication, CP:	Republic of Serbia		
Locality of publication, LP:	Autonomous Province of Vojvodina		
Publication year, PY:	2023.		
Publisher, PB:	Author's reprint		
Publication place, PP:	Faculty of Technical Sciences, Dositej Obradovic Sq. 6, Novi Sad		
Physical description, PD:	(chapters/pages/ref./tables/pictures/graphs/appendices)		
Scientific field, SF:	Biomedical Engineering		
Scientific discipline, SD:	Biomedical Engineering		
Subject/Key words, S/KW:	EEG, signal analysis, Emotiv, ML, prediction, evaluation		
UC			
Holding data, HD:	Library of Faculty of Technical Sciences		
Note, N:			
Abstract, AB:	First, the importance and use of measuring EEG signals during video game play is considered. Then the meanings of the signal wavebands are listed. It goes on to talk about the game itself, and the device used for recording. After that, there is a theoretical, and immediately after, a long practical passage through the problem. Finally, the obtained results are analyzed.		
Accepted by the Scientific Board on, ASB:			
Defended on, DE:			
Defended Board, DB:	President:		
	Member:		Menthor's sign
	Member, Mentor:	dr Sovilj Platon	

Садржај

Листинзи кода.....	9
Скраћенице.....	10
1. Увод	13
2. Значај EEG-а у видео игрицама	15
3. EEG опсези.....	19
3.0 Алфа (α) таласи	19
3.1 Бета 1 (β_1) таласи	20
3.2 Бета 2 (β_2) таласи	20
3.3 Гама (γ) таласи	21
3.4 Делта (δ) таласи	22
3.5 Тета (θ) таласи	22
3.6 Резиме о таласима	23
3.7 Снага EEG сигнала	24
4. Миноловац.....	27
4.0 Основно о Миноловцу	27
4.1 Логика и стратегије при игрању Миноловца	28
4.2 Увек решиви Миноловац.....	30
4.3 Анализа EEG-а током играња Миноловца	30
4.4 Решавање Миноловца помоћу ML	31
4.5 Негативни и позитивни аспекти ML у овом случају	33
5. Emotiv EPOS	35
5.0 Спецификације и кључне карактеристике система	35
5.1 EmotivPRO	37
5.2 JSON.....	37
6. Процесирање сигнала	39
6.0 Машинско учење	39
6.1 Прозорирање.....	41
6.2 Матрица обележја и вектор класа	42
6.3 Корелација и крос-варијанса	43
6.4 Класификација и регресија	45
6.5 Класификатор вектора подршке	46
6.6 Класификатор стабла одлучивања.....	47

6.7 Класификатор случајне шуме	48
6.8 Упоређивање класификатора	49
6.9 Theil-Sen регресор	50
6.10 Разлике између Линеарног и Theil-Sen регресора	51
6.11 Регресор на бази градијентног појачања	52
6.12 Регресор на бази мултислојног перцептрона	53
6.13 Упоређивање регресора	54
6.14 Евалуација перформанси класификатора	55
6.15 Евалуација перформанси регресора	57
7. Имплементација система путем Python-а	59
7.0. Учитавање неопходних података	59
7.1 Читање снимљених података о EEG-у	61
7.2 Извлачење података о амплитуди и снази	64
7.3 Креирање имена колона матрице обележја	65
7.4 Прозорирање EEG-а – над амплитудом и снагом	66
7.5 Извлачење обележја из EEG сигнала	67
7.6 Формирање матрице обележја	71
7.7 Класификација – предикција и евалуација	73
7.8 Рачунање усредњеног класификационог извештаја	77
7.9 Регресија – предикција и евалуација	78
7.10 Функција за чување вредности скоровања	83
7.11 Визуализација добијених метрика	83
7.12 За извршавање програма као регуларне скрипте	85
7.13 Провера услова неопходних за извршавање експеримента као регуларне скрипте	88
7.14 За извршавање програма као Streamlit апликације	92
7.15 Чување вредности свих параметара током извршавања кода	98
7.16 Помоћна функција за хипотетички дуготрајни експеримент	98
7.17 Покретање анализе и ML алгоритама	99
7.18 Главно извршавање програма	100
8. Резултати предикције и евалуације	103
8.0 Streamlit интерфејс	103

8.1 Анализа резултата предикције и евалуације.....	105
9. Закључак.....	109
Литература	111
Додатак – Листинг целокупног кода	113

Слике

Слика 1: Блок шема сложеног BCI система базираног на ML (са посебним акцентом на feedback-у).....	14
Слика 2: Пример класичне <i>neurofeedback</i> петље.....	17
Слика 3: Примери употребе neurofeedback-а – са фокусом на рехабилитацију.....	17
Слика 4: Матрица веза између менталног стања и различитих EEG опсега, добијена путем <i>неуроимагинг-а</i>	19
Слика 5: Кратак сажетак карактеристика EEG опсега	24
Слика 6: Амплитуда и снага EEG сигнала – различити домени; временски и фреквентни	25
Slika 7: Primer uspešno završene partije Minolovca	27
Слика 8: Пример одлучивања у Миноловцу на локалном нивоу	28
Слика 9: Ситуација када се сусрећемо са 50/50 случајевима.....	29
Слика 10: Вишеструке могућности за одређивање преосталих мина ...	29
Слика 11: Улога ML у предикцији	33
Слика 12: Emotiv EPOS <i>хедсет</i> и распоред електрода	35
Слика 13: Прилагодљивост <i>хедсет-а</i>; пример са малим дечаком	36
Слика 14: Изглед EmotivPRO интерфејса	37
Слика 15: Блок дијаграм врста ML-а	40
Слика 16: Сликовито појашњење матрице обележја и вектора класа .	43
Слика 17: Разумевање разлике између класификације и регресије – преко формата излаза.....	46
Слика 18: Разумевање разлике између класификације и регресије – преко груписања узорака.....	46
Слика 19: Support Vector Classifier	47
Слика 20: Decision Tree Classifier	48
Слика 21: Random Forest Classifier	49
Слика 22: Linear Regressor vs Theil-Sen Regressor	52
Слика 23: Gradient Boosting Regressor	53
Слика 24: Multilayer Perceptron (Neural Network)	54
Слика 25: Детаљан опис матрице конфузије	56
Слика 26: Неке од метрика које се користе за евалуацију регресорског модела	58
Слика 27: Streamlit – изглед апликације за подешавање параметара и покретање експеримента	104
Слика 28: Streamlit – промене на апликацији након клика на старт дугме	105
Слика 29: Streamlit – испис након успешног завршетка експеримента	105
Слика 30: Приказ последњих 7 колона из матрице обележја сачуване у CSV формату.....	106

Слика 31: (крајње лево) тачност класификације током 10 епоха (центар-лево) усредњена матрица конфузије кроз 10 епоха (центар-десно) усредњен класификациони извештај кроз 10 епоха (крајње десно) израчунати MSE, MAE и R2 током 10 епоха приликом употребе TSR алгоритма	107
Слика 32: Израчунати MSE, MAE и R2 током 10 епоха приликом употребе GBR алгоритма	107
Слика 33: Евалуација кроз 5 епоха – SVC и NNR са наизглед лоше подешеним параметрима	108
Слика 34: Контрола аватара у игрици Elden Ring путем EEG-а.....	109

Листинзи кода

Листинг 1 – увожење неопходних библиотека и њихових подструктуре	60
Листинг 2 – игнорисање свих врста упозорења.....	61
Листинг 3 – извлачење корисних података из CSV и JSON фајлова генерисаних од стране ЕРОС система	63
Листинг 4 – процесирање EEG амплитуда и снага.....	65
Листинг 5 – креирање листе имена колона за будућу матрицу обележја....	66
Листинг 6 – прозорирање EEG амплитуда и снага.....	67
Листинг 7 – извлачење статистичких обележја из EEG-а.....	69
Листинг 8 – креирање комплетне матрице обележја.....	72
Листинг 9 – класификација – предикција и евалуација исхода игре	76
Листинг 10 – рачунање средњих вредности свих поља класификационог извештаја	78
Листинг 11 – регресија – предикција и евалуација времена трајања партије	81
Листинг 12 – све вредности скорова за класификатор и регресор.....	83
Листинг 13 – графички приказ свих резултата евалуације	85
Листинг 14 – дефинисање параметара при регуларном покретању скрипте	87
Листинг 15 – проверавање вредности параметара при регуларном покретању скрипте	91
Листинг 16 – дефинисање параметара као Streamlit елемената	96
Листинг 17 – чување свих параметара у CSV-у	98
Листинг 18 – пример генератора свих нама релевантних комбинација boolean параметара.....	99
Листинг 19 – извршавање функција које образују ML алгоритам.....	100
Листинг 20 – покретање програма	102

Скраћенице

AI	Artificial Intelligence
AJAX	Asynchronous Javascript and XML
AMP	Amplitude
API	Application Programming Interface
AUC	Area Under the ROC Curve
BCI	Brain-Computer Interface
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
DTC	Decision Tree Classifier
EEG	Electroencephalography, Electroencephalogram
EPOC	/
ERP	Event Related Potentials
FN	False Negative
FP	False Positive
GBR	Gradient Boosting Regressor
GIF	Graphics Interchange Format
HTML	Hypertext Markup Language
JSON	Javascript Object Notation
LR	Linear Regression
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MATLAB	Matrix Laboratory
ML	Machine Learning
MLP	Multilayer Perceptron
MOBA	Multiplayer Online Battle Arena
MSE	Mean Squared Error
NaN	Not a Number
NNR	Neural Network Regressor
POW	Power
PNG	Portable Network Graphics
R2	R-Squared, Coefficient of Determination
RFC	Random Forest Classifier
RMSE	Root-Mean-Square Error

ROC	Receiver Operating Characteristic Curve
SMOTE	Synthetic Minority Over-Sampling Technique
SVC / M	Support Vector Classifier / Machine
TN	True Negative
TP	True Positive
TSR	Theil-Sen Regressor
VR	Virtual Reality
XML	Extensible Markup Language

Захвалница

Захвалајем се проф. Платону Савиљу и колеги Николи Петровићу на пруженом материјалу и асистенцији током израде овог рада.

1. Увод

Анализа електроенцефалографских (EEG) сигнала измерених током играња видео игрица представља фасцинантно поље пресека мноштва наука, између остalog: неурологије, биомедицине, математике, и психологије. Ово ново истраживачко подручје уноси значајне могућности за унапређење људског разумевања мозданог реаговања на интерактивне дигиталне стимулансе. У овом научно-истраживачком раду, дубље ће се залазити у анализу EEG сигнала у контексту играња видео игрица; разлажући ову тему на мање, разумљивије кораке.

Пре свега, кључно је прво разумети основе EEG сигнала – EEG је неинвазивна неуроимагинг техника која бележи групну (сумарну) електричну активност неурона из коре великог мозга. Електроде постављене на скалпу мере флукутације напона које произилазе из колективне активности хиљада неурона при извршавању било које мисли или радње. EEG пружа високу темпоралну резолуцију, омогућавајући нам да ухватимо активност мозга у стварном времену.

Видео игрице, у 21. веку, еволуирале су у широко прихваћен облик забаве, а и когнитивног ангажовања (тј. занимације). Играчи, када их користе, пролазе кроз комплексне виртуелне околине, доносе брзе одлуке и доживљавају широк спектар емоција током самог процеса играња. Ови интеракциони процеси чине видео игрице једним од идеалних поља за проучавање когнитивних процеса, пажње, регулације емоција и учења у контролисаном, али притом и динамичном окружењу.

Овде представљено истраживање се фокусира на иновативан приступ разумевању тога како когнитивни процеси утичу на перформансе у задацима који захтевају просторно размишљање и логичко закључивање. Коришћени су EEG сигнали прикупљени током играња Мајкрософтове игрице Minesweeper (у духу српског језика: Миноловац), путем комерцијализованог уређаја Emotiv EPOC.

Подаци прикупљени на овај начин су коришћени за обуку модела базiranog на вештачкој интелигенцији (енг. Artificial Intelligence; AI); тачније на машинском учењу (енг. Machine Learning; ML). Овај модел има задатак да на основу EEG сигнала предвиђа перформансе играча, конкретно: исход игрице (победа или пораз) и време потребно за завршетак (у секундама).

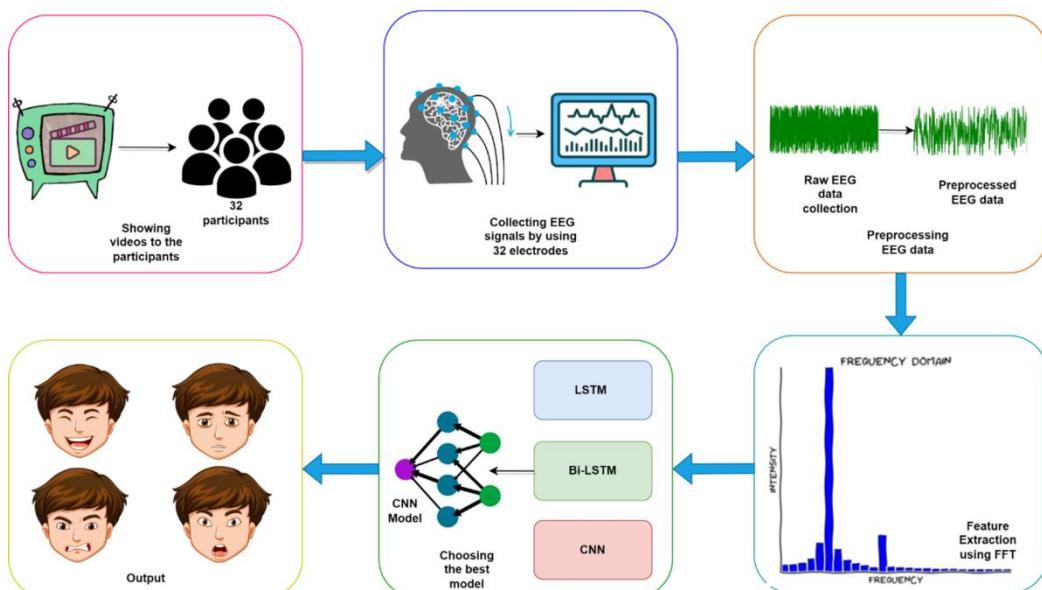
Интегрирани приступ не само да открива динамичне везе између когнитивних процеса, већ такође приказује базу за развој сопствене методологије за анализу и тумачење когнитивних стања на основу EEG сигнала. План истраживања укључује развој система за процесирање и синхронизацију EEG података, након тога за класификацију и регресију, и на крају за анализу резултата – како би се потврдила хипотеза о могућности предвиђања перформанси на основу EEG сигнала.

У општем случају, анализу EEG сигнала у контексту видео игрица можемо посматрати кроз следећи истраживачки процес:

- Дизајн самог експеримента – пројектовање експеримента који укључују специфичне играчке задатке или сценарије. Ови задаци могу варирати од једноставних тестова пажње до сложених изазова доношења одлука.

- Прикупљање података – истраживачи постављају EEG електроде на скалп испитаника током играња видео игрице, где систем потом непрекидно бележи активност мозга током играња.
- Екстракција карактеристика (илити обележја) – EEG сигнали су сирови и комплексни – да бисмо разумели податке, морамо издвојити релевантне карактеристике, као што су нпр. спектрална снага или корелација. Ове карактеристике пружају увид у различите аспекте активности мозга.
- Предобрада сигнала – EEG подаци често садрже шум, разне периодичне или апериодичне артефакте, и помераје базне линије сигнала. Предпроцесирање, укључујући филтрирање и уклањање артефаката, кључна је за осигурање квалитета података.
- Математичка анализа – захтева употребу напредних алата, и она је заправо најбитнији део овог рада. Овде се упоређују EEG карактеристике добијене употребом вредности параметара услова или просто различитих сигналних група, како би извукли за нас значајне закључке.

На Сл. 1 испод је приказана блок шема једног сложеног EEG система, који у себи садржи битне елементе и појмове о којима ће бити приче у даљем тексту.



Слика 1: Блок шема сложеног BCI система базираног на ML (са посебним акцентом на feedback-y)

2. Значај EEG-а у видео игрицама

Електроенцефалографија (енг. електроенцефалографија, електроенцефалограм; EEG) је врста електрофизиолошког снимања, која служи за прикуп Проучавање EEG сигнала измерених током играња видео игрица има широк појас примене; нпр. може помоћи у унапређењу дизајна игрица прилагођавањем искуства менталном стању играча, док са друге стране даје допринос и код дијагностиковања и лечења неких неуролошких оболења, попут поремећаја пажње или поремећаја везаних за стрес.

EEG сигнале можемо анализирати различитим методама, где свака пружа различите увиде у когнитивне процесе. Једна од њих која нам је јако битна је спектрална анализа, која разлаже EEG сигнале на различите фреквенцијске појасеве, као што су делта, тета, алфа, бета и гама – где је сваки појас повезан са специфичним функцијама мозга (нпр. алфа таласи су повезани са опуштањем, док бета таласи указују на активне когнитивне процесе).

Овде, иако немају улогу у овом раду, можемо споменути и Event Related Potentials (ERP); то су специфични обрасци EEG активности који се јављају као одговор на одређене догађаје или стимулансе унутар игрице. Они нам омогућавају да прецизно одредимо тачно време когнитивних процеса, као што су промене пажње или чување краткорочне меморије.

Јасно је да је разумевање тога како EEG сигнали повезују са когнитивним стањима кључно у анализи видео игрица. EEG може открити када и којој мери су играчи дубоко уроњени у игру или су, напротив, ометени. Повећана тета активност, нпр. може указивати на смањену пажњу, док виша бета активност може значити усмерену когнитивну обраду (размишљање у циљу проналаска решења). Надоградња овога би била регулација емоција – видео игрице често изазивају различите бурне емоције; EEG сигнали могу помоћи у идентификацији тренутака када играчи доживљавају узбуђење, задовољство, стрес или фрустрацију [1], помажући девелоперима игрица да прилагоде искуства како би изазвали одређене (обострано жељене) емоционалне реакције.

Даља надоградња на претходно споменуто би била примена анализе EEG-а ради добијања повратних информација у реалном времену. Процесирањем сигнала у току игрице, оне се активно могу прилагођавати когнитивном стању играча – ако се играч превише стресира, игрица ће на основу претходно дефинисаних параметарских услова увести смирујуће елементе или смањити ниво тежине како би побољшала забаву. Предност овог концепта је и већа прилагодљивост, зато што је неефикасно унапред дефинисати неке параметре као константе; јер сваки играч је ментално прича за себе, а ту треба урачунати и природну, свакодневну варијабилност сваке индивидуе.

Иако показује огроман потенцијал, анализа EEG сигнала у видео игрицама носи неколико изазова, укључујући висок степен варијабилности података и потребу за софицицираним методама анализе. Будућа истраживања имају за циљ да се суоче са овим изазовима и открију дубља сазнања око межданих одговора на видео игрице. Кроз ово истраживачко поље, циљ је да се разоткрију комплексни односи између људског мозга и разноразних интерактивних дигиталних искустава, што

даље отвара пут за иновације у игрицама и дубље разумевање човечијих когнитивних процеса.

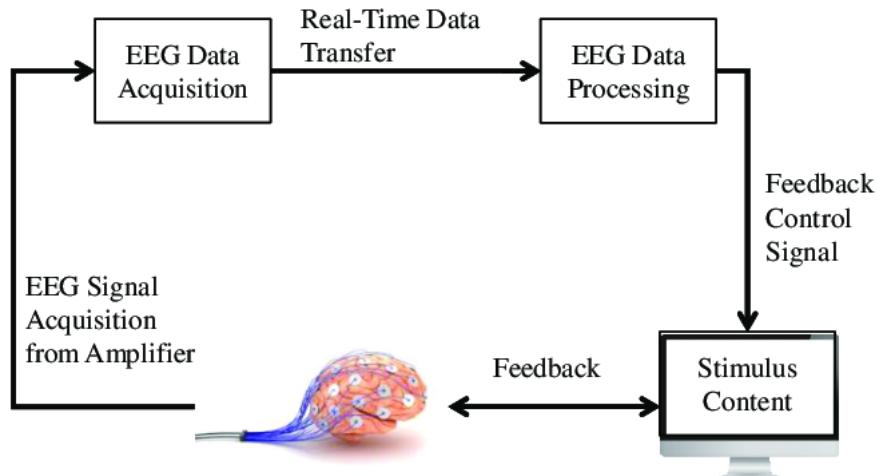
Наравно, употреба EEG у анализи видео игрица поставља и етичка питања у вези са приватношћу и информисаним пристанком. Истраживачи и развојни девелопери игрица се морају побринути да учесници буду свесни процеса прикупљања података и његових последица – заштита приватности података о мозгу играча је обавезан корак у сваком научно-истраживачком раду.

У последњих неколико година, индустрија видео игрица је препознала потенцијал EEG технологије да унапреди искуство играча, оптимизује дизајн игрица и пружи друге вредне увиде [2][3]. Мозак-рачунар интерфејси (енг. Brain-Computer Interface; BCI) омогућавају играчима да контролишу аспекте игрице путем својих мисли [4]. У контексту индустрије видео игрица, BCI може играчима пружити за сада незапамћени ниво уроњености (тј. удубљености). Играчи могу манипулисати објектима у игрици, навигирати виртуелним световима или доносити одлуке путем сигнала из свог мозга, стварајући тако ново, занимљивије и узбудљивије искуство играња.

Са становишта профита то је једна од кључних примена EEG у индустрији видео игрица – праћење ангажованости играча – анализом EEG-а можемо добити увиде у то како играчи реагују на игрицу у реалном времену. Ове информације омогућавају девелоперима да прилагоде механике игрице, и изазове и наративе како би ангажованост играча била на високом нивоу, а притом и конзистентна током целог искуства играња [5].

Фокусираност (илити фокус) је психолошко стање у којем појединци постају потпuno урођени у активност којом се тренутно баве, често описано као да су у зони. Истраживачи су користили анализу EEG-а како би проучавали маждану активност играча који доживљавају фокус током играња. Идентификацијом специфичних EEG образаца повезаних са фокусом, девелопери игрица могу дизајнирати искуства која олакшавају и одржавају ово високо задовољавајуће стање, побољшавајући ангажованост играча. Превелика фокусираност може бити лоша, те је за дизајн игрице битно и разумевање когнитивног оптерећења које играчи могу доживети. Ове информације помажу девелоперима да постигну равнотежу између изазовног играња и когнитивног напора.

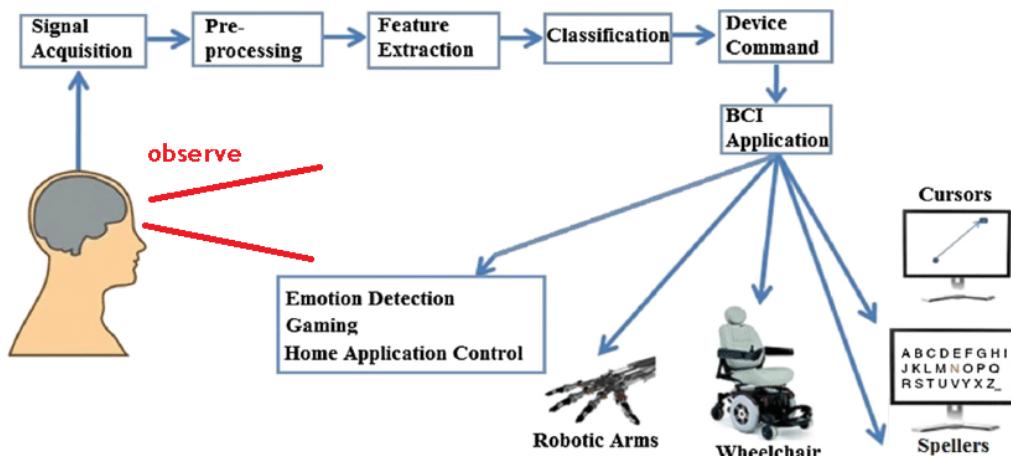
Neurofeedback игрице (Сл. 2 и 3) базиране на EEG-у, дизајниране за когнитивно обучавање (у циљу когнитивног јачања) су већ сада доволно развијене, па да се могу донекле комерцијализовати, а очекује се да ће постати и популарне. Ове игрице користе EEG сигнале у терапеутске сврхе, како би побољшале когнитивне функције као што су памћење, концентрација и опуштање. Играчи добијају повратне информације о мажданој активности у реалном времену, што им омогућава активно тренирање и унапређење својих когнитивних способности, чинећи играње не само забавним, већ и корисним за ментално здравље и благостање.



Слика 2: Пример класичне *neurofeedback* петље

Када зађемо мало дубље у ову конкретну тему, можемо посматрати и:

- Унапређење виртуелне реалности (енг. Virtual Reality; VR) – у свету VR игрица, EEG сигнали се користе за фино подешавање целокупног играчког искуства (утиска) [6].
- Проучавање зависности [7] – праћењем маждане активности код особа које показују знакове зависног играња, истраживачи су стекли увиде у неуралне корелате зависности. Ово истраживање може информисати превентивне мере и интервенције за особе које су подложне прекомерном конзумирању видео игрица.
- Утицај страних елемената – за нпр. истраживање ефекта које изазива рекламирање (енг. ad(s)) унутар игрице на пажњу и уроњеност играча.
- Когнитивне рехабилитационе игрице – у суштини су већ споменуте у тексту изнад, но овде се потенцира то да се користе у процесу опоравка од неуролошких повреда или стања. Ове игрице прилагођавају нивое тежине на основу EEG сигнала пацијента, осигуравајући да су вежбе за рехабилитацију истовремено ангажујуће и ефикасне.

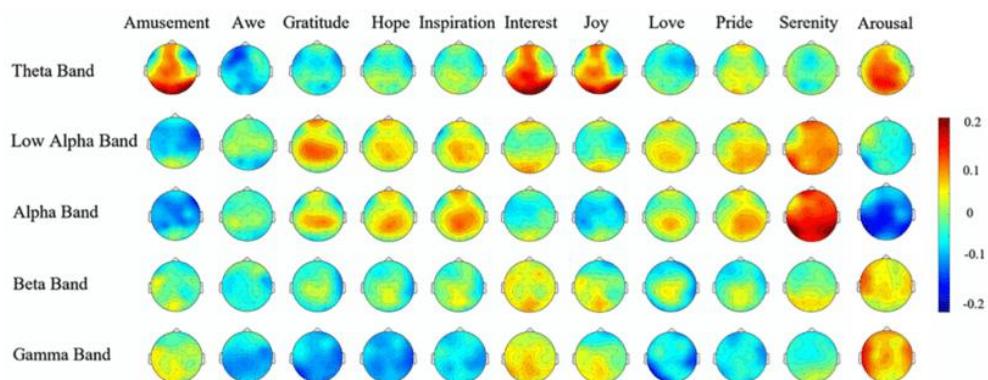


Слика 3: Примери употребе neurofeedback-а – са фокусом на рехабилитацију

Са даљим напретком технологије, може се очекивати комерцијализација минијатуризованих бежичних EEG уређаја, што ће омогућити далеко шири спектар студија. Такође, ML и AI ће побољшати аутоматизовану анализу EEG података, омогућавајући брже, тачније и прецизније увиде у њих.

3. EEG опсези

EEG фреквентни оспези (енг. bands) представљају различите фреквенције електричних осцилација које се могу уочити на снимцима. У овом раду они су искоришћени за формирање неких од обележја – зато што довољно дистинктни; сваки опсег има своје карактеристике и повезан је са одређеним стањима свести и менталним процесима (Сл. 4). Називе су добили из грчког алфабета, и то по редоследу открића.



Слика 4: Матрица веза између менталног стања и различитих EEG опсега, добијена путем неуроимагинг-а

3.0 Алфа (α) таласи

- Фреквенција – од 8 до 13 Hz.
- Амплитуда – од 5 до 50 μ V.
- Ритмичка природа – алфа таласи се карактеришу својом ритмичком, осцилаторном природом. Они показују редовнан, синусоидални образац, што их чини препознатљивим у односу на друге EEG фреквенције.
- Локација – јављају се претежно у задњим деловима мозга, пре свега у окципиталном режњу. Међутим, може се такође посматрати и у другим деловима мозга, зависно о индивиду и њихових когнитивних стања.
- Стања ума – најизраженији су када је особа будна, али опуштена и није активно укључена у интензивне когнитивне задатке. Често се повезују са стањем лутајућег ума, маштањем или мирним и будним менталним стањем.
- Затварање очију појачава алфа активност – када особе затворе очи, алфа таласи постају израженији, што указује на прелазак у опуштеније стање.

Током играња видео игрица, присуство или одсуство алфа таласа може пружити увиде у ниво пажње и ангажованости играча. Смањена алфа моћ у одређеним деловима мозга може указивати на повећану пажњу и когнитивно ангажовање (уз то иде и когнитивно оптерећење), сугеришући да је играч дубоко уроњен у игру. Такође, аналогно претходно реченом, могу открити ниво стреса или опуштања играча. Повећана алфа моћ, посебно у окципиталном региону, може указивати на стање опуштања, док смањење активности алфа може значити стрес или повећану емоционалну узбуђеност.

Дуготрајна висока активност алфа таласа, посебно у одсуству ангажујућих стимуланса, може указивати на ментални умор или досаду. Девелопери игрица могу користити ове информације да идентифikuју тренутке када играчи могу да се искључе из игрице због перципирање монотоније.

Пошто EEG технологија омогућава повратне информације у реалном времену и прилагодбу у видео игрицама, подаци о алфа таласима се могу користити да прилагоде ниво тежине игрице или да уведу умирујуће елементе као одговор на стрес играча, стварајући тако прилагођено и уживање играчко искуство.

3.1 Бета 1 (β 1) таласи

- Зову се и бета ниски таласи.
- Фреквенција – од 13 до 20 Hz.
- Амплитуда – од 5 до 30 μ V.
- Локација – активност ниског бета таласа се генерално распоређује по централним и задњим деловима мозга, укључујући сензомоторни кортекс.
- Когнитивно ангажовање – повезују се са когнитивним ангажовањем и менталном будношћу. Постају израженији када особа активно фокусира на неки задатак, или је ангажована у решавању проблема.
- Сензомоторна активност – играју улогу у координацији моторичких покрета; посебно финих моторичких вештина и прецизних радњи (нпр. било који рад који захтева употребу прстију шака).

При игрању видео игрица, повећање моћи ниског бета таласа може сугерисати да играч активно концентрише на задатке унутар игрице, као што су стратегија, доношење одлука или решавање новонасталих проблема (без обзира на то да ли су већ виђени). Пошто су повезани и са сензомоторном обрадом, њихова анализа може пружити увиде у перформансе моторичких вештина играча. Висок ниво активности ниског бета таласа може указивати на прецизну контролу радњи унутар игрице, као што су нишање, циљање или навигација кроз комплексне играчке околине.

Праћење промена у ниским бета таласима током времена може помоћи у процени учења и прилагођавања играча унутар игрице. Повећање снаге ниског бета таласа може указивати на то да играч постаје вештији у механици или стратегијама неке игрице. Иноврзно у односу на алфа, али са истим закључком: прекомерно висок ниво активности ниског бета таласа може указивати на когнитивно оптерећење или стрес током изазовних сценарија игрице – ове информације могу помоћи да се избалансира тежина игрице (ово конкретно важи за све опсеге).

3.2 Бета 2 (β 2) таласи

- Зову се и бета високи таласи.
- Фреквенција – од 20 до 30 Hz.
- Амплитуда – од 2 до 20 μ V.
- Локација – посматрају се у централним и фронталним деловима мозга.

- Когнитивно ангажовање – повезани са когнитивним процесима вишег реда и битним извршним функцијама. Постају израженији када појединци учествују у сложеним менталним задацима који захтевају пажњу, доношење одлука и обраду информација.
- Реакција на стрес – у одређеним контекстима, високи бета таласи такође могу бити повезани са реакцијама на стрес, посебно када су појединци суочени са изазовним или емоционално набијеним ситуацијама.

Улога високих бета таласа у анализи видео игрица се огледа у интензивној когнитивној обради и активном решавању сложених задатака. Повећање снаге високих бета таласа може указивати на то да је играч дубоко уроњен у стратешко размишљање, решавање проблема током сценарија који подразумевају висок притисак на перформансе играча, илити мултитаскинг унутар игрице.

Као и ниски бета таласи, висока бета активност може бити повезана са прецизном моторичком контролом; где је тајминг јако битан. Штавише, промене у обрасцима високих бета таласа током времена указују на учење и развој вештина играча. Повећање снаге високих бета таласа може да сугерише да играч постаје вештији у савладавању механике игрице или стратегијама.

3.3 Гама (γ) таласи

- Често се зову и гама активност, чиме се алудира на то да су један од најинтригантнијих и најкомплекснијих аспеката мождане активности (разуме се, није сва мождана активност обухваћена овде наведеним опсезима из грчког алфабета).
- Фреквенција – од 30 до 100 Hz; уз напомену да могу ићи и знатно преко ове стандардно навођене горње границе.
- Амплитуда – од 1 до 10 μ V.
- Локација – могу се појавити у различитим регијама мозга, зависно о конкретном когнитивном или сензорном процесу који се проучава.
- Когнитивни процеси – повезани су са разним когнитивним процесима [8], укључујући пажњу, опоравак меморије, перцепцију и решавање сложених проблема. Сматрају се кључним за повезивање различитих елемената сензорних информација у кохерентно перцептивно искуство.
- Сензорно процесирање – играју кључну улогу у синхронизацији неуронског пражњења између различитих региона мозга. Укључени су у интеграцију сензорних информација из различитих модалитета.

У анализи видео игрица гама таласи нам говоре о побољшаној перцепцији и пажњи – повећање гама снаге може указивати да је играч интензивно усмерен на визуелне и аудитивне сигнале игрице, побољшавајући њихов степен реактивности. Видео игрице често укључују комплексна сензорна искуства, комбинујући визуелне, аудитивне и понекад тактилне елементе. Аналогно њиховој биолошкој улози, гама таласи и овде помажу постизању синхронизације различитих стимуланса.

Процеси меморије су витални у видео игрицама, посебно за сећање на правила игрице, стратегије и просторне информације – анализом гама таласа можемо

открити неуронске механизме иза кодирања и опоравка меморије током играња. У стратегијским или слагалицама, играчи се баве решавањем сложених проблема – повећање гама активности може указивати на активно когнитивно процесирање везано за решавање изазова или слагалица унутар игрице. Посматрањем кохерентности гама таласа може се пружити увид у то како различите моздане области комуницирају током играња.

3.4 Делта (δ) таласи

- Првенствено се јављају током дубоког сна, али се могу појавити и током одређених будних стања. У снажној су вези са дубоким, спороталасним сном, повезаним са физичким и менталним опоравком током сна.
- Фреквенција – од 0.5 до 4 Hz.
- Амплитуда – од 20 до 200 μ V.
- Локација – најизраженији су у фронталним и централним регионима мозга током дубоког сна. У будним стањима, обично се јављају у регионима повезаним са специфичним когнитивним процесима.
- Когнитивни процеси – иако су делта таласи првенствено повезани са сном, могу се појавити и током будних стања у случајевима екстремне релаксације или повреде мозга. Када се виде у будном стању, често су повезани са оштећеном когнитивном функцијом.

Присуство делта таласа током будних стања, посебно у одсуству релаксације или активности која изазива сан, може указивати на умор или недостатак сна код играча, као и на неангажовање или досаду. Ова информација може бити вредна за процену благостања играча током продужених сесија играња. Анализа узорака делта таласа може помоћи идентификацији тренутака када је когнитивна функција играча компромитована због умора или других фактора.

3.5 Тета (θ) таласи

- Фреквенција – од 4 до 8 Hz.
- Амплитуда – од 10 до 100 μ V.
- Локација – посматрају се у различитим регијама мозга, укључујући фронталне, централне и темпоралне области.
- Когнитивна и емоционална стања – повезани су са разним когнитивним и емоционалним стањима, укључујући опуштање, сањарење, креативност и ране фазе сна. Често се јављају током задатака који захтевају пажњу, али не и активно решавање проблема.
- Памћење – имају значајну улогу у процесима меморије, посебно код кодирања и опоравка епизодичних меморија. Њихово присуство се повезује са формирањем нових меморија и реактивацијом постојећих.
- Емоције – учествују у процесирању емоција и могу бити повезани са регулацијом емоција и изражавањем емоционалних реакција.

У видео игрицама указују на умерено когнитивно ангажовање (тј. праве разлику у односу на дубљу концентрацију) – док играчи навигирају кроз играчко окружење, доносе одлуке или одржавају пажњу на наратив игрице. Тета таласи могу пружити увиде у то како играчи кодирају и употребљавају информације унутар игрице, као што су правила игрице, локације предмета или циљеви задатка.

3.6 Резиме о таласима

Алфа таласи:

- Ознака су опуштених стања, без узбуђења.
- Могу указивати на смањену пажњу или досаду током игрице.
- Вредни су за праћење релаксације играча и емоционалних стања током играња.

Бета 1 таласи:

- Указују на фокусирено когнитивно процесирање и пажњу.
- Корисни су за праћење доношења одлука, перформанси моторичких вештина и учења у игрицама.
- Вредни су за прилагођавање изазова игрице на основу когнитивног ангажовања.

Бета 2 таласи:

- Означавају више когнитивно процесирање и решавање проблема.
- Вредни су за процену сложеног решавања проблема, кодирања меморије и доношења одлука током играња.
- Могу помоћи у прилагођавању изазова игрице и награда на основу нивоа когнитивног ангажовања.

Гама таласи:

- Повезани са побољшаном перцепцијом, пажњом и сензорном интеграцијом.
- Указују на веома високо когнитивно ангажовање и брузу обраду информација.
- Корисни за разумевање сензорне обраде, меморијског кодирања и сложеног решавања проблема током играња.

Делта таласи:

- Повремено присуство може указивати на умор или когнитивно оштећење током игрице.
- Може указивати на добробит играча и потребу за паузама током продужених сесија игрица.
- Праћење делта активности помаже да се осигура здравље играча и когнитивно функционисање.

Тета таласи:

- Повезани са умереним когнитивним ангажовањем и релаксацијом.
- Играју улогу у кодирању меморије и поновном присећању током игрице.
- За разумевање когниције играча, процеса памћења и неких емоционалних одговора.

На Сл. 5 испод је дат још краћи, табеларни приказ свега што је речено.

Brainwave Type	Frequency Range (Hz)	State of the brain
Delta (δ)	0.1Hz to 3Hz	Deep, dreamless sleep, non-REM sleep, unconscious
Theta (θ)	4Hz to 7Hz	Intuitive, creative, recall, fantasy, imaginary, dream
Alpha (α)	8Hz to 12Hz	Relaxed, but not drowsy, tranquil, conscious
Low-range Beta (β)	12Hz to 15Hz	Formerly SMR, relaxed yet focused, integrated
Mid-range Beta (β)	16Hz to 20Hz	Thinking, aware of self & surroundings
High-range Beta (β)	21Hz to 30Hz	Alertness, agitation
Gamma (γ)	30Hz to 100+Hz	Motor Functions, higher mental activity

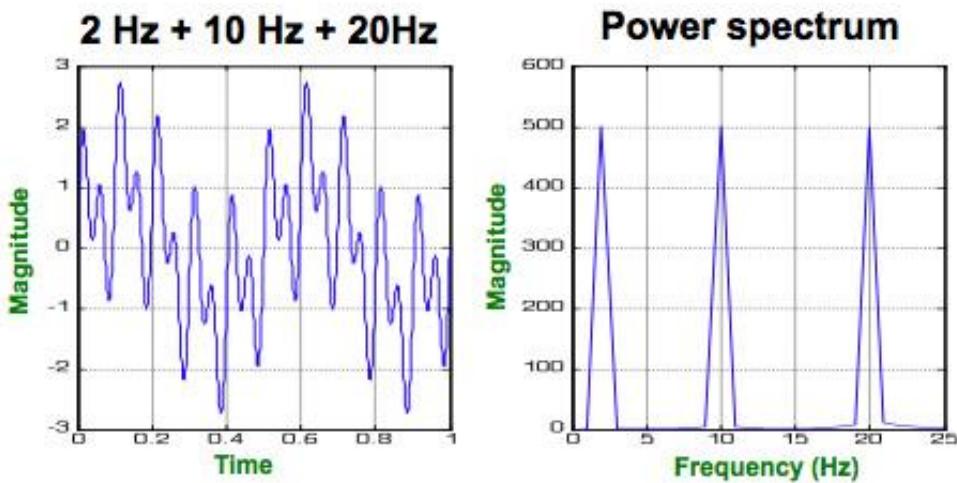
Слика 5: Кратак сајетак карактеристика EEG опсега

Са становишта ML алгоритма ове теоријске разлике у значењима различитих опсега нису битне, но битно је да знамо (да смо сигурни) да су нам обележјаовољно различита – да би имало смисла користити их.

3.7 Снага EEG сигнала

Снага EEG сигнала пружа додатне информације, изван саме амплитуда, а те информације су драгоцене за разумевање сложености неуронске активности у мозгу. Главни разлог због којег је снага EEG сигнала важна и потребна је то што даје увиде који зависе од фреквентног опсега – пошто снага квантификује снагу неуронских осцилација унутар различитих фреквентних опсега (Сл. 6). Амплитуде, с друге стране, дају општу меру јачине сигнала, али не праве разлику између доприноса различитих фреквенцијских компоненти. Анализом снаге у специфичним фреквентним опсезима, можемо да стекнемо увид у релативно ангажовање различитих когнитивних и неуронских процеса.

Спектар EEG сигнала је структуриран тако да има мању снагу на вишим фреквенцијама, што приближно следи $1/f$ образац – то значи да је снага сигнала обрнуто пропорционална фреквенцији, тј. да ниске фреквенције имају већу снагу него високе фреквенције.



Слика 6: Амплитуда и снага EEG сигнала – различити домени; временски и фреквентни

У вези стим, треба споменути neurofeedback и biofeedback – такви системи омогућавају појединцима да регулишу своју моздану активност пружањем повратних информација у реалном времену на основу промена снаге у одређеним фреквентним опсезима. То може имати терапеутско дејство код стања као што су хиперактивност, анксиозност и несаница.

Битна нам је и темпорална динамика; снага сигнала обухвата временску динамику моздане активности, тј. показује како се снага неуронских осцилација мења током времена, омогућавајући истраживачима да посматрају флуктуације у когнитивним стањима и одговоре на стимулусе. Овај временски аспект је кључан за проучавање процеса као што су пажња, памћење и емоционални одговори, који се увек динамички мењају и развијају.

Уз све то треба споменути и присуство сложене интеракција у склопу EEG-а, зато што током когнитивних задатака, може доћи до унакрсне спрече, где снага једног фреквентног опсега модулира амплитуду или фазу другог. Ове интеракције могу открити замршене односе између различитих когнитивних процеса, који се не могу адекватно регистровати само мерењем амплитуде.

Видео игрице се могу користити за стимулацију многих делова мозга, а снага EEG сигнала може пружити увид у то како се различити делови мозга активирају током играња видео игрица. Једна студија је спровела систематски преглед техника компјутерске EEG анализе приликом играња видео игрица [9]. Студија је открила да су коришћене технике заиста различите и да су приступи који се користе за BCI и когнитивну анализу веома слични. Такође је потврђено да се снага EEG сигнала може користити за процену како се људи осећају када играју видео игрицу.

Друга студија се фокусирала на откривање пажње у виртуелним окружењима користећи EEG сигнале [10]. Студија је осмишљена да идентификује кораке анализе и обраде EEG сигнала у односу на окружења виртуелне интеракције и да укаже на главне екстраховане карактеристике и алате повезане са системом имерзије који користи EEG сигнал.

Укратко, снага EEG сигнала је корисна алатка када се анализирају подаци прикупљени током играња видео игрица, која нам пружа далеко ширу слику о томе шта се догађа у нашим мозговима.

4. Миноловац

4.0 Основно о Миноловцу

Мајкрософтов Миноловац је класична, култна слагалица за једног играча која је унапред инсталација на различитим оперативним системима. Примарни циљ игрице је да се открију све скривене мина на табли без активирања иједне од њих – да би то постигли, играчи морају да користе логику и дедукцију [11].

Окружење:

- Игра се на табли (Сл. 7), која се састоји од квадрата (поља, ћелија) распоређених у редове и колоне.
- Величина табле може да варира, а најчешће предефинисане опције су 9x9, 16x16 и 16x30.
- На табли, која је на почетку скроз неоткривена, се налазе скривене мина – у неким верзијама игрице могуће је аутоматски изгубити након првог кликтаја.

Циљ:

- Главни циљ Миноловца је да се открију сви квадрати на мрежи који не садрже мина.
- Играч побеђује у игри када се открију сва поља која нису минска; другим речима, ако је мрежа потпуно откривена осим мина.



Slika 7: Primer uspešno završene partije Minolovca

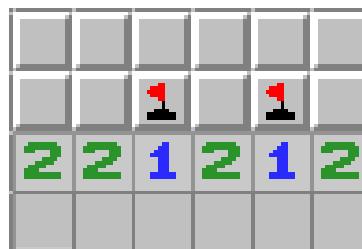
Ток игре:

- Почиње се кликом на било који квадрат на мрежи.

- Бројеви на откривеним квадратима служе као трагови, помажући играчу да одреди локације оближњих мина. Вредности могу бити од 1 до 8, што представља број суседних мина.
- Наставља се са кликтањем на квадрате, систематски откривајући бројеве и избегавајући mine.
- Ако играч кликне на квадрат који садржи мину, игрица је изгубљена и све mine се откривају.
- Десним кликом на квадрат поставља се заставица која указује на то да ту вероватно локација мине. Ово помаже при стратешком планирању.
- Да би победили, морају се исправно означити све локације мине и открили сва сигурна поља.

4.1 Логика и стратегије при игрању Миноловца

- Кључ Миноловца је коришћење бројева за одређивање локације мине.
- Користећи процес елиминације треба отворати сигурне квадрате и стратешки означавати mine.
- Ако квадрат има вредност 1, а само један неоткривени квадрат је у околних 8 квадрата, тај квадрат је вероватно нумерисана ћелија. Супротно томе, ако квадрат има вредност 8, то значи да су у свим околним квадратима mine.
- Да не би имали описане само најједноставније могуће примере, посматрајмо наредни (на Сл. 8 испод):
 - Ако се фокусирате на изоловани квадрат 2 (који је између два квадрата 1), тј. на 8 квадрата око њега – постоје 3 неоткривена поља у његовој околини (изнад њега), а пошто у 5 отворених поља око њега није било mine, то значи да су обе у та 3 неоткривена поља.
 - За одређивање локација мине треба искористити два 1 – ако бисмо рекли да су 2 mine на лева два поља од 3 неоткривена, добили би да су 2 mine и у околини левог 1, што је немогуће.
 - Идентично резоновање имамо и за десно 1; те долазимо до закључка да су 2 mine у околини 2 директно изнад два споменута 1.
 - Након овога, можемо кликнути (отворити) поље директно изнад референтног 2, а и поља лево и десно од два поља за која смо закључили да садрже mine.



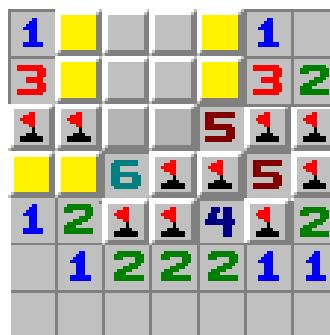
Слика 8: Пример одлучивања у Миноловцу на локалном нивоу

Бодовање:

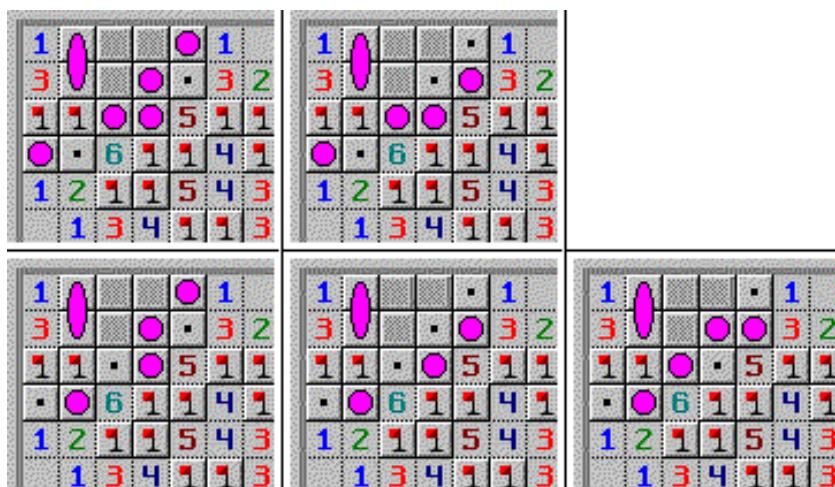
- Играчи се рангирају на основу њиховог времена завршетка.
- Неке верзије игрице такође имају још више нивоа тежине, са већим мрежама и више мина за додатни изазов.

Миноловац је игрица која комбинује логику, дедукцију и пажљиво доношење одлука. Користећи бројеве као трагове, играчи могу стратешки очистити мрежу док избегавају скривене мине. То је класична игрица која деценијама забавља играче и наставља да изазива умове својим замршеним правилима и игром.

У традиционалном Миноловцу није гарантовано да ће све конфигурације игрице бити решиве – тј. често ће се долазити у ситуацију где је немогуће одредити даљи ток игре на основу отворених поља. У просеку, у току сваке партије на табли величине 16x30 са 99 мина; што је стандард, биће око 4 таква случаја, где даљи прогрес зависи од среће. Ради поједностављења, нека је сваки од тих случајева 50/50, тј. постоје само два поља у оптицају, од којих једно садржи мину – добијамо да су шансе да игра буде успешно завршена свега 1 у 16 (и то под претпоставком да играч игра савршено). Сл. 9 приказује ситуацију где се на малом делу табле нашло на чак 3 таква случаја (напомена: посматра се горњи леви угао табле). На Сл. 10 је дочарана постојање вишеструких решења у истом том примеру (иста партија).



Слика 9: Ситуација када се сусрећемо са 50/50 случајевима



Слика 10: Вишеструке могућности за одређивање преосталих мина

4.2 Увек решиви Миноловац

Међутим, варијанта увек решивог Миноловца је дизајнирана на такав начин да се свака игрица може логички решити без прибегавања нагађању. Овај концепт додаје занимљив преокрет класичној игри, осигурувајући да без обзира колико сложена мрежа изгледа, увек постоји пут до победе кроз дедуктивно резоновање. Кључне карактеристике увек решивог Миноловца су:

- Детерминистички распоред – постављање мина није псеудо-случајно, тј. прате се одређена правила како би се обезбедила решивост. Исто тако, и међу нумерисаним пољима нема насумичних елемената. Као директне последице овога, добијамо да нема нагађања, али и да ће први кликтај бити одређен од стране програма, тј. играчу ће бити речено где да кликне иницијално, да би даље шара могла да се развија како треба.
- Дедуктивна логика – игрица се у потпуности ослања на логику и дедукцију. Играчи морају анализирати откривене бројеве на мрежи како би донели информисане одлуке о томе где су лоциране мине.
- Вишеструка решења – иако је свака игрица решива, и даље може постојати више валидних решења. Играчи морају пажљиво да бирају своје потезе да би напредовали ефикасно.

Ова верзија Миноловца додаје додатни слој класичној игри, чинећи је још више ослоњеном на дедуктивно закључивање и логичко размишљање. Гарантујући решивост, играчи могу приступити свакој игри са уверењем да постоји логичан пут до победе, под условом да пажљиво анализирају трагове на табли и примењују исправне стратегије за откривање мина и чишћење сигурних поља.

4.3 Анализа EEG-а током играња Миноловца

Анализирање маждане активности током играња увек решиве верзије игре може дати вредан увид у когнитивне процесе, доношење одлука и решавање проблема. Неки од закључака и сазнања која се могу извући из такве анализе су да:

- Подаци могу открити когнитивно оптерећење које играчи доживљавају приликом доношења одлука у Миноловцу. Прелазак са почетне неизвесности на развој логичких стратегија може се пратити кроз обрасце у неуронској активности.
- Сигнали могу указивати на укљученост мозга у памћење и присећање просторних информација о локацијама мина. Снага и кохерентност мжданих таласа могу показати како играчи кодирају и преузимају податке из мреже.
- Анализа помаже да се идентификују неуронски корелати процене ризика током игрице. Промене у мажданој активности могу открити емоционалне реакције, као што су анксиозност или олакшање, када играчи најђу на мине или безбедна поља.
- Праћење неуронских података у више игрица може пружити увид у криву учења у Миноловцу. Промене у мажданој активности могу одражавати побољшање вештина и развој ефикаснијих стратегија за решавање проблема.

- Варијабилност у одговорима може истаћи индивидуалне разлике у когнитивним приступима Миноловац. Прилагођени neurofeedback или програми когнитивног тренинга могу се развити на основу ових података како би се појединцима помогло да побољшају своје вештине решавања проблема и пажњу.

Предвиђање исхода партије и времена завршетка увек решиве верзије Миноловца, на основу прикупљених EEG сигнала уноси неколико изазова и ограничења. Док EEG може да пружи вредан увид у когнитивне процесе током игрице, предвиђање специфичних исхода за игру са великим прецизношћу је сложен задатак.

Изазови са којим се срећемо би био то да иако одређени когнитивни обрасци могу бити повезани са успешним играњем – један погрешан избор, настало услед нпр. претеране самоуверености, може довести до пораза. На време завршетка утиче комбинација фактора, укључујући сложеност табле за игру, претходан ниво познавања Миноловца и брзина доношења одлука (шта ако неко нпр. има потребу да више пута провери неке потезе). Иако одређени EEG обрасци могу бити у корелацији са когнитивним оптерећењем или брзином доношења одлука, предвиђање тачног времена завршетка је по природи компликовано.

Поред свега тога, индивидуалне разлике у когнитивним стратегијама и нивоима вештина могу да закомпликују напоре предвиђања. Тачно предвиђање исхода би вероватно захтевало комбинацију EEG података, алгоритама специфичних за игру и информација о покретима очију или миша. Тачно предвиђање времена завршетка захтевало би дубоко разумевање когнитивних процеса играча и специфичне карактеристике коришћене Миноловац верзије и табле. Чак и тада, прилагођавања стратегије или оклевања у реалном времену могу да уведу варијабилност коју сами EEG подаци не могу да покажу. Могућности за побољшање:

- Интегрисање EEG података са другим информацијама, као што су обрасци кретања миша, подаци о праћењу очију или додатне статистике из игрице (нпр. тренутни број откривених поља), може побољшати тачност предвиђања.
- ML алгоритми који се могу обучити на комбинацији EEG-а и додатних података како би се временом направила тачнија предвиђања (прикупљање значајног скupa података за обуку је кључно).
- Креирање појединачних профила, на основу играчевих историјских EEG података и перформанси игрице може понудити персонализована предвиђања. Временом би се систем могао прилагодити јединственим когнитивним обрасцима играча.

4.4 Решавање Миноловца помоћу ML

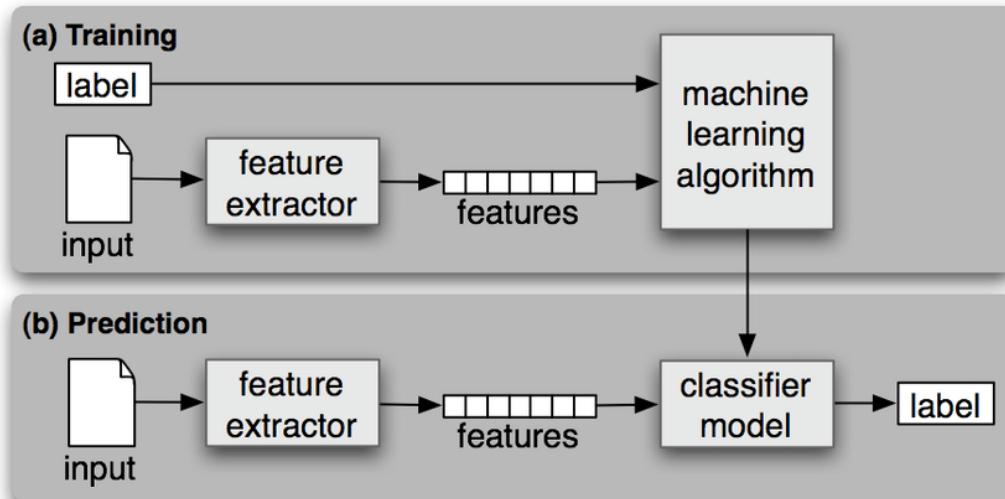
Коришћење ML за предвиђање исхода и времена завршетка Миноловца укључује подучавање рачунара да препозна обрасце у EEG подацима и другим релевантним информацијама за предвиђање [12]. Даље следи поједностављен опис свих корака у том процесу:

- Прикупљање података – EEG, статистика игрице и евентуално друге податке (као што су покрети миша) од испитаника док играју Миноловац.

- Претходна обрада података – филтрирање и процесирање података за анализу, осигурувајући да су у формату погодном за ML.
- Екстракција карактеристика – након идентификације, треба извучити важна обележја или обрасце из података, који би могли да помогну у предвиђању жељених параметара.
- Обука модела – употреба ML алгоритама за креирање предиктивних модела. Модели уче из издвојених обележја, повезујући EEG сигнале, статистику игрице и исходе.
- Тестирање модела – процена тачности модела, тако што ће се тестирати са новим, до тог тренутка невидљивим подацима.
- Поновне итерације – корекција модела и његових карактеристика на основу остварених перформанси. Овај поступак се понавља све док се добије жељена тачност.

ML омогућава рачунарима да праве предвиђања на основу откривених и запамћених образца у подацима. Храњењем модела EEG подацима и информацијама везаним за игру, он може научити да предвиђа да ли ће играч победити или изгубити и проценити колико ће времена бити потребно да се играча заврши. Овај приступ се може побољшати током времена како се више података прикупља и модел учи из њих. Неке од битних напомена везаних за употребу ML су:

- Одабир карактеристика – битно је одабрати најрелевантније информације из прикупљених података. Ово може укључивати и унос додатних података у експеримент, који можда на први поглед нису били битни.
- Одабир модела – доступни су различити алгоритми, сваки са својим предностима и манама. Избор правог алгоритма је кључан, и он зависи од сложености проблема и природе података.
- Подела података (Сл. 11) – да би се прецизно процениле перформансе модела, подаци се обично деле у два скупа: скуп за обуку и скуп за тестирање. Модел учи обрасце из података о обуци и затим се тестира на невидљивим подацима тестирања како би се проценила његова способност генерализације.



Слика 11: Улога ML у предикцији

- Хиперпараметарско подешавање – фино подешавање (енг. фине-туннинг) модела је важно за оптимизацију његових предиктивних могућности. Прилагођавање хиперпараметара, као што је брзина учења или дубина нпр. стабла одлучивања, може значајно утицати на тачност модела.
- Унакрсна валидација – тј. унакрсна провера, као што је к-фолд валидација, може се користити да би се осигурало да перформансе модела буду робусне у различитим подскуповима података. Ово помаже у спречавању прекомерног прилагођавања, где модел постаје превише специфичан за податке о обуци.
- Методе ансамбла – комбинација више модела ML да би се направила робуснија предвиђања. Шуме одлучивања, на пример, су скуп стабала одлучивања.
- Интерпретабилност – тј. тазумевање зашто модел даје одређена предвиђања – то може помоћи да се открије који EEG обрасци или статистика игрице су најутицајнији.
- Континуирано учење – како више података постаје доступно, модел се може стално ажурирати и поновно обучавати.

4.5 Негативни и позитивни аспекти ML у овом случају

Иако ML обећава за предвиђање исхода Миноловца и времена завршетка на основу EEG података и статистике игрице, оно такође долази са неколико ограничења, посебно за овај специфичан проблем. Ту се пре свега мисли на доступност и квалитет података за обуку и тестирање.

Прикупљање довольних и висококвалитетних EEG података, заједно са детаљном статистиком игрице, може бити изазовно. Ограничени подаци могу довести до прекомерног прилагођавања, где се модел добро понаша на подацима о обуци, али се бори да генерализује на нове ситуације. Развој и фине подешавање модела ML за овај специфичан проблем може захтевати велике ресурсе у смислу времена и стручности.

Миноловац укључује сложене когнитивне процесе, укључујући логичко резоновање, памћење и процену ризика. Тешко је прецизно ухватити ове процесе само путем EEG података. Поједностављења направљена од стране модела ML можда не представљају у потпуности сложеност људске спознаје и размишљања. С тим у вези, имамо (и раније спомињану) интериндивидуалну варијабилност – људи имају различите когнитивне стилове и стратегије када играју Миноловац (нешто што функционише за једног играча можда се не односи на другог).

Миноловац је динамична игрица и одлуке играча се могу брзо променити. Модели ML који се ослањају на статичне снимке EEG података могу се борити да одрже корак са природом игрице која се развија. Чак и са савршеним предвиђањима понашања играча, исходи Миноловца и даље могу бити неизвесни због инхерентне случајности игрице.

Једна јако битна ставка је и то што овде имамо ограничени контекст – ML модели се ослањају на информације које су им дате. У контексту Миноловца, они немају приступ важним контекстуалним знацима или намерама играча које би људски играч користио за доношење одлука.

Неки алгоритми ML су сложени и изазовни за тумачење. Разумевање зашто је модел направио одређено предвиђање може бити тешко, што може ограничити његову корисност за извлачење смислених увида. И на крају – са чисто техничког становишта: EEG сигнали су подложни шуму из различитих извора, као што су електричне сметње и артефакти помераја.

Наравно, сада треба навести и неке од позитивних аспеката коришћења ML за предвиђање исхода Миноловца и времена завршетка на основу EEG података и статистике игрице. Уз одговарајуће податке и избор модела, модели ML могу постићи разумну тачност предвиђања. Они тада пружају квантитативне процене исхода и времена завршетка, што може бити драгоценом за истраживање и примене. ML се може користити за креирање персонализованих модела предвиђања, узимајући у обзир индивидуалне разлике у когнитивним стратегијама и стиловима играња. Ово може довести до прецизнијих предвиђања за одређене играче. Пошто је то објективан и квантитативан начин за процену перформанси играча и когнитивних процеса – може бити посебно корисно за поређење појединача или праћење промена у перформансама током времена.

Модели могу лако да се прилагоде новим подацима и промени понашања играча. Како више података постаје доступно, модели се могу континуирано обучавати и побољшавати како би се побољшала тачност предвиђања. Једном обучени, модели могу аутоматизовати процес предвиђања, чинећи га ефикасним и скалабилним. Ово је посебно корисно за анализу великих скупова података или апликација у реалном времену. Чак штавише – модел може аутоматски да идентификује важна обележја унутар података, помажући истраживачима да одреде који аспекти EEG сигнала и статистике игрице су најрелевантнији за предвиђање.

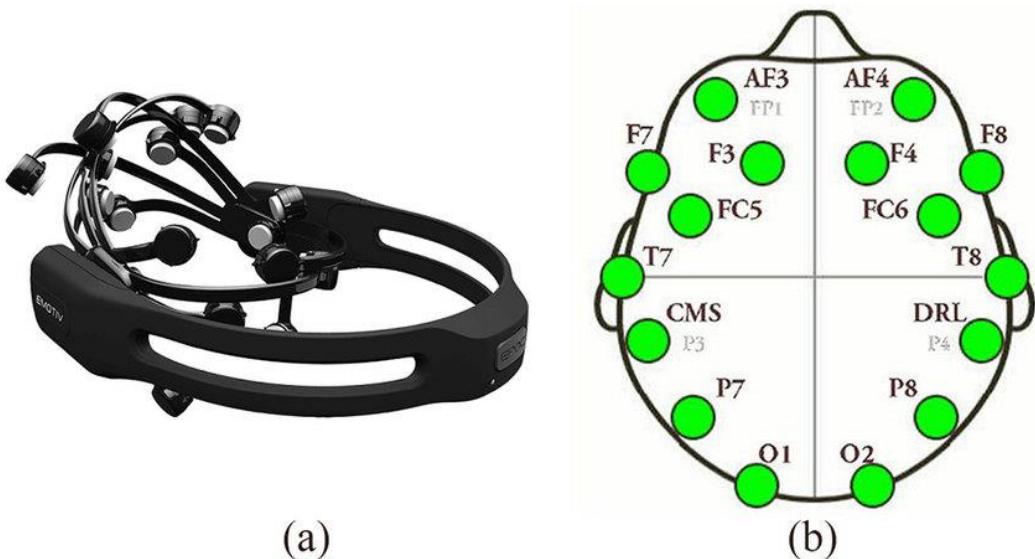
Осим предвиђања, ML може пружити истраживачима дубље разумевање когнитивних процеса укључених у игру Миноловац; може открити како се EEG обрасци односе на доношење одлука, памћење и процену ризика.

5. Emotiv EPOS

Emotiv EPOS је напредан неуротехнолошки уређај који се користи за снимање и анализу мозганих сигнала. Развијен је како би омогућио корисницима да интерагирају с рачунарима и другим уређајима помоћу својих мозганих активности, постоје верзије уређаја и за истраживачку и за потрошачку верзију – она истраживачког нивоа укључује више електрода и напредније функционалности. Овај уређај има низ примена, укључујући у видео игрицама, VR, истраживање когнитивних функција, neurofeedback терапији, итд. Помоћу EPOS уређаја, корисници могу контролисати виртуалне објекте, управљати интерфејсима без додира и притом пратити своју концентрацију и емоционално стање. Emotiv EPOS представља значајан напредак у BCI, отварајући врата за развој иновативних апликација и истраживање у разним областима, као и у побољшавању људских перформанси.

5.0 Спецификације и кључне карактеристике система

- Број електрода – EEG слушалице обично имају 14 или 16 електрода. Ове електроде су стратешки постављене преко главе да би ухватиле моздану активност из различитих региона (Сл. 12).



Слика 12: Emotiv EPOS *хедсет* и распоред електрода

- Сензор – сензорима на бази физиолошког раствора, који обезбеђују добар контакт са скалпом за добијање EEG сигнала. Слушалице су дизајниране тако да се могу подесити да удобно одговарају различитим величинама главе. Ово обезбеђује сигурну и поуздану везу између електрода и коже главе.
- Бежично повезивање – омогућава да се повеже са рачунаром или мобилним уређајем преко bluetooth-a. Ова бежична повезаност побољшава удобност и мобилност корисника током прикупљања EEG података.
- Фреквенција одабирања – променљива; обично подешена на 128 Hz.

- Резолуција – односи се на аналогно-дигиталну конверзију, и износи 14-бита; стандард је минимум 11 бита, а пожељно је да буде 12 или више.
- Жироскоп и акцелерометар – за снимање покрета и оријентације главе, респективно. Ови подаци се могу интегрисати у апликације које укључују виртуелну реалност, проширену стварност или препознавање покрета главом.
- Батерије – напајање се врши помоћу пуњиве литијум-полимерне батерије, која обично обезбеђује неколико сати непрекидног коришћења са једним пуњењем. Трајање батерије може да варира у зависности од употребе и верзије уређаја. То га чини погодним за теренске студије и експерименте који се спроводе ван традиционалног лабораторијског окружења.
- Компатибилност – Emotiv обезбеђује комплете за развој софтвера и Application Programming Interface-а за компатибилност са различитим платформама, укључујући Windows, macOS и Android. Ово омогућава програмерима да интегришу уређај у своје апликације и истраживачке пројекте.
- Софтвер – EmotivPRO (више о њему испод) софтвер се користи за визуелизацију података, обраду сигнала у реалном времену и развој апликација.
- Излазни подаци – чувају се у стандардним форматима као што су Comma Separated Values (CSV), или JavaScript Object Notation (JSON), што га чини компатибилним са широким спектром алата за анализу података и ML.
- Омогућава праћење EEG сигнала у реалном времену, што га чини погодним за различите апликације где је потребна тренутна повратна информација о мозданој активности.
- Користи се у широком спектру апликација, укључујући неуронаучна истраживања, когнитивни тренинг, интеракцију човека и рачунара, па и код играња игрица.
- Слушалице су дизајниране имајући на уму удобност корисника, са подесивим кашевима и лаганом конструкцијом како би се обезбедило сигурно и удобно пристајање током дуже употребе (Сл. 13).



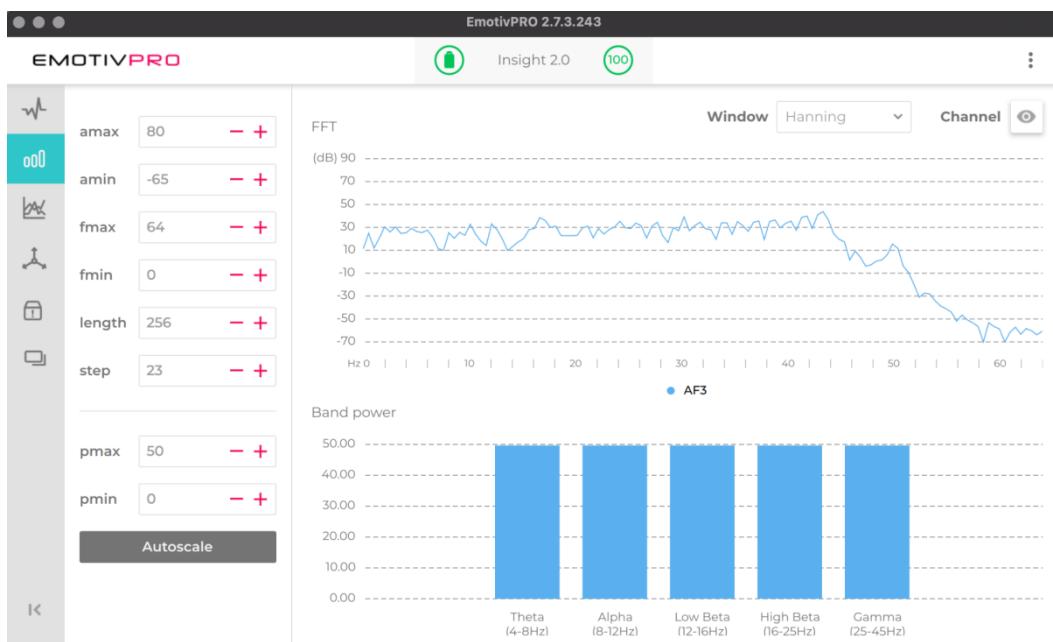
Слика 13: Прилагодљивост хеадсет-а; пример са малим дечаком

- Emotiv има онлине заједницу корисника који деле увиде и примене уређаја, као и документацију за све своје производе, што га чини вредним ресурсом за све особе који би да раде са њиховим уређајима.

5.1 EmotivPRO

EmotivPRO је софтверски пакет дизајниран да допуни Emotiv EEG слушалице – служи као свеобухватна платформа за прикупљање, анализу и истраживачке апликације EEG података. Компабилан је са различитим моделима слушалица, омогућавајући корисницима да снимају EEG сигнале у реалном времену или снимају сесије за каснију анализу. Софтверски пакет укључује алате за основну обраду сигнала, као што су филтрирање, уклањање артефаката и смањење шума. Ови кораци предобраде помажу у побољшању квалитета EEG података, чинећи их погодним за напредну анализу. Обезбеђује и алате за визуелизацију, путем којих корисници могу да посматрају EEG таласне облике, спектрограме и друге визуелне приказе мождане активности (Сл. 14).

За истраживаче и аналитичаре података, EmotivPRO нуди могућности за екстракцију обележја из EEG података. Ово је кључно за добијање смислених увида из можданых сигнала и може укључивати мере као што су спектрална снага у одређеним фреквентним опсезима или потенцијали повезани са догађајима (ERP). Компабилан је и са програмским језицима као што су MATLAB и Python, омогућавајући употребу ширег спектра алата за анализу и моделирање.



Слика 14: Изглед EmotivPRO интерфејса

У зависности од верзије и лиценце, EmotivPRO може укључити функције које обезбеђују безбедност података и усклађеност са прописима о приватности, што је посебно важно у здравственим и истраживачким окружењима.

5.2 JSON

JSON је текстуални формат за размену података који је лако читљив и разумљив како људима тако и рачунарима. JSON је постао популаран формат за пренос

података између сервера и клијената, као и за складиштење података. Подаци у њему се састоје од парова кључ-вредност, где је кључ увек низ карактера (илити стринг), а вредност може бити стринг, нумеричка вредност, логичка вредност, листа са елементима, објекат, или null.

- Стрингови су низови карактера између двоструких наводника. Стрингови могу садржавати било који знак, укључујући и есапе секвенце за посебне карактере (нпр. \ за двоструке наводнике унутар самог стринга).
- Бројеви су нумеричке вредности; како целе (интедџер) тако и децималне (флоат), уз напомену да нема посебних ознака за различите типове бројева – нпр. 34 или 1.618.
- JSON подржава логичке вредности труе и фалсе за означавање истинитости или неистинитости.
- Листе су уређени 1Д низови вредности које се налазе између угластих заграда [] и раздвајају се зарезима, нпр:
 - [црвена, зелена, плава]
- JSON објекти су ограничени витичастим заградама {} и састоје се од низа парова кључ-вредност. Кључ и вредност се раздвајају двотачком, а парови се раздвајају зарезима, нпр:
 - {име: Петар, презиме: Петровић, године: 30}
- Null вредност – користи се кључна реч null за представљање недостатка вредности или непостојећих вредности.
- Пример JSON објекта који садржи низ објеката:
 - {студенти: [{име: Петар, презиме: Петровић, године: 30}, {име: Ивана, презиме: Ивановић, године: 30}]}

JSON се често користи у web развоју за слање података између сервера и клијената у AJAX (asynchronous JavaScript and XML; XML – extensible markup language) захтевима, за конфигурацију апликација, као и за складиштење структуираних података у датотекама или базама података. JSON је популаран због своје једноставности и интероперабилности између различитих програмских језика и платформи.

6. Процесирање сигнала

Сваки сигнал, након што је филтриран у аналогном и дигиталном домену, треба додатно процесирати, не би ли га довели у нама погодан облик за анализу. То укључује разне технике које имају за циљ да одстране све оно што нам није потребно, а да притом и нагласе све оно што је битно.

6.0 Машино учење

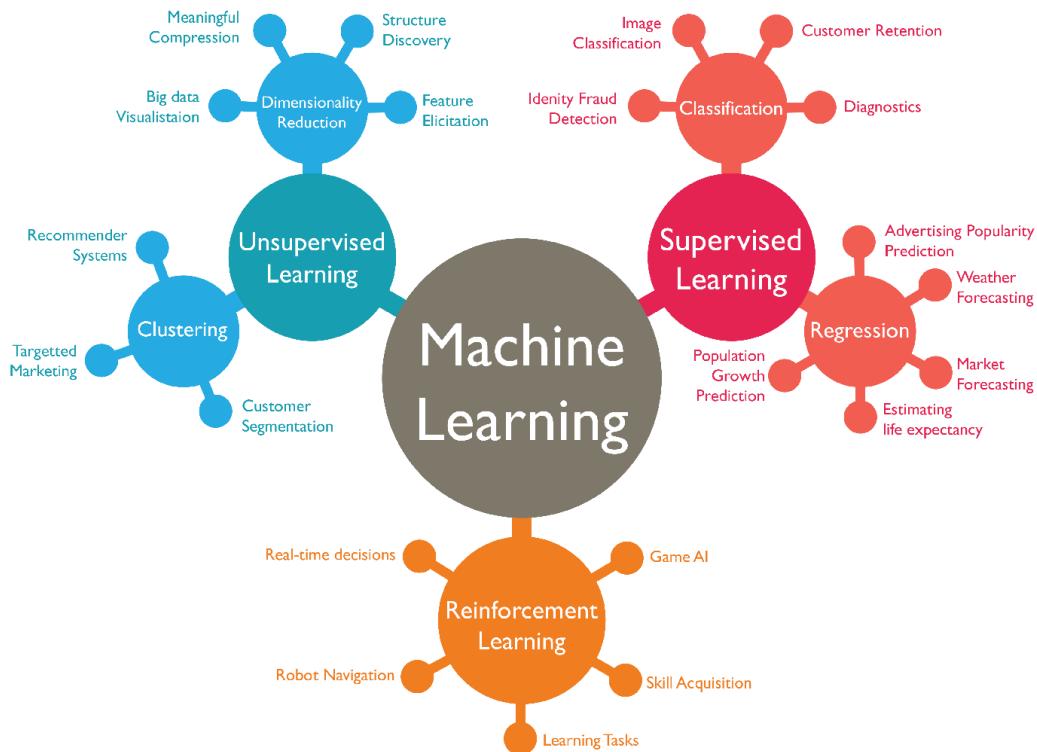
ML је једна од грана AI која се бави развојем техника и алгоритама који омогућавају рачунарима да науче и извлаче закључке из података без експлицитног програмирања. Овај концепт је кључан за анализу података и доношење информисаних одлука у многим областима, укључујући медицину, економију, маркетинг, индустрију, и многе друге.

На самом почетку, алгоритам за ML се тренира на основу скупа података. Овај скуп података може садржавати различите врсте информација, као што су текст, слике, бројеви или било која друга врста података. Током тренинга, алгоритам учи да препознаје обрасце и везе између података.

Примарни циљ ML је да се алгоритму омогући да генерализује своје знање и примени га на нове, непознате податке. Ово се постиже тако што алгоритам проналази скривене обрасце у тренинг подацима и користи их за доношење предвиђања или класификацију нових података.

Постоје различите врсте ML (Сл. 15), укључујући:

- Надгледано учење (енг. supervised learning) – овде се алгоритам тренира на паровима улаза и излаза – нпр. можемо тренирати модел да препозна слике мачака и паса на основу тренинг скупа слика које су већ означене као мачке или пси.
- Ненадгледано учење (енг. unsupervised learning) – овде алгоритам учи без тачних одговора – нпр. груписање (кластеровање) података може бити примењено како би се идентификовали природни узорци или сегменти у подацима.
- Појачано учење (енг. reinforcement learning) – овај тип учења се користи за развој система који доносе одлуке у окружењу како би максимизовали награде. Примери укључују управљање аутономним возилима или играње видео игрица.



Слика 15: Блок дијаграм врста ML-а

Примена ML у анализи података:

- Предвиђање будућих догађаја или вредности на основу постојећих података; нпр. предвиђање цена акција на берзи, временске прогнозе или вероватноће оболевања од одређене болести.
- Класификација података у различите категорије или класе; нпр. аутоматско препознавање спам е-поште или класификација пацијената према ризику од одређене болести.
- Сегментација сличних група или сегмената у великим скуповима података, што је корисно за персонализацију услуга или маркетиншке стратегије.
- Откривање аномалија, тј. идентификација необичних или сумњивих података који могу указивати на проблеме, као што су преваре у банковним трансакцијама или откривање грешака у производном процесу.

Свака примена ML за анализу података захтева одговарајуће припремање и обраду података. Ово укључује чишћење података од недостајућих вредности, скалирање и нормализацију података ради обезбеђивања конзистентних резултата, као и избор одговарајућих обележја које ће алгоритам користити за учење и предвиђање.

Један од изазова у ML је overfitting, што се догађа када модел превише учи тренинг податке и не успева добро да генерализује на нове податке. Да би се превазишао овај проблем, често се користи техникама као што су регуларизација и валидација модела. Такође, важно је напоменути да избор одговарајућег алгоритма за ML зависи од конкретног задатка и типа података са којима радите. Постоји много различитих алгоритама, укључујући алгоритме за дубоко учење (енг. deep learning) који су посебно моћни за анализу сложених података као што су слике, звукови или природни језик.

ML игра кључну улогу у анализи комплексних научних података, као што су геномски подаци, астрофизички подаци или анализа климатских модела. ML је моћан алат за анализу података који омогућава рачунарима да открију обрасце и везе које би били тешки или немогући за људски ум да идентификује. Његова примена је широка и обухвата многе аспекте нашег свакодневног живота и пословања, пружајући нам боље разумевање и информације за доношење одлука. Следе занимљиви примери примене ML:

- Го шампион против рачунара [13] – 2016. програм AlphaGo, развијен од стране DeepMind (део компаније Google), победио је светског шампиона у игри Go, једне од најкомплекснијих друштвених игрица. Ово је била значајна победа за AI, јер је Go захтевао дубоко разумевање стратегије и интуиције, нешто што је раније сматрано тешким за рачунаре.
- Биомедицина – алгоритми за дееп леарнинг су показали импресивне резултате у анализи медицинских слика, као што су рендгенски снимци и магнетна резонанца, што може помоћи у раном откривању болести.
- Ауто-индустрија – развој аутономних возила зависи у великој мери од ML. Аutomobili користе сензоре и камере како би прикупили податке из околине и донели одлуке у реалном времену. ML омогућава тим возилима да, између осталог, препознају саобраћајне знакове, пешаке и друге возила.
- Edge AI, или ML на уређајима, постаје све популарније. То значи да паметни уређаји, попут модерних телефона и различних кућних уређаја, могу обављати анализу података локално, без потребе за сталном везом са интернетом. Ово омогућава бржу и поуздану обраду података.
- Генеративни модели, који омогућавају рачунарима да стварају реалистичне слике, звуке и текстове. Ово се користи у креирању уметничких дела, синтези говора и још много чему.
- У спорту се може користити за побољшање перформанси и доношење одлука – тимови користе технологију за праћење играча и анализу њихових перформанси, као и за оптимизацију стратегије и тактике.
- Chat ботови и виртуални асистенти – пре свега ChatGPT, али и други chat ботови су свеприсутни на интернету и користе моделе ML за обраду природног језика и одговарање на корисничке упите. Виртуални асистенти, попут Siri, Cortana и Google Assistant, такође користе напредне технике.

Ове занимљивости илуструју ширину и дубину примене ML у свакодневном животу и различитим индустријама. Ова технологија наставља да се развија и трансформише наше друштво на различите начине.

6.1 Прозорирање

Прозорирање EEG сигнала је уобичајена техника која се користи у обради сигнала – оно подразумева дељење EEG сигнала на мање сегменте (илити прозоре) који се могу, а и не морају преклапати.

Тиме добијамо темпоралну анализу, пошто EEG сигнали могу да обухватају дуже периоде. Да би се извршила смислена анализа, често је потребно фокусирати се на краће временске интервале. Прозорирање нам омогућава да раздвоје сигнал на

сегменте којима се може управљати, од којих сваки представља одређени временски оквир. С тим у вези, у EEG студијама, често је фокус на специфичним догађајима или стимулусима – прозор омогућава да се изолују и анализирају EEG активност око тих догађаја, пружајући увид у потенцијале везане за догађаје (ERP).

EEG сигнали могу да испоље нестационарност, што значи да се њихове карактеристике мењају током времена. Прозорирање претпоставља стационарност унутар сваког сегмента, што олакшава примену метода статистичке и спектралне анализе које претпостављају стационарне сигнале.

За ML задатке и уопштено за препознавање образца [14] – прозори су од суштинског значаја за екстракцију карактеристика. Поделом сигнала на прозоре, релевантне карактеристике могу се израчунати из сваког сегмента, или упоређивањем сегмената, па се затим могу користити као улаз за моделе предвиђања.

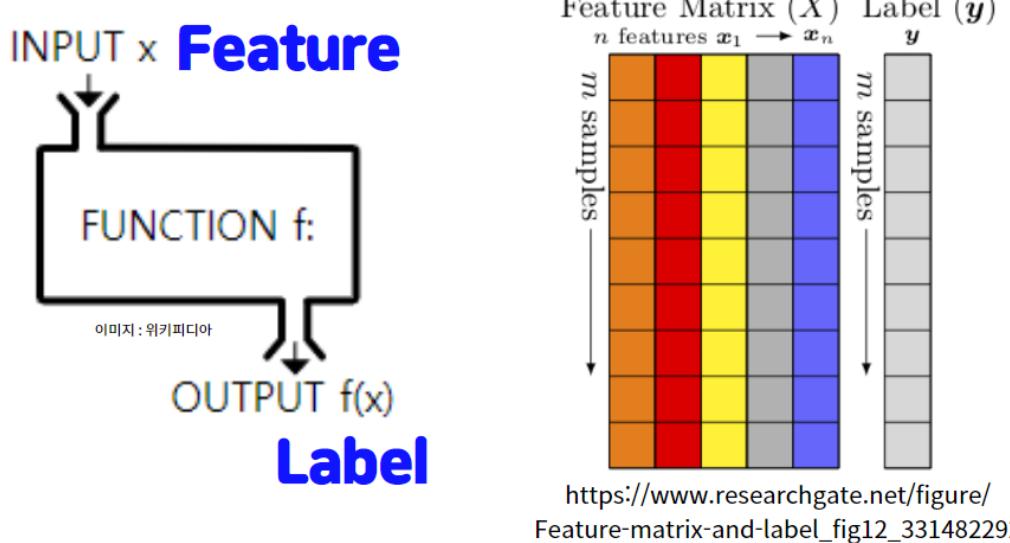
6.2 Матрица обележја и вектор класа

Матрица обележја и вектор класа играју битну улогу у изградњи и обуци ML модела. Матрица карактеристика је попут табеле у којој сваки ред представља другачији узорак, а свака колона представља обележје (део информације) који се односи на тај узорак.

У општем случају, сваки узорак одговара сегменту EEG података (добијеног прозорирањем), а обележја могу бити различите статистичке игре или нешто што је изведено из самих EEG података (нпр. средња вредност).

Матрица обележја помаже у организовању и структуирању података за ML (Сл. 16) – она нам даје начин да EEG информације представимо у формату који ML модел може да разуме. Конвертовањем EEG сигнала у матрицу обележја, креиратмо нумеричку репрезентацију која се може користити за предиктивно моделирање.

Вектори класа су у суштини ознаке које означавају категорије или класе којима сваки узорак у матрици обележја припада. У нашем случају, они могу представљати да ли је играч победио или изгубио игру или колико му је времена било потребно да заврши игру. Користимо их да научимо модел шта желимо да предвиди. Они дају „одговоре“ или циљне вредности за модел из којих може да учи. Када обучава модел, он упоређује своја предвиђања са векторима класе да би прилагодио своје унутрашње параметре и направио боља предвиђања.



Слика 16: Сликовито појашњење матрице обележја и вектора класа

Ако предвиђамо исходе Миноловца, вектор класе може да садржи две категорије: „победа“ и „пораз“, при чему сваки унос у вектору одговара томе да ли одговарајући сегмент EEG података представља победничку или изгубљену партију. За предвиђање времена завршетка, вектор класе може да садржи стварна времена завршетка (у секундама).

Анализа EEG сигнала често укључује издвајање карактеристика из необрађених EEG података и коришћење ових карактеристика за класификацију или предвиђање одређених исхода. На пример, једна студија је представила оквир за класификацију нивоа стручности играча помоћу носивих слушалица за EEG [15]. Студија је поставила хипотезу да је моздана активност стручњака и играча почетника различита, што се може класификовати коришћењем карактеристика фреквенцијског домена екстравалорних из EEG сигнала играча у игри. Приступ систематској редукцији канала представљен је коришћењем методе процене атрибута заснованог на корелацији. Овај приступ је довео до идентификације два значајна EEG канала, тачније: AF3 и P7, међу 14 канала доступних у Emotiv EPOC слушалицама. Карактеристике извучене из ова два EEG канала највише су допринеле класификацији нивоа стручности играча.

6.3 Корелација и крос-варијанса

Корелација је статистичка мера која се користи како би се утврдила веза између два или више скупа података. Ова мера квантификује степен линеарног односа између варијабли, који може бити позитиван, негативан или нулти, и изражава се као вредност између -1 и 1.

- Позитивна корелација – ако вредности две варијабле расту заједно; вредност близка 1 указује на снажну позитивну корелацију, док вредност близка 0 указује на слабу позитивну корелацију.
- Негативна корелација – ако вредности једне варијабле расту док вредности друге опадају; вредност близка -1 указује на снажну негативну корелацију, док вредност близка 0 указује на слабу негативну корелацију.

- Нулта корелација – ако нема јасне везе између варијабли, корелација ће бити близка нули.

Важно је напоменути да корелација не имплицира узрочну везу; само показује да постоје неки обрасци у подацима.

Корелација квантификује линеарни однос између два прозора, тј. помаже у процени колико су два прозора EEG сигнала слична или различита у смислу њихових образца и варијација. Висока позитивна корелација указује на синхронизацију, што значи да EEG сигнали у два прозора имају тенденцију да се мењају на координисан начин. Ово може бити значајно за идентификацију межданих активности које укључују координиране промене сигнала у различитим регионима. Насупрот томе, ниска или негативна корелација сугерише асинхронију, где EEG сигнали у два прозора показују различите обрасце или чак супротне варијације.

Из овога видимо да се EEG корелација се може користити за проучавање функционалне повезаности у мозгу. Помаже да се идентификује који региони мозга раде заједно или независно током одређених задатака или стања.

Крос-варијанса (енг. cross-covariance) је мера која квантификује заједничку варијабилност између две случајне варијабле. Она мери како се две варијабле заједно мењају. Формула за крос-варијансу изгледа овако:

$$Cov(X, Y) = \sum_{i=0}^n \frac{(X_i - \mu_X) \times (Y_i - \mu_Y)}{n - 1}$$

Где:

- $Cov(X, Y)$ представља крос-варијансу између варијабли X и Y.
- X_i и Y_i су појединачне вредности из склопа података X и Y.
- μ_X и μ_Y су средње вредности за склопове података X и Y.
- n представља број опажања у склоповима података.

Крос-варијанса може бити позитивна, негативна или нула, слично корелацији. Позитивна крос-варијанса значи да кад једна варијабла расте, друга такође обично расте – негативна значи да кад једна варијабла расте, друга обично опада – а нулта да две варијабле нису повезане.

Крос-варијанса не даје јасну информацију о јачини везе између варијабли као што то чини корелација, јер крос-варијанса зависи од размера и јединица мере варијабли. Када се жели квантifikовати веза између варијабли, корелација је преферирана мера.

Унакрсна коваријанса мери сличност између два склопа података истовремено узимајући у обзир њихове средње вредности. Често се користи да се процени како су два прозора EEG сигнала повезана у смислу њихових временских образца. Она пружа информације о било каквом временском кашњењу или кашњењу између два прозора EEG сигнала. У функционалној анализи, унакрсна коваријанса се може користити за проучавање интеракција између различитих EEG канала или региона мозга – где помаже да се идентификује који канали или региони утичу једни на друге и то којим временским редоследом.

Значај корелације и унакрсне коваријансе између прозора у анализи EEG података лежи у њиховој способности да ухвате односе и зависности између различитих сегмената или канала EEG сигнала.

6.4 Класификација и регресија

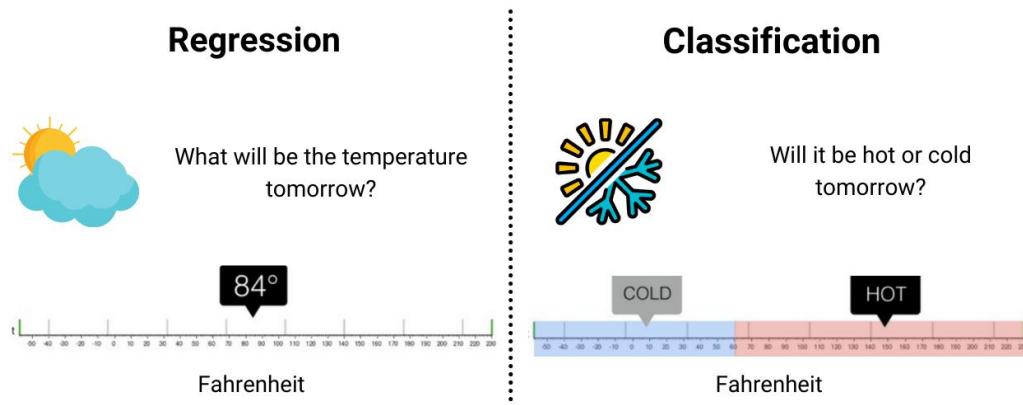
Класификација и регресија су две основне технике које се користе за предвиђање и евалуацију резултата. Класификација се користи када задатак предвиђања укључује додељивање узорака унапред дефинисаним категоријама или класама, те је погодна за проблеме код којих је исход дискретан или категоричан. У контексту предвиђања исхода Миноловца, може се користити за класификовање сваког појединачног EEG сигнала у категорије као што су „победа или „пораз. Класификациони алгоритми уче обрасце из матрице обележја и вектора класа да би извршили ове задатке. Уобичајени алгоритми класификације укључују: стабла одлучивања, машина са векторима подршке и неуронске мреже.

Регресија се користи када задатак предвиђања укључује процену континуиране вредности или количине; она је погодна за проблеме где је исход реалан број, као што су време, растојање или температура. У нашем случају регресија се користи за предвиђање времена (у секундама) које је потребно играчу да заврши игру. Алгоритми регресије уче односе између карактеристика и континуираних циљних вредности. Уобичајени алгоритми регресије укључују линеарну регресију, регресију стабла одлучивања и опет неуронске мреже.

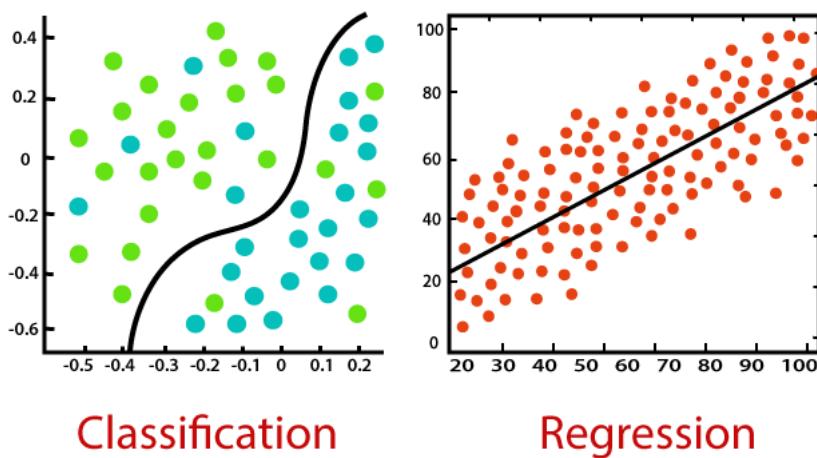
За евалуацију перформанси модела класификације користе се метрике као што су тачност, прецизност, опозив и F1 резултат. Тачност мери укупну исправност предвиђања, док се прецизност и памћење фокусирају на способност модела да исправно идентификује специфичне класе (нпр. исправно идентификује победничке игрице).

Регресиони модели се процењују коришћењем метрика као што су средња апсолутна грешка и средња средња квадратна грешка. Ове метрике квантификују колико су предвиђања модела близка стварним временима завршетка – ниже вредности указују на бољу тачност предвиђања.

Класификација и регресија су две врсте техника надгледаног учења које се могу користити за предвиђање исхода игрице и времена завршетка. У класификацији, циљ је да се предвиди категоричан исход, као што је да ли ће играч победити или изгубити утакмицу. У регресији, циљ је да се предвиди континуирани исход, као што је време које је потребно играчу да заврши игру (разлика је илустрована на Сл. 17 и 18).



Слика 17: Разумевање разлике између класификације и регресије – преко формата излаза



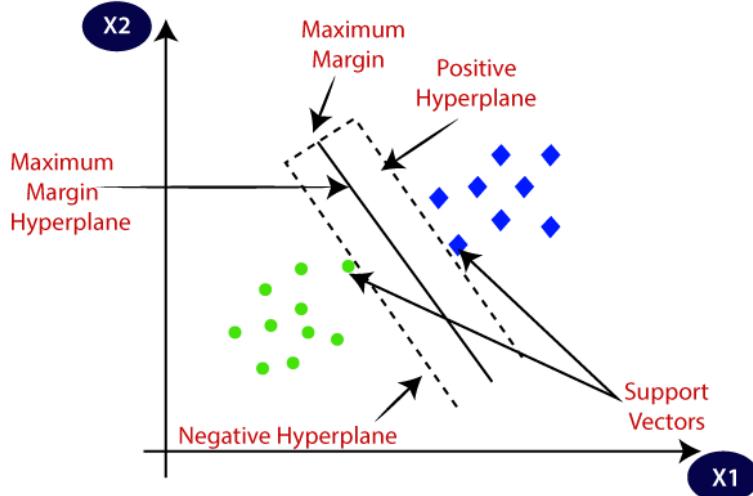
Слика 18: Разумевање разлике између класификације и регресије – преко груписања узорака

Један занимљив пример коришћења класификације и регресије за предвиђање исхода игрице и времена завршетка је из области е-спорта [16]. Истраживачки рад се фокусирао на изградњу предиктивних машина и модела дубоког учења како би се идентификовала исход Dota 2 MOBA (енг. Multiplayer Online Battle Arena) игрице користећи нову методу предвиђања вишеструких корака унапред. Истражена су и упоређена три модела: линеарна регресија, неуронске мреже и тип рекурентне неуронске мреже базиран на long short term memory.

6.5 Класификатор вектора подршке

SVM (Support Vector Machine) је моћан ML алгоритам који се користи за задатке класификације и регресије. Ради тако што пронађе хиперравнину која најбоље раздваја тачке података у различите класе или предвиђа нумеричке вредности. Оно што SVM чини јединственим је његов фокус на максимизирању маргине између најближих тачака података из различитих класа, познатих као вектори подршке. Ова максимизација маргине не само да побољшава генерализацију модела, већ и омогућава SVM-овима да рукују нелинеарним подацима кроз функције кернела, ефикасно мапирајући податке у просторе виших димензија ради раздвајања – те се широко користе у апликацијама као што су класификација слика, категоризација текста и препознавање образца.

SVC (Support Vector Classifier) је SVM који се користи за класификацију (Сл. 19) – ради на следећи начин: алгоритам тражи оптималну хиперраван која максимизује размак између две класе (позитивна и негативна класа). Ова хиперраван се назива Support Vector Hyperplane, а подржавачки вектори су инстанце из скупа за обуку које су јој најближе.



Слика 19: Support Vector Classifier

Предности:

- Ефикасан је за рад са подацима и великим бројем атрибута.
- Може се користити за класификацију на различитим подацима и подржава различите функције језика (kernel функције) за рад са нелинеарним подацима.
- Отпоран је на overfitting због своје природе у максимизовању размака између класа.

Мане:

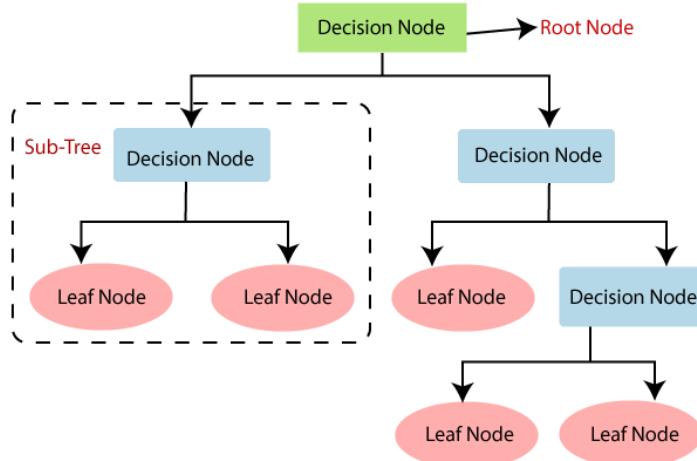
- Потребно је пажљиво изабрати параметре као што су C (који контролише меку или тврду маргину) и тип kernel функције.
- За велике скупове података и/или велики број атрибута, обучавање SVM може бити временски захтевно.
- Модели који се заснивају на SVM-у су често тешки за интерпретацију јер не пружају интуитиван увид у то како се донесе одлука.

У случају предикције исхода Миноловац партије, SVM класификатор може бити корисан ако постоје подаци о различитим аспектима игре (на пример, број потеза, величина игре, расположење мина и слично) и желите да предвидите исход партије (победа/пораз). SVC би могао да обучи модел да раздвоји податке на основу ових атрибута и предвиди исход партије.

6.6 Класификатор стабла одлучивања

DTC (Decision Tree Classifier) се користи за раздвајање података у различите класе на основу низа атрибута – DT је графички модел који се састоји од чворова и грана и приказује логичке одлуке за раздвајање података (Сл. 20).

Процес одлучивања у DT-у почиње са коренским чврором који представља најбитнију атрибут према којем се подаци деле. Сваки чврор представља тест на одређени атрибут, а гране од чврора ка чврору представљају могуће вредности тог атрибута. Процес се наставља кроз дрво, при чему се на крају долази до листова који представљају класе или излазне вредности.



Слика 20: Decision Tree Classifier

Предности:

- Лако интерпретиран и разуме се као ако-онда правило.
- Може се користити са категоричким и нумеричким подацима без великог напора.
- Може ефикасно радити са несиметричним и неизбалансираним подацима.

Мане:

- Overfitting – DTC може лако прелетети на тренинг подацима ако се не примене одговарајуће методе прекидања и управљања прелетањем.
- Мале промене у подацима могу довести до значајних промена у структури дрвета, што га чини нестабилним.

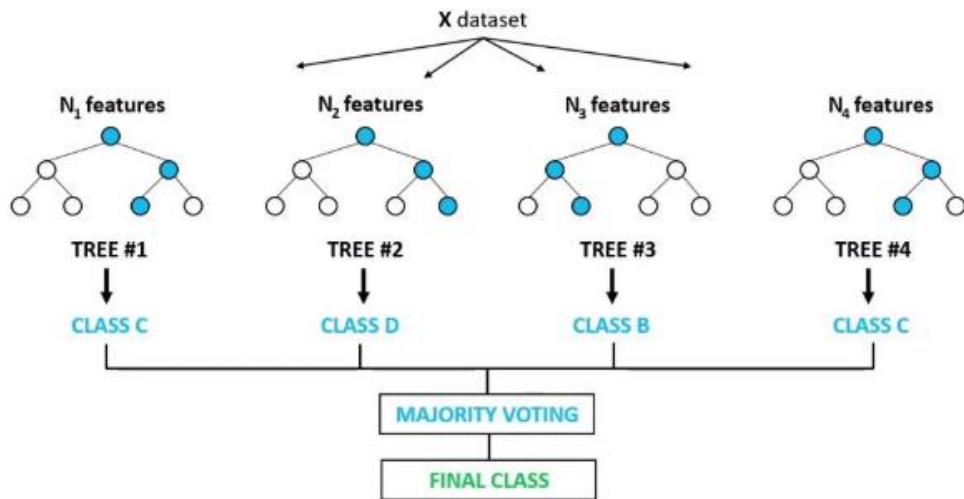
У случају предикције исхода Миноловац партије, DTC може бити корисан ако имате податке о различитим аспектима игре (на пример, број потеза, величина игре, расположење мина и слично) и желите да предвидите исход партије (победа/пораз). Можете тренирати дрво одлуке на овим подацима и користити га за класификацију партија. Међутим, треба бити опрезан у вези са прелетањем и несиметричним подацима, и можда ћете морати да примените технике као што су обрезивање (pruning) и уређивање података како бисте добили најбоље резултате.

6.7 Класификатор случајне шуме

Random Forest Classifier (RFC) представља ансамбл алгоритам у машинском учењу који се користи за класификацију и регресију. Овај алгоритам се заснива на идеји да се умноже DTC-ови – за боље предвиђање исхода.

Процес рада RFC-а почиње са генерирањем случајног скупа података из тренинг сета (подскуп података). Затим се над овим случајним скупом података изгражује неколико дрвета одлуке (Сл. 21). Свако од ових дрвета ради налично и предвиђа

класу за дати улазни примерак. Класификација се врши на основу гласања (вишељког множинског гласања), где сваки DTC има један глас за класу коју предвиђа – класа са највише гласова је финални исход класификације.



Слика 21: Random Forest Classifier

Предности:

- Метода је робустна на аутлајере и шум у подацима.
- Често постиже добре резултате без потребе за дубоким тј. комплексним моделима.
- Отпорност на прелетање (overfitting) – специфичност RFC-а се ограничава случајним избором подскупа података и атрибута, што помаже у смањењу прелетања.

Мане:

- Изградња великог броја DTC-ова може бити временски и рачунски захтевна операција.
- Консолидовање резултата из великог броја стабала може бити компликовано, што чини интерпретацију модела тежом.

У контексту предикције исхода Миноловац партије, главна предност ове методе у је њена способност да обрађује велики број фактора и да избегава прелетање. Требало би да се обрати пажња на хиперпараметре и важност атрибута како би се постигли најбољи резултати при коришћењу класификатора.

6.8 Упоређивање класификатора

Упоређивање SVC, DTC и RFC модела на малим скуповима података има своје специфичне аспекте и предности за сваки алгоритам:

- SVC – може да ради добро и на малим скуповима података, посебно када подаци нису линеарно раздвојиви. Има способност да максимизује размак између класа, што може бити битно за боље генерализовање. Због потребе за оптимизацијом и рачунском сложеношћу, може захтевати више времена за обучавање на малим скуповима података у односу на друге алгоритме.

Избор параметара као што је C и тип kernel функције може бити изазован, а на малим скуповима података лакше може долазити до оверфитовања.

- DTC – може ефикасно радити на малим скуповима података и обично не захтева дугу обучавању. Интерпретација резултата је једноставна и лако разумљива. Могуће је долазити до прелетања (overfitting) ако се дрво одлуке не обреже (prune) или ако је модел сувише дубок, што може бити ризик на малим скуповима података. Није најбољи избор ако подаци нису линеарни или ако постоји сложена зависност између атрибута.
- RFC – робустан модел – и добро ради на малим скуповима података, а такође спречава прелетање због ансамбл методологије. Има добро опште перформансе и способан је за рад са различитим типовима података. Обучавање великог броја дрвета одлуке може бити временски захтевно, али ово може бити амортизирано када радите на великим скуповима података.

Уопштено гледано, када се ради са малим скупом података, одабир алгоритма зависиће од природе података, претпоставки о моделирању и компутационих ресурса. SVC може бити прикладан ако су подаци сложени и нису линеарно раздвојиви. DTC је добар избор ако се жели једноставна интерпретација резултата. RFC може бити добар компромис између робустности и перформанси на малим скуповима података. У сваком случају, важно је практично тестирање свих алгоритама на конкретним подацима да би се одабрали најбољу опцију.

6.9 Theil-Sen регресор

Theil-Sen регресија (TSR), такође позната као Sen's slope estimator, представља метод регресије који се користи за анализу везе између две променљиве. Овај метод је непараметрички, што значи да не захтева претпоставке о расподели података и омогућава моделирање линеарних веза између променљивих без претпоставки о структури модела.

Основна идеја овог метода је заснована на израчунавању медијане склона (slope) између свих парова података. Први корак је израчунати склоне (нагибе) између свих парова података и затим изабрати медијану од тих склонова. Ова медијана склона представља средњу тачку нашег регресионог модела. То значи да се са TSR моделира средња вредност склонова између свих податковних парова.

Предности:

- Метода је робустна према присуству аномалија и аутлајера у подацима, што га чини погодним за анализу реалних података који често садрже шум или аномалије.
- Не захтева претпоставке о расподели података или о структури модела, што омогућава флексибилно моделирање везе између променљивих.
- Рачунски је ефикасан и лако се примењује.

Мане:

- Ова метода може губити информације о подацима, посебно ако постоји велика варијабилност у склону између података.

- У случају великог броја података, рачунски захтеви могу бити велики, што може утицати на брзину обраде.

Што се тиче употребе TSR-а у случају предикције времена траја Миноловац партије, ова метода би могла да буде корисна ако постоји потреба да се анализира веза између различитих фактора (као што су број потеза, величина игре, итд.) и времена потребног за решавање партије. Коришћењем TSR-а, може се анализирати ова веза без потребе за претпоставкама о облику функције. Могли би се истражити фактори који највише утичу на време решавања и израдити прогнозе на основу анализе варијабилности података. Међутим, треба имати на уму и мање овог метода, као и рачунске изазове ако постоји велики број података.

6.10 Разлике између Линеарног и Theil-Sen регресора

Линеарна регресија (енг. Linear regression; LR) је један од основних ML алгоритама који се користи за предвиђање нумеричких исхода. Моделира однос између зависне променљиве (оне коју желимо да предвидимо) и једне или више независних променљивих (карактеристика) уклапањем праве линије кроз тачке узорака. Циљ је пронаћи линију која минимизира збир квадрата разлика између предвиђених и стварних вредности. LR се широко користи у различitim областима, као што су економија, финансије и наука, за задатке као што су предвиђање цена, анализа трендова и процена утицаја, због своје једноставности и интерпретабилности.

Разлика између TSR и LR се односи на начин на који ови методи обрађују и моделирају везу између променљивих у статистичкој анализи и машинском учењу (Сл. 22):

Тип регресије:

- LR је метод који моделира линеарну везу између зависне и независних променљивих. Ова метода претпоставља да је веза између променљивих линеарна, што значи да се мењање независних променљивих узрокује промену зависне променљиве уз константан коефицијент.
- TSR је непараметрички метод који не захтева претпоставку о линеарности везе између променљивих. Она моделира везу између променљивих користећи медијану склона између парова података.

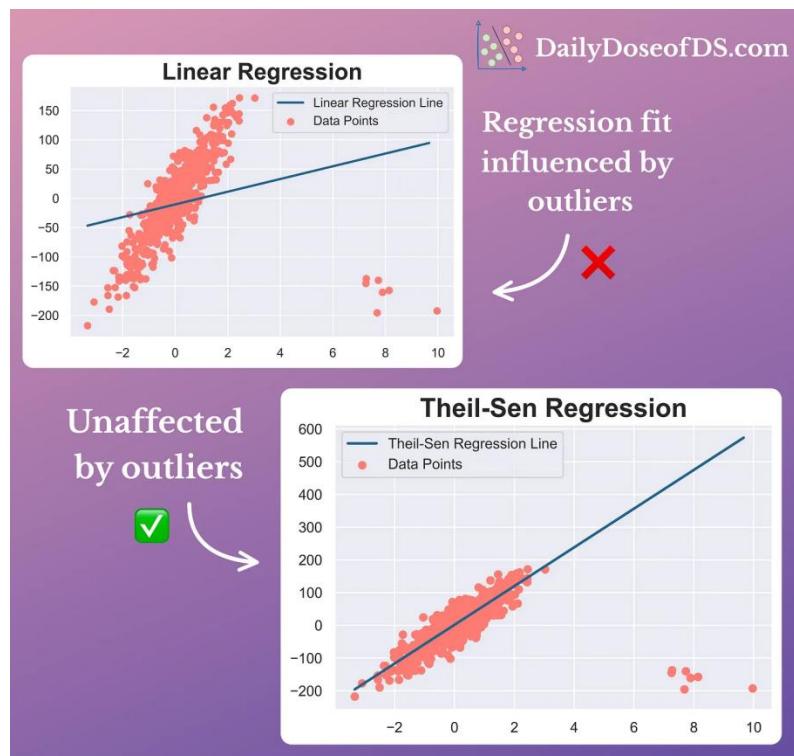
Робустност:

- LR је осетљива на аутлајере и не подноси аномалије у подацима, што може довести до нетачних модела ако су присутни аутлајери.
- TSR је робустан метод и може ефикасно радити с аутлајерима и аномалијама у подацима. Она претпоставља да је већина података тачна и моделира везу на основу медијане склонова, чиме се аутлајери минимизују.

Претпоставке:

- LR има неколико претпоставки, укључујући линеарну везу између променљивих, нормалну расподелу резидуала и хомоскедастичност. Ако се ове претпоставке не испуњавају, резултати могу бити нетачни.

- TSR не захтева такве претпоставке и може се користити у случају нормално нерасподељених и хетероскедастичних података.



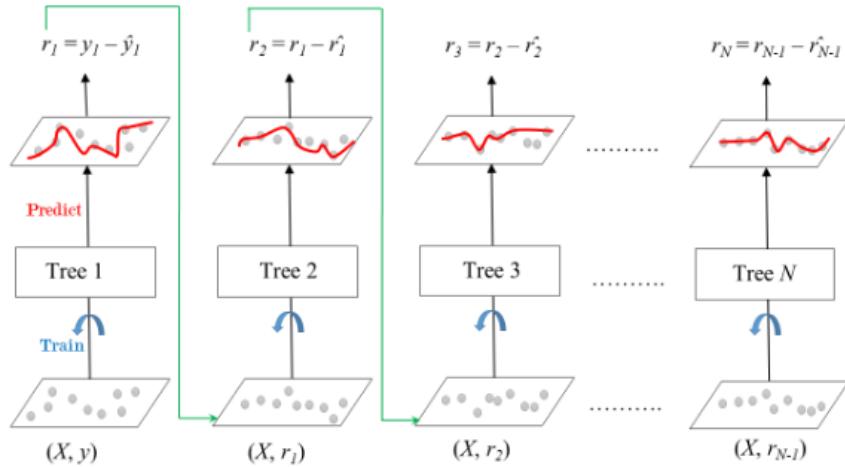
Слика 22: Linear Regressor vs Theil-Sen Regressor

Оба метода имају своје предности и мане, и њихов избор зависи од специфичних карактеристика података и поставке проблема које треба анализирати. У случају када постоје подаци са аутлајерима или када не треба да се претпоставља линеарна веза између променљивих, TSR регресија може бити бољи избор. LR, с друге стране, може бити прикладнија када су испуњене претпоставке о линеарности и нормалности расподеле података.

6.11 Регресор на бази градијентног појачања

Gradient Boosting регресија (GBR) представља моћан алгоритам за регресиону анализу и предвиђање непрекидних вредности. Овај метод припада групи ансамбл алгоритама, који комбинују више слабих модела да би се постигла већа предиктивна способност.

Процес GBR-а започиње коришћењем једноставног модела (нпр. дрво одговора), који се назива базни модел. Затим се разлика између реалних вредности и предвиђених вредности овог модела (погрешка) користи као основа за изградњу следећег модела. Следећи модел се гради тако да апроксимира остатак (погрешку) од претходног модела. Овај поступак се понавља за више итерација, док се не достигне задати број модела или одређен критеријум заустављања (Сл. 23). На крају, предвиђени резултати из свих модела се комбинују како би се добио финални резултат.



Слика 23: Gradient Boosting Regressor

Предности:

- Висока предиктивну способност – ова метода често постиже боље резултате од других регресионих алгоритама.
- Робустна је на аномалије и шум у подацима.
- Може се користити са разноразним типовима података.

Мане:

- Тренирање овог алгоритма може бити временски захтевно, посебно ако користите велики скуп података или комплексне моделе.
- Ако није адекватно контролисан, GBR може бити склон прелетању, што може довести до прекомерног прилагођавања тренинг подацима.

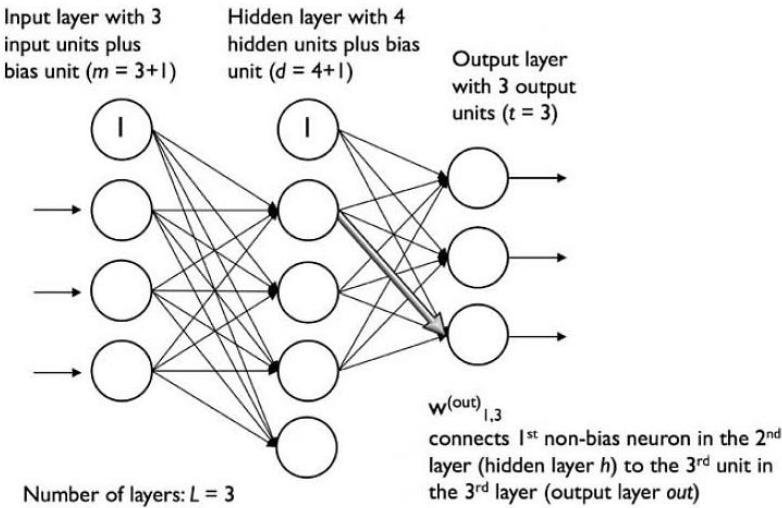
У случају предикције времена траја Миноловац партије, GBR може бити корисна ако постоји комплексан скуп података са различитим факторима који утичу на време траја партије. Ова метода би могла да моделира нелинеарне везе између фактора и времена траја игре. Треба обратити пажњу на оптимизацију параметара алгоритма и контролисање ризика од прелетања како бисте добили најтачније предикције.

6.12 Регресор на бази мултислојног перцептрона

Мултислојни перцептрон (MLP), такође познат као неуронска мрежа, представља тип машинског учења који се користи за регресионе и класификацијоне задатке. Основна идеја иза MLP-а је симулирање рада неурона у мозгу како би се обавили сложени задаци обраде података.

MLP се састоји од више слојева неурона (Сл. 24), укључујући улазни слој, скривене слојеве и излазни слој. Улазни слој приhvата вредности независних променљивих, а излазни слој предвиђа целовиту вредност зависне променљиве (у овом случају, време траја Миноловац партије). Скривени слојеви су између улазног и излазног слоја и служе за обраду података и екстракцију абстрактних фичера из података.

Тренирање MLP-а укључује процес учења у којем модел покушава да минимизује грешку између предвиђених и стварних вредности. Ово се постиже тако што се користи алгоритам оптимизације (као што је градијентни спуст) да се прилагоде вредности тежина у мрежи.



Слика 24: Multilayer Perceptron (Neural Network)

Предности:

- Способност моделирања сложених нелинеарних веза између променљивих.
- Може се конфигурисати број и величину скривених слојева и број неурона, што омогућава прилагођавање модела различитим задацима.
- Добре перформансе на великом броју података.

Мане:

- Захтев за великим бројем података – за успешно тренирање MLP-а, обично је потребан велики број података.
- Може бити склоан прелетању (overfitting) на тренинг подацима ако није адекватно регулисан.

У случају предикције времена траја Миноловац партије, MLP регресија може бити корисна ако постоји велики скуп података који укључује различите факторе који могу утицати на време траја игре.

6.13 Упоређивање регресора

Упоређивање TSR, MLP и GBR на малим скуповима података има важне импликације за избор регресионог модела:

- TSR – је робустан метод и обично добро ради на малим скуповима података. Не захтева претпоставке о расподели података или хомоскедастичности, што га чини придодатним алгоритмом на малим скуповима са нелинеарним или неунформним распределама. Може бити мање прецизан у односу на неке друге методе ако подаци и веза између променљивих нису линеарни.

- MLP – може моделирати сложене нелинеарне везе између променљивих и прилагодити се разноликим облицима података. Решавање проблема прецизности може бити постигнуто на малим скуповима података. Најосетљивији је на прелетање и може имати тенденцију ка оверфитовању на малим скуповима података ако се не користе адекватне технике регуларизације. Обично захтева више података и може бити тежак за обучавање на малим скуповима.
- GBR – може постићи врло високу тачност регресије и обично је робустан алгоритам. Постоји и адаптација на малим скуповима података, када се користи за моделирање комплексних зависности. Може бити склон прелетању на малим скуповима података ако није добро настројен или ако нема добро одабране параметре за регуларизацију.

Избор између ових метода зависи од природе података, циља регресије и компјутерских ресурса. Ако се располаже са малом количином података и потребан је робустан регресиони модел без значајних претпоставки о подацима, TSR може бити добар избор. MLP регресор се обично користи ако су подаци нелинеарни и ако има доволно података за тренирање модела. GBR може бити изабрано када је потребно постићи високу тачност регресије и ако можете обезбедити додатне податке за тренинг.

6.14 Евалуација перформанси класификатора

Евалуација предикције у контексту класификације и регресије је од суштинске важности за процену перформанси модела ML – она нам помаже да разумемо колико добро модел ради и омогућава идентификацију могућих унапређења.

У проблемима класификације, циљ је предвиђати класу илити категорију на основу улазних података. Следи неколико уобичајених метрика за евалуацију перформанси класификационих модела.

Матрица конфузије (или конфузиона матрица) је алат који се често користи у анализи перформанси класификационих модела у машинском учењу и статистици; за процену тачности модела тако што упоређује стварне вредности циљне променљиве (или класе) са предвиђеним вредностима које модел генерише (Сл. 25).

		ACTUAL VALUES	
		Positive	Negative
PREDICTED VALUES	Positive	TP	FP
	Negative	FN	TN

The predicted value is positive and its positive

Type I error : The predicted value is positive but it False

Type II error : The predicted value is negative but its positive

The predicted value is Negative and its Negative

Слика 25: Детаљан опис матрице конфузије

У нашем случају то је квадратна матрица која садржи четири основне вредности, базиране на две класе (назовимо их А и Б):

- True Positives(TP) – број тачно класификованих инстанци које припадају класи А.
- False Positives (FP) – број инстанци које су нетачно класификоване као да припадају А, иако припадају Б (такође се називају тип I грешке).
- True Negatives (TN) – број тачно класификованих инстанци које припадају класи Б.
- False Negatives (FN) – број инстанци које су нетачно класификоване као да припадају Б, иако припадају А (такође се називају тип II грешке).

Матрица конфузије даље омогућава анализу перформанси модела путем израчунавања различитих метрика, укључујући:

- Тачност (енг. accuracy) мери удео тачних предикција у односу на укупан број инстанци у тестном скупу података. Тачност је често прва метрика која се посматра, али може бити обмањујућа ако имате неуравнотежену заступљеност класа. Рачуна се као:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

- Прецизност (енг. precision) метрика мери колико је предикција позитивне класе тачна – што је корисно када је важно смањити FP предикције. Рачуна се као:

$$\frac{TP}{TP + FP}$$

- Одзив (енг. recall или true positive rate) – мери колико правих позитивних инстанци је модел успео да идентификује. Корисно је када је важно минимизовати FN предикције. Рачуна се као:

$$\frac{TP}{TP + FN}$$

- Специфичност (енг. specificity) – мери способност модела да идентификује све TN инстанце. Рачуна се као:

$$\frac{TN}{TN + FP}$$

- F1-скор – је хармонијски средњи однос између прецизности и одзива. Ова метрика је корисна када желите баланс између прецизности и одзива:

$$2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

- Area Under the Receiver Operating Characteristic Curve (AUC-ROC) – ROC крива графика приказује перформансе модела у односу на праг одлучивања, а AUC-ROC мери површину испод те криве. Ово је корисно за процену перформанси у ситуацијама са неравнотеженим класама.

Наведене метрике се приказују и кроз алат под називом класификациони извештај, који је уз матрицу конфузије комплементарни алат, који омогућава целокупно разумевање перформанси класификационих модела. Класификациони извештај је користан за брзи преглед укупних перформанси модела и може бити користан када треба упоредити више модела или експеримента.

Матрица конфузије се користи за дубљу анализу и дијагностику грешака модела, да би се идентификовали специфични проблеми, као што су нпр. класе са високим лажним позитивима или лажним негативима.

Класификациони извештај првенствено пружа информације о прецизности, одзиву, F1-скору и другим метрикама за сваку класу у класификацији. Осим тога, можете добити укупне метрике које обухватају све класе, док матрица конфузије даје детаљан приказ стварних и предвиђених вредности за сваку класу.

6.15 Евалуација перформанси регресора

У проблемима регресије, циљ је предвиђати континуиране вредности, обично нумерицке. Следи неколико уобичајених метрика за евалуацију перформанси регресионих модела (Сл. 26):

- Средња квадратна грешка (енг. mean squared error; MSE) – мери просечну квадратну разлику између стварних и предвиђених вредности. Веће вредности указују на већу грешку.

- Средња апсолутна грешка (енг. mean absolute error; MAE) – мери просечну апсолутну разлику између стварних и предвиђених вредности. Ова метрика је мање осетљива на outlier-e у подацима у поређењу са MSE.
- Средња апсолутна процењена грешка (енг. mean absolute percentage error) – мери просечну релативну грешку у односу на стварне вредности. Ова метрика је корисна када желимо разумети релативну тачност предикција.
- Р-квадрат (coefficient of determination; R^2) – мери колики део варијације у зависној варијабли је објашњен моделом. Вредност се креће од 0 (модел не објашњава варијацију) до 1 (модел савршено објашњава варијацију).

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}|$$

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where,

\hat{y} – predicted value of y
 \bar{y} – mean value of y

Слика 26: Неке од метрика које се користе за евалуацију регресорског модела

Евалуација се често врши на одвојеном тестном скупу података који модел није видео током тренинга како би се оценила његова способност генерализације на новим подацима. Избор одговарајућих метрика зависи од конкретног проблема и циља модела, а често је корисно користити више метрика како бисте добили целовиту слику о перформансама модела.

7. Имплементација система путем Python-a

У овом поглављу је дат детаљан опис целог кода, подељен на смислене целине. Треба напоменути да је првих 76 линија овде изостављено, зато што њихов садржај нема потребе описивати. На њима је приказана листа свих библиотека и њихових верзија, инсталираних на персоналном лаптопу који је коришћен за израду овог мастер рада – и налазе се у Додатку, на крају рада.

Неке од функција садрже једнолинијски коментар INNER директно испод назива – то је само помоћни индикатор, који нам говори да се оне позивају у склопу других функција (изузетак су две функције из потпоглавља 7.11, које су дефинисане у оквиру друге функције).

7.0. Учитавање неопходних података

Листинг 1 приказује увожење (импортовање) неопходних компонената, како би омогућили жељену манипулацију подацима, обраду сигнала, ML и евалуацију модела у оквиру Python окружења. У сваку скрипту треба увести све неопходне библиотеке, модуле, класе, функције (илити методе) и атрибуте – сви ти структурирани елементи нам омогућавају директну или индиректну имплементацију сложених функционалности и алгоритама који ће се користити.

Неки од њих долазе уз инсталацију самог Python-a, док већину треба инсталирати по потреби – Python је целокупно најкоришћенији програмски језик на свету, и самим тим постоји неограничен број различитих библиотека за (условно речено) све што некоме може пасти на памет. Испод слике следи кратак опис свега што се на њој види:

```
import os, json
from time import time
from datetime import datetime
from collections import Counter
from warnings import simplefilter

import numpy as np
from pandas import read_csv, DataFrame

from scipy.signal import correlate2d
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import(
    accuracy_score, confusion_matrix, classification_report,
    mean_squared_error, mean_absolute_error, r2_score)

from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier as DTC
from sklearn.ensemble import RandomForestClassifier as RFC

from sklearn.linear_model import TheilSenRegressor as TSR
from sklearn.neural_network import MLPRegressor as NNR
from sklearn.ensemble import GradientBoostingRegressor as GBR
```

```

from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE

import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
from seaborn import heatmap

import streamlit as st

```

Листинг 1 – увођење неопходних библиотека и њихових подструктуре

- os – модул који пружа функционалности за интеракцију са оперативним системом, укључујући: читање или писање у системско окружење, управљање директоријумима, читање информација о фајловима, итд.
- json – модул за рад са JSON подацима (да бисмо конвертовали податке из JSON формата у Python објекте).
- time – функција из истоименог time модула која враћа тренутно време изразено као број секунди од почетка међународно утврђене епохе.
- datetime – модул за рад са датумима и временима (за креирање, парсирање, форматирање или манипулисање датумима и временима).
- numpy – библиотека која пружа подршку за велике многодимензионалне низове и матрице, заједно с великим колекцијама математичких функција за операције с тим низовима.
- read_csv и DataFrame – функционалности из pandas библиотеке које користите за читање CSV фајлова и манипулисање са табеларним подацима, респективно.
- correlate2d – функција из scipy.signal модула која служи за израчунавање 2Д корелације.
- train_test_split – функција из sklearn.model_selection модула која се користи за поделу података на тренинг и тест скупове
- GridSearchCV – класа из sklearn.model_selection модула која служи за оптимизацију хиперпараметара модела.
- accuracy_score, confusion_matrix, classification_report, mean_squared_error, mean_absolute_error, r2_score – функције из sklearn.metrics модула које се користе за израчунавање различитих метрика перформанси модела.
- SVC, DecisionTreeClassifier, RandomForestClassifier, TheilSenRegressor, GradientBoostingRegressor, MLPRegressor – класе из различитих модула у оквиру sklearn библиотеке, свака представља различит алгоритам ML-а.
- StandardScaler – класа из sklearn.preprocessing модула која се користи за стандардизацију података.
- SMOTE – класа из imblearn.over_sampling модула која се користи за балансирање класа у неизбалансираним базама података.

- Counter – класа из collections модула која служи за бројање елемената у колекцијама.
- matplotlib.pyplot – модул за једноставније визуализације података; за креацију различитих типова графикона.
- MaxNLocator – класа из matplotlib.ticker модула која омогућава контролу локације ознака на осама графикона.
- heatmap – функција из seaborn библиотеке која креира топлотне мапе сигнала, које су корисне ако радимо са матричним подацима.
- streamlit – библиотека за креирање интерактивних веб апликација (приказивање података, графикона, интерактивних форми итд.)

За већину случајева су, уместо целих библиотека, убачени само њихови делови који ће се користити – што нам смањује потрошњу радне меморије (где се учитавају), чинећи програм бржим и ефикаснијим.

Треба обратити посебну пажњу на наредни делић кода, приказан на Листингу 2.

- simplefilter је функција из warnings модула која омогућава контролу тога како се различита упозорења приказују - уопштеним постављањем филтера за упозорења на ignore (без референцирања на неку конкретну групу упозорења) значи да ће сва упозорења бити игнорисана.
- Одмах испод налази се линија која користи ос модул да постави окружење PYTHONWARNINGS на ignore.

Коришћење оба метода за игнорисање упозорења истовремено је веома корисно, зато што simplefilter игнорише упозорења која су већ генерисана у текућој Python сесији, а PYTHONWARNINGS ће поставити окружење тако да се упозорења игноришу пре него што се Python сесија покрене. Укратко: коришћењем оба метода обезбеђено је да су сва упозорења игнорисана, без обзира на то када и како су генерисана.

```
simplefilter("ignore")
os.environ["PYTHONWARNINGS"] = "ignore"
```

Листинг 2 – игнорисање свих врста упозорења

Разлог за игнорисање упозорења је то што је комбинација сложености овде представљеног алгоритма и чињенице да се располагало са малим скупом података довела до тога да је при неким покретањима немогуће изменити код тако да немамо нежељене исписе (ако изузмемо избацање делова кода).

7.1 Читање снимљених података о EEG-у

Функција extract_data (Листинг 3) је дизајнирана да извуче податке из датотека у одређеном директоријуму. Прво се креирају три празне листе: csv_values_data, csv_intervals_data и json_annotations_data. Ове листе ће бити коришћене касније за чување података извучених из датотека, и то:

- снимљених EEG сигнала (CSV),

- информација о временским интервалима или ознакама (енг. маркерс) током EEG снимања (CSV),
- додатне информације о подацима, тј. метаподаци (JSON).

Затим, узима се путања ка директоријуму из system_parameters[path] и чува се у променљивој directory_path.

Након тога, приступа се креирању речника files_lists код кога су кључеви екstenзије датотека, а вредности иницијално празне листе. За сваку екstenзију датотеке у system_parameters[files extensions], функција пролази кроз све датотеке у директоријуму – ако се датотека завршава са том екstenзијом, додаје се у одговарајућу листу у files_lists.

```
def extract_data():

    csv_values_data = []
    csv_intervals_data = []
    json_annotations_data = []

    directory_path = system_parameters["path"]

    files_lists = {key: [] for key in system_parameters["files
extensions"]}

    for key, extension in system_parameters["files
extensions"].items():
        files_lists[key] = sorted(
            [file for file in os.listdir(directory_path)
             if file.endswith(extension)],
            )

    for i in range(len(files_lists["csv values files"])):
        df_values = read_csv(
            os.path.join(directory_path,
                         files_lists["csv values files"][i]),
            skiprows=1,
            )

        columns_to_drop = processing_parameters["csv values: columns to
drop"]

        for j in range(len(columns_to_drop)):
            if columns_to_drop[j] in df_values.columns:
                df_values = df_values.drop(
                    columns_to_drop[j],
                    axis=1,
                    )
        csv_values_data.append(df_values)

    df_intervals = read_csv(
```

```

        os.path.join(directory_path,
                      files_lists["csv intervals files"][i]),
                  )
    csv_intervals_data.append(
        df_intervals[processing_parameters[
            "csv intervals: timestamps column name"]].tolist()
    )

    with open(os.path.join(
        directory_path,
        files_lists["json annotations files"][i]),
        "r") as json_file:

        json_data = json.load(json_file)

        json_annotations_data.append(
            {key: json_data.get(key, "")}
            for key in processing_parameters["game annotations"],
        )

    return csv_values_data, csv_intervals_data, json_annotations_data

```

Листинг 3 – извлачење корисних података из CSV и JSON фајлова генерисаних од стране EPOS система

Код затим пролази кроз све CSV датотеке наведене у files_lists[csv values files]. За сваку датотеку, избацују се колоне означене у processing_parameters[csv values: columns to drop].

Затим се генерише путању до CSV датотеке користећи os.path.join (спајање directory_path и имена датотеке у јединствену путању).

Програм чита CSV датотеку помоћу функције read_csv, прескачући први ред (прво заглавље; у суштини постоје два дефинисана заглавља, што збуњује read_csv). Проверава се да ли су колоне које треба избацити присутне у DataFrame-у – ако јесу, избацују се преко методе drop. Након обраде, DataFrame се додаје у листу csv_values_data.

Затим, програм генерише путању до CSV датотеке са интервалима (на аналоган начин претходно описаном), па затим чита CSV датотеку користећи read_csv. Након читања, издваја се колона са временским ознакама (наведена у processing_parameters[csv intervals: timestamps column name]), конвертује се у листу и додаје је у листу csv_intervals_data.

Слично претходно споменутом, генерише се путања до JSON датотеке са аnotацијама, па се чита њен садржај путем методе json.load.

Затим се креира речник који садржи кључеве наведене у processing_parameters[game annotations], а вредности за те кључеве су извучене из JSON података - овај речник се затим додаје у листу json_annotations_data.

Након што је свака датотека обрађена, функција враћа три попуњене листе: csv_values_data, csv_intervals_data и json_annotations_data.

7.2 Извлачење података о амплитуди и снази

Функција processing (Листинг 4) је дизајнирана да обради податке извучене из CSV датотека – прво се креирају две празне листе: eeg_amplitude и eeg_power.

Затим се пролази кроз сваки DataFrame у csv_values_data и узимају се: индекс колоне за временске ознаке и индекси првих колона који садрже амплитуду и снагу EEG-а. То се постиже применом методе get_loc над колонама DataFrame-а, уз узимање имена колона из processing_parameters речника.

```
def processing(csv_values_data: list,
               csv_intervals_data: list,
               json_annotations_data: list):

    eeg_amplitude = []
    eeg_power = []

    for i in range(len(csv_values_data)):
        current_df = csv_values_data[i]

        timestamps_column_index = current_df.columns.get_loc(
            processing_parameters["csv values: timestamps column
name"]),
        )
        amp_1st_column_index = current_df.columns.get_loc(
            processing_parameters["csv values: amp 1st column name"]),
        )
        pow_1st_column_index = current_df.columns.get_loc(
            processing_parameters["csv values: pow 1st column name"]),
        )

        timestamps = current_df.iloc[:, timestamps_column_index].values

        indexes = np.where(
            (timestamps > csv_intervals_data[i][0]) &
            (timestamps < csv_intervals_data[i][1])
            )[0]

        eeg_amplitude.append(
            current_df.iloc[:, amp_1st_column_index :
amp_1st_column_index + 5].
            values[indexes],
            )
        eeg_power.append(
            current_df.iloc[:, pow_1st_column_index:].
            values[indexes],
            )

    column_names = create_columns_names(
        current_df,
        amp_1st_column_index,
```

```

    pow_1st_column_index,
)

eeg_amplitude_w, eeg_power_w = windowing(eeg_amplitude, eeg_power)

return eeg_amplitude_w, eeg_power_w, json_annotations_data,
column_names

```

Листинг 4 – процесирање EEG амплитуда и снага

Следећи део функције обрађује податке из DataFrame-а и издваја вредности амплитуде и снаге EEG сигнала у одређеним временским интервалима.

Прво, програм издваја вредности временских ознака из current_df и чува их у променљивој timestamps. Затим се креира маска која указује на то да ли је (свака индивидуално) временска ознака између две вредности у csv_intervals_data[i]. Налазе се индекси на којима маска враћа True, помоћу np.where. Након тога се издвајају вредности амплитуде и снаге EEG-а на тим индексима и додају се у листе eeg_amplitude и eeg_power, респективно.

Након што је сваки DataFrame обрађен, позива се функцију create_columns_names да, као што јој име каже, креира имена колона за обрађене податке. На излазу се добијају три листе: eeg_amplitude, eeg_power и c_n.

7.3 Креирање имена колона матрице обележја

Код креирања имена колона за будућу матрицу обележја (Листинг 5), прво се преузимају имена свих колона из current_df и чувају у листи columns_names – из које функција издваја имена колона за амплитуду EEG сигнала и чува их у новој листи: amp_names. Након тога се приступа формирању финалне листе, која се добија спајањем:

- имена колона за средњу вредност и варијансу амплитуде EEG сигнала (базирана на именима колона из CSV фајла са вредностима) (10 вредности),
- имена колона за корелацију и крос-коваријансу (4 вредности),
- имена колона за снагу EEG сигнала (25 вредности),
- имена колона наведена у processing_parameters[game annotations] (од 2 до 5 вредности).

Из овога можемо видети да ће матрица обележја имати између 41 и 44 колона.

```

def create_columns_names(current_df: DataFrame,
                        amp_1st_column_index: int,
                        pow_1st_column_index: int):
    # INNER
    columns_names = current_df.columns.tolist()

    amp_names = columns_names[amp_1st_column_index :
    amp_1st_column_index + 5]

    columns_names = [f"{j}.{stat}" for stat in ['mean', 'var']]

```

```

        for j in amp_names + ["Corr", "Xcov"]]\ 
            + columns_names[pow_1st_column_index : ]\ 
            + processing_parameters["game annotations"]

return columns_names

```

Листинг 5 – креирање листе имена колона за будућу матрицу обележја

7.4 Прозорирање EEG-а – над амплитудом и снагом

Функција windowing (Листинг 6) је дизајнирана да примени прозорску функцију над EEG амплитудама и снази. Прво се рачуна број узорака по прозору множењем дужине прозора (system_parameters[window_length]) и фреквенције одабирања (system_parameters[fs]). Резултат се заокружује на најближи цео број и конвертује у целобројни тип.

Затим, функција креира две празне листе: eeg_amplitude_w и eeg_power_w. Пролази се кроз сваки елемент (i) у eeg_amplitude и eeg_power и чувају се вредности амплитуде и снаге у eeg_amplitude_i и eeg_power_i, као и дужина eeg_amplitude_i.

Након тога, креирају се две нове празне листе: eeg_amplitude_i_w и eeg_power_i_w; које ће бити коришћене касније за чување обрађених података на нивоима појединачних прозора..

```

def windowing(eeg_amplitude: list,
              eeg_power: list):
    # INNER
    samples_per_window = np.round(
        system_parameters["window length"] * 
        system_parameters["fs"],
        ).astype(int)

    eeg_amplitude_w = []
    eeg_power_w = []

    for i in range(len(eeg_amplitude)):
        eeg_amplitude_i = eeg_amplitude[i]
        eeg_power_i = eeg_power[i]

        eeg_amplitude_i_len = len(eeg_amplitude_i)

        eeg_amplitude_i_w = []
        eeg_power_i_w = []

        for j in range(0, eeg_amplitude_i_len, samples_per_window):
            if j + samples_per_window <= eeg_amplitude_i_len:
                eeg_amplitude_i_w.append(
                    eeg_amplitude_i[j:j + samples_per_window],
                    )
                eeg_power_i_w.append(
                    eeg_power_i[j:j + samples_per_window],
                    )

```

```

        )
eeg_amplitude_w.append(
    np.array(eeg_amplitude_i_w),
)
eeg_power_w.append(
    np.array(eeg_power_i_w),
)

return eeg_amplitude_w, eeg_power_w

```

Листинг 6 – прозорирање EEG амплитуда и снага

Функција пролази кроз сваки узорак у `eeg_amplitude_i` и `eeg_power_i`, кораком величине `samples_per_window`. За сваки узорак, проверава се да ли је збир тренутног индекса и величине прозора мањи или једнак дужини `eeg_amplitude_i`. Ако јесте, то значи да постоји доволно узорака за цео тренутни прозор – ти узорци се издавају из `eeg_amplitude_i` и `eeg_power_i` и додају у листе `eeg_amplitude_i_w` и `eeg_power_i_w`.

Након што су сви узорци за тренутни сигнал обрађени, код додаје листе `eeg_amplitude_i_w` и `eeg_power_i_w` у листе `eeg_amplitude_w` и `eeg_power_w`, које су и излаз из функције. То су листе испуњене `numpy` низовима, који су у суштини исто листе (тј. низови података) – сваки унутрашњи низ одговара једном прозорираном амплитудном или снаге каналу, а спољашње листе одговарају појединачним мерењима (тј. партијама Миноловца).

7.5 Извлачење обележја из EEG сигнала

Функција `eeg_feature_extraction` (Листинг 7) извлачи обележја из EEG амплитуде и снаге. Прво се креирају празне листе за чување различитих врста статистичких обележја, затим се пролази кроз троструку `for` петљу – прво (i) се пролази кроз сваки елемент у `eeg_amplitude_w` и `eeg_power_w`. За сваки елемент, чувају се вредности амплитуде и снаге EEG сигнала у `eeg_amplitude_w_i` и `eeg_power_w_i`, такође и дужина `eeg_amplitude_w_i` у `eeg_amplitude_w_i_len`.

Након тога, дефинишу се празне листе за чување средњих вредности и варијанси амплитуде EEG сигнала и средњих вредности снаге EEG сигнала. Такође се дефинишу две квадратне матрице испуњене нулама, `correlation_matrix` и `cross_covariance_matrix`, које имају исту димензију као `eeg_amplitude_w_i`.

```

def eeg_feature_extraction(eeg_amplitude_w: list,
                           eeg_power_w: list,
                           json_annotations_data: list,
                           column_names: list):
    eeg_amplitude_means = []
    eeg_amplitude_vars = []

    eeg_amplitude_corr_means = []
    eeg_amplitude_corr_vars = []
    eeg_amplitude_xcov_means = []

```

```

eeg_amplitude_xcov_vars = []

eeg_power_means = []

for i in range(len(eeg_amplitude_w)):
    eeg_amplitude_w_i = eeg_amplitude_w[i]
    eeg_amplitude_w_i_len = len(eeg_amplitude_w_i)

    eeg_amplitude_means_i = []
    eeg_amplitude_vars_i = []

    eeg_power_w_i = eeg_power_w[i]
    eeg_power_means_i = []

    correlation_matrix = np.zeros(
        shape=(eeg_amplitude_w_i_len, eeg_amplitude_w_i_len),
        )

    cross_covariance_matrix = np.zeros_like(
        a=correlation_matrix,
        )

    for j in range(eeg_amplitude_w_i_len):
        current_window_amp = eeg_amplitude_w_i[j]

        eeg_amplitude_means_i.append(
            np.mean(current_window_amp, axis=0),
            )

        eeg_amplitude_vars_i.append(
            np.var(current_window_amp, axis=0),
            )

        current_window_pow = eeg_power_w_i[j]

        current_window_pow = np.mean(
            current_window_pow[
                ~np.isnan(current_window_pow[:, 0])],
            axis=0,
            )

        eeg_power_means_i.append(current_window_pow)

    for k in range(eeg_amplitude_w_i_len):
        correlation_matrix[j, k] = np.corrcoef(
            eeg_amplitude_w_i[j].flatten(),
            eeg_amplitude_w_i[k].flatten(),
            )[0, 1]

        if j != k:

```

```

        cross_covariance_matrix[j, k] = correlate2d(
            eeg_amplitude_w_i[j],
            eeg_amplitude_w_i[k],
            mode="valid",
            )[0, 0]

eeg_amplitude_means.append(
    np.mean(np.array(eeg_amplitude_means_i), axis=0),
)
eeg_amplitude_vars.append(
    np.mean(np.array(eeg_amplitude_vars_i), axis=0),
)
eeg_amplitude_corr_means.append(
    np.mean(correlation_matrix),
)
eeg_amplitude_corr_vars.append(
    np.var(correlation_matrix),
)
eeg_amplitude_xcov_means.append(
    np.mean(cross_covariance_matrix),
)
eeg_amplitude_xcov_vars.append(
    np.var(cross_covariance_matrix),
)
eeg_power_means.append(
    np.mean(np.array(eeg_power_means_i), axis=0),
)

eeg_features = np.hstack(
    tup=(np.array(eeg_amplitude_means),
        np.array(eeg_amplitude_vars),
        np.array(eeg_amplitude_corr_means).reshape(-1, 1),
        np.array(eeg_amplitude_corr_vars).reshape(-1, 1),
        (np.array(eeg_amplitude_xcov_means) / 1e10).reshape(-1,
1),
        (np.array(eeg_amplitude_xcov_vars) / 1e19).reshape(-1, 1),
        np.array(eeg_power_means),
    ),
)
return eeg_features, json_annotations_data, column_names

```

Листинг 7 – извлачење статистичких обележја из EEG-а

Затим се улази у угњеждену for петљу првог степена (j), где се пролази кроз сваки прозор у eeg_amplitude_w_i и eeg_power_w_i – вредности амплитуде и снаге EEG сигнала се чувају у current_window_amp и current_window_pow. Затим се рачунају средња вредност и варијанса current_window_amp и додају у листе eeg_amplitude_means_i и eeg_amplitude_vars_i, респективно.

Уклањају се NaN вредности из current_window_pow, па се онда рачуна средња вредност current_window_pow и додаје у листу eeg_power_means_i.

Даље се приступа угњежђеној for петљи другог степена (k), у склопу које се рачунају корелација и крос-коваријанса између свих прозора амплитуде EEG сигнала – за сваки прозор се рачуна корелациони коефицијент између eeg_amplitude_w_i[j] и eeg_amplitude_w_i[k] користећи функцију np.corrcoef, а резултат се чува у correlation_matrix.

Ако итератор j није једнак k, рачуна се крос-коваријанса између eeg_amplitude_w_i[j] и eeg_amplitude_w_i[k] користећи функцију correlate2d (резултат се чува у cross_covariance_matrix). Крос-коваријанса сигнала са самим собом нам не носи битне информације.

У следећем делу кода се за сваки прозор рачуна средња вредност eeg_amplitude_means_i и eeg_amplitude_vars_i и додаје се у листе eeg_amplitude_means и eeg_amplitude_vars.

Програм такође рачуна средње вредности и варијансе correlation_matrix и cross_covariance_matrix и додаје их у листе eeg_amplitude_corr_means и eeg_amplitude_corr_vars, и eeg_amplitude_xcov_means и eeg_amplitude_xcov_vars, респективно.

Затим, програм рачуна средњу вредност eeg_power_means_i и додаје је у листу eeg_power_means – средње вредности снаге канала на нивоу појединачних прозора. Сваки елемент у листи садржи по 25 вредности – за сваки од 5 електродних канала имамо по 5 вредности снаге, нпр:

- POW.AF3.Theta, POW.AF3.Alpha, POW.AF3.BetaL,
 POW.AF3.BetaH, POW.AF3.Gamma.

Сви споменути рачуни се врше по колонама, да би имали по један ред у матрици обележја за сваку партију Миноловца.

На крају се користи функција np.hstack за конкатенацију више низова у један хоризонтални (колона) низ. Ово се ради за све иницијално празно дефинисане листе са почетка ове функције:

- средње вредности амплитуде EEG сигнала (eeg_amplitude_means),
- варијансе амплитуде EEG сигнала (eeg_amplitude_vars),
- средње вредности корелационих матрица амплитуде EEG сигнала (eeg_amplitude_corr_means),
- варијансе корелационих матрица амплитуде EEG сигнала (eeg_amplitude_corr_vars),
- средње вредности матрица крос-коваријанса амплитуде EEG сигнала (eeg_amplitude_xcov_means), дељене са 1e10,
- варијансе матрица крос-коваријанса амплитуде EEG сигнала (eeg_amplitude_xcov_vars), дељене са 1e19,
- средње вредности снаге EEG сигнала (eeg_power_means).

Након што је сваки низ обрађен и спојен, функција враћа резултат у нечemu што доста подсећа на матрицу обележја EEG-а

Вредности које се односе на крос-коваријансу су у овом кораку и нормиране, зато што би иначе јако пуно одступале по реду величине од осталих..

7.6 Формирање матрице обележја

Функција `creating_the_feature_matrix` (Листинг 8) је дизајнирана да креира финализирану матрицу обележја, на нивоу целог експеримента. Прво, функција креира речник `game_features` где су кључеви имена карактеристика игре, а вредности су празне листе. Затим се пролази кроз сваку анотацију игре у `game_annotations` - за све постојеће се додају вредности за сваки кључ у одговарајућу листу у `game_features`.

Накнадно се приступа елементима речника, пошто за је за успешну анализу потребно да нам сви подаци буду у нумеричком облику, и то посебних формата.

Стога, прво се врши конверзија времена трајања партија из формата минут:секунд у укупан број секунди и чува их у `game_features[Time]`. Потом се мапирају исходи партија (Win или Lose) на бинарне вредности (1 или 0) и чувају у `game_features[Outcome]`.

```
def creating_the_feature_matrix(eeg_features: np.ndarray,
                                 game_annotations: list,
                                 column_names: list):

    game_features = {key: [] for key in game_annotations[0].keys()}

    for game_annotation in game_annotations:
        for key, values in game_annotation.items():
            game_features[key].append(values)

    time_in_sec = []
    for time_str in game_features["Time"]:
        minutes, seconds = map(int, time_str.split(":"))
        time_in_sec.append(minutes * 60 + seconds)
    game_features["Time"] = time_in_sec

    outcome_mapping = {"Win": 1, "Lose": 0}
    game_features["Outcome"] = [outcome_mapping[outcome]
                               for outcome in game_features["Outcome"]
                               ]

    difficulty_mapping = {"Easy": 10, "Normal": 100, "Advanced": 1000}
    keys_to_process = ["Difficulty", "Fields", "Mines"]

    if use_streamlit or \
            prediction_parameters["use non essential game annotations"]:
        for key in keys_to_process:
            if key in game_features:
                if key == "Difficulty":
                    game_features[key] =
                    [difficulty_mapping[difficulty]
```

```

        for difficulty in
game_features[key]]
    else:
        game_features[key] = [int(value) for value in
game_features[key]]

for key, values in game_features.items():
    game_features[key] = np.array(values).reshape(-1, 1)

game_features = np.hstack(
    tup=list(game_features.values()),
)

feature_matrix = DataFrame(
    data=np.hstack(
        tup=(eeg_features, game_features)),
    columns=column_names,
)

feature_matrix.to_csv(f"{current_datetime}_feature_matrix.csv",
index=False)

return feature_matrix

```

Листинг 8 – креирање комплетне матрице обележја

Даље се креира мапа тежина (difficulty_mapping) која мапира Easy, Normal и Advanced на 10, 100 и 1000, респективно. Ако је use_streamlit постављено на True, код пролази кроз сваки кључ у game_features:

- ако Difficulty постоји као кључ, програм мапира сваку тежину у game_features[Difficulty] користећи difficulty_mapping и ажурира вредности у game_features[Difficulty],
- ако Fields постоји као кључ, програм конвертује сваку вредност у game_features[Fields] у целобројни тип и ажурира вредности у game_features[Fields],
- ако Mines постоји као кључ, програм ради све аналогно као за Fields.

Овај конкретно део кода је уведен зато што је претходно дизајнирано да при коришћењу Streamlit апликације корисник може да одабере произвољан број обележја игре, док при стандардном покретању скрипте постоје само две опције: да се користе есенцијална обележја игре (за овај експеримент то су оно што желимо да предвидимо, тј. Outcome и Time) или да се користи свих пет обележја. Стога, без овога би наредни део кода не би радио, у општем случају.

У случају да не покрећемо скрипту преко Streamlit-a, а prediction_parameters[use non essential game annotations] је постављено на True, програм ће извршити сва три условна сегмента из претходног дела одједном (без услова).

Затим, код претвара сваку листу вредности у game_features у низ и помоћу reshape() методе мења његов облик тако да има само једну колону - ти низови се затим спајају у један хоризонтални низ.

Након тога, спајају се eeg_features и game_features у целовиту матрицу обележја. Коначно, та матрица се конвертује у DataFrame, при чему јој се напокон и додељују имена колона генерисана помоћу функције create_columns_names..

7.7 Класификација – предикција и евалуација

Функција classification_prediction_and_evaluation (Листинг 9), као што јој само име сугерише, је дизајнирана да изврши класификацију, предикцију и евалуацију исхода игре (тј. партије) на основу матрице обележја.

Прво, функција уклања колону Outcome из feature_matrix и чува је у game_outcomes – то су циљне вредности које треба предвидети.

Затим се креира речник classification_metrics за чување различитих метрика класификације: тачности, матрица конфузије и класификационог извештаја. Функција такође креира стринг outcomes_str за чување информација о стварним и предвиђеним исходима игре.

Пре уласка у for петљу, која се покреће онолики број пута колико корисник жели да постоји итерација експеримента (system_parameters[number of runs]), почиње да се мери време извршавања помоћу функције time().

```
def classification_prediction_and_evaluation(feature_matrix:  
DataFrame):  
    game_outcomes = feature_matrix.pop("Outcome")  
  
    classification_metrics = {  
        "accuracies": [],  
        "confusion_matrices": [],  
        "classification_reports": [],  
    }  
  
    outcomes_str = "GAME OUTCOMES:\n\n"  
  
    start = time()  
    for i in range(system_parameters["number of runs"]):  
        while True:  
            random_state = 42 if prediction_parameters["use fixed  
random state"] \  
                else np.random.randint(1000)  
  
            X_train, X_test, y_train, y_test = train_test_split(  
                feature_matrix.values, game_outcomes,  
                test_size=0.3, random_state=random_state,  
                stratify=game_outcomes if prediction_parameters["use  
stratify"] \  
                    else None,  
            )
```

```

class_counts = Counter(y_train)

if all(count >= 2 for count in class_counts.values()) \
    and len(set(y_test)) >= 2 and len(set(y_train)) >= 2:
    break

if prediction_parameters["use scaling and oversampling for
classification"]:
    scaler = StandardScaler()
    X_train= scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    smote = SMOTE(
        sampling_strategy="auto", random_state=random_state,
        k_neighbors=1, n_jobs=-1)
    X_train, y_train = smote.fit_resample(X_train, y_train)

    classifier_random_state = random_state \
        if prediction_parameters["use same random state for
classifier"] \ 
        else None

if use_streamlit:
    st_classifier = prediction_parameters["pick classifier"]
    combo_map = {
        "use SVC": ("use SVC", False, False),
        "use DTC": (False, "use DTC", False),
        "use RFC": (False, False, "use RFC"),
        }
    prediction_parameters.update(
        dict(zip(["use SVC", "use DTC", "use RFC"],
                combo_map[st_classifier])))
else:
    if prediction_parameters["use SVC"]:
        classifier = SVC(random_state=classifier_random_state)
        param_grid = {
            "C": [0.1, 1],
            "kernel": ["linear", "rbf"],
            "gamma": ["scale", "auto", 0.1],
            "class_weight": [None, "balanced"],
            }
    else:
        param_grid = {
            "criterion": ["gini", "entropy"],
            "max_depth": [None, 2],
            "min_samples_split": [2, 4],
            "min_samples_leaf": [1, 2],
            "max_features": ["auto", "sqrt"],
```

```

        }
    if prediction_parameters["use DTC"]:
        classifier = DTC(random_state=classifier_random_state)

    elif prediction_parameters["use RFC"]:
        classifier = RFC(random_state=classifier_random_state)
        param_grid["n_estimators"]:[100, 200]

    grid_search = GridSearchCV(
        classifier, param_grid, cv=3, n_jobs=-1,
        scoring=prediction_parameters["scoring for classifier grid
search"]),
    )

    grid_search.fit(X_train, y_train)
    best_parameters = grid_search.best_params_

    for param_name, param_value in best_parameters.items():
        setattr(classifier, param_name, param_value)

    classifier.fit(X_train, y_train)
    predicted_outcomes = classifier.predict(X_test)

    classification_metrics["accuracies"].append(
        accuracy_score(y_test, predicted_outcomes),
    )

    classification_metrics["confusion_matrices"].append(
        confusion_matrix(y_test, predicted_outcomes),
    )

    classification_metrics["classification_reports"].append(
        classification_report(y_test, predicted_outcomes,
output_dict=True),
    )

    outcomes_str += f"""
RUN {i+1}
\tpredicted
{np.where(predicted_outcomes == 1.0, "Win", "Lose")}
\treal
{np.where(y_test == 1.0, "Win", "Lose")}
"""

end = time()

with open(f"{current_datetime}_predictions.txt", "w") as f:
    f.write(f"Total execution time - {classifier}:\n \
{np.round(end - start, 3)} s\n\n")
    f.write(outcomes_str + "\n\n")

```

```

feature_matrix[ "Outcome" ] = game_outcomes

return feature_matrix, classification_metrics

```

Листинг 9 – класификација – предикција и евалуација исхода игре

Следећи део кода извршава поделу података на тренинг и тест сетове, користећи функцију `train_test_split`, и то у склопу условно бесконачне while петље.

`random_state` ће бити 42 ако је `prediction_parameters[use fixed random state]` постављено на True - у супротном се генерише случајни број између 0 и 1000. Овај параметар одређује почетне услове интерног генератора псеудо-случајних бројева, тј. омогућава контролу случајности и репродукцију резултата

Затим се користи функција `train_test_split` да подели `feature_matrix.values` (имена колона нам не играју улогу овде) и `game_outcomes` на тренинг и тест скупа - величина тест скупа ће увек бити 30 % укупних података. Ако је `prediction_parameters[use stratify]` постављено на True, користи се стратификована подела - како би се осигурало задржавање баланса између класа кроз различите скупове података. Такође се рачуна и број инстанци сваке класе у `y_train` користећи функцију `Counter`.

На крају, код проверава да ли свака класа има барем две инстанце у тренинг сету и да ли тест сет има барем две различите класе. Ако је то тачно, избећи ће се потенцијалне грешке у даљем извршавању, а програм прекида петљу и наставља са следећим кораком.

Наредни део кода примењује скалирање и пребалансирање на тренинг податке ако је `prediction_parameters[use scaling and oversampling for classification]` постављено на True.

Прво, код креира објекат `StandardScaler` и користи га да скалира `X_train` и `X_test`. Ово је важан корак у припреми података за многе алгоритме машинског учења.

Затим, код креира објекат SMOTE за пребалансирање тренинг података. Ова техника ради тако што генерише нове примере у мањинској класи тако што извршава интерполацију између постојећих примера. Затим се користи SMOTE да пребалансира `X_train` и `y_train`.

На крају, поставља се `random_state` за класификатор на исту вредност као и за функцију `train_test_split` ако је `prediction_parameters[use same random state for classifier]` постављено на True. У супротном, `random_state` за класификатор је None.

Затим се проверава да ли је `use_streamlit` постављено на True. Ако јесте, код извршава следеће кораке:

- Прво, код узима изабрани класификатор из `prediction_parameters[pick classifier]` и чува га у променљивој `st_classifier`.
- Затим, код креира мапу `combo_map` која мапира сваки могући избор класификатора на три вредности: `use SVC`, `use DTC`, и `use RFC`. Сваки од ових избора је или `False` или једнак изабраном класификатору.

- Након тога, код ажурира prediction_parameters тако што замени вредности за use SVC, use DTC, и use RFC са вредностима из combo_map које одговарају изабраном класификатору.

Као и у претходном потпоглављу, и овде се проверава да ли се користи Streamlit услед неслагања – пошто су код регуларног позива скрипте у самом старту дефинисани условни параметри за сва три класификатора, док је у Streamlit у старту дефинисан само један (онај који је одабран из падајуће листе). Овиме се дефинишу сва три условна параметра, те се избегавају грешке у извршавању наредног дела кода.

Затим је приказана креација класификатора и параметарске мреже за претрагу на основу параметара за предикцију.

- Ако је prediction_parameters[use SVC] постављено на True, код креира објекат SVC и параметарску мрежу за SVC. У супротном, код креира параметарску мрежу за класификаторе на основу дрвета одлучивања.
- Ако је prediction_parameters[use DTC] постављено на True, код креира објекат DTC.
- Ако је prediction_parameters[use RFC] постављено на True, код креира објекат RFC и додаје параметар n_estimators у параметарску мрежу.

Наредни део кода извршава класификацију, предвиђање и оцену на основу матрице карактеристика. Прво, програм креира објекат GridSearchCV са класификатором, параметарском мрежом, бројем фолдова за унакрсну валидацију и метриком оцене. Затим, код користи GridSearchCV да би обучио модел на тренинг подацима и нашао најбоље параметре. Затим, код ажурира параметре класификатора на основу најбољих параметара. Онда се поново обучава класификатор са ажурираним параметрима на тренинг подацима.

Након обуčавања, код користи класификатор да предвиди исходе за тест податке. Затим се приступа рачунању тачност предвиђања, која се додаје је листу classification_metrics[accuracies].

Код такође рачуна матрицу конфузије и извештај о класификацији за предвиђене исходе и додаје их у одговарајуће листе у classification_metrics..

Последњи део кода у овој функцији извршава следеће кораке:

- Додаје информације о предвиђеним и стварним исходима игре у стринг outcomes_str за сваки покренути циклус.
- Рачуна укупно време извршења користећи функцију time().
- Записује време извршења и исходе игре у текстуалну датотеку.
- Враћа колону Outcome у feature_matrix.
- На крају, функција враћа feature_matrix и classification_metrics.

7.8 Рачунање усредњеног класификационог извештаја

Функција calculate_the_average_classification_report (Листинг 10) рачуна просечни извештај о класификацији из листе извештаја о класификацији.

Прво, функција креира празан речник average_classification_report за чување просечних вредности.

Затим, функција пролази кроз сваки извештај о класификацији у classification_reports. За сваки извештај, функција додаје вредности у average_classification_report. Ако је вредност речник (што значи да је то под-извештај за одређену класу), функција додаје сваку вредност у под-извештају.

```
def calculate_the_average_classification_report(classification_reports):
    # INNER
    average_classification_report = {}

    for report in classification_reports:
        for key, value in report.items():
            if key not in average_classification_report:
                average_classification_report[key] = value
            else:
                if isinstance(value, dict):
                    for inner_key, inner_value in value.items():
                        average_classification_report[key][inner_key] += \
                            inner_value
                else:
                    average_classification_report[key] += value

    for key, value in average_classification_report.items():
        if isinstance(value, dict):
            for inner_key, inner_value in value.items():
                average_classification_report[key][inner_key] /= \
                    len(classification_reports)
        else:
            average_classification_report[key] /= len(classification_reports)

    return DataFrame(average_classification_report).transpose()
```

Листинг 10 – рачунање средњих вредности свих поља класификационог извештаја

Након што су сви извештаји обрађени, функција пролази кроз average_classification_report и делити сваку вредност са бројем извештаја, чиме се добија просечна вредност.

На крају, функција претвара average_classification_report у DataFrame и транспонује га (претвара редове у колоне и обрнуто), а затим га враћа.

7.9 Регресија – предикција и евалуација

Функција, regression_prediction_and_evaluation (Листинг 11), је дефинисана са два аргумента: feature_matrix и classification_metrics.

feature_matrix је DataFrame који садржи податке који ће бити коришћени за регресиону анализу. Прва линија кода у функцији уклања колону Time из овог DataFrame-a и чува је у променљивој completion_time.

classification_metrics је речник који садржи метрике класификације. Ова променљива тренутно није коришћена у делу кода који сте послали.

Затим, функција креира нови речник, regression_metrics, који садржи празне листе за вредности MSE, MAE и R2.

Променљива completion_times_str се иницијализује као стринг који почиње са COMPLETION TIMES:\n\n.

Коначно, функција покреће тајмер користећи функцију time() из модула time, и вредност чува у променљивој start.

```
def regression_prediction_and_evaluation(feature_matrix: DataFrame,
                                         classification_metrics: dict):
    completion_time = feature_matrix.pop("Time")

    regression_metrics = {
        "mse_values": [],
        "mae_values": [],
        "r2_values": []
    }

    completion_times_str = "COMPLETION TIMES:\n\n"

    start = time()
    for i in range(system_parameters["number of runs"]):
        random_state = 42 if prediction_parameters["use fixed random state"] \
            else np.random.randint(1000)

        X_train, X_test, y_train, y_test = train_test_split(
            feature_matrix.values, completion_time,
            test_size=0.3, random_state=random_state)

        if prediction_parameters["use augmentation for regression"]:
            X_train_augmented = np.vstack((X_train, X_train + 0.01 *
                np.random.standard_normal(X_train.shape)))
            y_train_augmented = np.hstack((y_train, y_train))
        else:
            X_train_augmented, y_train_augmented = X_train, y_train

        regressor_random_state = random_state \
            if prediction_parameters["use same random state for regressor"] \
            else None

        if use_streamlit:
```

```

st_regressor = prediction_parameters["pick regressor"]
combo_map = {
    "use TSR": ("use TSR", False, False),
    "use NNR": (False, "use NNR", False),
    "use GBR": (False, False, "use GBR"),
}
prediction_parameters.update(
    dict(zip(["use TSR", "use NNR", "use GBR"],
            combo_map[st_regressor])))

if prediction_parameters["use TSR"]:
    regressor = TSR(random_state=regressor_random_state)

elif prediction_parameters["use NNR"]:
    regressor = NNR(random_state=regressor_random_state)
    param_grid = {
        "alpha": [0.1, 0.5, 1.0],
        "hidden_layer_sizes": [(50,), (100,), (50, 50), (100,
50)],
        "activation": ["relu", "tanh"],
        "learning_rate_init": [0.001, 0.01],
    }

elif prediction_parameters["use GBR"]:
    regressor = GBR(random_state=regressor_random_state)
    param_grid = {
        "n_estimators": [100, 200],
        "learning_rate": [0.01, 0.05],
        "max_depth": [2, 3],
        "min_samples_split": [2, 4],
        "min_samples_leaf": [1, 2],
    }

if not prediction_parameters["use TSR"]:
    grid_search = GridSearchCV(
        regressor, param_grid, cv=3, n_jobs=-1,
        scoring=prediction_parameters["scoring for regressor
grid search"])

    grid_search.fit(X_train_augmented, y_train_augmented)

    best_parameters = grid_search.best_params_
    for param_name, param_value in best_parameters.items():
        setattr(regressor, param_name, param_value)

    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train_augmented)
    X_test = scaler.transform(X_test)

    regressor.fit(X_train, y_train_augmented)
    predicted_completion_times = regressor.predict(X_test)

```

```

        regression_metrics["mse_values"].append(
            mean_squared_error(y_test, predicted_completion_times))
        regression_metrics["mae_values"].append(
            mean_absolute_error(y_test, predicted_completion_times))
        regression_metrics["r2_values"].append(
            r2_score(y_test, predicted_completion_times))

    completion_times_str += f"""
RUN {i+1}
\tpredicted
{predicted_completion_times.astype(int)}
\treal
{y_test.astype(int).values}
"""

end = time()

with open(f"{current_datetime}_predictions.txt", "a") as f:
    f.write(f"Total execution time - {regressor}:\n \
        {np.round(end - start, 3)} s\n\n")
    f.write(completion_times_str)

feature_matrix["Time"] = completion_time

return classification_metrics, regression_metrics

```

Листинг 11 – регресија – предикција и евалуација времена трајања партије

Следећи део кода је петља која се извршава system_parameters[number of runs] пута. У сваком пролазу петље, следеће се дешава:

random_state се поставља на 42 ако је prediction_parameters[use fixed random state] тачно, у супротном се бира случајни број између 0 и 1000.

Подаци се деле на тренинг и тест скупове користећи функцију train_test_split из модула sklearn.model_selection. Тест скуп је 30% укупних података.

Ако је prediction_parameters[use augmentation for regression] тачно, тренинг подаци се аугментирају додавањем малог шума (0.01 пута стандардна нормална дистрибуција). У супротном, тренинг подаци остају непромењени.

regressor_random_state се поставља на исту вредност као random_state ако је prediction_parameters[use same random state for regressor] тачно, у супротном је постављено на None.

Наредни део кода проверава да ли је use_streamlit тачно. Ако јесте, онда се извршава следећи код:

- Променљива st_regressor се поставља на вредност prediction_parameters[pick regressor].
- Креира се речник combo_map који мапира имена регресора на тројке буловских вредности или стрингова.

- prediction_parameters се ажурира тако што се вредности за кључеве use TSR, use NNR и use GBR постављају на основу мапирања из combo_map за изабрани регресор.

Затим се проверава који регресор је изабран и на основу тога иницијализира одговарајући регресор са датим random_state:

- Ако је prediction_parameters[use TSR] тачно, користи се TSR регресор.
- Ако је prediction_parameters[use NNR] тачно, користи се NNR регресор. Такође, креира се param_grid за претрагу параметара. Овај речник садржи различите вредности за параметре који ће бити искоришћени у процесу оптимизације модела.
- Ако је prediction_parameters[use GBR] тачно, користи се GBR регресор. Слично као и за NNR, креира се param_grid за претрагу параметара.

Ако је prediction_parameters[use TSR] False, извршава се следећи код:

- Креира се објекат GridSearchCV из модула sklearn.model_selection. Овај објекат ће бити коришћен за претрагу најбољих параметара за дати регресор користећи унакрсну валидацију (са 3 фолда). Метрика која се користи за оцену модела је негативна средња квадратна грешка.
- Покреће се претрага најбољих параметара позивом методе fit на објекту grid_search са аугментираним тренинг подацима.
- Најбољи параметри се чувају у променљивој best_params.
- Коначно, сваки од најбољих параметара се поставља за дати регресор користећи функцију setattr.

Затим се прво креира објекат StandardScaler из модула sklearn.preprocessing. Овај објекат ће бити коришћен за стандардизацију података тако да имају средњу вредност 0 и стандардну девијацију 1.

Тренинг подаци се стандардизују позивом методе fit_transform на објекту scaler са аугментираним тренинг подацима. Тест подаци се трансформишу користећи исти scaler.

Регресор се тренира на стандардизованим тренинг подацима позивом методе fit. Предвиђања за времена завршетка се рачунају за тест податке позивом методе predict на обученом регресору.

За свако предвиђање, рачунамо MSE, MAE и R2 скор упоређивањем предвиђених времена завршетка са стварним временима завршетка из тест података. Ове вредности се додају у одговарајуће листе у речнику regression_metrics.

Последњи део кода у овој функцији одрађује следеће:

- За сваки пролаз петље, предвиђена и стварна времена завршетка се додају у стринг completion_times_str.
- Време завршетка петље се рачуна користећи функцију time() из модула time, а време извршавања се рачуна као разлика између времена почетка и времена завршетка.

- Предвиђена и стварна времена завршетка, као и укупно време извршавања, се уписују у текстуалну датотеку.
- Колона Time се враћа у feature_matrix.
- Функција враћа classification_metrics и regression_metrics.

7.10 Функција за чување вредности скоровања

Функција scorings_for_grid_search (Листинг 12) служи да чува све подржане одлучујуће аргументе који се користе код претраживања мреже параметара у циљу проналажења оптималних.

```
def scorings_for_grid_search():
    # INNER
    scorings_for_classifier = (
        "accuracy", "balanced_accuracy", "top_k_accuracy",
        "f1", "f1_macro", "f1_micro", "f1_samples", "f1_weighted",
        "jaccard", "jaccard_macro", "jaccard_micro", "jaccard_samples",
        "jaccard_weighted",
        "neg_brier_score", "neg_log_loss",
        "precision", "average_precision", "precision_macro",
        "precision_micro", "precision_samples",
        "precision_weighted",
        "recall", "recall_macro", "recall_micro", "recall_samples",
        "recall_weighted",
        "roc_auc", "roc_auc_ovo", "roc_auc_ovo_weighted",
        "roc_auc_ovr", "roc_auc_ovr_weighted",
    )
    scorings_for_regressor = (
        "explained_variance", "max_error",
        "neg_mean_absolute_error",
        "neg_mean_absolute_percentage_error",
        "neg_mean_squared_log_error", "neg_mean_poisson_deviance",
        "neg_mean_squared_error", "neg_mean_squared_log_error",
        "neg_median_absolute_error", "neg_root_mean_squared_error",
        "r2",
    )
    return scorings_for_classifier, scorings_for_regressor
```

Листинг 12 – све вредности скорова за класификатор и регресор

7.11 Визуализација добијених метрика

Функција metrics_visualization (Листинг 13) служи за визуализацију метрика класификације и регресије. Она прима два аргумента: classification_metrics и regression_metrics, који су речници са вредностима метрика.

Прво се креира опсег бројева од 1 до броја покретања система. Ова променљива ће вероватно бити коришћена као x-оса на графиконима.

Затим, креирамо фигуру са шест подграфикона (2 реда и 3 колоне) и постављамо јој величину на 12 инча по ширини и 7 инча по висини. Такође, постављамо размак између подграфикона на 0.35 (како хоризонтално, тако и вертикално).

Две помоћне (унутрашње) функције, `plot_1D_metric` и `plot_2D_metric`, су дефинисане за цртање 1D и 2D графика, респективно. Оне примају различите аргументе укључујући податке за цртање, ознаке оса, наслов графика и додатне параметре за прилагођавање изгледа графика.

```
def metrics_visualization(classification_metrics: dict,
                           regression_metrics: dict):
    runs_range = range(1, system_parameters["number of runs"] + 1)

    fig, axs = plt.subplots(nrows=2, ncols=3)
    fig.set_size_inches(w=12, h=7)
    fig.subplots_adjust(wspace=0.35, hspace=0.35)

    def plot_1D_metric(ax, x, y, xlabel, title, color):
        ax.plot(x, y, color=color, linewidth=3)
        ax.set_xlabel(xlabel)
        ax.set_title(title)
        ax.grid(True)
        ax.xaxis.set_major_locator(MaxNLocator(integer=True))

    def plot_2D_metric(ax, data, xlabel, ylabel, title, fmt, cmap):
        heatmap(data, annot=True, fmt=fmt, cmap=cmap, cbar=False,
                ax=ax)
        ax.set_xlabel(xlabel)
        ax.set_ylabel(ylabel)
        if "report" in title:
            ax.yaxis.set_label_coords(-0.2, 0.8)
        ax.set_title(title)

    plot_1D_metric(
        ax=axs[0, 0], x=runs_range,
        y=classification_metrics["accuracies"],
        xlabel="run", title="Accuracy over runs", color="orange")

    plot_2D_metric(
        ax=axs[0, 1],
        data=np.mean(
            classification_metrics["confusion_matrices"], axis=0),
        xlabel="Predicted labels", ylabel="True labels",
        title="Avg confusion matrix", fmt=".2f", cmap="Greens")

    plot_2D_metric(
        ax=axs[0, 2],
        data=calculate_the_average_classification_report(
```

```

        classification_metrics["classification_reports"]),
xlabel="Metrics", ylabel="Classes",
title="Avg classification report", fmt=".2f", cmap="BuPu")

plot_1D_metric(
    ax=axs[1, 0], x=runs_range, y=regression_metrics["mse_values"],
    xlabel="run", title="MSE over runs", color="blue")

plot_1D_metric(
    ax=axs[1, 1], x=runs_range, y=regression_metrics["mae_values"],
    xlabel="run", title="MAE over runs", color="green")

plot_1D_metric(
    ax=axs[1, 2], x=runs_range, y=regression_metrics["r2_values"],
    xlabel="run", title="R2 over runs", color="red")

plt.savefig(f"{current_datetime}__evaluations.png")
plt.show()

```

Листинг 13 – графички приказ свих резултата евалуације

Следећи део кода користи помоћне функције plot_1D_metric и plot_2D_metric да би визуализовао метрике класификације:

- plot_1D_metric – ова функција се користи да би се нацртала прецизност (accuracy) током покретања. x-оса представља број покретања, док у-оса представља вредности прецизности.
- plot_2D_metric – користи се два пута: први пут да би се нацртала матрица конфузије (x-оса представља предвиђене лабеле, док у-оса представља стварне лабеле), а други пут се користи да би се нацртао извештај о класификацији (x-оса представља метрике, док у-оса представља класе).

Наредни део кода користи помоћну функцију plot_1D_metric да би визуализовао метрике регресије. Први пут се користи да би се нацртала MSE током покретања. Други пут се користи да би се нацртала MAE током покретања. Трећи пут се користи да би се нацртао R^2 скор током покретања. За сваки графикон, x-оса представља број покретања, док у-оса представља вредности одговарајућих метрика.

Фигура се чува као PNG датотека – име датотеке је тренутни датум и време, уз пратећи стринг __evaluations.png.

plt.show() – ова команда приказује фигуру у засебном прозору (само при регуларном покретању) .

7.12 За извршавање програма као регуларне скрипте

У оквиру функције if_running_a_regular_script (Листинг 14) се прво дефинише речник pred_parameters са 16 елемената – који садржи различите параметре који се користе за предикцију:

- use non essential game annotations – одлучује да ли ће се користити неесенцијалне играчке анотације.
- use fixed random state – одлучује да ли ће се користити фиксирани случајни стање.
- use stratify – одлучује да ли ће се користити стратификација.
- use scaling and oversampling for classification – одлучује да ли ће се користити скалирање и превише узорковање за класификацију.
- use same random state for classifier – одлучује да ли ће се користити исти случајни стање за класификатор.
- use SVC, use DTC, и use RFC – ови параметри одлучују који класификатор ће се користити.
- use same random state for regressor – одлучује да ли ће се користити исти случајни стање за регресор.
- use augmentation for regression – одучује да ли ће се користити аугментација за регресију.
- use TSR, use NNR, и use GBR – ови параметри одлучују који регресор ће се користити.
- scoring for classifier grid search и scoring for regressor grid search – ови параметри одређују која метрика ће се користити за оцену класификатора и регресора током претраге мрежа.

```

def if_running_a_regular_script():
    # 13 + 2 elemenata
    pred_parameters = {
        "use non essential game annotations" : False,
        "use fixed random state" : False,
        "use stratify" : False,
        "use scaling and oversampling for classification" : False,
        "use same random state for classifier" : False,
        "use SVC" : True,
        "use DTC" : False,
        "use RFC" : False,
        "use same random state for regressor" : False,
        "use augmentation for regression" : False,
        "use TSR" : False,
        "use NNR" : True,
        "use GBR" : False,
        "scoring for classifier grid search" : "accuracy",
        "scoring for regressor grid search" :
        "neg_mean_squared_error",
    }

    # 6 elemenata
    proc_parameters = {
        "csv values: columns to drop" : ["OriginalTimestamp"],
    }

```

```

    "csv values: timestamps column name" : "Timestamp",
    "csv values: pow 1st column name" : "POW.AF3.Theta",
    "csv values: amp 1st column name" : "EEG.AF3",
    "csv intervals: timestamps column name" : "timestamp",
    "game annotations" : ["Time", "Outcome"],
}

# 5 (3) elemenata
sys_parameters = {
    "path" : r"C:\Users\neman\Desktop\master_python\newdata",
    "number of runs" : 3,
    "window length" : 2.0,
    "fs" : 128,
    "files extensions" : {
        "csv values files" : ".md.mc.pm.fe.bp.csv",
        "csv intervals files" : "intervalMarker.csv",
        "json annotations files" : ".json",
    },
}

if pred_parameters["use non essential game annotations"]:
    proc_parameters["game annotations"].extend(
        ["Difficulty", "Fields", "Mines"])

check, errors_text = check_various_conditions(
    pred_parameters, proc_parameters, sys_parameters)

if check:
    return pred_parameters, proc_parameters, sys_parameters, None
else:
    return None, None, None, errors_text

```

Листинг 14 – дефинисање параметара при регуларном покретању скрипте

Даље се у функцији дефинишу још два речника, proc_parameters и sys_parameters, који држе различите параметре за обраду података и системска подешавања, респективно.

proc_parameters садржи шест елемената:

- csv values: columns to drop – одређује које колоне треба избацити из CSV датотека са вредностима.
- csv values: timestamps column name – одређује име колоне која садржи временске ознаке у CSV датотекама са вредностима.
- csv values: pow 1st column name и csv values: amp 1st column name – ови параметри одређују имена првих колона за POW и AMP вредности у CSV датотекама.
- csv intervals: timestamps column name – одређује име колоне која садржи временске ознаке у CSV датотекама са интервалима.

- game annotations – одређује које анотације треба користити из игрица.

sys_parameters је речник са пет елемената:

- path – одређује путању до директоријума где су подаци.
- number of runs – одређује број покретања система.
- window length – одређује дужину прозора за обраду података.
- fs – одређује учестаност узорковања података.
- files extensions – овај параметар је и сам речник који одређује екstenзије датотека за различите типове података.

Наредни део кода проверава да ли ће се користити неесенцијалне играчке анотације. Ако ће, онда се листа proc_parameters[game annotations] проширује са додатним анотацијама: Difficulty, Fields, и Mines.

Затим, функција check_various_conditions се позива са pred_parameters, proc_parameters, и sys_parameters као аргументима. Ова функција враћа две вредности: check и errors_text. check је буловска вредност која указује да ли су сви услови испуњени, док errors_text садржи поруку о грешци ако неки услов није испуњен.

На крају, ако је check тачно, функција враћа pred_parameters, proc_parameters, и sys_parameters, и None. У супротном, враћа None, None, None, и errors_text.

7.13 Провера услова неопходних за извршавање експеримента као регуларне скрипте

Функција check_various_conditions (Листинг 15) је дефинисана са три улазна аргумента: pred_parameters, proc_parameters и sys_parameters.

Функција почиње иницијализацијом стринга failed_vconditions са вредношћу \n\nGreške:\n. Ова променљива ће вероватно бити коришћена да сакупи поруке о грешкама које се јављају током провере различитих услова.

Затим, функција креира две листе, scorings_for_classifier и scorings_for_regressor, које садрже имења метрика које се користе за оцену класификатора и регресора, респективно.

```
def check_various_conditions(pred_parameters: dict,
                             proc_parameters: dict,
                             sys_parameters: dict):
    # INNER
    failed_vconditions = "\n\nGreške:\n"

    scorings_for_classifier, scorings_for_regressor =
    scorings_for_grid_search()

    vcondition_0 = os.path.exists(sys_parameters["path"])

    vcondition_1 = all(
```

```

        [isinstance(value, bool) for key, value in
pred_parameters.items()
            if key not in ["scoring for classifier grid search",
                           "scoring for regressor grid search"]])

vcondition_1a = pred_parameters["scoring for classifier grid
search"] in \
    scorings_for_classifier

vcondition_1b = pred_parameters["scoring for regressor grid
search"] in \
    scorings_for_regressor

vcondition_2 = np.sum([pred_parameters["use SVC"],
                      pred_parameters["use DTC"],
                      pred_parameters["use RFC"]])
) == 1

vcondition_3 = np.sum([pred_parameters["use TSR"],
                      pred_parameters["use NNR"],
                      pred_parameters["use GBR"]])
) == 1

vcondition_4 = True
try:
    if isinstance(int(sys_parameters["number of runs"]), int):
        pass
except:
    vcondition_4 = False

vcondition_5 = sys_parameters["number of runs"] > 0

vcondition_6 = isinstance(sys_parameters["window length"], float)

vcondition_7 = 0.1 <= sys_parameters["window length"] <= 5

vcondition_8 = sys_parameters["fs"] in [2**i for i in range(7, 10 +
1)]

vcondition_9 = True
for key, value in proc_parameters.items():
    if key in ["csv values: columns to drop", "game annotations"]:
        if isinstance(value, list):
            if key == "game annotations" and len(value) == 0:
                vcondition_9 = False
                break
        else:
            vcondition_9 = False
            break
    for list_value in value:

```

```

        outer_break = False
        if not isinstance(list_value, str):
            outer_break = True
            vcondition_9 = False
            break
        if outer_break:
            break
    else:
        if not isinstance(value, str):
            vcondition_9 = False
            break

vcondition_10 = True
for suffix in sys_parameters["files extensions"].values():
    if not isinstance(suffix, str):
        vcondition_10 = False
        break
    elif suffix == sys_parameters["files extensions"]["json annotations files"]:
        if not suffix.endswith(".json"):
            vcondition_10 = False
            break
    elif not suffix.endswith(".csv"):
        vcondition_10 = False
        break

vcondition_11 = (len(pred_parameters) == 16 and
                 len(proc_parameters) == 6 and
                 len(sys_parameters) == 5 and
                 len(sys_parameters["files extensions"]) == 3)

if not vcondition_0:
    failed_vconditions += "- Zadata je nepostojeća putanja.\n"

if not vcondition_1:
    failed_vconditions += """- Parametri koji se koriste u sklopu
predikcija
(osim scoring-a) moraju svi biti definisani kao boolean-i.\n"""

if not vcondition_1a:
    failed_vconditions += """- Metoda za ocenjivanje klasifikatora
pri traženju
optimalnih parametara nije nađena u predefinisanom skupu.\n"""

if not vcondition_1b:
    failed_vconditions += """- Metoda za ocenjivanje regresora pri
traženju
optimalnih parametara nije nađena u predefinisanom skupu.\n"""

if not vcondition_2:

```

```

        failed_vconditions += "- Isključivo jedan klasifikator mora
biti aktivan.\n"

    if not vcondition_3:
        failed_vconditions += "- Isključivo jedan regresor mora biti
aktivan.\n"

    if not vcondition_4:
        failed_vconditions += "- Broj iteracija sistema mora biti
intedžer.\n"

    if not vcondition_5:
        failed_vconditions += "- Broj iteracija sistema mora biti >
1.\n"

    if not vcondition_6:
        failed_vconditions += "- Dužina prozora mora biti float
vrednost.\n"

    if not vcondition_7:
        failed_vconditions += "- Nedozvoljena dužina prozora.\n"

    if not vcondition_8:
        failed_vconditions += "- Nedozvoljena frekvencija
odabiranja.\n"

    if not vcondition_9:

        failed_vconditions += """- Svi parametri koji označavaju imena
kolona
        moraju biti str (sem imena kolona za izbacivanje i anotacija
igre;
        oni moraju biti liste ispunjene str).\n"""

    if not vcondition_10:
        failed_vconditions += "- Sufiksi fajlova nisu pravilno
definisani.\n"

    if not vcondition_11:
        failed_vconditions += "- Rečnici parametara nisu originalne
dužine.\n"

    if failed_vconditions != "\n\nGreške:\n":
        return False, failed_vconditions
    else:
        return True, None

```

Листинг 15 – проверавање вредности параметара при регуларном покретању скрипте

Следећи део кода просто проверава различите услове:

- vcondition_0 услов проверава да ли путања коју сте навели у sys_parameters[path] постоји.
- vcondition_1 услов проверава да ли су све вредности у pred_parameters буловске, осим за scoring for classifier grid search и scoring for regressor grid search.
- vcondition_1a и vcondition_1b услови проверавају да ли је вредност scoring for classifier grid search у листи scorings_for_classifier и да ли је вредност scoring for regressor grid search у листи scorings_for_regressor.
- vcondition_2 и vcondition_3 услови проверавају да ли је тачно само један од параметара use SVC, use DTC, и use RFC, и да ли је тачно само један од параметара use TSR, use NNR, и use GBR.
- Услов vcondition_4 се иницијално поставља на True. Затим, покушава да конвертује вредност sys_parameters[number of runs] у целобројну вредност. Ако конверзија не успе, vcondition_4 се поставља на False.
- vcondition_5 услов проверава да ли је број покретања система већи од 0.
- vcondition_6 услов проверава да ли је дужина прозора децимални број.
- vcondition_7 услов проверава да ли је дужина прозора између 0.1 и 5.0.
- vcondition_8 услов проверава да ли је фреквенција одабирања једнака једној од предефинисаних вредности.
- Услов vcondition_9 се иницијално поставља на True, а затим се проверавају различите услови за вредности у proc_parameters. Ако било који од тих услова није испуњен, vcondition_9 се поставља на False.
- Услов vcondition_10 се иницијално поставља на True, а затим се проверавају различити услови за вредности у sys_parameters[files extensions]. Ако било који од ових услова није испуњен, vcondition_10 се поставља на False.
- vcondition_11 услов проверава да ли речници pred_parameters, proc_parameters и sys_parameters имају тачан (предефинисан) број елемената.

Након тога, само треба проверити путем негације да ли иједан од услова није задовољен. Пролази се редом кроз све претходно наведене, и за сваки ће постојати карактеристичан стринг који ће се исписивати у терминалу.

Ако failed_vconditions није једнако \n\nGreške:\n, то значи да је било грешака. У том случају, функција враћа False и поруку о грешци. У супротном, то значи да није било грешака – у том случају, функција враћа True и None.

7.14 За извршавање програма као Streamlit апликације

Функција if_running_a_streamlit_app (Листинг 16) користи се за подешавање изгледа и структуре Streamlit апликације. Ево шта сваки део кода ради:

- `st.set_page_config(layout="wide")` – подешава конфигурацију странице тако да користи широки изглед.
- `edit_streamlit_style` – садржи CSS стилове који се користе за прилагођавање изгледа апликације. Специфично, она скрива главни мени и подножје, и додаје простор између горњег дела странице и првог дела.
- `st.markdown(body=edit_streamlit_style, unsafe_allow_html=True)` – примењује CSS стилове дефинисане у `edit_streamlit_style`.
- `col_1, _, col_2, col_3 = st.columns(spec=[8, 1, 5, 2])` – креира четири колоне на страницама. Првој колони је додељено 8 јединица ширине, другој 1 јединица ширине (али се не користи), трећој колони је додељено 5 јединица ширине, а четвртој колони је додељено 2 јединице ширине.
- `col_1.text(body=)` и `col_2.text(body=)` – овим линијама се додају празни текстуални блокови у прву и трећу колону.
- `col_1.subheader(body="Pomoćni alat za master tezu")` – додаје поднаслов Помоћни алат за мастер тезу у прву колону.
- `col_2.info(body="Nemanja Peruničić, B1 3/22, icon=□")` – ова линија додаје информациони блок са текстом Немања Перуничић, Б1 3/22 и иконом студентске капе у трећу колону.
- `try: col_3.image(image="ftn_logo_fun.gif") except: pass` – ова линија покушава да дода визуелни елемент (слику) у четврту колону. Ако слика не може бити учитана (нпр. ако фајл не постоји), код ће игнорисати грешку и наставити са извршавањем.

```
def if_running_a_streamlit_app():
    st.set_page_config(layout="centered")

    edit_streamlit_style = """
        <style>
            MainMenu {visibility: hidden;}
            footer {visibility: hidden;}
            #root > div:nth-child(1) > div > div > div > div > \
            section > div {padding-top: 2rem;}
        </style>
    """
    st.markdown(body=edit_streamlit_style, unsafe_allow_html=True)

    col_1, _, col_2, col_3 = st.columns(spec=[8, 1, 5, 2])

    col_1.text(body="")
    col_1.subheader(body="Pomoćni alat za master tezu")

    col_2.text(body="")
    col_2.info(body="Nemanja Peruničić, B1 3/22", icon="▣")

    try:
        col_3.image(image="ftn_logo_fun.gif")
    
```

```

except:
    pass

st.divider()
_, col_a, _, col_b = st.columns(spec=[1, 14, 1, 12])

with col_a:
    scorings_for_classifier, scorings_for_regressor =
scorings_for_grid_search()

pred_parameters = {
    "use fixed random state" : st.checkbox(
        label="use fixed random state",
        ),
    "divider 1" : st.divider(
        ),
    "use stratify" : st.checkbox(
        label="use stratify",
        ),
    "use scaling and oversampling for classification" :
st.checkbox(
        label="use scaling and oversampling for
classification",
        ),
    "use same random state for classifier" : st.checkbox(
        label="use same random state for classifier",
        ),
    "pick classifier" : st.selectbox(
        label="pick classifier",
        options=["use SVC", "use DTC", "use RFC"],
        ),
    "use grid search for classifier": st.checkbox(
        label="use grid search for classifier",
        ),
    "scoring for classifier grid search" : st.selectbox(
        label="pick classifier scoring",
        options=scorings_for_classifier,
        ),
    "divider 2" : st.divider(
        ),
    "use same random state for regressor" : st.checkbox(
        label="use same random state for regressor",
        ),
    "use augmentation for regression" : st.checkbox(
        label="use augmentation for regression",
        ),
    "pick regressor" : st.selectbox(
        label="pick regressor",
        options=["use TSR", "use NNR", "use GBR"],
        ),
}

```

```

    "use grid search for regressor": st.checkbox(
        label="use grid search for regressor",
        ),
    "scoring for regressor grid search" : st.selectbox(
        label="pick regressor scoring",
        options=scorings_for_regressor,
        ),
    "fake divider" : st.text(
        body="",
        ),
    }
try:
    st.image(image="eeg_potato.jpg")
except:
    pass

with col_b:
    proc_parameters = {
        "csv values: columns to drop" : st.multiselect(
            label="csv values: columns to drop",
            options=["OriginalTimestamp"],
            default=["OriginalTimestamp"],
            ),
        "csv values: timestamps column name" : st.selectbox(
            label="csv values: timestamps column name",
            options=["Timestamp"],
            ),
        "csv values: pow 1st column name" : st.selectbox(
            label="csv values: pow 1st column name",
            options=["POW.AF3.Theta"],
            ),
        "csv values: amp 1st column name" : st.selectbox(
            label="csv values: amp 1st column name",
            options=["EEG.AF3"],
            ),
        "csv intervals: timestamps column name" : st.selectbox(
            label="csv intervals: timestamps column name",
            options=[["timestamp"]],
            ),
        "game annotations" : st.multiselect(
            label = "game annotations",
            options=["Time", "Outcome", "Difficulty", "Fields",
            "Mines"],
            default=[["Time", "Outcome"]],
            help = "Ne izostaviti obavezne parametre!"),
        }

    sys_parameters = {
        "path" : st.selectbox(
            label="path",

```

```

        options=["C:/Users/neman/Desktop/master_python/newdata"
    ],
        ),
    "divider 1" : st.divider(
        ),
    "number of runs" : st.slider(
        label="number of runs",
        min_value=1, max_value=50, step=1, value=5,
    ),
    "window length" : st.slider(
        label="window length",
        min_value=0.1, max_value=5.0, step=0.1, value=2.0,
    ),
    "fs" : st.select_slider(
        label="sampling frequency",
        options=[2**i for i in range(7, 10 + 1)], value=2**7,
    ),
    "divider 2" : st.divider(
        ),
    "files extensions" : {
        "csv values files" : st.selectbox(
            label="csv values files",
            options=[ ".md.mc.pm.fe.bp.csv" ],
        ),
        "csv intervals files" : st.selectbox(
            label="csv intervals files",
            options=[ "intervalMarker.csv" ],
        ),
        "json annotations files" : st.selectbox(
            label="json annotations files",
            options=[ ".json" ],
        ),
    },
}
st.divider()

return pred_parameters, proc_parameters, sys_parameters

```

Листинг 16 – дефинисање параметара као Streamlit елемената

- st.divider() – додаје хоризонталну линију (разделник) на страницу.
- _, col_a, _, col_b = st.columns(spec=[1, 14, 1, 12]) – креира четири нове колоне. Друга колона (col_a) има 14 јединица ширине, а четврта колона (col_b) има 12 јединица ширине. Прва и трећа колона су празне и служе као маргине.
- scorings_for_classifier и scorings_for_regressor – ове две листе садрже могуће опције за оцену модела класификације и регресије.

- pred_parameters – овај речник дефинише неколико параметара који се могу подесити у Streamlit интерфејсу. Сваки параметар је представљен као пар клуч-вредност где је клуч име параметра, а вредност је Streamlit вицет који омогућава кориснику да изабере вредност за тај параметар. На пример, use fixed random state : st.checkbox(label=use fixed random state) – додаје потврдни оквир који кориснику омогућава да одабере да ли жели да користи фиксни случајни статус.

У наредном коду се само листају даље елементи pred_parameters.

- scoring for regressor grid search : st.selectbox(label=pick regressor scoring, options=scorings_for_regressor) – додаје изборник који кориснику омогућава да одабере који скоринг метод жели да користи за претрагу мреже регресора. Опције су дефинисане у scorings_for_regressor листи.
- fake divider : st.text(body=) – додаје празан текстуални блок који служи као вештачки разделник.
- try: st.image(image=eeg_potato.jpg) except: pass – покушава да дода визуелни елемент (слику) на страницу. Ако слика не може бити учитана (на пример, ако фајл не постоји), код ће игнорисати грешку и наставити са извршавањем.

У col_b се додаје још опција за конфигурацију обраде CSV података. Корисник може да одабере које колоне да отпадну, име колоне за временске ознаке, и имена првих колона за POW и AMP вредности.

Затим се додаје још неколико опција за конфигурацију обраде CSV података:

- csv intervals: timestamps column name – додаје изборник који кориснику омогућава да одабере име колоне за временске ознаке у CSV интервалима. Једина доступна опција је timestamp.
- game annotations – додаје вишеструки изборник који кориснику омогућава да одабере које играчке анотације жели да укључи. Подразумеване опције су Time и Outcome. Овде се једино у коду јавља и параметар за помоћна упутства кориснику, који се на екрану приказује у виду засеченог кружића са знаком питања уз средини - преласком курсора миша преко кружића ће се појавити дефинисани текст.

Затим, код креира две листе суфикса за CSV и JSON фајлове:

- suffixes = [.md.mc.pm.fe.bp.csv, intervalMarker.csv, .json] – дефинише листу суфикс који се користе у фајловима.
- csv_suffixes је листа која садржи само суфикс који се користе у CSV фајловима.
- json_suffixes је листа која садржи само суфикс који се користе у JSON фајловима.

На крају је приказано још опција за конфигурацију системских параметара, укључујући путању до фајлова, број покретања, дужину прозора, учсталост узорковања и екстензије фајлова. Након што су сви параметри подешени, функција враћа три речника са параметрима за предикцију, обраду и систем.

7.15 Чување вредности свих параметара током извршавања кода

Функција remember_parameters (Листинг 17) комбинује три речника параметара у један речник – ово се постиже коришћењем оператора **, који распакује речник. Затим, функција уклања све ставке из речника чијији клјучеви садрже реч divider.

Након тога, функција претвара combined_dict у DataFrame објекат, који има две колоне: Key и Value.

На крају, функција чува DataFrame у .csv датотеку чије име је датум и време када је функција позвана (представљено променљивом current_datetime), заједно са стрингом __parameters.csv. Параметар index=False спречава да се индекс DataFrame објекта чува у .csv датотеку.

```
def remember_parameters():
    # INNER
    combined_dict = {**prediction_parameters,
                     **processing_parameters,
                     **system_parameters}
    combined_dict = {k: v for k, v in combined_dict.items() if
"divider" not in k}

    df = DataFrame(list(combined_dict.items()), columns=[ 'Key' ,
'Value' ])
    df.to_csv(f"{current_datetime}__parameters.csv", index=False)
```

Листинг 17 – чување свих параметара у CSV-у

7.16 Помоћна функција за хипотетички дуготрајни експеримент

Функција experimental (Листинг 18) генерише листу буловских вредности које представљају све могуће комбинације параметара за предвиђање. Ево шта сваки део кода ради:

- combo_prediction_parameters = [] – иницијализује празну листу у коју ће бити додате комбинације параметара за предвиђање.
- x = f{how_many_parameters:02d}b – форматира стринг тако да представља бинарни број са фиксираним бројем цифара (дужином) који је одређен вредношћу how_many_parameters.
- for i in range(2^how_many_parameters) – пролази кроз сваки број у опсегу од 0 до $2^{how_many_parameters}$.
- binary = format(i, x) – конвертује тренутни број у бинарни формат користећи претходно форматирани стринг x.
- if any(binary[range_classifiers].count('1'), binary[range_regressors].count('1')) == 1 – осигуруја да се узима само оне комбинације где постоји тачно једна '1' у деловима стринга који су одређени са range_classifiers и range_regressors.
- boolean_list = [bool(int(bit)) for bit in binary] – конвертује сваку цифру у бинарном стрингу у буловску вредност и ставља их у листу.

- `combo_prediction_parameters.append(boolean_list)` – додаје тренутну листу буловских вредности у главну листу.

```
def experimental(how_many_parameters, range_classifiers,
range_regressors):
    # INNER
    combo_prediction_parameters = []
    x = f"{how_many_parameters:02d}b"

    for i in range(2^how_many_parameters):
        binary = format(i, x)

        if any(binary[range_classifiers].count('1'),
               binary[range_regressors].count('1')) == 1:

            boolean_list = [bool(int(bit)) for bit in binary]
            combo_prediction_parameters.append(boolean_list)

    return combo_prediction_parameters
```

Листинг 18 – пример генератора свих нама релевантних комбинација boolean параметара

7.17 Покретање анализе и ML алгоритама

Функција `machine_learning_system` (Листинг 19) служи само да позове разне претходно дефинисане функције. У њој је каскадни позив неколико функција, који прво извршава класификацијону предикцију и процену на матрици карактеристика, затим регресиону предикцију и процену на резултатима класификацијоне предикције и процене, и на крају визуализацију метрика.

Испод се налазе и информације о почетним линијама свих функција (унутрашње су дате са табуларним размаком).

```
def machine_learning_system():
    metrics_visualization(
        *regression_prediction_and_evaluation(
            *classification_prediction_and_evaluation(
                creating_the_feature_matrix(
                    *eeg_feature_extraction(
                        *processing(
                            *extract_data(
                                ))))))))

    _ = """
start lines:

    extract_data - 113
    processing - 171
        create_columns_names - 218
        windowing - 234
    eeg_feature_extraction - 273
    creating_the_feature_matrix - 375
```

```
classification_prediction_and_evaluation - 428
    calculate_the_average_classification_report - 551
regression_prediction_and_evaluation - 577
    scorings_for_grid_search - 685
metrics_visualization - 712
```

```
if_running_a_regular_script - 771
    check_various_conditions - 827
if_running_a_streamlit_app - 968
remember_parameters - 1125
"""
```

Листинг 19 – извршавање функција које образују ML алгоритам

7.18 Главно извршавање програма

У овом потпоглављу је последњи, и главни део Python скрипте који се извршава само када се скрипта покрене директно, а не када се увезе као модул (Листинг 20):

- use_streamlit = True – ова променљива је кључна – она одређује да ли ће се користити Streamlit за извршавање скрипте.
- current_datetime = datetime.now().strftime("%y-%m-%d_%H-%M-%S") – ова линија кода добија тренутни датум и време и форматира га у стринг.
- if not use_streamlit – одлучује који део кода ће се извршити на основу вредности променљиве use_streamlit.
- (prediction_parameters, processing_parameters, system_parameters, errors) = if_running_a_regular_script() – ако use_streamlit није True, покреће се стандардна функција – ако постоје грешке, оне се исписују.

```
if __name__ == "__main__":
    # !!!
    use_streamlit = True
    _ = """
    if True >> u terminalu >> "streamlit run nemanja_perunicic_mas.py"

    if False >> proverite vrednosti parametara u funkciji:
        if_running_a_regular_script() >> pa potom standardno Run
dugme/komanda
"""

if not use_streamlit:
    (prediction_parameters,
     processing_parameters,
     system_parameters,
     errors
    ) = if_running_a_regular_script()

    current_datetime = datetime.now().strftime("%y-%m-%d_%H-%M-%S")
```

```

        if errors:
            print(errors)
        else:
            remember_parameters()
            machine_learning_system()

    else:
        (prediction_parameters,
         processing_parameters,
         system_parameters
        ) = if_running_a_streamlit_app()

    status = st.empty()
    start = st.button(
        label="**:green[START THE SYSTEM]** 🚀",
        use_container_width=True,
    )

    if start:
        status.subheader("🕒 🚧 🚧 🚧 🚧 🚧 🔁")

        current_datetime = datetime.now().strftime("%y-%m-%d_%H-%M-%S")
        remember_parameters()
        machine_learning_system()

        status.subheader("🕒 ... gotovo! ➡ Možete skrolovati nadole...")

    try:
        st.image(f"{current_datetime}_evaluations.png")
    except:
        pass
    try:
        with open(f"{current_datetime}_predictions.txt", "r") as f:
            st.divider()
            st.text(body=f.read())
    except:
        pass
    try:
        st.divider()
        st.table(data=read_csv(f"{current_datetime}_feature_matrix.csv"))
    except:
        pass
    try:
        st.divider()

```

```

    st.table(data=read_csv(f"{current_datetime}__parameters
.csv"))
except:
    pass

```

Листинг 20 – покретање програма

Затим следи алтернативни део if __name__ == __main__: блока који се извршава када је use_streamlit постављено на True:

- (prediction_parameters, processing_parameters, system_parameters) = if_running_a_streamlit_app() – ако се користи Streamlit, ова функција се покреће и враћа три вредности.
- remember_parameters() – ова функција се покреће да би сачувала параметре.
- status = st.empty() – креира празан Streamlit виџет који може бити ажуриран касније.
- start = st.button(...) – креира дугме у Streamlit апликацији – када корисник кликне на дугме, вредност start постаје True.
- if start: ... – ако је корисник кликнуо на дугме, тај блок кода се извршава.
- status.subheader() – ажурира претходно креирани празан виџет са новим текстом.
- machine_learning_system() – ова функција се покреће да би извршила главни део система за машинско учење.
- status.subheader – након што је систем за машинско учење завршен, виџет се поново ажурира са новим текстом.

Финални део кода приказује различите резултате који су генерисани током извршавања система за машинско учење:

- st.image(f{current_datetime}__evaluations.png) - ова линија кода покушава да прикаже слику са оценама које је генерисао систем. Ако слика не постоји, изузетак ће бити покренут и игнорисан.
- with open(f{current_datetime}__predictions.txt, r) as f: st.text(body=f.read()) - овај део кода покушава да отвори текстуални фајл са предикцијама које је генерисао систем и да их прикаже. Ако фајл не постоји, изузетак ће бити покренут и игнорисан.
- st.table(data=read_csv(f{current_datetime}__feature_matrix.csv)) - ова линија кода покушава да учита .csv фајл са матрицом карактеристика коју је генерисао систем и да је прикаже као табелу. Ако фајл не постоји, изузетак ће бити покренут и игнорисан. Аналогни поступак се користи и за чување вредности параметара.

Сваки од ових делова кода је обавијен у try/except блок да би се избегло заустављање извршавања кода у случају грешке при отварању фајла.

8. Резултати предикције и евалуације

8.0 Streamlit интерфејс

Streamlit је бесплатна и јавно доступна Python библиотека која омогућава брзо изградњу и дељење веб апликација за ML и науку о подацима. Streamlit је дизајниран за инжењере и омогућава креацију комплекснијих апликација уз релативно мало линија кода.

Ова библиотека је постала популарна због комбинације једноставности и ефикасности; Streamlit омогућава брузу прототипизацију идеја и њихово представљање на начин који је интуитиван и привлачен за корисника. Подржан је и широк спектар Python библиотека.

Сваки елемент на веб страница, било да је то текст, слика или графикон, може се контролисати помоћу Python кода, тј. није неопходно учити JavaScript или CSS да би се направила интерактивна веб апликација. Streamlit такође има функционалност за аутоматско освежавање, тј. након измена у коду, веб страница ће аутоматски повући нове промене (ово наравно не важи за све врсте промена).

Такође је и оптимизован за брзину – користи кеш меморију да би смањио време потребно за извршавање скрипти, што га чини идеалним за рад са великим скуповима података.

Streamlit представља моћан алат који омогућава научницима података и инжењерима машинског учења да деле своје открића на приступачан и ефикасан начин.

У оквиру овог потпоглавља треба споменути и CSS – то је језик за стилске листове који се користи за описивање презентације документа написаног у језику за обележавање као што су HTML или XML (укључујући XML дијалекте, нпр. XHTML). CSS омогућава контролу тачно како ће HTML елементи изгледати у прегледачу, представљајући вашу обележеност користећи било који дизајн који вам се свиђа.

Када се користи уз Streamlit, CSS може да помогне у прилагођавању изгледа веб апликација – омогућавајући далеко детаљније подешавање изгледа елемената; глобално, групно или појединачно. Управљање изгледом веб странице путем CSS-а је било приказано на Листингу 16, у оквиру потпоглавља 7.14.

На Сл. 27 је дат приказ целе веб апликације која се отвара приликом извршавања скрипте када се у терминалу зада команда `streamlit run nemanja_perunicic_mas.py`. Неколико напомена у вези са овим:

- Све што се види на свим сликама у склопу овог целог поглавља, па и на слици испод, је генерирано путем Streamlit команди, приказаних у оквиру потпоглавља 7.14 и 7.18 (разуме се, када је `use_streamlit` True).
- Измењена је линија кода 969 – `st.set_page_config` позива `centered`, уместо `wide`, да би слика могла лепо да стане на Word страницу.
- Лого Факултета Техничких Наука је уживо покретан (.gif).
- Слике се могу позивати и са интернета (овде су у истом директоријуму).

Pomoćni alat za master tezu

Nemanja Peruničić,
B1 3/22



use fixed random state

csv values: columns to drop
OriginalTimestamp... x ✖

use stratify

csv values: timestamps column name
Timestamp ✖

use scaling and oversampling for classification

csv values: pow 1st column name
POW.AF3.Theta ✖

use same random state for classifier

csv values: amp 1st column name
EEG.AF3 ✖

pick classifier
use RFC

pick classifier scoring
accuracy

csv intervals: timestamps column name
timestamp ✖

use same random state for regressor

game annotations
Time x Outcome x
Fields x Difficulty x
Mines x

use augmentation for regression

path
C:/Users/neman/Desktop/maste...

pick regressor
use TSR

number of runs
5 1 50

pick regressor scoring
neg_mean_squared_error

window length
2.00 0.10 5.00

sampling frequency
128 128 1024



csv values files
.md.mc.pm.fe.bp.csv

csv intervals files
intervalMarker.csv

json annotations files
.json

START THE SYSTEM   

Слика 27: Streamlit – изглед апликације за подешавање параметара и покретање експеримента

На Сл. 28 је приказ промена у апликацији када се кликне на старт дугме, а на Сл. 29 је текст који се исписује након завршетка експеримента.



Слика 28: Streamlit – промене на апликацији након клика на старт дугме



Слика 29: Streamlit – испис након успешног завршетка експеримента

У истом директоријуму у ком се налази скрипта је стављен и фолдер .streamlit, који садржи фајл config.toml – у коме се могу дефинисати приказ и понашање веб странице која се отвара. То су предефинисани називи, које Streamlit аутоматски чита при покретању, и путем тог конфигурационог фајла је намештено да се свима аутоматски отвара светла тема (ради поклапања боје позадине слика са бојом позадине странице).

8.1 Анализа резултата предикције и евалуације

Дата апликација је робусна и прилагођена за лаку и разумљиву употребу – и у стању је да брзо генерише доста резултата. Регресија ће и у општем случају имати мању тачност, због природе проблема, па тако и овде – међутим, треба напоменути да је услед употребе доста различитих параметара, тј. динамичких вредности (уместо статичких), постигнут безбедан пут ка добним резултатима – напротив постоји доста комбинација које воде до пожељних евалуација. Класификација је значајно лакша за предвиђање, код ње је могуће доћи до 100 % тачности без икаквог смисленог (тактичког) подешавања параметара).

У току извршавања програма генеришу се четири излаза:

- матрица обележја у виду сачуваног CSV фајла (Сл. 30),
- предвиђена и стварна времена завршетка партије у програмском терминалу,
- списак вредности свих параметара у CSV-у,
- графички приказ евалуационих метрика класификације и регресије (Сл. 31 и 32).

AL	AM	AN	AO	AP	AQ	AR
POW.AF4.BetaH	POW.AF4.Gamma	Difficulty	Fields	Mines	Time	Outcome
0.681882316	0.366978781	10	81	10	30	1
0.599132994	0.349315269	10	81	10	18	1
0.64292025	0.363795538	10	81	10	17	1
...	0.809868705	0.47446508	10	81	10	17
0.816497614	0.447658824	10	81	10	31	1
0.663769999	0.37987001	100	256	40	97	1
0.614998133	0.332608611	100	256	40	87	1
...	0.613079596	0.339095736	100	256	40	93
0.647201201	0.34973459	100	256	40	71	1
0.634315721	0.297583607	100	256	40	97	0
0.62227209	0.29575819	1000	480	99	155	0
0.821651938	0.4809895	1000	480	99	293	1
0.593723486	0.308242627	1000	480	99	107	0

Слика 30: Приказ последњих 7 колона из матрице обележја сачуване у CSV формату

На пример, можемо посматрати четврту итерацију извршавања регресије за случај када је коришћен GBR алгоритам. Видимо нагло погоршање свих метрика (нпр. пад R^2 у негативу), а све то је последица огромне разлике између предвиђеног и стварног времена завршетка за једну од Advanced партија (види се и на претходној слици; за преко 200 секунди је промашио):

- Предвиђена времена: [67.7, 105.3, 36.9, 84.4]
- Стварна времена: [293.0, 97.0, 30.0, 71.0]

Графици који се тичу класификације који су приказани на Сл. 31 су добијени без употребе стандардизације и балансирања података – укључивањем било ког од та два у игру добићемо 100%-ни учинак класификације, што свакако није реално.

У нашем случају, класификациони модел је тестиран на скупу од 4 инстанце, од којих је једна припадала класи 0, а три класи 1. Ова матрица показује да модел није успео да тачно класификује инстанцу која припада класи 0 (постоји један FP резултат), али је тачно класификовао све инстанце које припадају класи 1.

Класификациони извештај показује да модел има прецизност од 75% за класу 1, што значи да је од свих инстанци које је модел класификовао као класу 1, 75% су биле тачне (нема лажно позитивних). Међутим, за класу 0, прецизност је 0%, што значи да модел није успео да тачно класификује ниједну инстанцу ове класе.

Повратна информација (енг. рекалл) за класу 1 је 100%, што значи да је модел успео да идентификује све инстанце ове класе (нема лажно негативних). За класу 0, повратна информација је 0%, што значи да модел није успео да идентификује ниједну инстанцу ове класе.

F1 скор је хармонијски просек прецизности и поврата информација, и даје јединствену меру перформанси модела за сваку класу. У овом случају, F1 скор за класу 1 је 86%, док је за класу 0 0%.

Укупна тачност модела, која се израчунава као број тачно класификованих инстанци подељен са укупним бројем инстанци, износи 75%.

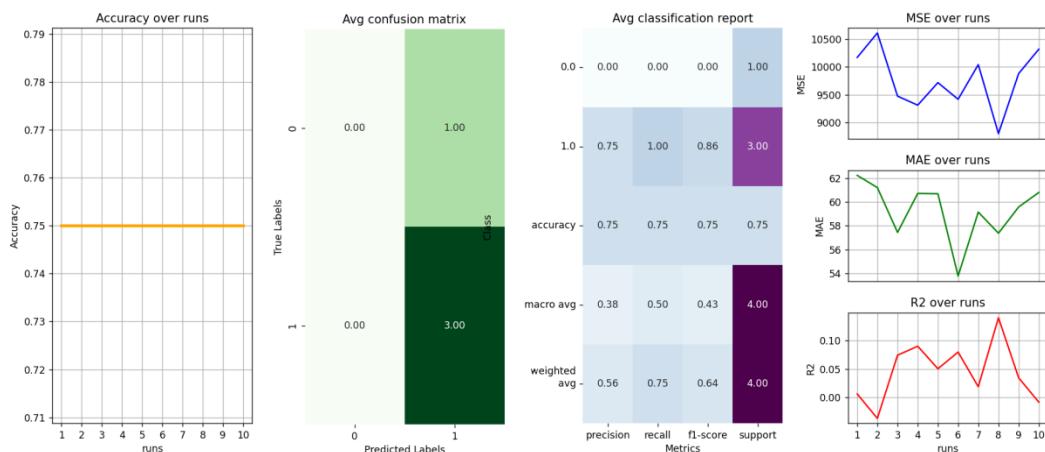
Макро просечне вредности се израчунавају као непондерисани просек вредности за сваку класу, док се пондерисане просечне вредности израчунавају тако што се

вредности за сваку класу помноже са бројем инстанци те класе пре него што се израчуна просек.

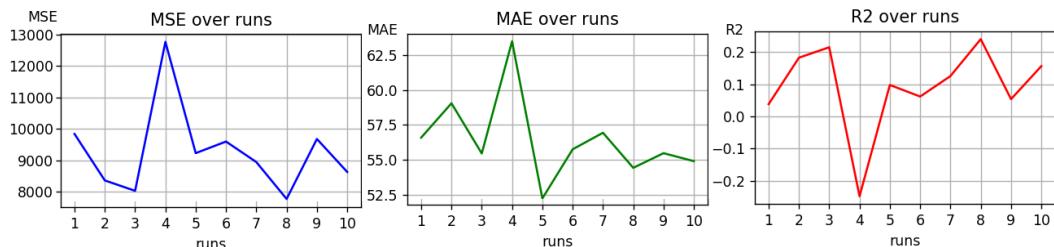
На основу ових резултата, може се закључити да модел има добре перформансе при класификацији инстанце класе 1, али не успева да тачно класификује инстанце класе 0 (у нашем случају је то мањински заступљена класа – а притом се по правилу јавља баш код дужих партија).

За разлику од класификације, код регресије су резултати далеко разумљиви и сва три нам у суштини говоре исту исту. R² ће увек ићи од негативне бесконачности до 1 (вредност 1 би била идеална), док вредности MSE и MAE зависе од конкретних експеримената. У нашем случају постојала су два проблема:

- Мали скуп података за обуку (и тестирање).
- Неуравнотеженост међу подацима – што се највише види код 12-те партије, тј. 12-тог реда у матрици обележја – а и код небалансираних класа.



Слика 31: (крајње лево) тачност класификације током 10 епоха | (центар-лево) усредњена матрица конфузије кроз 10 епоха | (центар-десно) усредњени класификациони извештај кроз 10 епоха | (крајње десно) израчунати MSE, MAE и R2 током 10 епоха приликом употребе TSR алгоритма

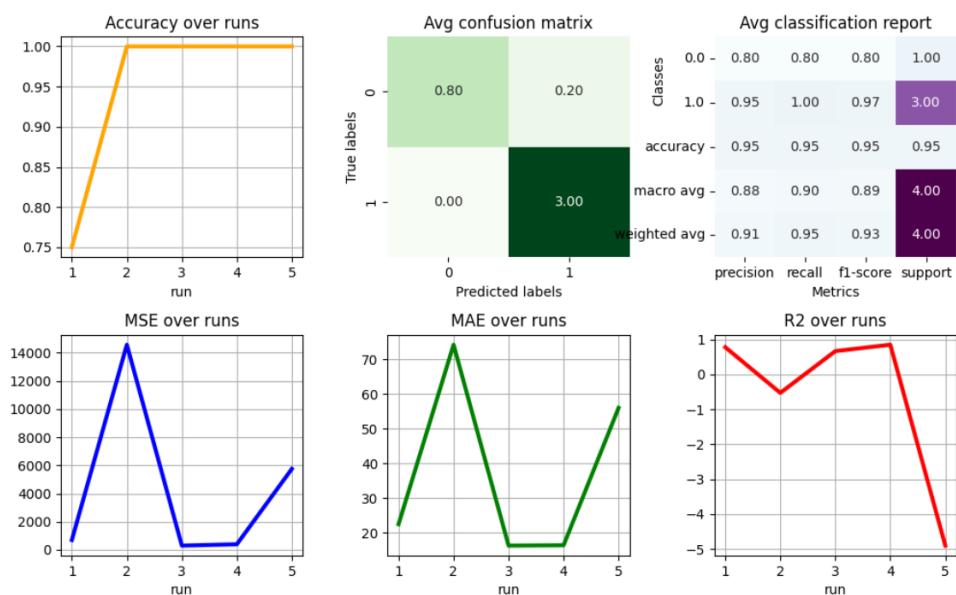


Слика 32: Израчунати MSE, MAE и R2 током 10 епоха приликом употребе GBR алгоритма

Са друге стране – употребом наредне комбинације параметара се дошло до R² скора који је био близу 1 (Сл. 33):

- use fixed random state False
- use stratify False
- use scaling and oversampling for classification True
- use same random state for classifier False

- pick classifier use SVC
- use grid search for classifier True
- scoring for classifier grid search accuracy
- use same random state for regressor False
- use augmentation for regression True
- pick regressor use NNR
- use grid search for regressor False
- scoring for regressor grid search explained_variance
- csv values: columns to drop ['OriginalTimestamp']
- csv values: timestamps column name Timestamp
- csv values: pow 1st column name POW.AF3.Theta
- csv values: amp 1st column name EEG.AF3
- csv intervals: timestamps column name timestamp
- game annotations ['Time', 'Outcome', 'Fields']
- path C:/Users/neman/Desktop/master_python/newdata
- number of runs 5
- window length 1.9
- fs 128
- files extensions {'csv values files': '.md.mc.pm.fe.bp.csv', 'csv intervals files': 'intervalMarker.csv', 'json annotations files': '.json'}



Слика 33: Евалуација кроз 5 епоха – SVC и NNR са наизглед лошем подешеним параметрима

9. Закључак

Истраживање и развој индустрије видео игрица континуирано напредује, а истраживање путем EEG-а доприноси том развоју. Кроз EEG студије, девелопери могу боље разумети како играчи интерагују са игрицама, пружајући вредне податке за будући дизајн игрица и иновације. Такође, истраживање путем EEG-а може довести до развоја нових играчких искустава, замућујући границе између забаве и научног истраживања. EEG сигнали су пронашли кључну улогу у индустрији видео игрица, трансформишући начин на који се игрице дизајнирају и доживљавају – искориштавањем моћи EEG технологије, може се стварати више урођених, ангажованијих и прилагодљивих искустава играња, што на крају побољшава задовољство играча и отвара врата узбудљивим новим могућностима у интерактивној забави.

Анализа EEG сигнала пружила је драгоцене увиде у свет видео игрица, обогаћујући наше разумевање ангажованости играча, когнитивног оптерећења и потенцијала терапеутских искустава играња.

Искориштавањем EEG технологије, истраживачи и девелопери игрица настављају да гурају границе дизајна игрица и интеракције са играчима, коначно побољшавајући искуство играња на различите начине (нпр. на Сл. 34 испод видимо напредну контролу аватара у сложеној, модерној игрици).



Слика 34: Контрола аватара у игрици *Elden Ring* путем EEG-а

Конкретно – анализа EEG сигнала током играња Миноловца, са циљем предвиђања исхода игрице и времена завршетка користећи ML, има велики значај у областима неуронауке, когнитивне психологије и интеракције човека и рачунара.

Ово истраживање пружа јединствени прозор у когнитивне процесе у игри када се појединци баве изазовним менталним задацима као што је Миноловац. Она баца светло на то како мозак реагује на сложене сценарије решавања проблема и управља њима, откривајући неуронске основе доношења одлука, памћења и просторног расуђивања. Способност предвиђања исхода игрице и времена завршетка показује практичне примене разумевања когнитивних процеса током

игрице. То знање се може проширити на домене у којима је брзо и ефикасно доношење одлука кључно, као што су реаговање у ванредним ситуацијама, логистика и стратешко планирање.

Дешифровањем неуронских маркера повезаних са перформансама Миноловца, ово истраживање може омогућити развој персонализованих система повратних информација. Ти системи могу помоћи појединцима да унапреде своје вештине решавања проблема нудећи увид у своје когнитивне процесе у реалном времену, потенцијално револуционирајући приступе образовању и обуци.

Разумевање начина на који мозак ступа у интеракцију са рачунарским интерфејсом током игрице је од суштинског значаја за дизајнирање интуитивнијег софтвера који је прилагођен кориснику. Увиди стечени из EEG анализе могу да усмере развој адаптивних интерфејса који одговарају на когнитивно стање корисника, на крају побољшавајући корисничко искуство и продуктивност.

Коришћењем ML, предвиђање исхода Миноловца и времена завршетка може постати прецизније и прилагодљивије. Комбинација EEG података са другим релевантним информацијама, избор одговарајућих карактеристика и пажљив избор алгоритама за ML могу резултирати моделима који пружају вредан увид у перформансе играча и когнитивне процесе током игрице. ML обезбеђује објективност, прилагодљивост и аутоматизацију, омогућавајући истраживачима и програмерима да стекну вредне увиде и потенцијално побољшају искуство играња. Иако изазови постоје, позитивни аспекти ML у овом контексту обећавају да ће унапредити наше разумевање когнитивних процеса током игрице.

Успешна предикција валидира EEG као поуздан алат за праћење когнитивних активности, што аутоматски јача основу за коришћење EEG-а у различитим апликацијама, укључујући напредне BCI системе, процену менталног здравља и рехабилитацију.

Укратко, анализа EEG сигнала током игрице Миноловац је вредан подухват који продубљује наше разумевање људске спознаје, доношења одлука и решавања проблема. Нуди практичан увид са потенцијалом да унапреди различите аспекте интеракције човека и рачунара, образовања, па чак и здравствене заштите, постављајући преседан за одговорно истраживање и примену EEG технологије у будућности.

Литература

- [1] Jihyeon Ha, Wanjoo Park, Sang In Park, Chang-Hwan Im, Laehyun Kim, EEG response to game-craving according to personal preference for games, *Social Cognitive and Affective Neuroscience*, Volume 16, Issue 9, October 2021, Pages 995–1005, <https://doi.org/10.1093/scan/nsaa131>
- [2] Choi E, Shin SH, Ryu JK, Jung KI, Kim SY, Park MH. Commercial video games and cognitive functions: video game genres and modulating factors of cognitive enhancement. *Behav Brain Funct.* 2020 Feb 3;16(1):2. doi: 10.1186/s12993-020-0165-z. PMID: 32014027; PMCID: PMC6996164.
- [3] L. Cabañero-Gómez, R. Hervas, J. Bravo, and L. Rodriguez-Benitez, ‘Computational EEG Analysis Techniques When Playing Video Games: A Systematic Review’, Proceedings, vol. 2, no. 19, 2018.
- [4] Liao, Lun-De & Chen, Chi-Yu & Wang, I-Jan & Chen, Sheng-Fu & Li, Shih-Yu & Chen, Bo-Wei & Chang, Jyh-Yeong & Lin, Chin-Teng. (2012). Gaming control using a wearable and wireless EEG-based brain-computer interface device with novel dry foam-based sensors. *Journal of neuroengineering and rehabilitation*. 9. 5. 10.1186/1743-0003-9-5.
- [5] Chen, D., James, J., Bao, F.S., Ling, C., Fan, T. (2016). Relationship Between Video Game Events and Player Emotion Based on EEG. In: Kurosu, M. (eds) Human-Computer Interaction. Novel User Experiences. HCI 2016. Lecture Notes in Computer Science(), vol 9733. Springer, Cham. https://doi.org/10.1007/978-3-319-39513-5_35
- [6] Ren, Shen & Babiloni, Fabio & Thakor, Nitish & Bezerianos, Anastasios. (2016). Real-Time Workload Assessment Using EEG Signals in Virtual Reality Environment. 10.1007/978-3-319-32703-7_259.
- [7] L. Wang, X. Ding, W. Zhang and S. Yang, Differences in EEG Microstate Induced by Gaming: A Comparison Between the Gaming Disorder Individual, Recreational Game Users and Healthy Controls, in *IEEE Access*, vol. 9, pp. 32549-32558, 2021, doi: 10.1109/ACCESS.2021.3060112.
- [8] Yang K, Tong L, Shu J, Zhuang N, Yan B, Zeng Y. High Gamma Band EEG Closely Related to Emotion: Evidence From Functional Network. *Front Hum Neurosci.* 2020 Mar 24;14:89. doi: 10.3389/fnhum.2020.00089. PMID: 32265674; PMCID: PMC7107011.
- [9] B. Kerous, F. Skola, and F. Liarokapis, ‘EEG-Based BCI and Video Games: A Progress Report’, *Virtual Real.*, vol. 22, no. 2, pp. 119–135, Jun. 2018.
- [10] R. H. C. e. Souza and E. L. M. Naves, ‘Attention Detection in Virtual Environments Using EEG Signals: A Scoping Review’, *Frontiers in Physiology*, vol. 12, 2021.
- [11] Becerra, D.J. (2015). Algorithmic Approaches to Playing Миноловац.
- [12] Sinha, Y., Malviya, P., & Nayak, R.K. (2021). Fast constraint satisfaction problem and learning-based algorithm for solving Миноловац. ArXiv, abs/2105.04120.

- [13] <https://www.theguardian.com/technology/2016/mar/09/google-deepmind-alphago-ai-defeats-human-lee-sedol-first-game-go-contest> (pristupljeno 30.08.2023.)
- [14] Amin HU, Mumtaz W, Subhani AR, Saad MNM, Malik AS. Classification of EEG Signals Based on Pattern Recognition Approach. *Front Comput Neurosci.* 2017 Nov 21;11:103. doi: 10.3389/fncom.2017.00103. PMID: 29209190; PMCID: PMC5702353.
- [15] Hafeez T, Umar Saeed SM, Arsalan A, Anwar SM, Ashraf MU, Alsubhi K. EEG in game user analysis: A framework for expertise classification during gameplay. *PLoS One.* 2021 Jun 18;16(6):e0246913. doi: 10.1371/journal.pone.0246913. PMID: 34143774; PMCID: PMC8213131.
- [16] Akhmedov, K., & Phan, A. (2021). Machine learning models for DOTA 2 outcomes prediction. *ArXiv*, abs/2106.01782.

Додатак – Листинг целиковног кода

```
_ = """
Python ver (collections, datetime, os, time, warnings) 3.10.11
pip freeze:

altair==5.1.2
attrs==23.1.0
blinker==1.6.3
cachetools==5.3.1
certifi==2023.7.22
cftime==1.6.2
charset-normalizer==3.2.0
click==8.1.7
colorama==0.4.6
contourpy==1.1.0
cycler==0.11.0
decorator==5.1.1
fonttools==4.41.1
gitdb==4.0.10
GitPython==3.1.37
idna==3.4
imbalanced-learn==0.11.0
importlib-metadata==6.8.0
Jinja2==3.1.2
joblib==1.3.1
jsonschema==4.19.1
jsonschema-specifications==2023.7.1
kiwisolver==1.4.4
lxml==4.9.3
markdown-it-py==3.0.0
MarkupSafe==2.1.3
matplotlib==3.7.2
mdurl==0.1.2
mne==1.4.2
mne-connectivity==0.5.0
netCDF4==1.6.4
numpy==1.22.1
packaging==23.1
pandas==2.0.3
Pillow==10.0.0
platformdirs==3.9.1
polib==1.2.0
pooch==1.7.0
protobuf==4.24.4
pyarrow==13.0.0
pydeck==0.8.1b0
Pygments==2.16.1
pyparsing==3.0.9
```

```

python-dateutil==2.8.2
python-docx==0.8.11
pytz==2023.3
referencing==0.30.2
requests==2.31.0
rich==13.6.0
rpds-py==0.10.4
scikit-learn==1.3.0
scipy==1.11.1
seaborn==0.12.2
six==1.16.0
smmap==5.0.1
streamlit==1.27.2
tenacity==8.2.3
threadpoolctl==3.2.0
toml==0.10.2
toolz==0.12.0
tornado==6.3.3
tqdm==4.65.0
typing_extensions==4.8.0
tzdata==2023.3
tzlocal==5.1
urllib3==2.0.4
validators==0.22.0
watchdog==3.0.0
xarray==2023.7.0
zipp==3.17.0
"""

import os, json
from time import time
from datetime import datetime
from collections import Counter
from warnings import simplefilter

import numpy as np
from pandas import read_csv, DataFrame

from scipy.signal import correlate2d
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import(
    accuracy_score, confusion_matrix, classification_report,
    mean_squared_error, mean_absolute_error, r2_score)

from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier as DTC
from sklearn.ensemble import RandomForestClassifier as RFC

from sklearn.linear_model import TheilSenRegressor as TSR
from sklearn.neural_network import MLPRegressor as NNR

```

```

from sklearn.ensemble import GradientBoostingRegressor as GBR

from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE

import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
from seaborn import heatmap

import streamlit as st

simplefilter("ignore")
os.environ["PYTHONWARNINGS"] = "ignore"

def extract_data():

    csv_values_data = []
    csv_intervals_data = []
    json_annotations_data = []

    directory_path = system_parameters["path"]

    files_lists = {key: [] for key in system_parameters["files_extensions"]}

    for key, extension in system_parameters["files_extensions"].items():
        files_lists[key] = sorted(
            [file for file in os.listdir(directory_path)
             if file.endswith(extension)],
            )

    for i in range(len(files_lists["csv values files"])):

        df_values = read_csv(
            os.path.join(directory_path,
                        files_lists["csv values files"][i]),
            skiprows=1,
            )

        columns_to_drop = processing_parameters["csv values: columns to drop"]

        for j in range(len(columns_to_drop)):
            if columns_to_drop[j] in df_values.columns:
                df_values = df_values.drop(
                    columns_to_drop[j],
                    axis=1,
                    )
        csv_values_data.append(df_values)

    df_intervals = read_csv(

```

```

        os.path.join(directory_path,
                      files_lists["csv intervals files"][i]),
                      )
    csv_intervals_data.append(
        df_intervals[processing_parameters[
            "csv intervals: timestamps column name"]].tolist()
    )

    with open(os.path.join(
        directory_path,
        files_lists["json annotations files"][i]),
        "r") as json_file:

        json_data = json.load(json_file)

        json_annotations_data.append(
            {key: json_data.get(key, "")}
            for key in processing_parameters["game annotations"],
        )

    return csv_values_data, csv_intervals_data, json_annotations_data
}

def processing(csv_values_data: list,
               csv_intervals_data: list,
               json_annotations_data: list):

    eeg_amplitude = []
    eeg_power = []

    for i in range(len(csv_values_data)):
        current_df = csv_values_data[i]

        timestamps_column_index = current_df.columns.get_loc(
            processing_parameters["csv values: timestamps column name"],
            )
        amp_1st_column_index = current_df.columns.get_loc(
            processing_parameters["csv values: amp 1st column name"],
            )
        pow_1st_column_index = current_df.columns.get_loc(
            processing_parameters["csv values: pow 1st column name"],
            )

        timestamps = current_df.iloc[:, timestamps_column_index].values

        indexes = np.where(
            (timestamps > csv_intervals_data[i][0]) &
            (timestamps < csv_intervals_data[i][1])
            )[0]

```

```

        eeg_amplitude.append(
            current_df.iloc[:, amp_1st_column_index : amp_1st_column_index + 5].
            values[indexes],
        )
        eeg_power.append(
            current_df.iloc[:, pow_1st_column_index:].
            values[indexes],
        )

    column_names = create_columns_names(
        current_df,
        amp_1st_column_index,
        pow_1st_column_index,
    )

    eeg_amplitude_w, eeg_power_w = windowing(eeg_amplitude, eeg_power)

    return eeg_amplitude_w, eeg_power_w, json_annotations_data, column_names

def create_columns_names(current_df: DataFrame,
                        amp_1st_column_index: int,
                        pow_1st_column_index: int):
    # INNER
    columns_names = current_df.columns.tolist()

    amp_names = columns_names[amp_1st_column_index : amp_1st_column_index + 5]

    columns_names = [f"{j}.{stat}" for stat in ['mean', 'var']
                    for j in amp_names + ["Corr", "Xcov"]] \
                    + columns_names[pow_1st_column_index : ] \
                    + processing_parameters["game annotations"]

    return columns_names

def windowing(eeg_amplitude: list,
             eeg_power: list):
    # INNER
    samples_per_window = np.round(
        system_parameters["window length"] *
        system_parameters["fs"],
    ).astype(int)

    eeg_amplitude_w = []
    eeg_power_w = []

    for i in range(len(eeg_amplitude)):
        eeg_amplitude_i = eeg_amplitude[i]
        eeg_power_i = eeg_power[i]

```

```

eeg_amplitude_i_len = len(eeg_amplitude_i)

eeg_amplitude_i_w = []
eeg_power_i_w = []

for j in range(0, eeg_amplitude_i_len, samples_per_window):
    if j + samples_per_window <= eeg_amplitude_i_len:
        eeg_amplitude_i_w.append(
            eeg_amplitude_i[j:j + samples_per_window],
        )
        eeg_power_i_w.append(
            eeg_power_i[j:j + samples_per_window],
        )

eeg_amplitude_w.append(
    np.array(eeg_amplitude_i_w),
)
eeg_power_w.append(
    np.array(eeg_power_i_w),
)

return eeg_amplitude_w, eeg_power_w

```

```

def eeg_feature_extraction(eeg_amplitude_w: list,
                           eeg_power_w: list,
                           json_annotations_data: list,
                           column_names: list):
    eeg_amplitude_means = []
    eeg_amplitude_vars = []

    eeg_amplitude_corr_means = []
    eeg_amplitude_corr_vars = []
    eeg_amplitude_xcov_means = []
    eeg_amplitude_xcov_vars = []

    eeg_power_means = []

    for i in range(len(eeg_amplitude_w)):
        eeg_amplitude_w_i = eeg_amplitude_w[i]
        eeg_amplitude_w_i_len = len(eeg_amplitude_w_i)

        eeg_amplitude_means_i = []
        eeg_amplitude_vars_i = []

        eeg_power_w_i = eeg_power_w[i]
        eeg_power_means_i = []

        correlation_matrix = np.zeros(

```

```

shape=(eeg_amplitude_w_i_len, eeg_amplitude_w_i_len),
)

cross_covariance_matrix = np.zeros_like(
    a=correlation_matrix,
)

for j in range(eeg_amplitude_w_i_len):
    current_window_amp = eeg_amplitude_w_i[j]

    eeg_amplitude_means_i.append(
        np.mean(current_window_amp, axis=0),
    )

    eeg_amplitude_vars_i.append(
        np.var(current_window_amp, axis=0),
    )

    current_window_pow = eeg_power_w_i[j]

    current_window_pow = np.mean(
        current_window_pow[
            ~np.isnan(current_window_pow[:, 0])],
        axis=0,
    )

    eeg_power_means_i.append(current_window_pow)

    for k in range(eeg_amplitude_w_i_len):
        correlation_matrix[j, k] = np.corrcoef(
            eeg_amplitude_w_i[j].flatten(),
            eeg_amplitude_w_i[k].flatten(),
        )[0, 1]

        if j != k:
            cross_covariance_matrix[j, k] = correlate2d(
                eeg_amplitude_w_i[j],
                eeg_amplitude_w_i[k],
                mode="valid",
            )[0, 0]

eeg_amplitude_means.append(
    np.mean(np.array(eeg_amplitude_means_i), axis=0),
)
eeg_amplitude_vars.append(
    np.mean(np.array(eeg_amplitude_vars_i), axis=0),
)
eeg_amplitude_corr_means.append(
    np.mean(correlation_matrix),
)

```

```

        eeg_amplitude_corr_vars.append(
            np.var(correlation_matrix),
        )
        eeg_amplitude_xcov_means.append(
            np.mean(cross_covariance_matrix),
        )
        eeg_amplitude_xcov_vars.append(
            np.var(cross_covariance_matrix),
        )
        eeg_power_means.append(
            np.mean(np.array(eeg_power_means_i), axis=0),
        )

eeg_features = np.hstack(
    tup=(np.array(eeg_amplitude_means),
        np.array(eeg_amplitude_vars),
        np.array(eeg_amplitude_corr_means).reshape(-1, 1),
        np.array(eeg_amplitude_corr_vars).reshape(-1, 1),
        (np.array(eeg_amplitude_xcov_means) / 1e10).reshape(-1, 1),
        (np.array(eeg_amplitude_xcov_vars) / 1e19).reshape(-1, 1),
        np.array(eeg_power_means),
    ),
)
)

return eeg_features, json_annotations_data, column_names

def creating_the_feature_matrix(eeg_features: np.ndarray,
                                game_annotations: list,
                                column_names: list):

    game_features = {key: [] for key in game_annotations[0].keys()}

    for game_annotation in game_annotations:
        for key, values in game_annotation.items():
            game_features[key].append(values)

    time_in_sec = []
    for time_str in game_features["Time"]:
        minutes, seconds = map(int, time_str.split(":"))
        time_in_sec.append(minutes * 60 + seconds)
    game_features["Time"] = time_in_sec

    outcome_mapping = {"Win": 1, "Lose": 0}
    game_features["Outcome"] = [outcome_mapping[outcome]
                               for outcome in game_features["Outcome"]
                               ]
    ]

difficulty_mapping = {"Easy": 10, "Normal": 100, "Advanced": 1000}
keys_to_process = ["Difficulty", "Fields", "Mines"]

```

```

if use_streamlit or \
    prediction_parameters["use non essential game annotations"]:
    for key in keys_to_process:
        if key in game_features:
            if key == "Difficulty":
                game_features[key] = [difficulty_mapping[difficulty]
                                      for difficulty in game_features[key]]
            else:
                game_features[key] = [int(value) for value in
                                      game_features[key]]

for key, values in game_features.items():
    game_features[key] = np.array(values).reshape(-1, 1)

game_features = np.hstack(
    tup=list(game_features.values()),
    )

feature_matrix = DataFrame(
    data=np.hstack(
        tup=(eeg_features, game_features)),
    columns=column_names,
    )

feature_matrix.to_csv(f"{current_datetime}_feature_matrix.csv", index=False)

return feature_matrix


def classification_prediction_and_evaluation(feature_matrix: DataFrame):
    game_outcomes = feature_matrix.pop("Outcome")

    classification_metrics = {
        "accuracies": [],
        "confusion_matrices": [],
        "classification_reports": [],
    }

    outcomes_str = "GAME OUTCOMES:\n\n"

    start = time()
    for i in range(system_parameters["number of runs"]):
        while True:
            random_state = 42 if prediction_parameters["use fixed random state"] \
                else np.random.randint(1000)

            X_train, X_test, y_train, y_test = train_test_split(
                feature_matrix.values, game_outcomes,

```

```

    test_size=0.3, random_state=random_state,
    stratify=game_outcomes if prediction_parameters["use stratify"] \
        else None,
    )

class_counts = Counter(y_train)

if all(count >= 2 for count in class_counts.values()) \
    and len(set(y_test)) >= 2 and len(set(y_train)) >= 2:
    break

if prediction_parameters["use scaling and oversampling for
classification"]:
    scaler = StandardScaler()
    X_train= scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    smote = SMOTE(
        sampling_strategy="auto", random_state=random_state,
        k_neighbors=1, n_jobs=-1)
    X_train, y_train = smote.fit_resample(X_train, y_train)

classifier_random_state = random_state \
    if prediction_parameters["use same random state for classifier"] \
    else None

if use_streamlit:
    st_classifier = prediction_parameters["pick classifier"]
    combo_map = {
        "use SVC": ("use SVC", False, False),
        "use DTC": (False, "use DTC", False),
        "use RFC": (False, False, "use RFC"),
    }
    prediction_parameters.update(
        dict(zip(["use SVC", "use DTC", "use RFC"],
                combo_map[st_classifier])))
else:
    param_grid = {
        "C": [0.1, 1],
        "kernel": ["linear", "rbf"],
        "gamma": ["scale", "auto", 0.1],
        "class_weight": [None, "balanced"],
    }

```

```

        "min_samples_split": [2, 4],
        "min_samples_leaf": [1, 2],
        "max_features": ["auto", "sqrt"],
    }
    if prediction_parameters["use DTC"]:
        classifier = DTC(random_state=classifier_random_state)

    elif prediction_parameters["use RFC"]:
        classifier = RFC(random_state=classifier_random_state)
        param_grid["n_estimators"]:[100, 200]

    grid_search = GridSearchCV(
        classifier, param_grid, cv=3, n_jobs=-1,
        scoring=prediction_parameters["scoring for classifier grid search"],
    )

    grid_search.fit(X_train, y_train)
    best_parameters = grid_search.best_params_

    for param_name, param_value in best_parameters.items():
        setattr(classifier, param_name, param_value)

    classifier.fit(X_train, y_train)
    predicted_outcomes = classifier.predict(X_test)

    classification_metrics["accuracies"].append(
        accuracy_score(y_test, predicted_outcomes),
    )

    classification_metrics["confusion_matrices"].append(
        confusion_matrix(y_test, predicted_outcomes),
    )

    classification_metrics["classification_reports"].append(
        classification_report(y_test, predicted_outcomes, output_dict=True),
    )

    outcomes_str += f"""
RUN {i+1}
\tpredicted
{np.where(predicted_outcomes == 1.0, "Win", "Lose")}
\treal
{np.where(y_test == 1.0, "Win", "Lose")}
"""

end = time()

with open(f"{current_datetime}_predictions.txt", "w") as f:
    f.write(f"Total execution time - {classifier}:\n \
{np.round(end - start, 3)} s\n\n")
    f.write(outcomes_str + "\n\n")

```

```

feature_matrix["Outcome"] = game_outcomes

return feature_matrix, classification_metrics

def calculate_the_average_classification_report(classification_reports):
    # INNER
    average_classification_report = {}

    for report in classification_reports:
        for key, value in report.items():
            if key not in average_classification_report:
                average_classification_report[key] = value
            else:
                if isinstance(value, dict):
                    for inner_key, inner_value in value.items():
                        average_classification_report[key][inner_key] += \
                            inner_value
                else:
                    average_classification_report[key] += value

    for key, value in average_classification_report.items():
        if isinstance(value, dict):
            for inner_key, inner_value in value.items():
                average_classification_report[key][inner_key] /= \
                    len(classification_reports)
        else:
            average_classification_report[key] /= len(classification_reports)

    return DataFrame(average_classification_report).transpose()

def regression_prediction_and_evaluation(feature_matrix: DataFrame,
                                         classification_metrics: dict):
    completion_time = feature_matrix.pop("Time")

    regression_metrics = {
        "mse_values": [],
        "mae_values": [],
        "r2_values": []
    }

    completion_times_str = "COMPLETION TIMES:\n\n"

    start = time()
    for i in range(system_parameters["number of runs"]):
        random_state = 42 if prediction_parameters["use fixed random state"] \
            else np.random.randint(1000)

```

```

X_train, X_test, y_train, y_test = train_test_split(
    feature_matrix.values, completion_time,
    test_size=0.3, random_state=random_state)

if prediction_parameters["use augmentation for regression"]:
    X_train_augmented = np.vstack((X_train, X_train + 0.01 *
                                    np.random.standard_normal(X_train.shape)))
    y_train_augmented = np.hstack((y_train, y_train))
else:
    X_train_augmented, y_train_augmented = X_train, y_train

regressor_random_state = random_state \
    if prediction_parameters["use same random state for regressor"] \
    else None

if use_streamlit:
    st_regressor = prediction_parameters["pick regressor"]
    combo_map = {
        "use TSR": ("use TSR", False, False),
        "use NNR": (False, "use NNR", False),
        "use GBR": (False, False, "use GBR"),
    }
    prediction_parameters.update(
        dict(zip(["use TSR", "use NNR", "use GBR"],
                combo_map[st_regressor])))
}

if prediction_parameters["use TSR"]:
    regressor = TSR(random_state=regressor_random_state)

elif prediction_parameters["use NNR"]:
    regressor = NNR(random_state=regressor_random_state)
    param_grid = {
        "alpha": [0.1, 0.5, 1.0],
        "hidden_layer_sizes": [(50,), (100,), (50, 50), (100, 50)],
        "activation": ["relu", "tanh"],
        "learning_rate_init": [0.001, 0.01],
    }

elif prediction_parameters["use GBR"]:
    regressor = GBR(random_state=regressor_random_state)
    param_grid = {
        "n_estimators": [100, 200],
        "learning_rate": [0.01, 0.05],
        "max_depth": [2, 3],
        "min_samples_split": [2, 4],
        "min_samples_leaf": [1, 2],
    }

if not prediction_parameters["use TSR"]:
    grid_search = GridSearchCV(
        regressor, param_grid, cv=3, n_jobs=-1,

```

```

        scoring=prediction_parameters["scoring for regressor grid
search"])

grid_search.fit(X_train_augmented, y_train_augmented)

best_parameters = grid_search.best_params_
for param_name, param_value in best_parameters.items():
    setattr(regressor, param_name, param_value)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train_augmented)
X_test = scaler.transform(X_test)

regressor.fit(X_train, y_train_augmented)
predicted_completion_times = regressor.predict(X_test)

regression_metrics["mse_values"].append(
    mean_squared_error(y_test, predicted_completion_times))
regression_metrics["mae_values"].append(
    mean_absolute_error(y_test, predicted_completion_times))
regression_metrics["r2_values"].append(
    r2_score(y_test, predicted_completion_times))

completion_times_str += f"""
RUN {i+1}
\tpredicted
{predicted_completion_times.astype(int)}
\treal
{y_test.astype(int).values}
"""

end = time()

with open(f"{current_datetime}__predictions.txt", "a") as f:
    f.write(f"Total execution time - {regressor}:\n \
{np.round(end - start, 3)} s\n\n")
    f.write(completion_times_str)

feature_matrix["Time"] = completion_time

return classification_metrics, regression_metrics

def scorings_for_grid_search():
    # INNER
    scorings_for_classifier = (
        "accuracy", "balanced_accuracy", "top_k_accuracy",
        "f1", "f1_macro", "f1_micro", "f1_samples", "f1_weighted",
        "jaccard", "jaccard_macro", "jaccard_micro", "jaccard_samples",
        "jaccard_weighted",
        "neg_brier_score", "neg_log_loss",

```

```

    "precision", "average_precision", "precision_macro",
    "precision_micro", "precision_samples", "precision_weighted",
    "recall", "recall_macro", "recall_micro", "recall_samples",
    "recall_weighted",
    "roc_auc", "roc_auc_ovr", "roc_auc_ovr_weighted",
    "roc_auc_ovr", "roc_auc_ovr_weighted",
)
scorings_for_regressor = (
    "explained_variance", "max_error",
    "neg_mean_absolute_error", "neg_mean_absolute_percentage_error",
    "neg_mean_squared_log_error", "neg_mean_poisson_deviance",
    "neg_mean_squared_error", "neg_mean_squared_log_error",
    "neg_median_absolute_error", "neg_root_mean_squared_error",
    "r2",
)
return scorings_for_classifier, scorings_for_regressor

def metrics_visualization(classification_metrics: dict, regression_metrics: dict):
    runs_range = range(1, system_parameters["number of runs"] + 1)

    fig, axs = plt.subplots(nrows=2, ncols=3)
    fig.set_size_inches(w=12, h=7)
    fig.subplots_adjust(wspace=0.35, hspace=0.35)

    def plot_1D_metric(ax, x, y, xlabel, title, color):
        ax.plot(x, y, color=color, linewidth=3)
        ax.set_xlabel(xlabel)
        ax.set_title(title)
        ax.grid(True)
        ax.xaxis.set_major_locator(MaxNLocator(integer=True))

    def plot_2D_metric(ax, data, xlabel, ylabel, title, fmt, cmap):
        heatmap(data, annot=True, fmt=fmt, cmap=cmap, cbar=False, ax=ax)
        ax.set_xlabel(xlabel)
        ax.set_ylabel(ylabel)
        if "report" in title:
            ax.yaxis.set_label_coords(-0.2, 0.8)
        ax.set_title(title)

    plot_1D_metric(
        ax=axs[0, 0], x=runs_range, y=classification_metrics["accuracies"],
        xlabel="run", title="Accuracy over runs", color="orange")

    plot_2D_metric(
        ax=axs[0, 1],
        data=np.mean(

```

```

        classification_metrics["confusion_matrices"], axis=0),
        xlabel="Predicted labels", ylabel="True labels",
        title="Avg confusion matrix", fmt=".2f", cmap="Greens")

plot_2D_metric(
    ax=axs[0, 2],
    data=calculate_the_average_classification_report(
        classification_metrics["classification_reports"]),
    xlabel="Metrics", ylabel="Classes",
    title="Avg classification report", fmt=".2f", cmap="BuPu")

plot_1D_metric(
    ax=axs[1, 0], x=runs_range, y=regression_metrics["mse_values"],
    xlabel="run", title="MSE over runs", color="blue")

plot_1D_metric(
    ax=axs[1, 1], x=runs_range, y=regression_metrics["mae_values"],
    xlabel="run", title="MAE over runs", color="green")

plot_1D_metric(
    ax=axs[1, 2], x=runs_range, y=regression_metrics["r2_values"],
    xlabel="run", title="R2 over runs", color="red")

plt.savefig(f"{current_datetime}__evaluations.png")
plt.show()

```

```

def if_running_a_regular_script():
    # 13 + 2 elemenata
    pred_parameters = {
        "use non essential game annotations" : False,
        "use fixed random state" : False,
        "use stratify" : False,
        "use scaling and oversampling for classification" : False,
        "use same random state for classifier" : False,
        "use SVC" : True,
        "use DTC" : False,
        "use RFC" : False,
        "use same random state for regressor" : False,
        "use augmentation for regression" : False,
        "use TSR" : False,
        "use NNR" : True,
        "use GBR" : False,
        "scoring for classifier grid search" : "accuracy",
        "scoring for regressor grid search" : "neg_mean_squared_error",
    }

    # 6 elemenata
    proc_parameters = {
        "csv values: columns to drop" : ["OriginalTimestamp"],

```

```

    "csv values: timestamps column name" : "Timestamp",
    "csv values: pow 1st column name" : "POW.AF3.Theta",
    "csv values: amp 1st column name" : "EEG.AF3",
    "csv intervals: timestamps column name" : "timestamp",
    "game annotations" : ["Time", "Outcome"],
}

# 5 (3) elemenata
sys_parameters = {
    "path" : r"C:\Users\neman\Desktop\master_python\newdata",
    "number of runs" : 3,
    "window length" : 2.0,
    "fs" : 128,
    "files extensions" : {
        "csv values files" : ".md.mc.pm.fe.bp.csv",
        "csv intervals files" : "intervalMarker.csv",
        "json annotations files" : ".json",
    },
}

if pred_parameters["use non essential game annotations"]:
    proc_parameters["game annotations"].extend(
        ["Difficulty", "Fields", "Mines"])

check, errors_text = check_various_conditions(
    pred_parameters, proc_parameters, sys_parameters)

if check:
    return pred_parameters, proc_parameters, sys_parameters, None
else:
    return None, None, None, errors_text

def check_various_conditions(pred_parameters: dict,
                             proc_parameters: dict,
                             sys_parameters: dict):
    # INNER
    failed_vconditions = "\n\nGreške:\n"

    scorings_for_classifier, scorings_for_regressor = scorings_for_grid_search()

    vcondition_0 = os.path.exists(sys_parameters["path"])

    vcondition_1 = all(
        [isinstance(value, bool) for key, value in pred_parameters.items()
         if key not in ["scoring for classifier grid search",
                        "scoring for regressor grid search"]])

    vcondition_1a = pred_parameters["scoring for classifier grid search"] in \
        scorings_for_classifier

```

```

vcondition_1b = pred_parameters["scoring for regressor grid search"] in \
    scorings_for_regressor

vcondition_2 = np.sum([pred_parameters["use SVC"],
                      pred_parameters["use DTC"],
                      pred_parameters["use RFC"]])
) == 1

vcondition_3 = np.sum([pred_parameters["use TSR"],
                      pred_parameters["use NNR"],
                      pred_parameters["use GBR"]])
) == 1

vcondition_4 = True
try:
    if isinstance(int(sys_parameters["number of runs"]), int):
        pass
except:
    vcondition_4 = False

vcondition_5 = sys_parameters["number of runs"] > 0

vcondition_6 = isinstance(sys_parameters["window length"], float)

vcondition_7 = 0.1 <= sys_parameters["window length"] <= 5

vcondition_8 = sys_parameters["fs"] in [2**i for i in range(7, 10 + 1)]

vcondition_9 = True
for key, value in proc_parameters.items():
    if key in ["csv values: columns to drop", "game annotations"]:
        if isinstance(value, list):
            if key == "game annotations" and len(value) == 0:
                vcondition_9 = False
                break
        else:
            vcondition_9 = False
            break
    for list_value in value:
        outer_break = False
        if not isinstance(list_value, str):
            outer_break = True
            vcondition_9 = False
            break
        if outer_break:
            break
    else:
        if not isinstance(value, str):
            vcondition_9 = False

```

```

        break

vcondition_10 = True
for suffix in sys_parameters["files extensions"].values():
    if not isinstance(suffix, str):
        vcondition_10 = False
        break
    elif suffix == sys_parameters["files extensions"]["json annotations
files"]:
        if not suffix.endswith(".json"):
            vcondition_10 = False
            break
        elif not suffix.endswith(".csv"):
            vcondition_10 = False
            break

vcondition_11 = (len(pred_parameters) == 16 and
                 len(proc_parameters) == 6 and
                 len(sys_parameters) == 5 and
                 len(sys_parameters["files extensions"]) == 3)

if not vcondition_0:
    failed_vconditions += "- Zadata je nepostojeća putanja.\n"

if not vcondition_1:
    failed_vconditions += """- Parametri koji se koriste u sklopu predikcija
(osim scoring-a) moraju svi biti definisani kao boolean-i.\n"""

if not vcondition_1a:
    failed_vconditions += """- Metoda za ocenjivanje klasifikatora pri
traženju
optimalnih parametara nije nađena u predefinisanom skupu.\n"""

if not vcondition_1b:
    failed_vconditions += """- Metoda za ocenjivanje regresora pri traženju
optimalnih parametara nije nađena u predefinisanom skupu.\n"""

if not vcondition_2:
    failed_vconditions += "- Isključivo jedan klasifikator mora biti
aktivran.\n"

if not vcondition_3:
    failed_vconditions += "- Isključivo jedan regresor mora biti aktivran.\n"

if not vcondition_4:
    failed_vconditions += "- Broj iteracija sistema mora biti intedžer.\n"

if not vcondition_5:
    failed_vconditions += "- Broj iteracija sistema mora biti > 1.\n"

```

```

if not vcondition_6:
    failed_vconditions += "- Dužina prozora mora biti float vrednost.\n"

if not vcondition_7:
    failed_vconditions += "- Nedozvoljena dužina prozora.\n"

if not vcondition_8:
    failed_vconditions += "- Nedozvoljena frekvencija odabiranja.\n"

if not vcondition_9:

    failed_vconditions += """- Svi parametri koji označavaju imena kolona
moraju biti str (sem imena kolona za izbacivanje i anotacija igre;
oni moraju biti liste ispunjene str).\n"""

if not vcondition_10:
    failed_vconditions += "- Sufiksi fajlova nisu pravilno definisani.\n"

if not vcondition_11:
    failed_vconditions += "- Rečnici parametara nisu originalne dužine.\n"

if failed_vconditions != "\n\nGreške:\n":
    return False, failed_vconditions
else:
    return True, None

def if_running_a_streamlit_app():
    st.set_page_config(layout="centered")

    edit_streamlit_style = """
        <style>
            MainMenu {visibility: hidden;}
            footer {visibility: hidden;}
            #root > div:nth-child(1) > div > div > div > \
            section > div {padding-top: 2rem;}
        </style>
        """
    st.markdown(body=edit_streamlit_style, unsafe_allow_html=True)

    col_1, _, col_2, col_3 = st.columns(spec=[8, 1, 5, 2])

    col_1.text(body="")
    col_1.subheader(body="Pomoćni alat za master tezu")

    col_2.text(body="")
    col_2.info(body="Nemanja Peruničić, B1 3/22", icon="💡")

    try:
        col_3.image(image="ftn_logo_fun.gif")

```

```

except:
    pass

st.divider()
_, col_a, _, col_b = st.columns(spec=[1, 14, 1, 12])

with col_a:
    scorings_for_classifier, scorings_for_regressor =
scorings_for_grid_search()

pred_parameters = {
    "use fixed random state" : st.checkbox(
        label="use fixed random state",
        ),
    "divider 1" : st.divider(
        ),
    "use stratify" : st.checkbox(
        label="use stratify",
        ),
    "use scaling and oversampling for classification" : st.checkbox(
        label="use scaling and oversampling for classification",
        ),
    "use same random state for classifier" : st.checkbox(
        label="use same random state for classifier",
        ),
    "pick classifier" : st.selectbox(
        label="pick classifier",
        options=["use SVC", "use DTC", "use RFC"],
        ),
    "use grid search for classifier": st.checkbox(
        label="use grid search for classifier",
        ),
    "scoring for classifier grid search" : st.selectbox(
        label="pick classifier scoring",
        options=scorings_for_classifier,
        ),
    "divider 2" : st.divider(
        ),
    "use same random state for regressor" : st.checkbox(
        label="use same random state for regressor",
        ),
    "use augmentation for regression" : st.checkbox(
        label="use augmentation for regression",
        ),
    "pick regressor" : st.selectbox(
        label="pick regressor",
        options=["use TSR", "use NNR", "use GBR"],
        ),
    "use grid search for regressor": st.checkbox(
        label="use grid search for regressor",
        )
}

```

```

        ),
        "scoring for regressor grid search" : st.selectbox(
            label="pick regressor scoring",
            options=scorings_for_regressor,
            ),
        "fake divider" : st.text(
            body="",
            ),
        }
try:
    st.image(image="eeg_potato.jpg")
except:
    pass

with col_b:
    proc_parameters = {
        "csv values: columns to drop" : st.multiselect(
            label="csv values: columns to drop",
            options=["OriginalTimestamp"],
            default=[ "OriginalTimestamp" ],
            ),
        "csv values: timestamps column name" : st.selectbox(
            label="csv values: timestamps column name",
            options=[ "Timestamp" ],
            ),
        "csv values: pow 1st column name" : st.selectbox(
            label="csv values: pow 1st column name",
            options=[ "POW.AF3.Theta" ],
            ),
        "csv values: amp 1st column name" : st.selectbox(
            label="csv values: amp 1st column name",
            options=[ "EEG.AF3" ],
            ),
        "csv intervals: timestamps column name" : st.selectbox(
            label="csv intervals: timestamps column name",
            options=[ "timestamp" ],
            ),
        "game annotations" : st.multiselect(
            label = "game annotations",
            options=[ "Time", "Outcome", "Difficulty", "Fields", "Mines" ],
            default=[ "Time", "Outcome" ],
            help = "Ne izostaviti obavezne parametre!" ),
        }

sys_parameters = {
    "path" : st.selectbox(
        label="path",
        options=[ "C:/Users/neman/Desktop/master_python/newdata" ],
        ),
    "divider 1" : st.divider(

```

```

        ),
    "number of runs" : st.slider(
        label="number of runs",
        min_value=1, max_value=50, step=1, value=5,
    ),
    "window length" : st.slider(
        label="window length",
        min_value=0.1, max_value=5.0, step=0.1, value=2.0,
    ),
    "fs" : st.select_slider(
        label="sampling frequency",
        options=[2**i for i in range(7, 10 + 1)], value=2**7,
    ),
    "divider 2" : st.divider(
    ),
    "files extensions" : {
        "csv values files" : st.selectbox(
            label="csv values files",
            options=[".md.mc.pm.fe.bp.csv"],
        ),
        "csv intervals files" : st.selectbox(
            label="csv intervals files",
            options=["intervalMarker.csv"],
        ),
        "json annotations files" : st.selectbox(
            label="json annotations files",
            options=".json",
        ),
    },
},
st.divider()

return pred_parameters, proc_parameters, sys_parameters

def remember_parameters():
    # INNER
    combined_dict = {**prediction_parameters,
                     **processing_parameters,
                     **system_parameters}
    combined_dict = {k: v for k, v in combined_dict.items() if "divider" not in k}

    df = DataFrame(list(combined_dict.items()), columns=['Key', 'Value'])
    df.to_csv(f"{current_datetime}__parameters.csv", index=False)

def experimental(how_many_parameters, range_classifiers, range_regressors):
    # INNER
    combo_prediction_parameters = []
    x = f"{how_many_parameters:02d}"b"

```

```

for i in range(2^how_many_parameters):
    binary = format(i, x)

    if any(binary[range_classifiers].count('1'),
           binary[range_regressors].count('1')) == 1:

        boolean_list = [bool(int(bit)) for bit in binary]
        combo_prediction_parameters.append(boolean_list)

return combo_prediction_parameters


def machine_learning_system():
    metrics_visualization(
        *regression_prediction_and_evaluation(
            *classification_prediction_and_evaluation(
                creating_the_feature_matrix(
                    *eeg_feature_extraction(
                        *processing(
                            *extract_data(
                                ))))))))

    _ = """
start lines:

extract_data - 113
processing - 171
    create_columns_names - 218
    windowing - 234
eeg_feature_extraction - 273
creating_the_feature_matrix - 375
classification_prediction_and_evaluation - 428
    calculate_the_average_classification_report - 551
regression_prediction_and_evaluation - 577
    scorings_for_grid_search - 685
metrics_visualization - 712



---


if_running_a_regular_script - 771
    check_various_conditions - 827
if_running_a_streamlit_app - 968
remember_parameters - 1125
"""

if __name__ == "__main__":
    # !!!
    use_streamlit = True
    _ = """          ^^^^^^
if True >> u terminalu >> "streamlit run nemanja_perunicic_mas.py"

```

```

if False >> proverite vrednosti parametara u funkciji:
if_running_a_regular_script() >> pa potom standardno Run dugme/komanda
"""

if not use_streamlit:
    (prediction_parameters,
     processing_parameters,
     system_parameters,
     errors
    ) = if_running_a_regular_script()

current_datetime = datetime.now().strftime("%y-%m-%d_%H-%M-%S")

if errors:
    print(errors)
else:
    remember_parameters()
    machine_learning_system()

else:
    (prediction_parameters,
     processing_parameters,
     system_parameters
    ) = if_running_a_streamlit_app()

status = st.empty()
start = st.button(
    label="**:green[START THE SYSTEM]** 🚀",
    use_container_width=True,
)

if start:
    status.subheader("🚀 🚀 🚀 🚀 🚀 🚀 🔜")

    current_datetime = datetime.now().strftime("%y-%m-%d_%H-%M-%S")
    remember_parameters()
    machine_learning_system()

    status.subheader("🚀 ... gotovo! ➡ Možete skrolovati nadole...")

try:
    st.image(f"{current_datetime}_evaluations.png")
except:
    pass
try:
    with open(f"{current_datetime}_predictions.txt", "r") as f:
        st.divider()
        st.text(body=f.read())
except:

```

```
        pass
try:
    st.divider()
    st.table(data=read_csv(f"{current_datetime}__feature_matrix.csv"))
except:
    pass
try:
    st.divider()
    st.table(data=read_csv(f"{current_datetime}__parameters.csv"))
except:
    pass

#           \__(*)__/    ta-da
```