VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

Magistro Baigiamasis Darbas

EEG pikų aptikimas dirbtinio intelekto metodais

Atliko:
Ričardas Mikelionis parašas

Vadovas:
Dr. Andrius Vytautas Misiukas
Misiūnas

Vilnius
2021

Turinys

Sutartinis terminų žodynas

- *EEG* (angl. Electroencephalogram) – elektroencefalograma.
- *EKG* (angl. Electrocardiogram) – elektrokardiograma.
- *ANN* (angl. Artificial Neural Network) – Dirbtinis neuroninis tinkas.
- *DNN* (angl. Deep Neural Network) – Gilusis dirbtinis neuroninis tinklas.
- *CNN* (angl. Convolutional Neural Network) – Konvoliucinis dirbtinis neuroninis tinklas.
- *Vilnelė* (angl. Wavelet) – Į bangą panaši osciliacija, kurios amplitudė prasideda ties 0, paklya ir tuomet vėl grįžta į 0.
- *TPR* (angl. True Positive Rate) – Teisingai klasifikuotų pozityvių duomenų rinkinių kiekis.
- *FPR* (angl. False Positive Rate) – Neteisingai klasifikuotų pozityvių duomenų rinkinių kiekis.
- *Jautrumas* (angl. Sensitivity) – Skaičius nurodantis kiek proporcingai teigiamai žymimų duomenų rinkinių buvo klasifikuoti teisingai.
- *Konkretumas* (angl. Specificity) – Skaičius nurodantis kiek proporcingai neigiamai žymimų duomenų rinkinių buvo klasifikuoti teisingai.
- *ReLU* (angl. Rectified Linear Unit) – Išlyginto tiesinio vieneto aktyvacijos funkcija.
- *2D CNN* – ANN modelis, kurio pagrindą sudaro vienas ar keli dvidimensiai konvoliuciniai sluoksniai.
- *ROC* (angl. Receiver operating characteristic) –grafikas rodantis klasifikatoriaus jautrumo ir konkretumo tarpusavio ryšį.
- *DF* (angl. Decision Function) – atraminių vektorių klasifikatoriaus metodas, padedantis ieškoti hipererdvės.

Santrauka

Šiame darbe buvo analizuotas dirbtinio neuroninio tinklo, kaip įrankio skirto analizuoti elektroencefalogramą, efektyvumas. Pagrindinis šio tyrimo objektas – 40-200 ms. ilgio EEG darinys – pikas. Pikai dažnai naudojami įvairių epilepsijos pobūdžių diagnozei. Šiame darbe buvo naudojami EEG pikai gauti iš realių pacientų su Rolandinės epilepsijos (angl. Rolandic epilepsy) diagnoze.

EEG pikų aptikimui buvo pasirinkti 5 skirtingi klasifikavimo modeliai: 2D konvoliuciniai neuroniniai tinklai (2D CNN), logistinė regresija, sprendimų medžiai, atraminių vektorių klasifikatoriai bei AdaBoost metaalgoritmas. Signalų analizė buvo atlikta naudojant duomenis iš 18 EEG kanalų naudojant tiek neapdorotus tiek vilnelių transformacija filtruotus duomenis, rezultatus koreguojant per dirbtinio neuroninio tinklo modelio parametrus.

2D konvoliucinio neuroninio tinklo (2D CNN) pagalba buvo pasiektas 0.973 tikslumas, 0.9605 jautrumas, 0.3592 konkretumas. Logistinės regresijos modelio pagalba buvo pasiektas 0.7088 tikslumas, 0.7415 jautrumas, 0.3703 konkretumas. Sprendimų medžio modelio pagalba buvo pasiektas 0.8535 tikslumas, 0.9064 jautrumas, 0.3074 konkretumas. Atraminių vektorių klasifikatorių pagalba buvo pasiektas 0.883 tikslumas, 0.9408 jautrumas, 0.2851 konkretumas. AdaBoost metaalgoritmo pagalba buvo pasiektas 0.8650 tikslumas, 0.9179 jautrumas, 0.3185 konkretumas.

Geriausius rezultatus pavyko gauti naudojant 2D konvoliucinį neuroninį tinklą, naudojant vilnelių transformacija filtruotus duomenis.

Summary

EEG Spike Detection Using Artificial Neural Networks

In this paper, an analysis of artificial neural network (ANN) effectivenes, when used as a tool to analyse electroencephalograms (EEG), is presented. Main target of this analysis is a 40-200 ms. long EEG spike. EEG spikes are usually used in epilepsy diagnositcs. This paper analyses EEG spike information obtained from real life patients diagnosed with Rolandic epilepsy.

Five different types of classifiers were chose for EEG spike detection: 2D Convolutional Neural Networks (2D CNN), Logistic Regression classifier, Decision Tree classifier, Support vector machine classifer as well as AdaBoost metaalgorithm. Signal analysis was conducted on data, that was extracted from 18 EEG channels and was used unprocessed as well as preprocessed using wavelet transform and results optimisation was done only by changing various ANN parameters.

2D Convolutional Neural Network (2D CNN) archieved 0.973 accuracy, 0.9605 sensitivity and 0.3592 specificity values. Logistic regresion model archieved 0.7088 accuracy, 0.7415 sensitivity and 0.3703 specificity values. Decision Tree model archieved 0.8535 accuracy, 0.9064 sensitivity and 0.3074 specificity values. Support Vector machine classifier archieved 0.883 accuracy, 0.9408 sensitivity and 0.2851 specificity values. AdaBoost metaalgorithm archieved 0.8650 accuracy, 0.9179 sensitivity and 0.3185 specificity values.

Python programming language and several open-source libraries for machine learning such as TensorFlow, Keras as well as Scikit-Learn were used to archieve the results described in the paper. EEG were provided in the European Data Format (EDF) which were converted into CSV and divided into one spike length (maximum length of 200 ms. was chosen) intervals which were used for training of the neural network models.

As seen from this paper 2D CNN returned the best results of 0.973 accuracy, 0.9605 sensitivity and 0.3592 specificity when used with preprocessed data.

Įvadas

Encefalogragija – be jokios abejonės, yra bene svarbiausias įrankis neurologinių sutrikimų, ypač susijusių su epilepsija, diagnozavime. Sutrikimai encefalogramoje (*toliau – EEG*) dažnai pasirodo, kaip nereguliari veikla iškylanti virš bazinio signalo. Tačiau dėl duomenų kiekio vizuali sutrikimų paieška yra itin sudėtingas bei laiko reikalaujantis procesas. Štai pavyzdžiui elektrokardiograma (*toliau – EKG*) širdies veiklos sutrikimams nustatyti yra struktūriškai panaši, tačiau dauguma testų gauti EKG, netrunka nė minutės, todėl ir vizuali rankinė, ar kompiuterinė duomenų analizė netrunka ilgai. Kita vertus EEG testai trunka apie 20-40 minučių [5], todėl duomenų kiekis ir analizės laikas išauga eksponentiškai.

EEG tyrimo metu matuojama smegenų veikla, fiksuojami elektriniai signalai (virpesiai). Gauti signalai yra interpretuojami, signalo interpretacija gali nusakyti smegenų aktyvumą, bei sąmoningumo būklę. Dažniausiai EEG tyrimai naudojami epilepsijos ar kitų centrinės nervų sistemos ligų simptomų paieškai bei diagnozei.

Pagal 10-20 matavimo standartą išgautoje EEG duomenys sudaromi iš 21 kanalo, kiekviename kanale ieškoma signalo struktūrinių vienetų vadinamų pikais. Pikai EEG yra simptomas didžiąjai daliai centrinės nervų sistemos sutrikimų, tad gydytojai turi praleisti itin daug laiko ne tik ieškant pikų EEG signale, tačiau ir juos analizuojant siekiant diagnozuoti ligos pobūdį. Viena tokių ligų – gerybinė epilepsija dar vadinama rolandine epilepsija.

Dirbtinis intelektas (angl. *artificial intelligence*) – šiandien itin plačiai naudojamas įrankis tiek mokslinių tyrimų, tiek komercinių produktų srityse. Įrankių, kaip, dar 2015-ais *Google* išleistos [17] giliojo mokymosi (angl. *deep learning*), atviro kodo (angl. *open-source*) platformos, *Tensorflow* [1] bei *Scikit-Learn* kompiuterinio mokymosi (angl. *machine learning*) bibliotekos [21] prieinamumas ir naudojimosi paprastumas toliau skatina inovacijas.

Šiandien dirbtinio intelekto įrankiai yra itin populiarūs sprendžiant įvairias sudėtingas su duomenų analize susijusias užduotis. Įvairūs kompiuterinio mokymosi metodai šiandien sugeba spręsti ne tik įvairias matematiškai sudėtingas užduotis, tačiau plačiai pritaikomi ir kasdieniame gyvenime.

Vis kylanti populiarume pastaruoju metu esanti kompiuterinio mokymosi atšaka – gilieji neuroniniai tinklai (angl. *Deep Neural Network*) (toliau – DNN). Jais įprastai naudojamasi įvairioms su vaizdų apdorojimu susijusioms problemoms spręsti. Tačiau mažiau naudojama sritis, kuriai DNN puikiai pritaikyti yra įvairių signalų, tame tarpe ir medicinos signalų analizė. Tokio tipo analizė bus nagrinėjama šiame darbe.

Siekiant palengvinti gydytojų darbą, siekiama sukurti kompiuterinio mokymosi sistemą sugebančia atrinkti kandidatus į pikus.

Darbo tikslas ir uždaviniai

Šio darbo tikslas – atlikti automatinio EEG pikų paieškos tyrimą, panaudojant keletą populiarių kompiuterinio mokymosi įrankių. Tikslui pasiekti buvo suformuluoti tokie uždaviniai:

1. Išanalizuoti literatūrą apibūdinančią EEG struktūrą.
2. Išanalizuoti literatūrą apibūdinančia signalų analizę.
3. Išanalizuoti literatūrą apibūdinančią skirtingus kompiuterinio mokymosi metodus.
4. Paruošti EEG duomenis kompiuterinio mokymosi modelio apmokymui.
5. Apmokyti keletą dirbtinių 2D konvoliucinių neuroninių tinklų EEG pikų atpažinimo.
6. Palyginti apmokymo bei klasifikavimo tikslumą bei efektyvumą.
7. Apmokyti keletą kompiuterinio mokymosi modelių EEG pikų atpažinimo.
8. Palyginti apmokymo bei klasifikavimo tiklsumą bei efektyvumą.

1. Dalykinė sritis

Elektroencefalogrfija – metodas, kuriuo apie skalpą pritvirtintais elektrodais įrašomi smegenų neuronų veiklos sukeliami elektriniai virpesiai [5]. Medicinos srityje encefalografija vadinamas toks smegenų veiklos įrašinėjimas trumpą laiko tarpą, dažniausiai 20-40 minučių ir vykdoma įrašant keleto skirtingų elektrodų pritvirtintų prie skalpo duomenis, taip gaunama elektroencefalograma (toliau – *EEG*). Elektroencefalografija dažniausiai sutinkama epilepsijos diagnostikoje, nes epileptinė veikla EEG matosi, kaip anomalijos, kurias galima analizuoti.

Įprasta skalpo EEG gaunama ant skalpo pritvirtinant elektrodus konduktyvios medžiagos pagalba, įprastai skalpą paruošiant negyvų odos ląstelių pašalinimu. Didžioji dauguma sistemų naudoja elektrodus, kurių kiekvienas yra pritvirtinamas atskiru laidu. Taip pat yra sistemų, kuriose elektrodai pritvirtinami kepurėlės ar tinklelio viduje, tokios sistemos sutinkamos, kai yra poreikis tankiam elektrodų išsidėstymui.

Dažniausiai sutinkama yra tarptautinė 10-20 sistema. Sistemos pagalba yra užtikrinama, jog elektrodų pavadinimai yra vienodi daugybėje skirtingų laboratorijų. Daugumoje medicininių taikymų yra naudojami 19 elektrodų smegenų veiklai bei po vieną elektrodą įžeminimui ir sistemos baziniam įverčiui. Kiekvienas elektrodas prijungiamas prie skirtingo stiprintuvo, kai tuo tarpu sistemos bazinis elektrodas prijungiamas prie visų, taip sudarydamas elektrodų porą kiekvienam stiprintuvui. Sustiprintas signalas kiekvienam elektrodui tuomet yra filtruojamas ir skaitmenizuojamas taip sudarant galutinę EEG.

1.1. Tarptautinė 10-20 Elektroencefalografijos sistema

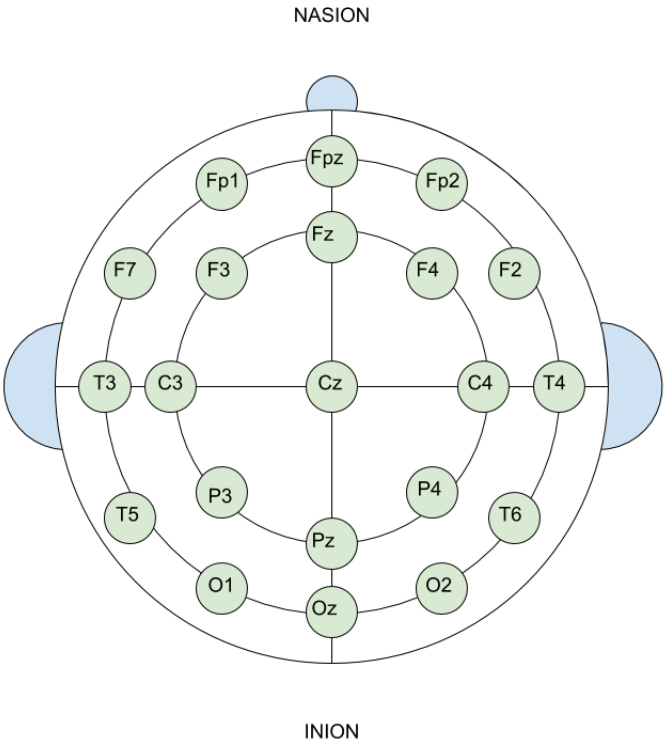
1958 m. Tarptautinė Encefalografijos bei Klinikinės Neurofiziologijos Federacija (angl. *International Federation in Electroencephalography and Clinical Neurophysiology*) pradėjo naudoti standartizuotą elektrodų išdėstymo metodą vadinamą 10-20 sistema, vaizduojama 1 paveikslėlyje. Ši sistema standartizavo fizinį elektrodų išdėstymą ant skalpo. Galva yra padalinama į proporcingai nutolusius atstumus nuo esminių kaukuolės darinių, kaip nosies sujungimas (angl. *Nasion*) bei žemiausia kaukuolės vieta (angl. *Inion*), siekiant kuo optimaliau padengti visus smegenų regionus [26].

Pavadinimas 10-20 paimtas pagal tai, jog toks šios sistemos proporcinis atstumas procentais tarp ausų ir nosies, kur lipdomi elektrodai. Elektrodai žymimi pagal smegenų sritį, kurią jie dengia: F – Kaktinė sritis (angl. *Frontal lobe*), C – Centrinė sritis (angl. *Central lobe*), T – Smilkininė sritis (angl. *Temporal lobe*), P – Momeninė sritis (angl. *Parietal lobe*) bei O – Pakaušinė sritis (angl. *Occipital lobe*). Prie raidžių žymėjimui naudojami nelyginiai skaičiai kairėje galvos pusėje, bei lyginiai skaičiai dešinėje galvos pusėje žr. 1 paveikslėlį. Kairė bei dešinė pusės parenkamos iš paciento pozicijos.

1.2. EEG taikymai

Medicinoje ir tyrimuose žmonių bei gyvūnų EEG, pasak R. D. Bickford, galima panaudoti [26]:

- stebint budrumą, komą bei smegenų mirtį;
- aptinkant po galvos sužalojimų, insulto, smegenų auglio bei kt. pažeistas smegenų sritis;
- tiriant epilepsiją bei nustatant epilepsijos priepolių priežastį;
- tiriant epilepsijos vaistų poveikį;



1 pav. Tarptautinės 10-20 sistemos elektrodų išsidėstymas bei etiketės. (Nubraižyta su *Google Drawings*)

- stebint žmonių bei gyvūnų smegenų raidą;
- testuojant spazmus, kaip vaistų šalutinį efektą;
- tiriant miego sutrikimus.

Elektroencefalografija medicinoje taip pat naudojama komos, encefalopatijos bei smegenų mirties diagnozei. EEG gali būti naudojama, kaip metodas diagnozuoti auglius, insultą bei kitus židininius smegenų veiklos sutrikimus, tačiau šiandien EEG tam nėra taip plačiai naudojama dėl išplitusių magnetinio rezonanso tomografijos bei kompiuterinės tomografijos technikų.

EEG bangų grupės

Esant normaliai paciento smegenų veiklai vyrauja viena iš keleto smegenų aktyvumo grupių (žiūrėti 1 lentelę). Vienu metu būdinga vienas iš pateiktų grupių priklausomai nuo paciento būsenos. Dažnis, kita vertus, priklauso nuo kitų veiksnių, kaip lytis, amžius, budrumo lygis. Esant smegenų veiklos sutrikimams šie dažniai gali keistis.

Kaip minėta pateiktoje lentelėje EEG galima išgauti matuojant paciento smegenų veiklą tiek budrumo, tiek miego busenose, tiek pacientui esant be sąmonės. Mažiems vaikams EEG matavimai dažnai atliekami esant miego būsenoje, kol suaugusiems bei vyresniems vaikams žymiai dažniau šie mataviai bus aliekami pacientui esant budumo būsenoje.

EEG Artefaktai

Dažna EEG signalo analizės technika yra artefaktų paieška. Įprastai artefaktais laikomi aukštesnės amplitudės ir kitokios formos, nei visas signalas, signalo gabalai. Artefaktai EEG būna dviejų tipų: sukelti paciento arba techniniai. Paciento sukelti artefaktai yra bet kokie fiziologiniai

1 lentelė. Smegenų aktyvumo bangų grupės

Bangos pavadinimas	Dažnis	Pobūdis
Beta	>13Hz	Dažniausiai sutinkama, jei pacientas yra atmerkęs akis, aktyviai reaguoja į aplinką.
Alpha	8-13Hz	Dažniausiai sutinkama, jei pacientas yra budrus, tačiau atsipalaidavęs, užmerkęs akis.
Theta	4-8Hz	Dažniausiai sutinkama, jei pacientas yra miego būsenoje.
Delta	0.5-4Hz	Dažniausiai sutinkama, jei pacientas yra gilaus miego būsenoje, ar be sąmonės.

signalai, kurie gali smarkiai paveikti EEG. Techniniai artefaktai, tokie, kaip kintamosios elektros srovės (angl. *AC power*) triukšmas gali būti pašalinti, pavyzdžiui naudojant trumpesnius elektrodų laidus. Dažniausi EEG artefaktų šaltiniai yra klasifikuojami taip [26]:

Paciento sukelti:

- Bet kokie kūno judesiai;
- Širdies stimuliatorius;
- Paspartėjęs pulsas;
- Staigūs akių judesiai;
- Prakaitavimas;

Techniniai:

- 50/60 Hz dažnis;
- Sujudinti elektrodą jungiantys laidai;
- Netinkamas laido ir elektrodo kontaktas;
- Per didelis ar per mažas kiekis elektrodo tepalo;
- Nusidėvėjęs akumuliatorinis įtaisas.

EEG pikai

Yra daugybė neurologinių sutrikimų pasižyminčių EEG pikais [25], vienas iš jų – gerybinė (rolandinė) epilepsija, kurios pikai ir analizuojami šiame darbe. 3 paveikslėlyje atvaizduojama 200ms. ilgio EEG iškarpa vaizduojanti piką. Pikas ryškiausiai matomas ties 538955ms. F3 bei C3 kanaluose.

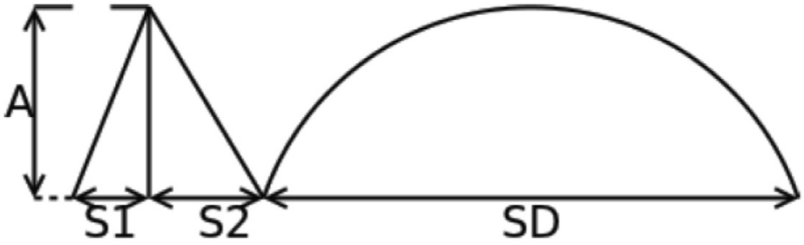
Pikas turi atsikartoti bent dviejuose gretimuose kanaluose, taip pat jis bent du kartus turi iškilti virš bazinio signalo. Pikas gali turėti tiek teigiamą, tiek neigiamą amplitudę.

Siekiant atpažinti pikus pagrindinė problema yra atskirti epilepsiforminius pikus nuo nepatologinių smailių bangų [23]. Gerybinės (rolando) epilepsijos pikai pasižymi šiomis charakteristikomis:

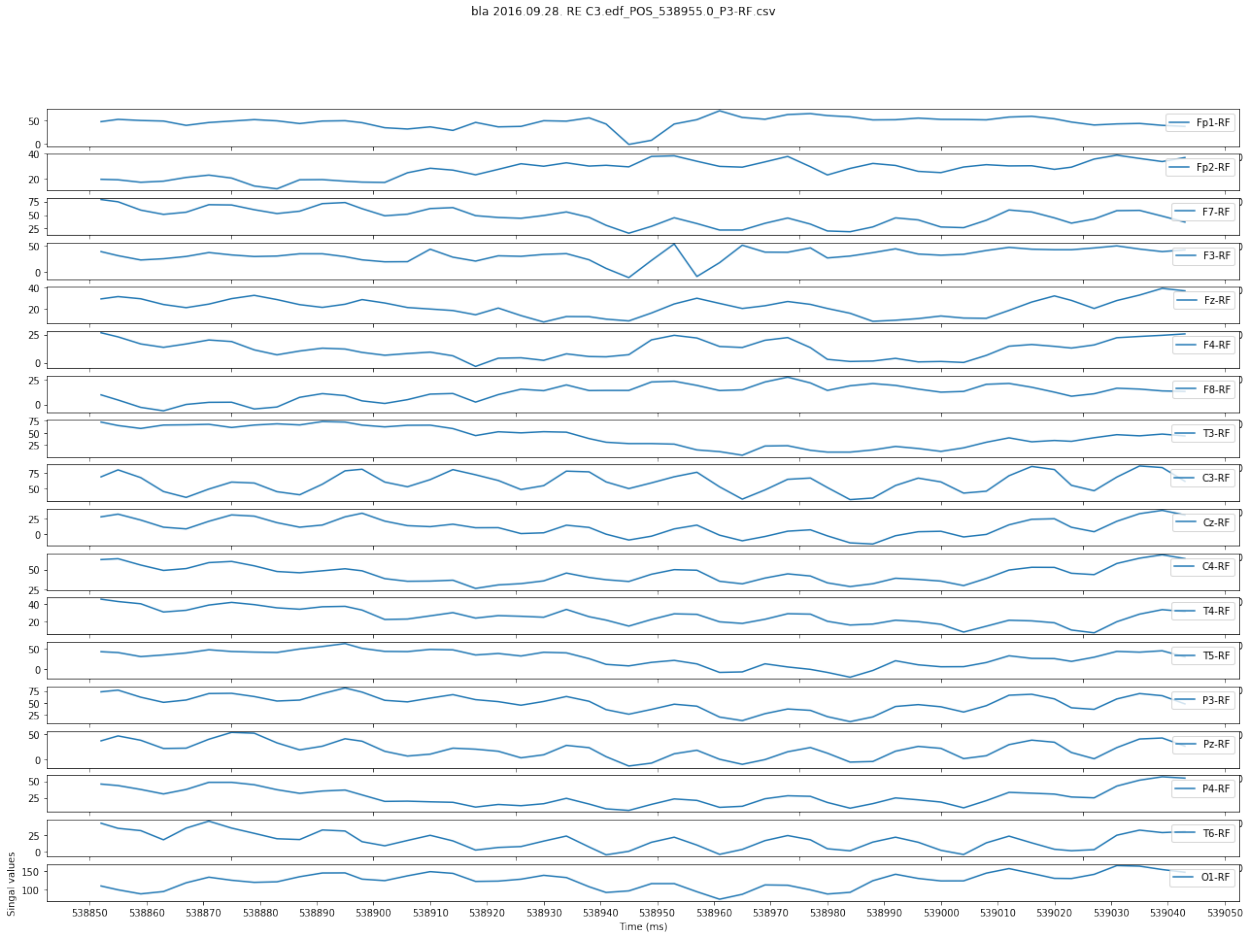
- Piko trukmė 40-200 ms.
- Piko amplitudė bent du kartus aukštesnė už dominuojančios bangos amplitudę.
- Nuožulnus besileidžiantis piko bangos šlaitas.
- Pikas kerta bazinę liniją.
- Pikai registruojami C3/C4 bei T3/T4 elektroduose.

Rolandinius pikus atskirti nuo kitų epilepsiforminių iškrovų padeda išskirti šios charakteristikos (žiūrėti 2 paveikslėlį):

- Piko trukmė (S1 + S2).
- Amplitudė (A).
- Aštrumas (S1 / S2).
- Lėtos bangos trukmė (SD).



2 pav. Rolandinės epilepsijos piko charakteristikos [23]



3 pav. 200ms. trukmės EEG iškarpa.

1.3. Susijusių darbų analizė

Pikų paieška be jokios abejonės yra binarinės klasifikacijos pobūdžio uždavinys. Tokio pobūdžio uždavinys, kur išanalizavus EEG atkarpą gaunamas rezultatas nusakantis, ar ta atkarpa savyje turi abnormalią EEG veiklą – piką, ar tiesiog reguliarią smegenų veiklą. Šiame poskyryje aprašomi darbai, kurie analizuoja šią problemą.

Pikų aptikimas ir kiti su epilepsija susiję tyrimai jau daugelį metų siejami su kompiuterinio mokymosi metodais, dar 1995 m. Ozdamar ir Kalayci [15] panaudojo Daub–4 bei Daub–20 grupių vilnelių transformacijas apdoroti signalui prieš apmokant dirbtinį neuroninį tinklą pikų aptikimo. Darbe autoriams pavyko pasiekti 91.4% klasifikavimo tikslumą signalą prieš tai apdorojus Daub–20 vilnelių transformaciją ir 90.2% klasifikavimo tikslumą signalą prieš tai apdorojus Daub–4 vilnelių transformacija.

Pikų aptikimą panaudojant vilnelių transformacijas signalui 2008m. toliau pratęsė Indiradevi, Elias, Sathidevim Nayak bei Radhakrishnan [14]. Autoriai darbe toliau naudojo Daub–4 grupės vilnelių transformacijas signalo apdorojimui prieš apmokant sistemą. Sistema apmokyta transformuotais duomenimis buvo itin efektyvi: 90.5% tikslumas, 91.7% jautrumas bei 89.3% konkretumas.

Vienas iš susijusių darbų, ir turbūt svarbiausias šio darbo kontekste apibūdina EEG signalo piko parametrus ir būdus apibūdinti signalo piką. Misiūnas, Meškauskas bei Samaitienė darbe [18] apie išvestinius EEG parametrus, tokius, kaip signalo piko lygio linijos (angl. *baseline*), piko pakilimo statumas – įkalnė (angl. *upslope*), piko nusileidimo statumas – nuokalnė (angl. *downlsope*) bei EEG sinalo piko plotis (angl. *width*). Darbe taip pat pristatomi automatiniai metodai naudojntys šias metrikas ir skirti pikų EEG signalų atpažinimui.

Tolesniame šių autorių darbe pateikiamas prieš tai buvusių darbų rezultatų pritaikymas tolesniai EEG analizei [2]. Autoriai pateikia keletą automatinės EEG analizės variantų naudojant dirbtinius neuroninius tinklus bei atraminių vektorių klasifikatorių (angl. *support vector machine*). Klasifikatoriai apmokomi remiantis praeitų darbų rezultatais, pirma morfologinių filtru atrenkant pikus, po to iš jų išrenkant parametrus, ir juos panaudojant vizualiai panašių pikų, kuriuos itin sunku atskirti, išskyrimui.

Šiame darbe bus atliekama pikų analizė pasitelkiant dirbtinį neuroninį tinklą, kol neuroniniai tinklai yra dažniau naudojami epilepsijos tipui nustatyti, o pikai aptinkami naudojant kitus metodus. Viename tokių darbų autoriai Misiūnas, Meškauskas bei Juozapavičius pateikia EEG pikų atpažinimo algoritmą naudojant morfologinį filtrą [19]. Algoritmas paremtas morfologijos filtrais, tačiau rezultatui pagerinti gale dar nadojami ir papildomi filtrai išvalantys EEG artefaktus. Tyrimo metu buvo sukurtas įrankis paremtas aprašytuoju algoritmu leidžiantis palengvinti EEG parametrų apskaičiavimą, kurie dažniausiai apskaičiuojami rankiniu būdu.

Galiausiai autorių Deveikis bei Meškauskas elektroencefalogramų analizės darbe autoriai palygina keletą skirtingų dirbtinių neuroninių tinklų pritaikytų atpažinti EEG pikus [22]. Darbe EEG pikams atpažinti naudojami dviejų tipų dirbtiniai neuroniniai tinklai: neuroninis tinklas be grįžtamųjų ryšių – FNN (angl. feed-forward neural network) bei neuroninis tinklas su grįžtamaisiais ryšiais – RNN (angl. recurrent neural network). Analizuojami šių neuroninių tinklų EEG pikų atpažinimo gebėjimo tikslumai, bei kaip šie sprendimai padeda išvengti klaidingai indentifikuotų pikų.

1.4. Darbe naudoti duomenys

Tyrimui bus naudojami realių pacientų duomenys iš Vilniaus Santaros Vaikų Ligoninės. Duomenys surinkti ir dalis jų sužymėti gydytojos dr. Rūtos Samaitienės. Duomenys padengia kelias dešimtis skirtingų pacientų su idiopatine Rolando epilepsija bei struktūrine epilepsija dėl galvos smegenų pažeidimų. Dalis pateiktų EEG duomenų jau turi sužymėtus pikus, bei darinius, kurie vizualiai panašūs į piką, tačiau tokiais nėra.

Šiam tyrimui bus naudojami tik EEG turinčios sužymėtus pikų duomenis. Darbui buvo surinkti 12-os 4–10 metų amžiaus pacientų sergančių gerybine (Rolandine) epilepsija EEG duomenys gauti 2010-2018 metų laikotarpiu, kaip matyti 2 lentelėje. Kiekvienam EEG kanalui buvo pažymėta bent keletas pikų, bei darinių, kurie nėra pikai, tačiau yra vizualiai panašūs į piką.

2 lentelė. EEG duomenų apžvalga

Paciento nr.	**Lytis**	**Gimimo metai**	**EEG gavimo metai**	**Ligos Pobūdis**
1	V	2009	2016	RE
2	V	2012	2018	RE
3	V	2010	2016	RE
4	M	2008	2016	RE
5	V	2009	2016	RE
6	V	2008	2015	RE
7	M	2003	2010	RE
8	M	2008	2018	RE
9	V	2014	2018	RE
10	V	2010	2016	RE
11-1	V	2011	2018	RE
11-2	V	2011	2018	RE
12	M	2015	2018	RE

1.5. Programavimo kalba bei įrankiai

Šiame poskyryje bus aptariami visi įrankiai bei bibliotekos naudotos šiame darbe. Pateikiama jų nauda darbui, bei pasirinkimo priežastys.

Python

Python – tai nemokama, atviro kodo, bendro pobūdžio ir galinga programavimo kalba. Python programavimo kalbos kodas yra lengvai skaitomas, bei vengia sudėtingesnių programavimo kalbų skirybos ženklų bei įvairių anotacijų, vietoj to kodo kompiliavimo tvarka valdoma kodą indentuojant. Dėl kalbos paprastumo galima daugiau pastangų teikti į kompiuterinio mokymosi problemos aprašymą bei tobulinimą vietoj aiškinimosi, kaip veikia sudėtingas kodas.

Apart intuityvumo Python kalbai apstu bibliotekų bei įvairių plėtinių, skirtų skirtingoms dalykinėms sritims, žymiai palengvinančių darbą. Tokių bibliotekų implementacija kode padeda išvengti situacijos, kuomet daug laiko ir pastangų išeikvojama programuojant jau egsiztuojančiu įrankius užuot prototipuojant galutinį produktą.

Python programavimo kalba yra nepriklausoma nuo platformos, tad leidžia programuotojams bei tyrėjams kodą paleisti ant bet kurio kompiuterio be pakeitimų (arba su labai minimaliais pakeitimais). Python rašyta programinė įranga gali būti lengvai perkompiluojama ir paprastai išleidžiama bet kuriai platformai (macOS, Windows, Linux) be papildomų kalbos interpretatorių.

Python programavimo kalba yra itin populiari, tad visos bibliotekos yra puikiai ir detaliai dokumentuotos. Internete apstu python naudojimosi pamokų, išleista daugybė knygų. Python dirbtinio intelekto entuziastai turi savo forumus, kur dalinasi įvairiais sprendimais, padeda vieni kitiems išspręsti dažnas problemas.

EDFBrowser

„EDFBrowser“ – tai nemokamas, atviro kodo įrankis skirtas atidaryti įvairių signalų, kaip EEG, EKG ir kitų, duomenų failų greitai peržiūrai [28]. Įrankis pasirinktas dėl to, jog veikia ant visų populiariausių operacinių sistemų (macOS, Windows, Linux) bei suteikė galimybę greitai atidaryti EEG failus pateiktus .edf formatu, peržiūrėti failų antraštes, redaguoti kanalų pavadinimus.

MNE

„MNE“ – tai atviro kodo „Python“ programavimo kalbai skirta biblioteka. Pagrindinės paketo paskirtis yra darbas su EEG tipo duomenimis, jų apdorojimas bei vizualizacija [9]. Ši biblioteka pasirinkta dėl puikių darbo su .edf formato failais funkcijų, galimybės nuskaityti .edf failą, jį konvertuoti į kitus duomenų formatus, filtruoti pagal, laiką kanalus, bei atvaizduoti.

Keras

„Keras“ – tai giliojo mokymo biblioteka skirta „Python“ programavimo kalbai. Keras biblioteka integruojasi su „Tensorflow“ [1], ir pateikia supaprastintą prieigą prie giliojo mokymo metodų [4]. Vienas pagrindinių „Keras“ privalumų tai, kad galima modelį apmokyti naudojant arba procesorių arba vaizdo pokštę, šiuo atveju kompiuteryje nesant dedikuotam (žiūrėti 3 lentelę) vaizdo plokštės procesoriui buvo būtinybė modeliui apmokant naudoti procesoriaus skaičiavimo resursus.

Scikit-Learn

„Scikit-Learn“ – tai įrankių skirtų duomenų analizei biblioteka skirta „Python“ programavimo kalbai [21]. „Scikit-Learn“ biblioteka pateikia paprastus naudoti duomenų klasifikavimo metodus, tokius kaip atraminių vektorių klasifikatorius (angl. Support Vector Machine – *SVM*), sprendimų medžiai (angl. Decision Trees) bei daugelį kitų. Su klasifikatoriais lengva dirbti, jei lengvai keičiami tarpusavyje, veikia sparčiai, tad lengva iteruoti ieškant tinkamiausios parametrų konfigūracijos.

NumPy

„NumPy“ – tai mokslinių skaičiavimų įrankių biblioteka skirta „Python“ programavimo kalbai [12]. Buvo pasirinkta dėl didžiulės metodų darbui su masyvais bei matricomis bibliotekos. EEG duomenys buvo konvertuojami į 2D masyvą, tad kilo poreikis greitai veikiančiai bibliotekai sugebančiai susitvarkyti su dideliu kiekiu tokio pobūdžio duomenų.

PyWavelets

„PyWavelets“ – tai atviro kodo vilnelių transormacijos įrankių biblioteka skirta „Python“ programavimo kalbai [10]. Biblioteka pasirinkta dėl itin didelio kiekio palaikomų vilnelių šeimų grupių bei spartaus veikimo.

MatplotLib

„MatplotLib“ – tai išsami „Python“ programavimo kalbai skirta biblioteka suteikianti įrankius statinių, animuotų bei interaktyvių vizualizacijų kūrimui [13]. Šios bibliotekos pagalba buvo sugeneruoti signalo atkarpos vaizdai, bei darbe pateikiamos ROC kreivės.

1.6. Kompiuterinis mokymasis

Šiame mokslinio darbo poskyryje pateikiami įvairių kompiuterinio mokymosi (angl. *machine learning*) modelių analizė. Išsamiai aptariami darbe naudoti prižiūrimo mokymosi (angl. *supervised learning*) algoritmai: 2D konvoliuciniai neuroniniai tinklai, logistinė regresija, sprendimų medžiai, atraminių vektorių klasifikatorius bei AdaBoost metaalgoritmas.

3 lentelė. Techninės įrangos specifikacija

Komponentas	**Tipas/Specifikacija**
Operacinė sistema	macOS Big Sur, version 11.1
Procesorius	2.3GHz Dual-Core Intel i5
Operatyvioji atmintis	16GB 2133MHz LPDDR3
Grafikos Procesorius	Intel Iris Plus Graphics 640 1535MB
Kietasis diskas	251GB APPLE SSD SM0256L

Konvoliucinis neuroninis tinklas

Konvoliucinis neuroninis tinklas (angl. *C*onvolutional Neural Network) – vienas iš ANN pagrįstų giliojo mokymosi metodų, plačiausiai naudojamų vaizdų klasifikavimo užduotyse. Konvoliuciniai neuroniniai tinklai (toliau CNN) yra įpatingas giliojo neuroninio tinklo tipas apdorojantis duomenis išsidėstančius tinkleliu, tad puikiai tinkamas darbui tiek su signalų duomenimis, kuriuos galime laikyti 1D tinkleliu, tiek vaizdų duomenimis, kuriuos galime laikyti 2D taškų tinkleliu [8].

CNN struktūriškai yra panašus į daugiasluoksnį perceptroną, tačiau bent viename iš sluoksnių aliekama matematinė konvoliucijos matematinę operacija.

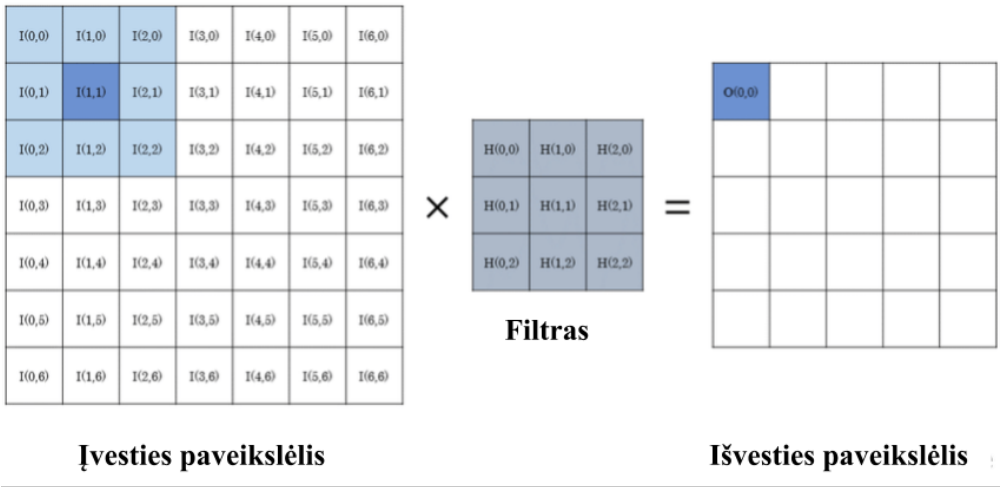
Dirbtiniai neuroniniai tinklai (toliau ANN) struktūriškai imituoja žmogaus nervų sistemos neuronų ryšius. ANN dažniausiai yra sudaromi iš daugelio pilnai sujungtų sluoksnių (angl. *fully-connected layer*).

Pagrindinis CNN įvesties elementas yra tenzorius (angl. *tensor*). Paprastas pavyzdys būtų tenzorius sudarytas iš vaizdo ir atitinkantis (H, W, 3) struktūrą. Čia H – eilučių kiekis (vaizdo aukštis taškais), W – stulpelių kiekis (vaizdo plotis taškais) bei 3 reprezentuoja įprasto paveikslėlio spalvinių kanalų kiekį (RGB). Ši iš tenzorių sudaryta įvestis panaudojama kiekviename ANN sluoksnyje ir priklausomai nuo sluoksnių struktūros bei pasirinkimo tarp sluoksnių transformuojama.

Kitas itin svarbus CNN elementas – konvoliucinis sluoksnis. Pavyzdžiui apdorojamas paveikslėlis (taškų matrica), kurios rezoliucija yra 7 x 7, o konvoliucijos operacijai pasirinktas filtras buvo 3 x 3, tuomet bus apdorojamos matricos 3 x 3 dydžio poaibiai padengiantys visą duomenų rinkinį, kiekvienam tokiam poaibiui atliekama konvoliucijos operacija ir gaunama nauja 5 x 5 dydžio matrica, kaip parodyta 4 paveikslėlyje.

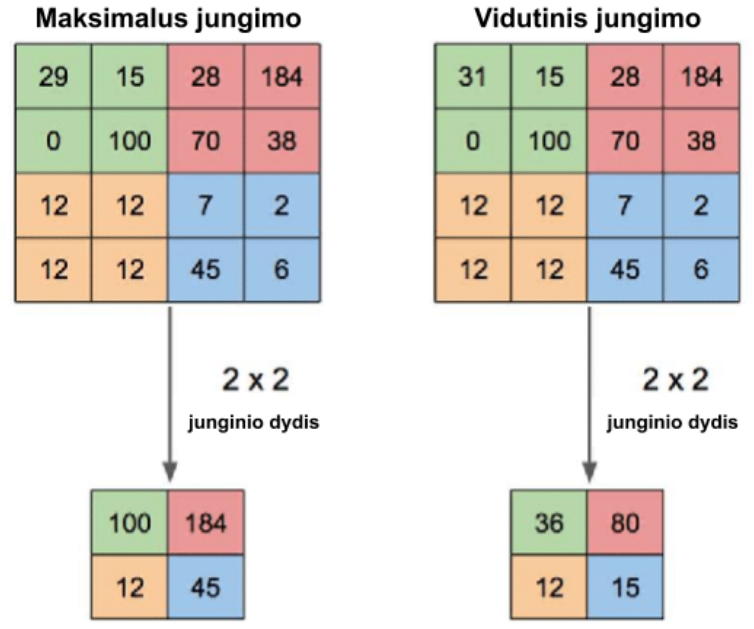
Įprastai tokia matricos operacija bus vykdoma padengiant visą matricą kiekvieną kartą filtrą pastumiant tik vieną tašką į dešinę arba žemyn. Toks modelis, gal kiek painiai, vadinamas bežingsniu (angl. *non-strided*), žingsnio dydis valdomas, dažnai didesnis žingsnis renkamas norint paspartinti CNN apmokymo spartą, tačiau padidinus žingsnį gali būti paaukota dalis modelio klasifikavimo tikslumo.

Jungimo sluoksnis – kitas itin svarbus CNN sluoksnis. Šis sluoksnis sumažina konvoliucinio sluoksnio rezultatu gautą matricą, dar labiau sumažinant kompiuterio resursų raikalavimą duomenų apdorojimui. Tačiau pagrindinė priežastis naudoti šį sluoksni – dominuojančių duomenų rinkinio atributų išrinkimas. Jungimo sluoksnis taip pat padeda sumažinti duomenų triukšmą.



4 pav. Konvoliucijos operacija 7 x 7 dydžio matricai [3].

Yra dviejų tipų jungimo sluoksniai: maskimalaus jungimo (angl. *max pooling*) ir vidutinio jungimo (angl. *average pooling*). Maksimalaus jungimo sluoksnis ir nurodyto filtro dydžio matricos išrenka maksimalią reikšmę, kai tuo tarpu vidutinio jungimo sluoksnis apskaičiuoja visų filtro dydžio matricoje pakliuvusių reikšmių vidurkį, kaip vaizduojama 5 paveikslėlyje.



5 pav. Skirtingų jungiamųjų sluoksnių vizualizacija [30].

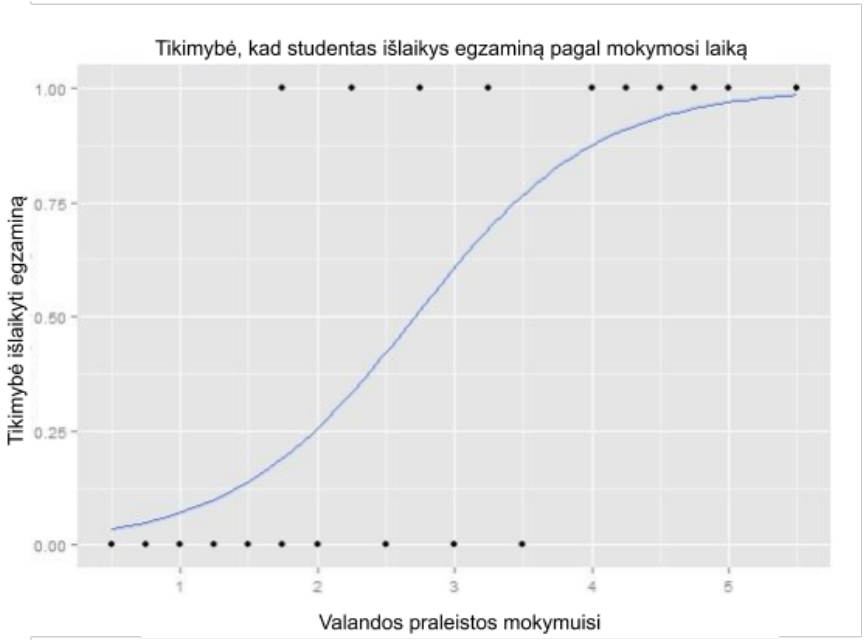
Taip apdoroti duomenys gali toliau būti naudojami įprastuose ANN sluoksniuose ir jais apmokomi klasifikavimo. Įprastai duomenys toliau apdorojami plokšinimo sluoksnio (angl. *flatten*) ir matrica paverčiama į duomenų vektorių (1D masyvą). Toks duomenų vektorius toliau yra teikiamas neuroniniam tinklui be grįžtamųjų ryšių (angl. *feed-forward neural network*).

Viena iš priežasčių, kodėl šiam darbui buvo pasirinkti CNN yra tai, kad CNN puikiai susidoroja su signalo pobūdžio duomenimis, ir galima teigti, jog CNN savo veikimu yra panašiausias metodas į rankinį pikų žymėjima, jeigu gydytojo pikų paiešką laikytume vaizdų analize.

Logistinė regresija

Viena pagrindinių logistinės regresijos panaudojimo sričių yra tikimybės, kad kažkas įvyks, arba tikimybės, kad pacientas bus diagnozuotas tam tikra liga įvertinimas [27]. Logistinė regresija yra dažnas statistinis metodas sąntykio, tarp pacientą apibūdinančių duomenų bei diagnozės, nustatymui.

Viena pagrindinių priežasčių, kodėl buvo pasirinktas šis modelis yra tai, jog modelis sukurtas spręsti binarinio klasifikavimo užduotį. Kitaip tariant pateiktiems duomenims grąžina tikimybė (tarp 0 ir 1) nurodančia priklausomybę vienai iš dviejų klasių, kaip vaizduojama 6 paveikslėlyje. EEG pikų atpažinimas šiame darbe traktuojama, kaip binarinės klasifikacijos užduotis.



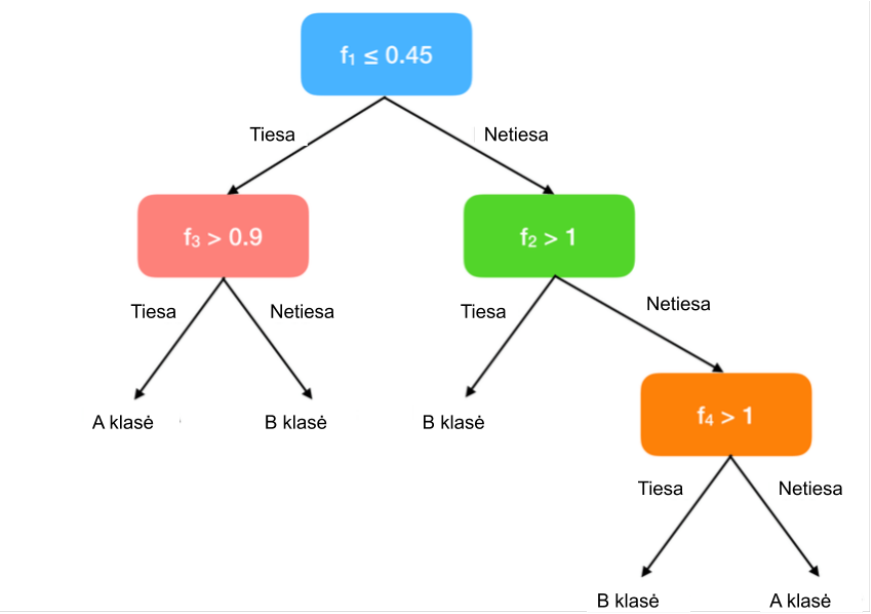
6 pav. Logistinės regresijos klasifikavimo pavydys. (Iliustracija yra vieša ir nėra saugoma autorinių teisių)

Sprendimų medžiai

Spendimų medis (angl. *Decision Tree*) – medžio struktūros klasifikacinis modelis. Sprendimų medis yra vienas paprasčiausių klasifikacinių modelių ir yra gana lengvai perprantamas, netgi ne ekspertams [24].

Sprendimų medis veikia duomenų atributų skaldymo principu. Sprendimų medžio algoritmas pasirenka geriausią atributą, kaip medžio šaknį ir iš ten skyla į šakas kiekvienam kitam atributui taip padengdamas visą duomenų rinkinį, kol galop pasiekia lapus, kuriose saugoma ne atributo bet klasifikacijos informacija. Taip sukuriami keliai nuo šaknies iki klasifikacijos kiekvienai atributų kombinacijai, kaip pavaizduota 7 paveikslėlyje.

Spendimų medžiai buvo pasirinkti, nes juos itin paprasta sukonstruoti, modelio apmokymas reikalauja itin mažai resursų, o rezultatai dažnai būna palyginamai geri kitiems, labiau, kompiuterio resursų prasme, reikliems modeliams.

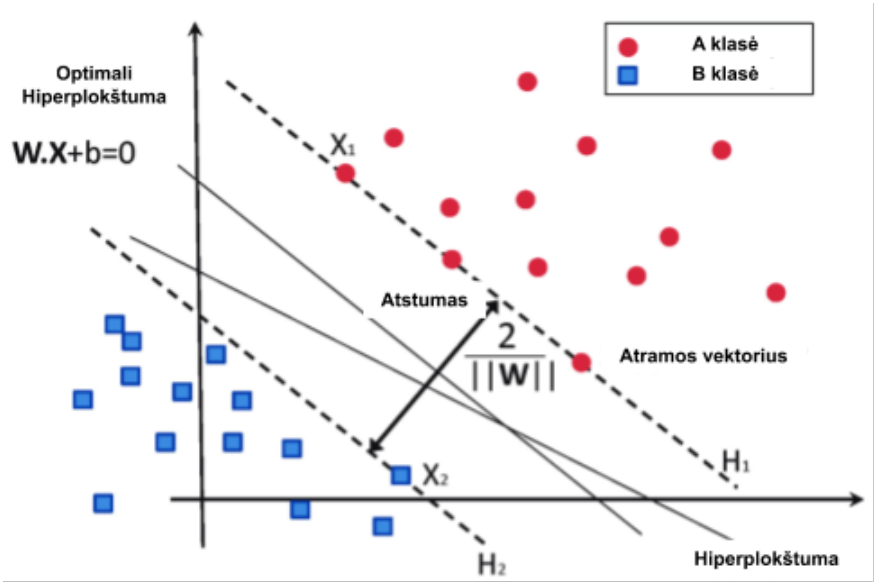


7 pav. Sprendimų medžio klasifikavimo pavyzdys [20] .

Atraminių vektorių klasifikatorius

Atraminių vektorių klasifikatorius (angl. *Support Vector machine*) – klasifikatorius, kurio algoritmo pagrindinis tiklas yra rasti optimalią hiperplokštumą (angl. *hyperplane*), kuri kuo geriau padalintų n-mačius duomenis į dvi išsiskiriančias klases. Jeigu duomenys yra n-mačiai, bus ieškoma $(n-1)$-dimensinės hiperplokštumos atskiriančios tiriamas klases. Kol tokių hiperplokštumų gali būti ne viena, optimizuojant modelį, ieškoma plokštuma, kurios atstumas (angl. *margin*) nuo bet kurių dviejų skirtingų klasių duomenų taškų būtų maksimalus [29].

Paprastas klasifikavimo atraminių vektorių klasifikacija pavyzdys pateiktas 8 paveikslėlyje, čia dvimatėje erdvėje duomenys dalinami hiperplokštuma, kuri yra tiesė.



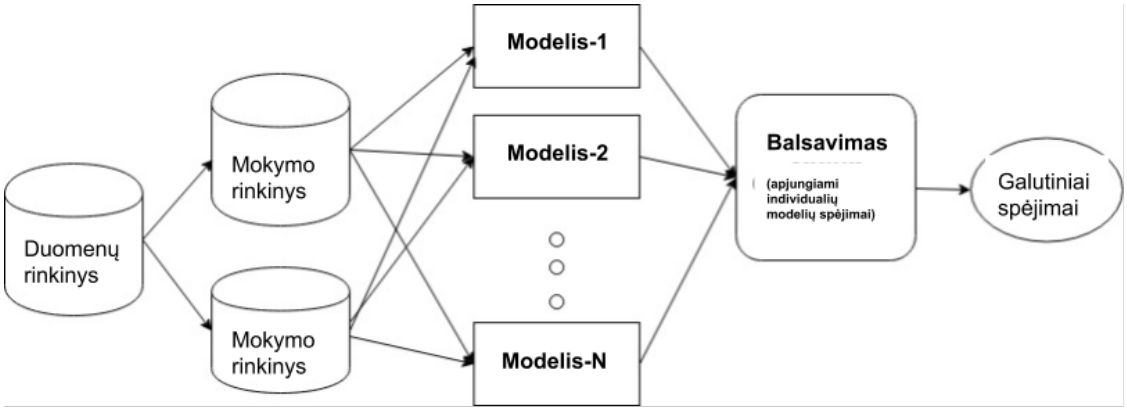
8 pav. Duomenų klasifikavimas naudojant atraminių vektorių klasifikatorių [7].

Atraminių vektorių klasifikatorius buvo pasirinktas, nes dėl savo veikimo principo – duomenų dalinimo į kategorijas bei prižiūrimojo mokymosi principų. Atraminių vektorių klasifikatorius yra

populiarus sprendimas binarinės klasifikacijos problemai.

Adaboost

AdaBoost (angl. *Adaptive Boosting*) – kitaip, nei kiti šiame poskyryje pateikti algoritmai pats nėra klasifikatorius, o klasifikavimo metaalgoritmas. Pagrindinis AdaBoost veikimo principas yra pasitelkiant daugybę prastai klasifikuojančių algoritmų (klasifikuojančių geriau nei 50% tikslumu) iš šios grupės sudaryti vieną geresnį klasifikatorių [6], kaip pavaizduota 9 paveikslėlyje.



9 pav. Adaboost metaalgoritmo veikimo pavyzdys. (Iliustracija yra vieša ir nėra saugoma autorinių teisių)

AdaBoost buvo pasirinktas, nes jis pats nėra klasifikatorius, tačiau naudoja kitus klasifikavimo algoritmus, kaip pagrindą. AdaBoost galima duoti kitus binariniam klasifikavimui tinkamus algoritmus šių rezultatui gerinti.

2. Duomenų apdorojimas

EEG duomenys tyrimui buvo pateikti .edf formatu [16], ir gauti naudojant 10-20 EEG sistemą. Prieš pradedant darbą su duomenimis, duomenų failų antrašės buvo redaguotos naudojant EDFBrowser [28] programą ir suvienodintos, tam, kad sutaptų EEG kanalų pavadinimai ir atitiktų formatą – „kanalas-RF“ (pvz. *FP1-RF*). Tuomet skirtingų pacientų failai buvo palyginti, tam kad būtų įsitikinta, jog visi failai buvo išgauti vienodais metodais, atitinka kanalų kiekis, duomenų dažnis, bei žymėjimas. 4 lentelėje pateikti šios analizės rezultatai.

Dėl ne vienodo kanalų kiekio tyrimas buvo atliekamas naudojant 18 kanalų tarpusavyje persidengiančių tarp visų 12 pacientų duomenų rinkinių: *Fp1-RF, Fp2-RF, F7-RF, F3-RF, Fz-RF, F4-RF, F8-RF, T3-RF, C3-RF, Cz-RF, C4-RF, T4-RF, T5-RF, P3-RF, Pz-RF, P4-RF, T6-RF, O1-RF*.

4 lentelė. EEG kanalų pasiskirstymas .edf failuose

Pacientas	**EEG kanalų kiekis**	**EEG kanalai**
1	21	C-RF, Fp1-RF, Fp2-RF, F7-RF, F3-RF, Fz-RF, F4-RF, F8-RF, T3-RF, C3-RF, Cz-RF, C4-RF, T4-RF, T5-RF, P3-RF, Pz-RF, P4-RF, T6-RF, O1-RF, O2-RF, MK-RF
2	23	F-RF, Fp1-RF, Fp2-RF, F7-RF, F3-RF, Fz-RF, F4-RF, F8-RF, T3-RF, C3-RF, Cz-RF, C4-RF, T4-RF, T5-RF, P3-RF, Pz-RF, P4-RF, T6-RF, O1-RF, O2-RF, 31HL-RF, 32HL-RF, MK-RF
3-12	26	F-RF, H-RF, A1-RF, Fp1-RF, Fp2-RF, A2-RF, F7-RF, F3-RF, Fz-RF, F4-RF, F8-RF, T3-RF, C3-RF, Cz-RF, C4-RF, T4-RF, T5-RF, P3-RF, Pz-RF, P4-RF, T6-RF, O1-RF, O2-RF, 31HL-RF, 32HL-RF, MK-RF

Darbe buvo naudojami tik gydytojo sužymėti duomenys. Kiekvienas .edf failas buvo konvertuotas į .csv formatą, su kuriuo yra lengviau dirbti Python aplinkoje, ir sukarpytas į 200 ms. gabalus aprėpiančius maksimalią vieno piko trukmę, viso išrenkant 50 įrašų per kanalą.

Paruošti duomenys buvo padalinti į du duomenų rinkinius skirtus mokymui ir testavimui 70% 30%. Kiekvienas duomenų segmentas, buvo sužymėtas (angl. labeled), kaip 0 – jeigu atstovauja netikrų pikų rinkiniui arba 1 – jeigu yra tikras EEG pikas.

Dižiausia problema su kuria buvo susidurta dirbant su šiais duomenimis – nesubalansuotas duomenų kiekis. Tyrimai šiame darbe atliekami su gydytojos sužymėtais pikais ir į piką panašiais dariniais, kurie pasiskirsto taip: 929 pikai ir 90 ne pikų.

3. Pikų atpažinimas

Šiame skyriuje aprašomi metodai naudoti pikų atpažinimui, metrikos kuriomis buvo vadovautasi, jei rezultatų gerinimo procesas.

Metrikos

Darbe pagrindinėmis laikomos šios metrikos:

- **Jautrumas** (angl. *Sensitivity*) – skaičius parodantis, kiek proporcingai teigiamai žymimų duomenų rinkinių buvo klasifikuoti teisingai.
- **Konkretumas** (angl. *Specificity*) – skaičius parodantis, kiek proporcingai neigiamai žymimų duomeny rinkinių buvo klasifikuoti teisingai.
- **Tikslumas** (angl. *Accuracy*) – skaičius parodantis, kiek proporcingai visoms klasifikacijoms duomenų rinkinių buvo klasifikuoti teisingai.
- **AUC** (angl. *Area Under Curve*) – skaičius parodantis, kaip tikėtina, jog modeliui suteikta nauja reikšmė bus klasifikuojama teisingai.

Tikslumas apskaičiuojamas formule:

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

Jautrumas apskaičiuojamas formule:

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

Konkretumas apskaičiuojamas formule:

$$Konkretumas = \frac{TN}{TN + FN}$$

Čia TP – teigiamai žymimos reikšmės, kurios buvo klasifikuotos, kaip teigiamos, TN – neigiamai žymimos reikšmės, kurios buvo klasifikuojamos, kaip neigiamos, FP – neigiamai žymimos reikšmės, kurios buvo klasifikuotos, kaip teigiamos bei FN – teigiamai žymimos reikšmės, kurios buvo klasifikuotos, kaip neigiamos.

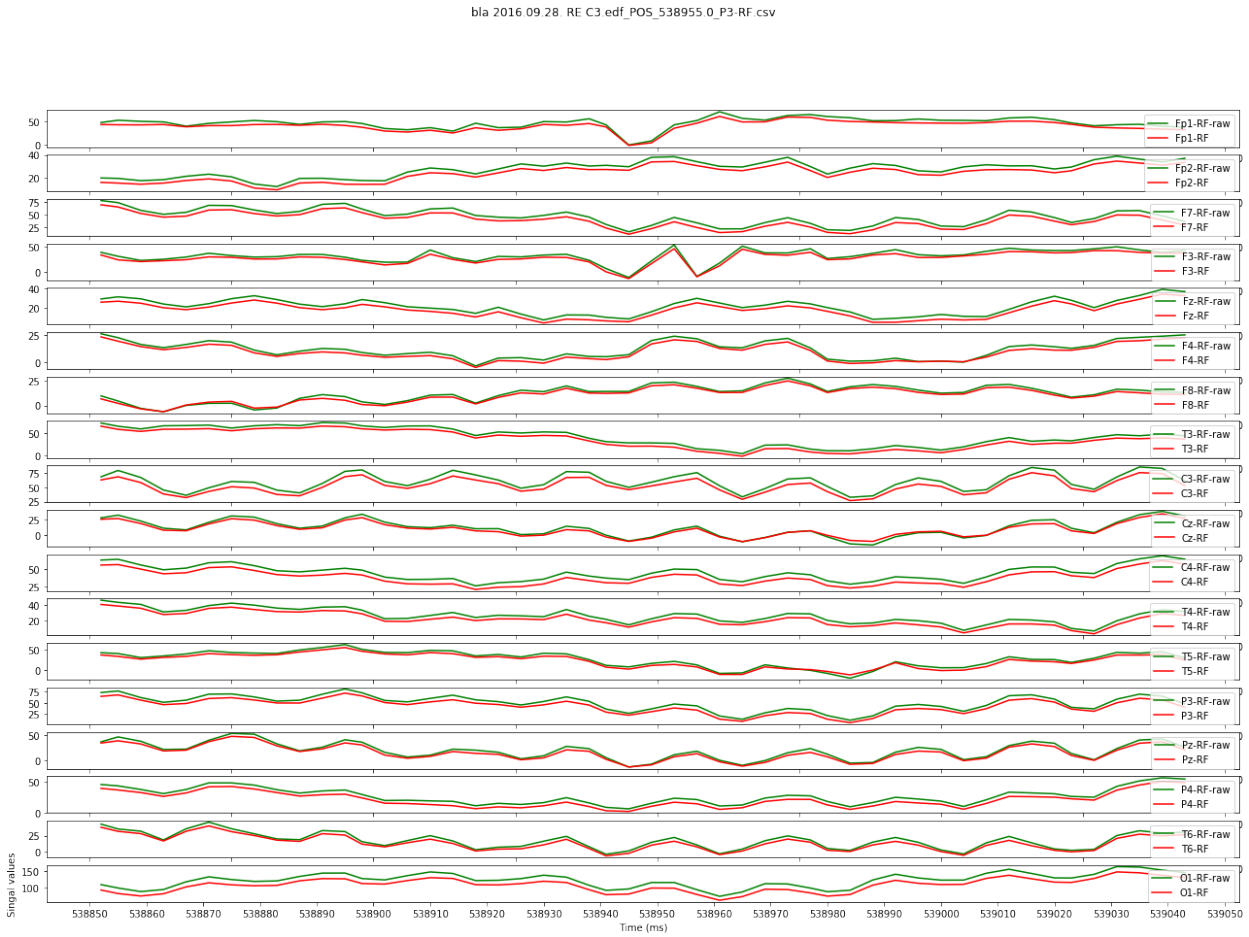
Turint tokį nesubalansuotą duomenų rinkinį itin svarbu atsižvelgti ne tik į modelio klasifikacijos tikslumą. Jeigu modelis sugeba atpažinti tik vieną iš dviejų klasių, ir tos klasės duomenų yra žymiai daugiau, tuomet tikslumo metrika bus proporcingai iškreipta. Žemas tikslumas visuomet rodo prastą klasifikavimo kokybę, tačiau aukštas tikslumas dar nereiškia geros klasifikavimo kokybės. Tokiu atveju itin svarbu atsižvelgti į kitas dvi metrikas jautrumą, bei specifiškumą, šiuo atveju aukštas jautrumas rodys, jog modelis puikiai sugeba atpažinti teigiamą piko atvejį, o aukštas specifiškumas – neigiamą. Didelis skirtumas tarp šių reikšmių, net esant aukštam tikslumui parodys tikrąją modelio binarinės klasifikacijos situaciją.

Signalo apdorojimas

Dažnai pasitaikanti pikų atpažinimo strategija yra signalo nutriukšminimas prieš apdorojant duomenis klasifikatoriumi. Vilnelių transformacijos yra itin dažnas EEG signalo nutriukšminimo metodas pikų aptikimo procese [11].

Šiuo darbu siekiant gauti kuo geresnius rezultatus pasirinktos „Daubechies“ vilnelių šeimos transformacijos „DB4“ bei „DB20“, kurios apibūdinamos, kaip itin efektyvios nutriukšminant duomenis prieš pikų atpažinimą [15].

Pradinė slenkstinė vertė (angl. *Treshold*) abiejų šeimų transformacijoms buvo pasirinkta – 0.1, kaip vaizduojama 10 paveikslėlyje. Transformacijos buvo taikomos visiems kanalų duomenims, ne tik tiems, kuriuose buvo žymimas pikas.



10 pav. 200ms. trukmės EEG iškarpa apdorota DB4.

3.1. Konvoliuciniai 2D neuroniniai tinklai

Darbe įgyvendintas kryžminio patikrinimo metodas (angl. *cross-validation*), duomenys buvo dalinami 10 kartų, kiekvieną kartą juos skirstant į 70% duomenų modelio apmokymui ir 30% duomenų testavimui. Šis metodas buvo taikomas norint gauti kuo tikslesnius klasifikavimo rezultatus, todėl visos darbe pateikiamos metrikos yra 10 kryžminio patikrinimo iteracijų vidurkis.

Darbe sprendžiamas binarinės klasifikacijos uždavinys, tad klasifikacijos rezultatai grąžinami, kaip tikimybė, jog duomenų rinkinys priklauso klasei. Čia buvo pasirinkta tolerancijos reikšmė lygi 0.5, jei kuri nors tikimybė yra lygi ar aukštesnė tolerancijos reikšmei, tos klasės nariu ir klasifikuojamas duomenų rinkinys.

CNN modelio grupės dydis (angl. *batch size*) yra lygus 70. Ši reikšmė parodo, koks duomenų kiekis bus naudojamas apmokant modelį vienoje iteracijoje. Abiejuose modelio sluoksniuse pradinis branduolių kiekis buvo lygus 16, o branduolių dydis – 2.

Pradinis CNN modelis buvo sudarytas iš dviejų 16 branduolių kiekio, bei 2 branduolių dydžio sluoksnių, po kurių sekė po vieną 2 branduolių dydžio jungiamąjį sluoksnį. Po konvoliucinių bei jungiamųjų sluoksnių sekė lyginamasis sluoksnis, bei du „Dense“ tipo daugiasluoksnio perceptrono sluoksniai. Modelio architektūra pateikta 18 lentelėje, dokumento B priede.

Abiems konvoliuciniams sluoksniams buvo naudojama išlyginto tiesinio vieneto aktyvacijos funkcija (angl. *Rectified Linear Unit (ReLU)*). Šiandien tai plačiausiai naudojama aktyvacijos funkcija. Funkcijos veikimo principas yra itin paprastas, nes ji tiesiog užnulina neigiamas reikšmes:

$$f(x) = max(0, x).$$

Pirminiai rezultatai

Pirmasis eksperimentas atliktas su 2D CNN modeliu buvo atliktas keičiant branduolių kiekį, kaip pavaizduota 5 lentelėje. Šiame eksperimente buvo naudojami vilnelių transformacijomis neapdoroti duomenys.

5 lentelė. Modelio rezultatai keičian branduolių kiekį

Branduolių kiekis	**AUC**	**Tikslumas**	**Jautrumas**	**Konkretumas**
16	0.59	0.989	0.9584	0.2296
32	0.61	0.989	0.9648	0.2696
64	0.61	0.986	0.9684	0.2592
128	**0.63**	**0.976**	**0.9587**	**0.2925**
256	0.60	0.965	0.9734	0.2185

Jau pirmasis bandymas su 16 branduolių grąžino neblogus rezultatus. Aukštas tikslumas ir jautrumas parodo, jog modelis puikiai sugeba išrinkti pikus, tačiau prastai skiria ne pikus. Bandymai buvo baigti po 256 branduolių, nes didinant branduolių kiekį toliau rezultatai prastėjo.

Geriausius rezultatus pavyko gauti su modeliu turėjusiu 128 branduolius abiejuose sluoksniuose (žiūrėti 6 lentelę), nors tikslumas ir jautrumas šiuo atveju nebuvo patys geriausi, tačiau šiuo atveju pavyko išgauti geriausią klasifikavimo konkretumą.

Rezultatų gerinimas

Pirmas žingsnis modelio rezultatų gerinime buvo modelį apmokyti naudojant duomenis apdorotus vilnelių transformacija. Bandymai buvo atliekami tiek su DB4 tiek su DB20 grupių trans-

6 lentelė. Modelio grąžinusio geriausius rezultatus architektūra

Sluoksnio tipas	Išeities forma	Parametrų skaičius
conv2d_1 (Conv2D)	(None, 49, 17, 128)	640
max_pooling2d_1 (MaxPooling2D)	(None, 24, 8, 128)	0
conv2d_2 (Conv2D)	(None, 27, 7, 128)	65664
max_pooling2d_2 (MaxPooling2D)	(None, 11, 3, 128)	0
flatten_1 (Flatten)	(None, 442)	0
dense_1 (Dense)	(None, 100)	422500
dense_2 (Dense)	(None, 2)	202

formacijomis apdorotais duomenimis. Pradinė slenkstinė vertė buvo 0.1, siekiant gauti geresnius rezultatus, ji buvo mažinta, kaip pateikta 7 lentelėje.

7 lentelė. Modelio rezultatai su skirtingomis vilnelių transformacijomis

Vilnelių šeima	Slenkstinė vertė	AUC	Tikslumas	Jautrumas	Konkretumas
DB4	0.1	0.61	0.970	0.9512	0.2740
DB20	0.1	0.60	0.976	0.9724	0.2222
DB4	0.05	0.60	0.970	0.9709	0.2370
DB20	0.05	0.61	0.959	0.9799	0.2333
DB4	**0.03**	**0.64**	**0.980**	**0.9526**	**0.3296**
DB20	0.03	0.62	0.977	0.9634	0.2703
DB4	0.02	0.61	0.966	0.9566	0.2629
DB20	0.02	0.62	0.976	0.9616	0.2814
DB4	0.01	0.63	0.979	0.9584	0.2925
DB20	0.01	0.61	0.976	0.9670	0.2481

Geriausius rezultatus su modeliu pavyko išgauti naudojant DB4 grupės transformacijas su 0.03 slenkstine verte. Šiuo atveju pats signalas buvo transformuotas minimaliai ypač apie piką (žiūrėti A priedą), tačiau transformuoti duomenys padėjo pasiekti didesnį tikslumą bei konkretumą. Visi tyrimai toliau buvo atliekami su duomenimis apdorotais naudojant šią grupę, bei slenkstinę vertę.

Siekiant įsitikinti, ar pirmojo eksperimento, kurio metu išrinktas modelis su 128 branduolių kiekiu abiejuose sluoksniuose, rezultatai vis dar validūs naudojant transformuotus duomenis. Eksperimentas buvo pakartotas su transformuotu signalu (žiūrėti 8). Rezultatai toliau išlieka tokie pat ir tolimesni eksperimentai buvo vykdomi naudojant modelį su 128 branduolių kiekiu abiejuose sluoksniuose.

8 lentelė. Pirmojo eksperimento pakartojimas su DB4

Branduolių kiekis	AUC	Tikslumas	Jautrumas	Konkretumas
16	0.59	0.982	0.9612	0.2185
32	0.60	0.989	0.9594	0.2444
64	0.62	0.989	0.9770	0.2629
128	**0.64**	**0.980**	**0.9526**	**0.3296**
256	0.60	0.969	0.9659	0.2296

Siekiant toliau pagerinti rezultatus su 2D CNN buvo keičiamas branduolio dydis kiekviename

sluoksnyje. Pirma buvo atliekami eksperimentai keičiant tik pirmojo sluoksnio branduolio dydį, tuomet gavus gerą rezultatą toliau keičiami antrojo sluoksnio branduolių dydžiai. Eksperimento rezultatai vaizduojami 9 lentelėje.

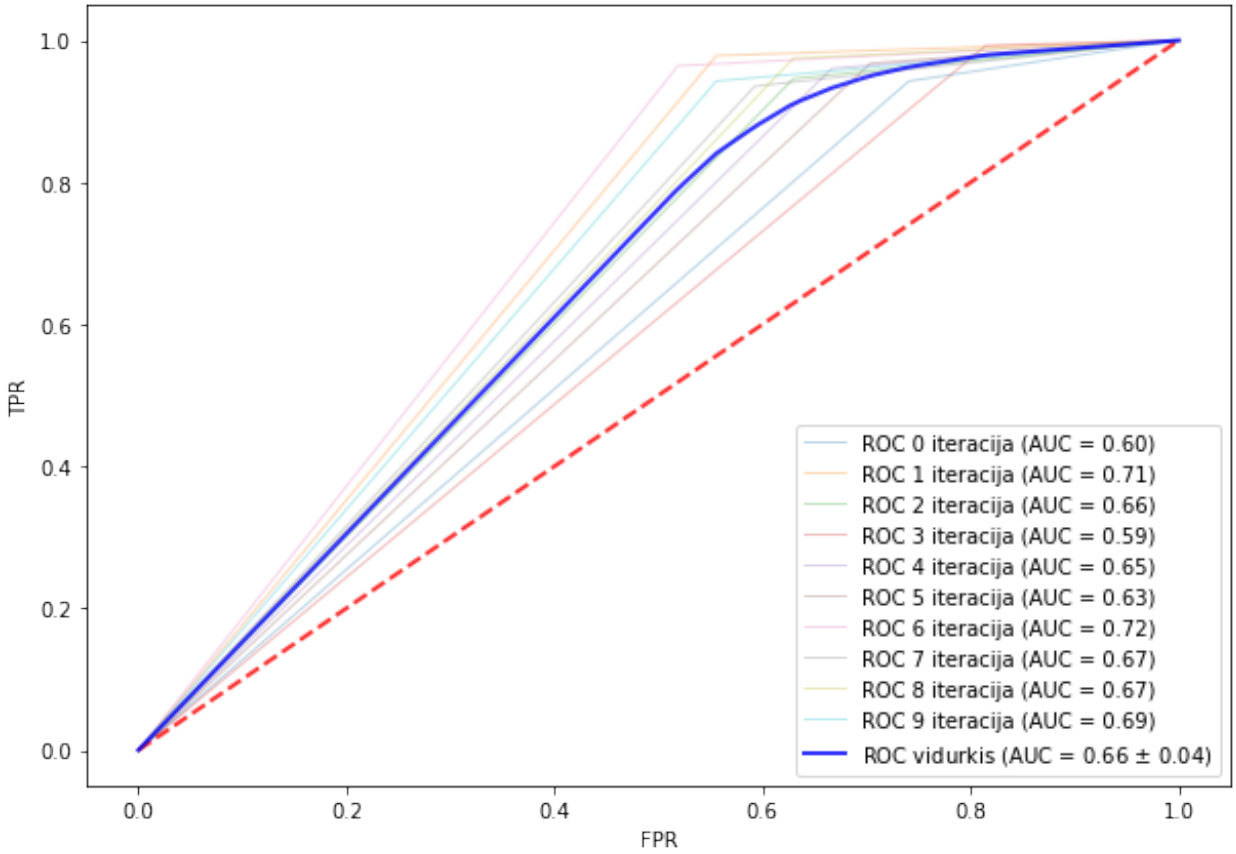
9 lentelė. Rezultatų gerinimas keičiant branduolių dydžius

Pirmasis sluoksnis	**Antrasis sluoksnis**	**AUC**	**Tikslumas**	**Jautrumas**	**Konkretumas**
2, 2	2, 2	0.64	0.980	0.9526	0.3296
3, 3	2, 2	0.62	0.978	0.9612	0.2814
4, 4	2, 2	0.60	0.965	0.9799	0.2222
5, 5	2, 2	0.63	0.980	0.9767	0.2740
6, 6	**2, 2**	**0.66**	**0.973**	**0.9605**	**0.3592**
7, 7	2, 2	0.63	0.978	0.9516	0.3037
8, 8	2, 2	0.66	0.977	0.9698	0.3444
9, 9	2, 2	0.59	0.964	0.9842	0.2000
6, 6	3, 3	0.66	0.981	0.9627	0.3518
6, 6	4, 4	0.60	0.977	0.9706	0.2259
6, 6	5, 5	0.60	0.981	0.9698	0.2370

Didinant pirmojo sluoksnio branduolių dydį tikslumo ir jautrumo reikšmės svyravo itin nedaug, tačiau pakėlus branduolio dydį iki 6 buvo gauta geriausia konkretumo reikšmė (žr. 11 paveikslėlį). Manipuliuojant antrojo sluoksnio branduolių dydžiu rezultatų pagerinti toliau nepavyko. Geriausius rezultatus su 2D CNN pavyko gauti naudojant modelį su dviem 2D CNN sluoksniais turinčiais 128 branduolius, su skirtingais branduolio dydžiais, žiūrėti 10 lentelę.

10 lentelė. Modelio grąžinusio pagerintus rezultatus architektūra

Sluoksnio tipas	**Išeities forma**	**Parametrų skaičius**
conv2d_1 (Conv2D)	(None, 45, 13, 128)	4736
max_pooling2d_1 (MaxPooling2D)	(None, 22, 6, 128)	0
conv2d_2 (Conv2D)	(None, 21, 5, 128)	65664
max_pooling2d_2 (MaxPooling2D)	(None, 10, 2, 128)	0
flatten_1 (Flatten)	(None, 2560)	0
dense_1 (Dense)	(None, 100)	256100
dense_2 (Dense)	(None, 2)	202



11 pav. CNN su architektūra grąžinusia geriausią rezultatą (žr. 10 lentelę) ROC grafikas.

3.2. Logistinė regresija

Dirbant su logistine regresija toliau buvo naudotas kryžminio patikrinimo metodas, duomenys dalinami 10 kartų bei skirstomi į 70% grupę modelio apmokymui, bei 30% grupę modelio testavimui. Poskyryje pateikiami rezultatai yra visų kryžminio patikrinimo metodo iteracijų vidurkiai.

Modeliui papildomai buvo pateikiami parametrai nurodantys duomenų sumaišymą (bibliotekoje parametras vadinamas – random state), optimizacijos algoritmą (bibliotekoje parametras vadinamas – solver), maksimalių iteracijų kiekį, bei problemos pobūdį (bibliotekoje parametras vadinams – multi-class). Visi pirminiai eksperimentai buvo vykdomi be duomenų sumaišymo ($randomstate = 0$) bei pasirinkus 1000, kaip maksimalų iteracijų kiekį.

Optimizacijos algoritmai:

- **Newton-CG** – Niutono metodas. Niutono metodai naudoja Hessian matricą, tad veikia lėtai dideliems duomenų rinkiniams, nes skaičiuojamos antros eilės išvestinės.
- **L-BFGS** – Limited-memory Broyden–Fletcher–Goldfarb–Shanno metodo trumpinys. Kaip ir niutono metodas skaičiuoja antros eilės išvestinių matricą, tačiau atmintyje saugo tik paskutines keletą iteracijų, tad reikalauja mažiau operatyviosios atminties. Veikia lėtai dideliems duomenų rinkiniams.
- **Sag** – Stochastic Average Gradient Descent metodo trumpinys. Gradientų nusileidimo algoritmo variantas. Greitai veikia ir su dideliais duomenų rinkiniais.
- **Saga** – Sag variantias, veikiantis sparčiau, nei sag.
- **Liblinear** – Library for Large Linear Classification trumpinys. Metodas naudoja koordinačių nusileidimo algoritmą. Puikiai tinka problemoms turinčioms daug dimensijų, tačiau prastai

veikia binarinio klasifikavimo problemoms.

Problemos pobūdis:

- **OvR** – One vs. Rest trumpinys. Nurodžius šį problemos pobūdį atliekama paprasta binarinė klasifikacija.
- **Multinomial** – Daugianario praradimo funkcija. Nurodžius šį problemos pobūdį pritaikoma daugianario praradimo funkcija kiekvienai iš klasifikuojamų kategorijų, net jei dirbama su binarine klasifikacija.

Pirminiai rezultatai

Pirmas eksperimentas buvo atliekamas modelui nurodant kiekvieną optimizacijos algoritmo bei problemos pobūdžio kombinaciją. Siekiant paprastesnio eksperimentų įvertinimo šiame žingsnyje maksimalių iteracijų kiekis, bei duomenų sumaišymo reikšmės buvo paliktos tokios pat. Visi modeliai buvo apmokyti naudojant duomenis apdorotus DB4 grupės vilnelių transformacija. Rezultatai pateikiami 11 lentelėje.

Eksperimentus naudojant liblinear optimizacijos algoritmą pavyko atlikti tik naudojant ovr, nes šis algoritmas palaiko tik binarinės klasifikacijos problemas.

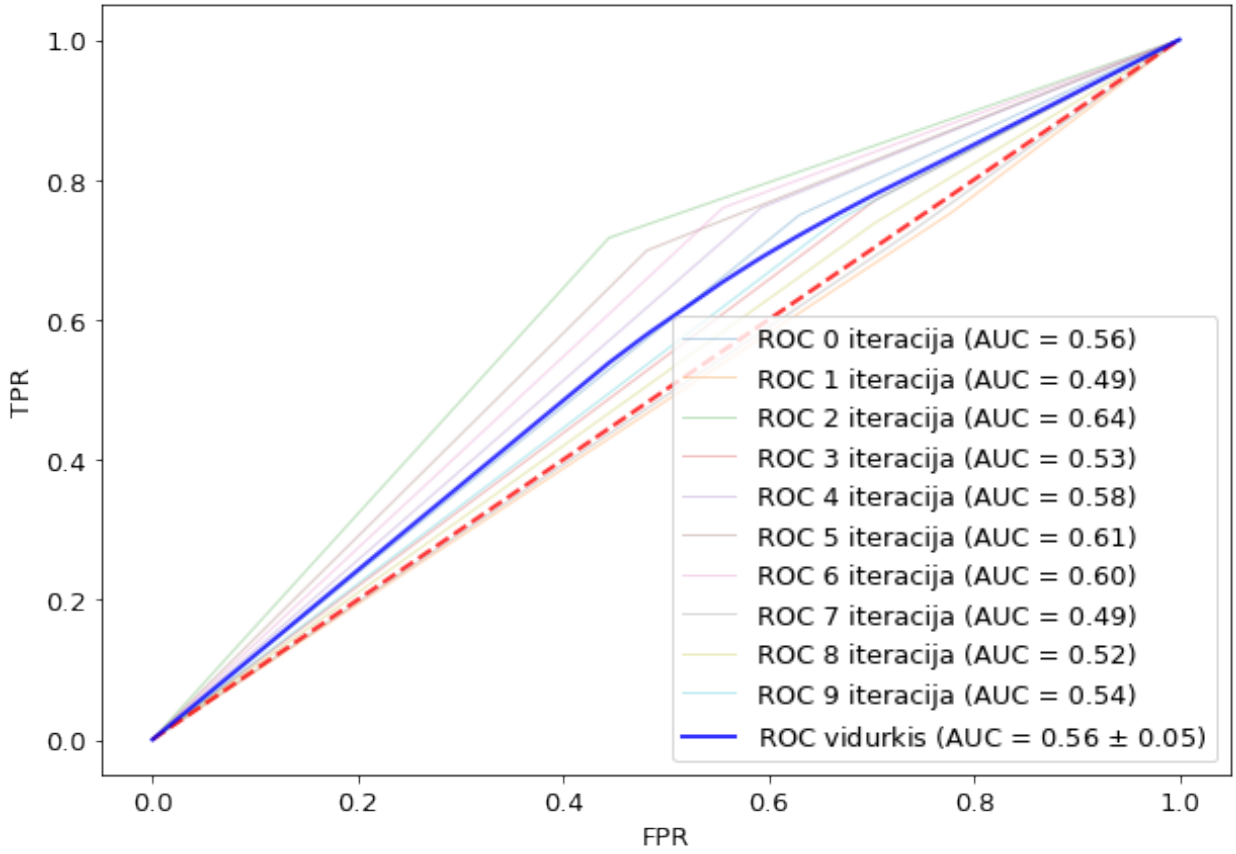
11 lentelė. Logistinės regresijos rezultatai

Solver	Multi-Class	Max Iteration	Tikslumas	Jautrumas	Konkretumas	AUC
newton-cg	multinomial	1000	0.8039	0.8738	0.2111	0.54
newton-cg	ovr	1000	0.8166	0.8752	0.2111	0.54
lbfgs	multinomial	1000	0.6346	0.6541	0.2111	0.54
lbfgs	ovr	1000	0.6323	0.6516	0.4333	0.54
sag	multinomial	1000	0.7026	0.7354	0.3629	0.55
sag	ovr	1000	0.7091	0.7422	0.3666	0.55
saga	**multinomial**	**1000**	**0.7088**	**0.7415**	**0.3703**	**0.56**
saga	ovr	1000	0.7137	0.7476	0.3629	0.56
liblinear	multinomial	1000	-	-	-	-
liblinear	ovr	1000	0.6401	0.6609	0.4259	0.54

Geriausius rezultatus pavyko gauti naustačius *„solver“* parametro reikšmę – *„saga“* bei *„multi-class“* parametro reikšmę – *„multinomial“*. Kol logistinės regresijos modeliai naudojantys niutono metodą grąžino geresnius tikslumo bei jautrumo rezultatus, o panaudojus Limited-memory Broyden–Fletcher–Goldfarb–Shanno metodą modelis grąžino geresnį konkretumo rezultatą buvo pasirinkta kombinacija grąžinusi balansą tarp tikslumo, jautrumo bei konkretumo metrikų reikšmių (žr. 12 paveikslėlį).

Rezultatų gerinimas

Toliau eksperimentai buvo atliekami manipuliuojant maksimalaus iteracijų kiekio, bei duomenų sumaišymo atributais modelyje. Tačiau keičiant šias reikšmes nebuvo pasiekti jokie reikšmingi modelio efektyvumo pagerėjimai.



12 pav. Geriausią rezultatą grąžinusio logistinės regresijos modelio ROC grafikas.

3.3. Sprendimų medžiai

Šiame poskyryje aprašomas eksperimente kurio metu buvo naudotas sprendimų medžio klasifikatoriaus modelis. Modeliui apmokyti buvo naudojami EEG duomenys apdoroti DB4 grupės vilnelių transformacija. Kiekviename iš bandymų buvo pateikiami šie papildomi modelio parametrai: funkcija matuojanti duomenų dalinimo kokybę (angl. *criterion*), strategija parodanti, kaip dalinti kiekvieną viršūnę (angl. *splitter*).

Funkcijos matuojančios duomenų dalinimo kokybę:

- **Gini** – Gini indeksas (angl. *gini impurity*). Matuoja, kaip dažnai duomenų rinkinio elementas bus klasifikuotas neteisingai jam parinkus atsitiktinės vertės etiketę. Jei gini indeksas pasiekia 0, tuomet ta viršūnė toliau nebus dalinama ir bus traktuojama, kaip unikali klasė.
- **Entropy** – Entropija matuoja, kaip pasirinkti duomenų rinkinio atributai koreliuoja su klase kurią bandoma klasifikuoti. Panašiai, kaip ir gini indeksas, mažesnė entropija reiškia konkretesnę klasifikavimo klasę.

Pirminiai rezultatai

Pirmas eksperimentas buvo atliekamas modeliui nurodant kiekvienos funkcijos matuojančios dalinimo kokybę bei dalinimo strategijos kombinaciją. Viso apmokyti keturi sprendimų medžio modeliai. Rezultatai pateikiami 12 lentelėje.

Geriausius rezultatus pavyko išgauti naudojant entropijos dalinimo kokybės funkciją su atsitiktinė (angl. *random*) dalinimo strategija. Kol kitos funkcijų kombinacijos modelyje grąžino aukštesnes konkretumo metrikas, tačiau entropijos ir atsitiktinės strategijos kombinacija grąžino

12 lentelė. Decision Tree rezultatai

Criterion	Splitter	Tikslumas	Jautrumas	Konkretumas	AUC
Gini	Best	0.8457	0.9003	0.2814	0.59
Entropy	Best	0.8640	0.9218	0.2666	0.59
Gini	Random	0.8437	0.9010	0.2518	0.58
Entropy	**Random**	**0.8647**	**0.9222**	**0.2703**	**0.60**

aukščiausią tikslumą, bei jautrumą, kartu su antru didžiausiu konkretumu, todėl ši kombinacija ir buvo pasirinkta, kaip grąžinusi geriausią rezultatą.

Rezultatų gerinimas

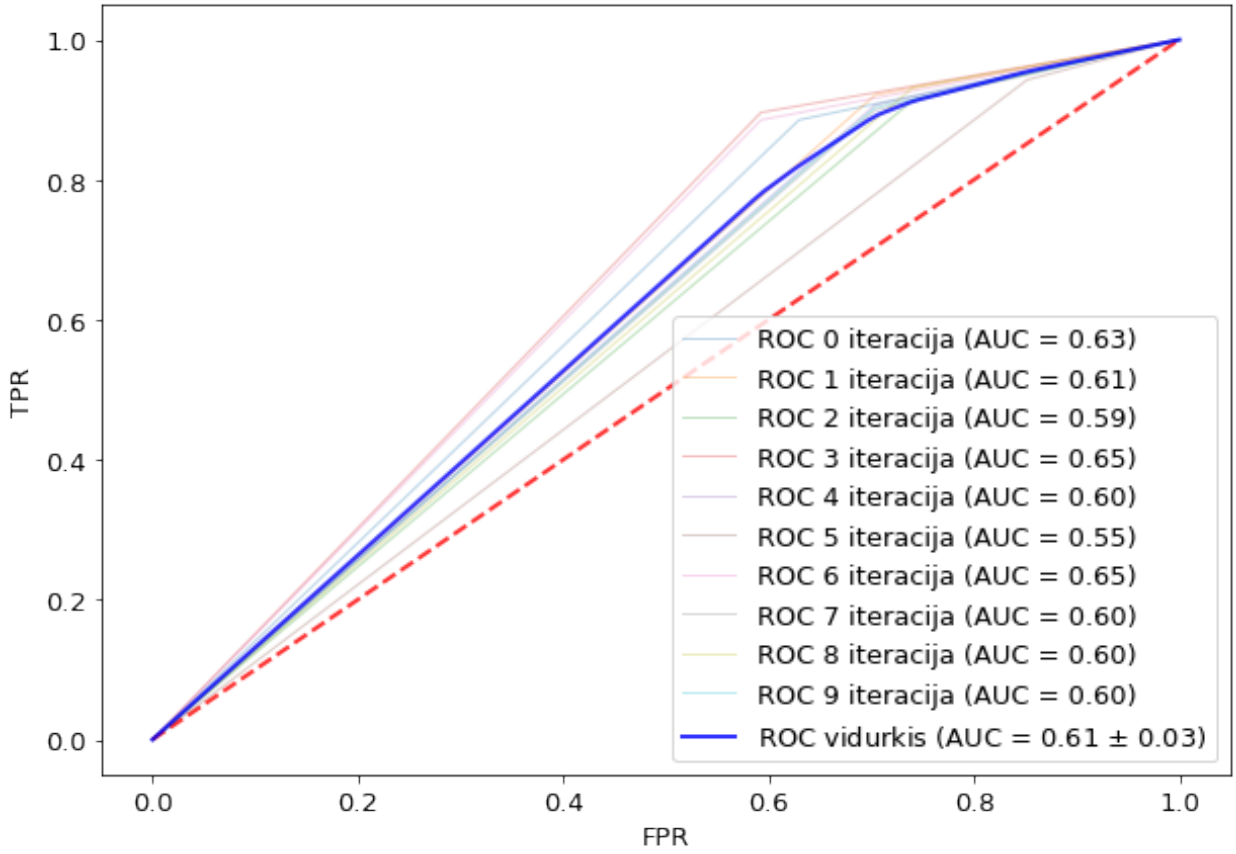
Toliau, siekiant pagerinti gautus rezultatus, eksperimentai buvo pakartoti naudojant smarkiai atsitiktinių medžių klasifikatorių (angl. *Etremely Randomized Trees Classifier*). Smarkiai atsitiktinių medžių klasifikatorius naudoja daugybę prastai klasifikuojančių spendimų medžių modelių gauti geresniam rezultatui.

Ekspermientai aprašyti 13 lentelėje buvo atliekami naudojant visas numatytas modelio reikšmes išskyrus dalinimo kokybę bei dalinimo strategiją.

13 lentelė. Extra Tree rezultatai

Criterion	Splitter	Tikslumas	Jautrumas	Konkretumas	AUC
Gini	Best	0.8568	0.9114	0.2925	0.60
Entropy	Best	0.8604	0.9240	0.2037	0.56
Gini	**Random**	**0.8535**	**0.9064**	**0.3074**	**0.61**
Entropy	Random	0.8584	0.9154	0.2703	0.59

Šiuo atveju rezultatai nėra galutiniai, nes pats atsitiktinių medžių klasifikatorius nėra iš savęs naudingas, o tikroji jo nauda atsiskleidžia naudojant jį kitame klasifikatoriuje mokančiame grupuoti daugybės klasifiatorių rezultatus. Geriausią rezultatą grąžino modelis, kuriame buvo nurodyta *Gini* indekso dalinimo kokybės funkcija bei atsitiktinės dalinmo strategijos kombinacija (žr. 13 paveiklsėlį). Ši konfigūracija bus naudojama tolesniems tyrimams su klasifikatorius agreguojančiais algoritmais.



13 pav. Geriausius rezultatus grąžinusio sperndimų medžio modelio ROC grafikas.

3.4. Atraminių vektorių klasifikatoriai

Šiame poskyryje aprašomas eksperimentas kurio metu naudotas atraminių vektorių klasifikatoriaus modelis. Modeliui apmokyti buvo naudojami EEG duomenys apdoroti DB4 grupės vilnelių transformacija. Kiekvienam iš bandymų buvo keičiami papildomi modelio parametrai nurodantys branduolio funkciją, bei sprendimų funkcijos forma.

Branduolio funkcijos:

- **RBF** – Radial basis function, tai viena paprastesnių skirstymo branduoliais forma, itin plačiai naudojama dėl jos panašumo Gauso paskirstymo funkcijai.
- **Linear** – Tiesinio skirstymo branduoliais funkcija yra bene pati parasčiausia tokio pobūdžio funkcija. Dažniausiai naudojama duomenų rinkiniams turintiems didelį kiekį parametrų, kurios galima paprastai padalinti į dvi grupes.
- **Poly** – Polinominio skirstymo branduoliais funkcija. Polinominio skirstymo funkcija žiūri ne tik į vieno duomenų rinkinio įvestį, tačiau į keletą duomenų rinkinių kombinacijų. Veikia greičiau, nei rbf, tačiau naudjama tuomet, kai negalima duomenų padalinti į dvi grupes tiesia linija.
- **Sigmoid** – Sigmoidinės funcijos skirstymo branduoliais funkcija. Ši funkcija savo veikimu panašiausia į dirbtinį neuroninį tinklą. Atraminių vektorių klasifikatoriaus ir šios funkcijos kombinacija dažnai naudojama paveikslėlių atpažinimo užduotims spręsti.

Pirminiai rezultatai

Pirmas eksperimentas buvo atliekamas modeliui nurodant kiekvienos branduolio funkcijos bei sprendimų funkcijos (14 lentelėje žymimos „DF“) kombinaciją. Viso buvo apmokyti 8 atraminių vektorių klasifikatoriumi paremti modeliai.

14 lentelė. SVC rezultatai

Kernel	DF	Max Iteration	Tikslumas	Jautrumas	Konkretumas	AUC
rbf	ovo	1000	0.9117	1.0000	0.0000	0.50
rbf	ovr	1000	0.9117	1.0000	0.0000	0.50
linear	**ovo**	**1000**	**0.7892**	**0.8480**	**0.1814**	**0.51**
linear	**ovr**	**1000**	**0.7892**	**0.8480**	**0.1814**	**0.51**
poly	ovo	1000	0.9068	0.9878	0.0703	0.53
poly	ovr	1000	0.9068	0.9878	0.0703	0.53
sigmoid	ovo	1000	0.8813	0.9598	0.0703	0.52
sigmoid	ovr	1000	0.8813	0.9598	0.0703	0.52

Rezultatai parodo, jog vienas prieš vieną (angl. *one vs. one (ovo)*) sprendimo funkcija yra visiškai neefektyvi sprendžiant pikų atpažinimo užduotį. Geriausią rezultatą pavyko išgauti su tiesine branduolių dalinimo funkcija, tačiau klasifikavimo konkretumas išgautas šiuo modeliu buvo itin prastas.

Rezultatų gerinimas

Siekiant pagerinti rezultatus eksperimentui buvo pasirinkta kita atramos vektorių klasifikavimo atšaka – Nu-Atramos vektorių klasifikatorius (angl. *Nu-Support Vector classifier*). Šio klasifikatoriaus veikimo principas panašus į paprasto artamos vektorių klasifikatoriaus, tačiau čia per „nu“ parametrą galima valdyti atramos vektorių kiekį modelyje.

Per keletą eksperimentų buvo nusistovėta ties $nu = 0.03$ reikšme. Visi eksperimentai naudojo tą pat nu reikšmę, ir buvo manipuliuojami tik keičiant branduolio funkciją. Rezultatai pateikiami 15 lentelėje.

15 lentelė. NuSVC rezultatai

Kernel	DF	Max Iteration	Tikslumas	Jautrumas	Konkretumas	AUC
rbf	**ovr**	**1000**	**0.8830**	**0.9408**	**0.2851**	**0.61**
linear	ovr	1000	0.7241	0.7670	0.2814	0.52
poly	ovr	1000	0.6016	0.6207	0.4037	0.51
sigmoid	ovr	1000	0.4019	0.3892	0.5333	0.46

Čia kitaip, nei paprasto atramos vektorių modelio atveju geriausią rezultatą grąžino modelis naudojantis rbf funkciją. Kur parasto atramos vektorių klasifikatoriaus atveju modelis aiškiai persimokė (angl. *overfitted*), kontroliuojant vektorių kiekį pavyko išgauti žymiai geresnį rezultatą.

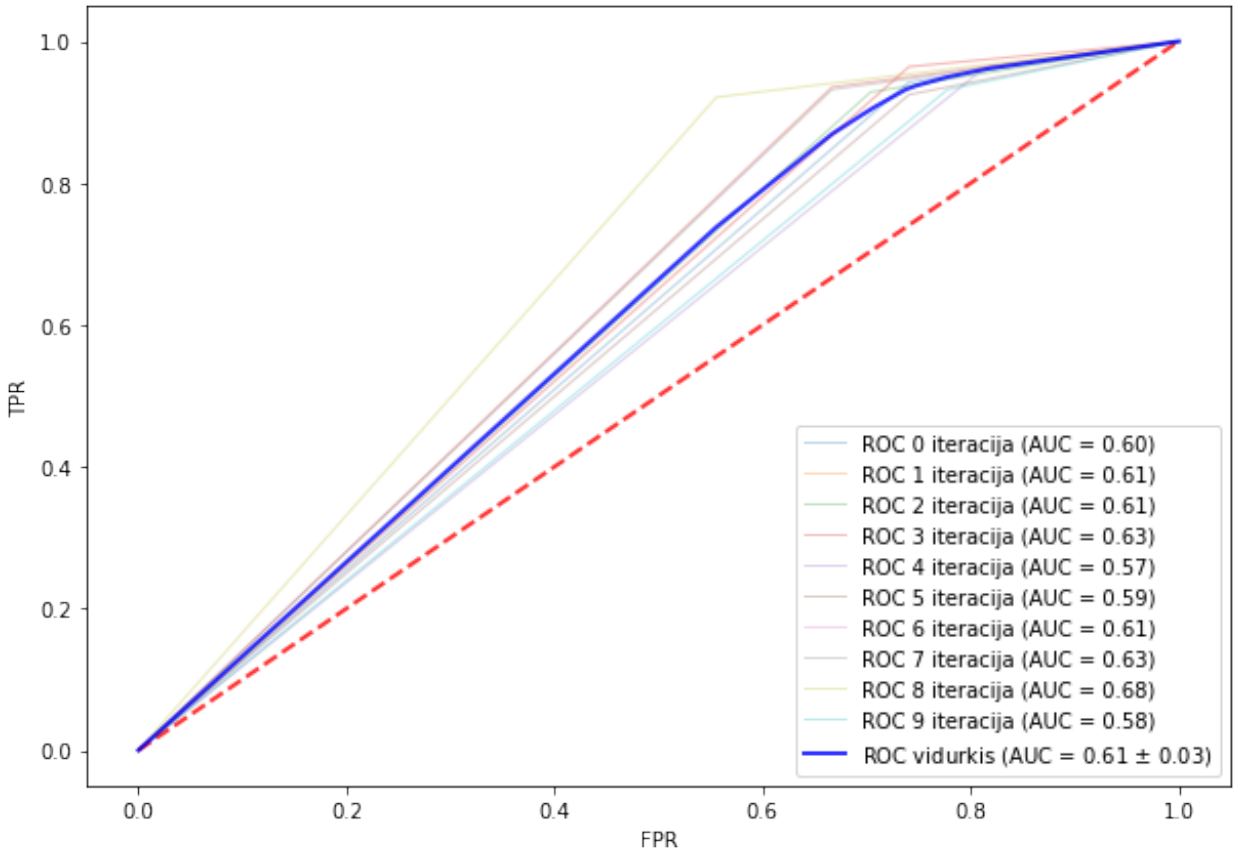
Dar vienas atramos vektorių modelio klasifikatoriaus atvejis yra tiesinis atramos vektorių klasifikatorius (angl. *Linear Support Vector Classifier*). Šis klasifikatorius yra naujas itin greitas paprasto atramos vektorių klasifikatoriaus variantas, kurio pagrindinė paskirtis yra daugiaklasė

16 lentelė. LinearSVC rezultatai

Loss	Multiclass	Tikslumas	Jautrumas	Konkretumas	AUC
squared_hinge	ovr	0.6382	0.6620	0.3925	0.53
hinge	ovr	0.6359	0.6584	0.4037	0.53
squared_hinge	crammer_singer	0.6336	0.6519	0.4444	0.55
hinge	**crammer_singer**	**0.6310**	**0.6487**	**0.4481**	**0.55**

klasifikacija su didžiuliais duomenų rinkiniais. Šis klasifikatorius nėra pritaikytas pikų aptikimo problemai spręsti, tačiau ir su juo buvo atlikti keletas bandymų aprašytų 16 lentelėje.

Visi bandymai su tiesiniu atramos vektorių klasifikatoriumi grąžino bene prasčiausius rezultatus. Tad šių eksperimentų metu geriausi rezultatai buvo pasiekti naudojant nu-atramos vektorių klasifikatoriumi naudojant „rbf“ branduolio funkciją. Rezultatai atvaizduojami 14 paveikslėlyje.



14 pav. Geriausius reultatus grąžinusio atraminių vektorių klasifikatoriaus modelio ROC grafikas.

3.5. AdaBoost

Šiame poskyryje aprašomi eksperimentai kurių metu buvo naudojamas AdaBoost klasifikavimo metaalgoritmas. Visi eksperimentai atlikti pasirenkant geriausiai pasirodžiusio klasifikatoriaus konfigūraciją (po vieną iš sprendimų medžių, Logistinės regresijos bei atramos vektorių mašinos) ir manipuliuojant algiritmo parametrais stengiantis pasiekti geriausią įmanomą rezultatą.

Eksperimentai buvo atliekami manipuliuojant koks algortimas bus naudojamas klasifikavimo metu, bei koks klasifikatorius bus optimizuojamas.

Algoritmai:

- **SAMME** – Konkrečių reikšmių algoritmas, grąžinantis 0 arba 1.
- **SAMME.R** – Tikimybių algoritmas, grąžinantis priklausymo tam tikrai klasei tikimybę. Įprastai veikia greičiau, nei SAMME bei turi už jį mažesnę paklaidą.

Pirminiai rezultatai

Pradinis eksperimentas buvo atliekamas metaalgoritmui nurodant kiekvieno nagrinėto klasifikatoriaus variantą, bei vieną iš dviejų algoritmų. Taip viso išbandyti 6 AdaBoost variantai, kurių rezultatai išdėstyti 17 lentelėje.

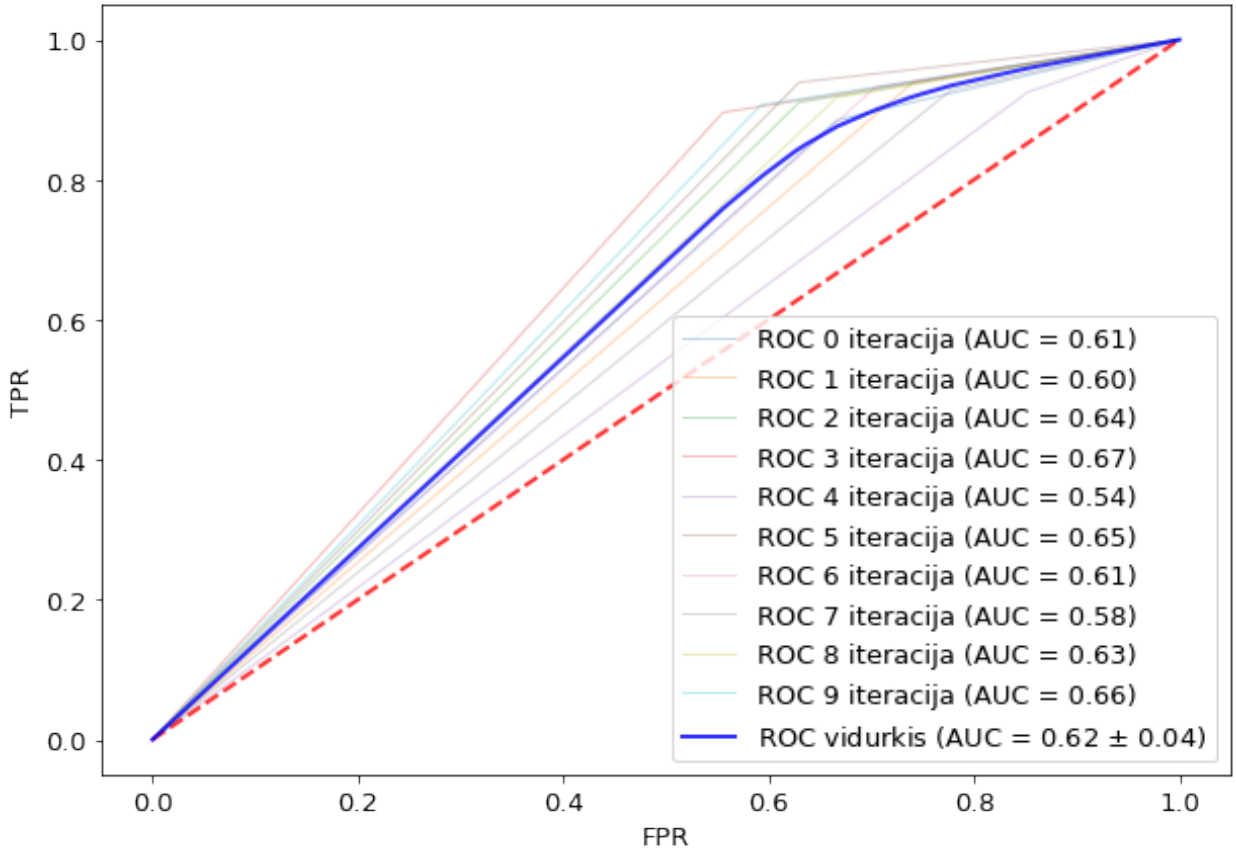
17 lentelė. Adaboost rezultatai

Bazė	Learning Rate	Algorithm	Tikslumas	Jautrumas	Konkretumas	AUC
DecTree	1	SAMME	0.8470	0.9021	0.2777	0.59
DecTree	**1**	**SAMME.R**	**0.8650**	**0.9179**	**0.3185**	**0.62**
LogReg	1	SAMME	0.8075	0.8652	0.2111	0.54
LogReg	1	SAMME.R	0.8075	0.8652	0.2111	0.54
NuSVC	1	SAMME	-	-	-	-
NuSVC	1	SAMME.R	-	-	-	-

Rezultatų lentelė parodo, kad eksperimentai su Nu-atramos vektorių klasifikatoriumi buvo nepavykę. Adaboost nepalaiko nu-atramos vektorių klasifikatoriaus naudojimo kartu su „SAMME.R“ algoritmu, o algoritmui „SAMME“ metaalgoritmas grąžino pranešimą, jog klasifikatoriaus rezultatai yra prastesni, nei atsitiktinio, todėl su juo negali toliau dirbti. Čia geriausią rezultatą pavyko išgauti naudojant smarkiai atsitiktinių medžių klasifikatorių (žr. 16 paveikslėlį). Naudojant šį klasifikatorių, kaip bazę buvo gautas geresnis rezultatas, nei su įprastu sprendimų medžiu, nors ir nežymiai.

Rezultatų gerinimas

Toliau buvo bandyta gerinti AdaBoost naudojančio smarkiai atsitiktinių medžių klasifikatorių, kaip bazę rezultatus keičiant mokymosi greitį, bei iteracijų kiekį. Tačiau šių parametrų manipuliacija negrąžino jokių reikšmingų rezultatų.



15 pav. Geriausius rezultatus grąžinusios AdaBoost konfigūracijos ROC grafikas.

Išvados ir rekomendacijos

Šiame darbe buvo siekiama atlikti pacientų sergančių gerybine (rolandine) epilepsija elektroencefalogramų tyrimą, siekiant išmokyti įvairius modelius klasifikuoti signalo gabalą, kaip piką ir jį atskirti nuo vizualiai panašių, tačiau jokios medicininės naudos neturiunčius darinius (darbe vadinamus ne pikais). Siekiant šių tiklsų pirmiausia aprašyta elektoencefalografijos sritis (žr. 1 skyrių) taip pat aprašyta tarptautinė 10-20 sistema (žr. 1.1 skyrių) ir praktiniai EEG taikymai (žr. 1.2 skyrių). Toliau darbe buvo apžvelgiami ir analizuojami realūs duomenys (žr. 1.4 skyrių), bei įrankiai (žr. 1.5 skyrių) kurių pagalba buvo atliekamas tyrimas. Šiame darbe buvo gauti tokie rezultatai bei išvados:

- Apmokius 2D CNN bei šiam pritaikius kryžminės validacijos metodą buvo pasiektas 0.976 tikslumas, 0.9587 jautrumas, 0.2925 konkretumas, bei 0.63 AUC metrika (žr. 3.1 skyrių).
- Apmokius 2D CNN naudojant vilnelių transformacija apdorotus duomenis buvo pasiektas 0.973 tikslumas, 0.9605 jautrumas, 0.3592 konkretumas, bei 0.66 AUC metrika (žr. 3.1 skyrių).
- Signalo duomenims pritaikius vilnelių transformacijas pavyko gauti geresnius rezultatus, nes, dėl pasirinktų vilnelių šeimos, duomenys buvo transformuojami minimaliai, ypač apie piką. Taip iš esmės išryškinant esminę signalo dalį bei sumažinant triukšmą.
- Apmokius logistinės regresijos modelį naudojant vilnelių transformacija apdorotus duomenis buvo pasiektas 0.7088 tikslumas, 0.7415 jautrumas, 0.3703 konkretumas, bei 0.56 AUC metrika (žr. 3.2 skyrių).
- Apmokius sprendimų medžio modelį naudojant vilnelių transformacija apdorotus duomenis buvo pasiektas 0.8535 tikslumas, 0.9064 jautrumas, 0.3074 konkretumas, bei 0.61 AUC metrika (žr. 3.3 skyrių).
- Apmokius atraminių vektorių klasifikatoriumi paremtą modelį naudojant vilnelių transformacija apdorotus duomenis buvo pasiektas 0.883 tikslumas, 0.9408 jautrumas, 0.2851 konkretumas, bei 0.61 AUC metrika (žr. 3.4 skyrių).
- Įgyvendinus atraminių vektorių klasifikatorių, jog pikų atpažinimo užduočiai spręsti sprendimo funkcija vienas prieš vieną (angl. *one vs. one (ovo)*) yra visiškai neefektyvi (žr. 14 lentelę).
- Atraminių vektorių klasifikatoriumi nepavyko išgauti geresnių rezultatų, nei su 2D CNN, tačiau verta atkreipti dėmesį į *„NuSVC“* klasifikatorių, kurio pagalba buvo gautas pati didžiausia klasifikavimo konkretumo reikšmė, nors naudojant šį algoritma gautos bene žemiausios tikslumo bei jautrumo metrikos(žr. 15 lentelę). Buvo pasiektas 0.4019 tikslumas, 0.3892 jautrumas, 0.5333 konkretumas.
- Atrinkus geriausias kiekvieno klasifikatoriaus konfigūracijas ir jas sudėjus į AdaBoost metaalgoritmą buvo pasiektas 0.8650 tikslumas, 0.9179 jautrumas, 0.3185 konkretumas, bei 0.62 AUC metrika (žr. 3.5 skyrių).
- Geriausius EEG pikų paieškos rezultatus pavyko pasiekti naudojant 2D konvoliucinį neuroninį tinklą. Sudarytą iš dviejų konvoliucinių bei dviejų „Dense“ tipo sluoksnių. Tačiau

net ir su geriausią rezultatą pasiekusiu modeliu toliau dirbti būtų sunku, nes jis nors ir puikiai atpažįsta jam paduotus signalo gabalus kuriuose vaizduojamas EEG pikas, kaip parodo 0.9605 jautrumo reikšmė, tačiau vos trečdalį į piką panašių darinių (ne pikų) sugeba klasifikuoti teisingai, kaip rodo 0.3592 konkretumas, tad modelis kartu su pikais klasifikuoja ir daug šiukšlių.

- Ištyrus įvairių kompiuterinio mokymosi algoritmų veikimą šiai problemai, hipotezė pasitvirtino tik iš dalies. Pasitelkian šiame darbe aprašomus 5 algoritmus nepavyko gauti modelio, kuris vienu metu sugebėtų atpažinti pikus ir juos atskirti nuo ne pikų su pakankamomis jautrumo bei konkretumo reikšmėmis. Darbu siekta gauti modelį, kurio jautrumo bei konkretumo metrikos siekė bent > 0.5.

Ateities tyrimų planas

Toliau dirbant su šiais duomenimis galima būtų naudoti kitus nutriukšminimo metodus, taip pat šiame darbe neišbandytas kitas vilnelių šeimas signalo transformacijai.

Siekiant geresnių klasifikavimo rezultatų galima darbą tęsti bandant sukurti antraeilį klasifikatorių priimantį pirmojo klasifikatoriaus rezultatus, kaip potencialias reikšmes ir jas klasifikuodamas iš naujo.

Pasiekus pakankamą EEG pikų atpažinimo tikslumą galima darbą toliau plėtoti ir ligos pagal aptiktą piką nustatymo srityje.

Literatūros šaltiniai

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Rūta Samaitienė Andrius Vytautas Misiukas Misiūnas, Tadas Meškauskas. Algorithm for automatic EEG classification according to the epilepsy type: Benign focal childhood epilepsy and structural focal epilepsy. *Biomedical Signal Processing and Control*, 48:118 – 127, 2019.

[3] Chaim Baskin, Natan Liss, Avi Mendelson, and Evgenii Zheltonozhskii. Streaming architecture for large-scale quantized neural networks on an fpga-based dataflow platform. 07 2017.

[4] François Chollet et al. Keras. https://keras.io, 2015.

[5] Tim Cox. Source of EEG activity, 2007.

[6] Yoav Freund and Robert E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[7] Esperanza García-Gonzalo, Zulima Fernández-Muñiz, Paulino Jose Garcia Nieto, Antonio Sánchez, and Marta Menéndez. Hard-rock stability analysis for span design in entry-type excavations with learning classifiers. *Materials*, 9:531, 06 2016.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[9] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti Hämäläinen. Meg and eeg data analysis with mne-python. *Frontiers in Neuroscience*, 7:267, 2013.

[10] Filip Wasilewski Kai Wohlfahrt Aaron O'Leary. Gregory R. Lee, Ralf Gommers. Pywavelets: A Python package for wavelet analysis. *Journal of Open Source Software*, 4(36):1237, 2019.

[11] Jonathan J. Halford. Computerized epileptiform transient detection in the scalp electroencephalogram: Obstacles to progress and the example of computerized ecg interpretation. *Clinical Neurophysiology*, 120(11):1909 – 1915, 2009.

[12] Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[13] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[14] K.P. Indiradevi, Elizabeth Elias, P.S. Sathidevi, S. Dinesh Nayak, and K. Radhakrishnan. A multi-level wavelet approach for automatic detection of epileptic spikes in the electroencephalogram. *Computers in Biology and Medicine*, 38(7):805 – 816, 2008.

[15] T. Kalayci and O. Ozdamar. Wavelet preprocessing for automated neural network detection of eeg spikes. *IEEE Engineering in Medicine and Biology Magazine*, 14(2):160–166, 1995.

[16] B. Kemp, A. Varri, A. C. Rosa, K. D. Nielsen, and J. Gade. A simple format for exchange of digitized polygraphic recordings. *Clinical Neurophysiology*, 82: 391–393, 1992.

[17] Cade Metz. *Google Just Open Sourced TensorFlow, Its Artificial Intelligence Engine*, 2015 (skaityta March 3, 2020).

[18] Andrius Vytautas Misiukas Misiūnas, Tadas Meškauskas, and Ruta Samaitienė. Derivative parameters of electroencephalograms and their measurement methods. *Lietuvos matematikos rinkinys*, 57, 12 2016.

[19] Andrius Vytautas Misiukas Misiūnas, Tadas Meškauskas, and Algimantas Juozapavičius. On the implementation and improvement of automatic eeg spike detection algorithm. *Lietuvos matematikos rinkinys*, 56, 2015.

[20] Ioannis Mollas, Grigorios Tsoumakas, and Nick Bassiliades. Lionforests: Local interpretation of random forests through path selection, 11 2019.

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

[22] Tadas Meškauskas Rokas Mykolas Deveikis. Analysis of electroencephalograms : Application of artificial neural networks for detection of epileptic discharges, 2018.

[23] Rūta Samaitienė. Rolando epilepsija sergančių vaikų EEG pakitimų, miego bei elgesio sutrikimų ir klinikinių charakteristikų sąsajos, 2013.

[24] Claude Sammut and Geo rey I. Webb, editors. *Encyclopedia of Machine Learning and Data Mining*. Springer Reference. Springer, New York, 2 edition, 2017.

[25] William O. Tatum. *Handbook of EEG Interpretation*. Demos Medical Publishing, 2014.

[26] Teplan M. FUNDAMENTALS OF EEG MEASUREMENT. *Measurement Science Review*, 2(2):1–11, 2002.

[27] Juliana Tolles and William J Meurer. Logistic regression: Relating patient characteristics to outcomes. *JAMA*, 316(5):533—534, August 2016.

[28] Teunis van Beelen. Edfbrowser, 2020.

[29] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

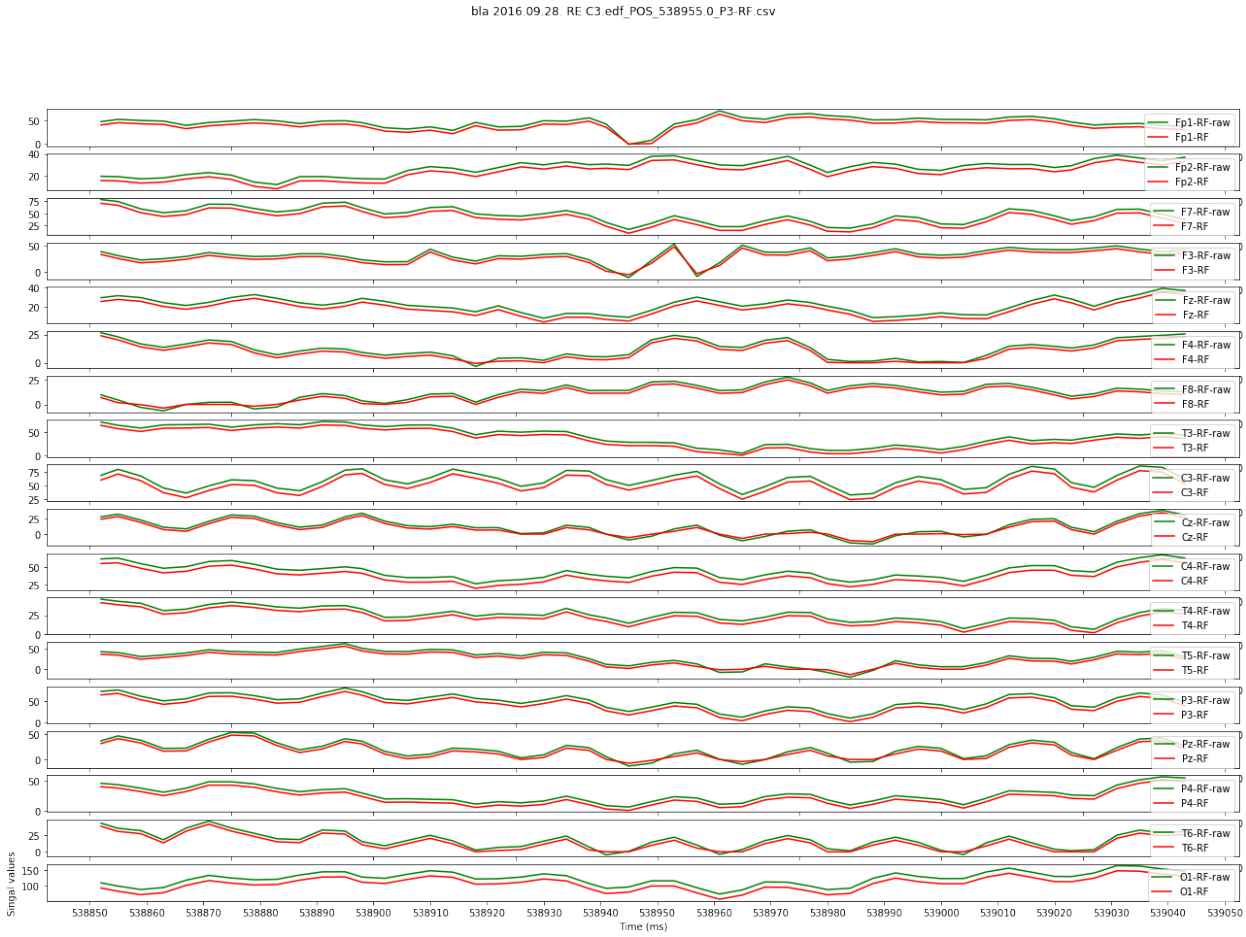
[30] Muhamad Yani, S Irawan, and M.T. S.T. Application of transfer learning using convolutional neural network method for early detection of terry's nail. *Journal of Physics: Conference Series*, 1201:012052, 05 2019.

Priedai

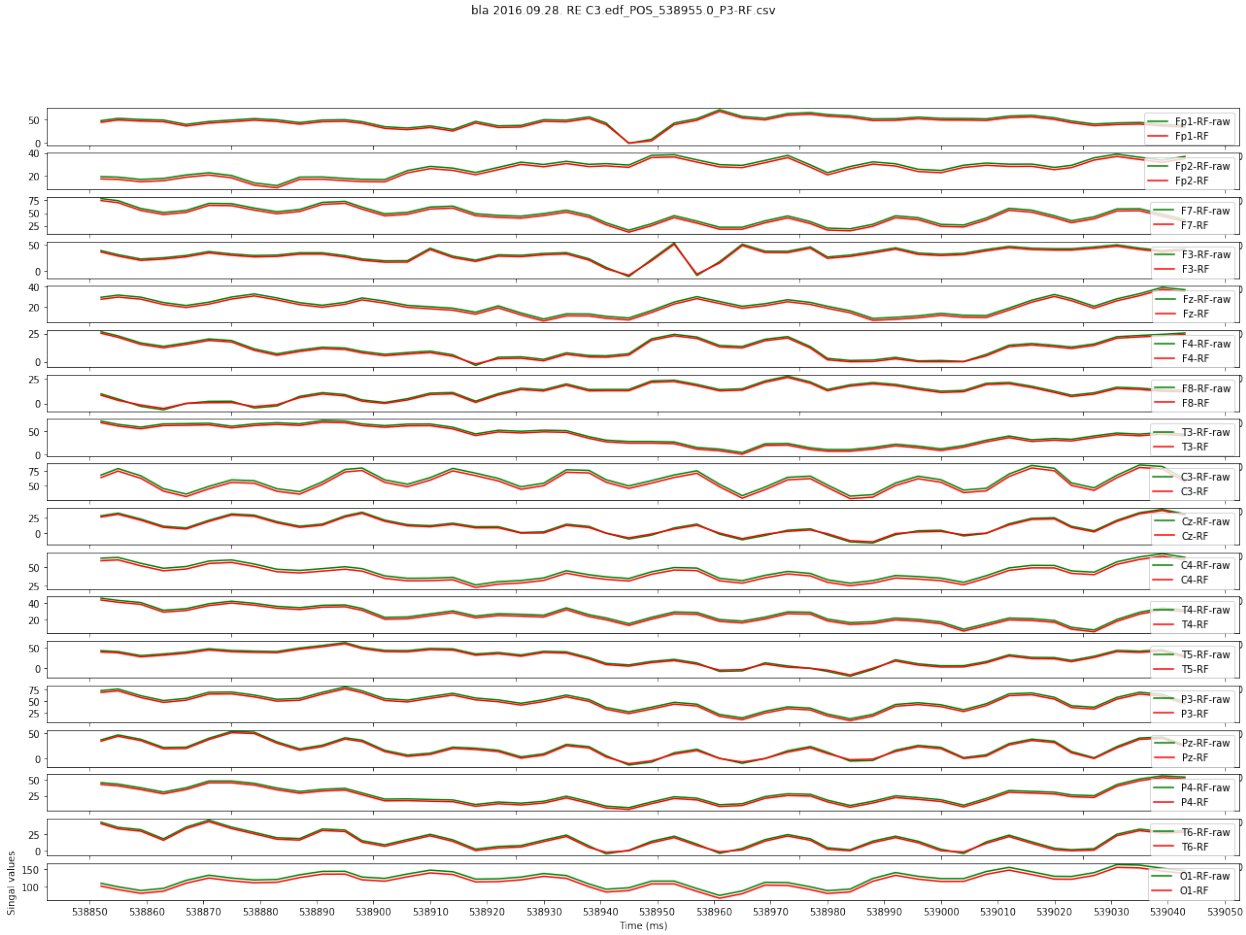
Dokumentą sudaro du priedai: A priede sutalpintos visos signalo vizualizacijos gautos atliekant vilnelių transformacijos tyrimo dalį.

B priede sutalpinti tarpinių CNN modelių architektūros.

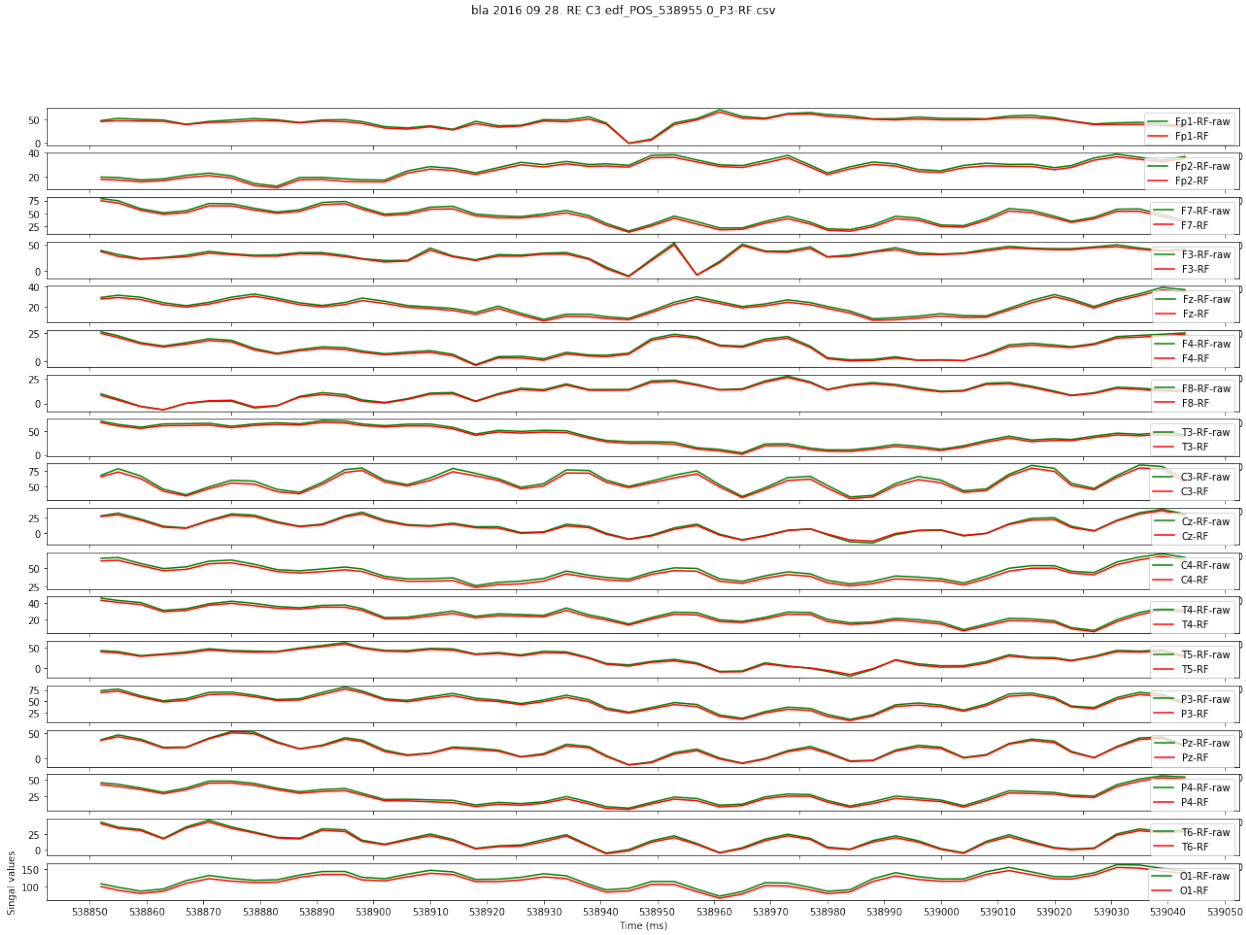
A. Vilnelių transformacija apdoroti signalo segmentai



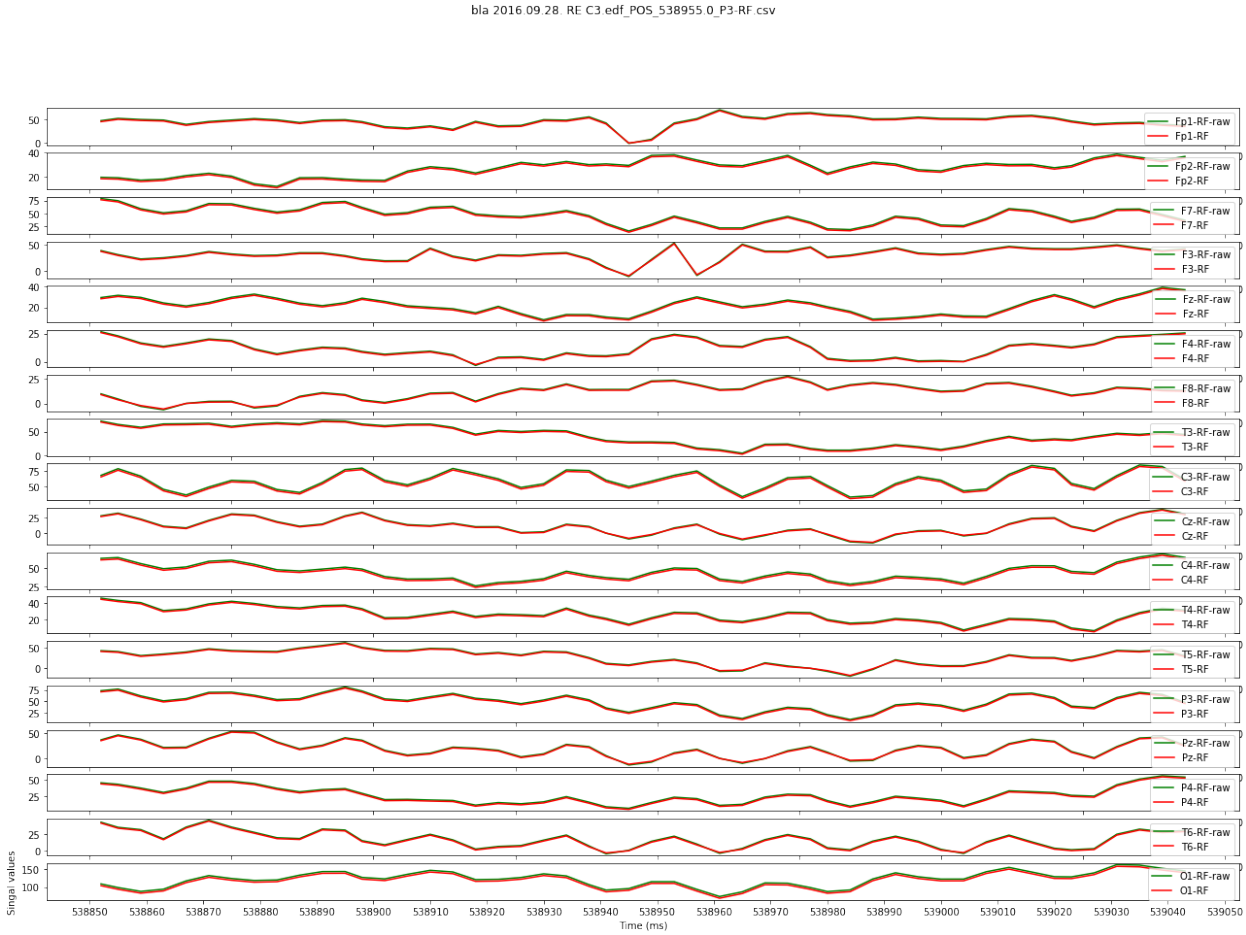
16 pav. 200ms. trukmės EEG iškarpa apdorota DB20 su 0.1 slenksčiu.



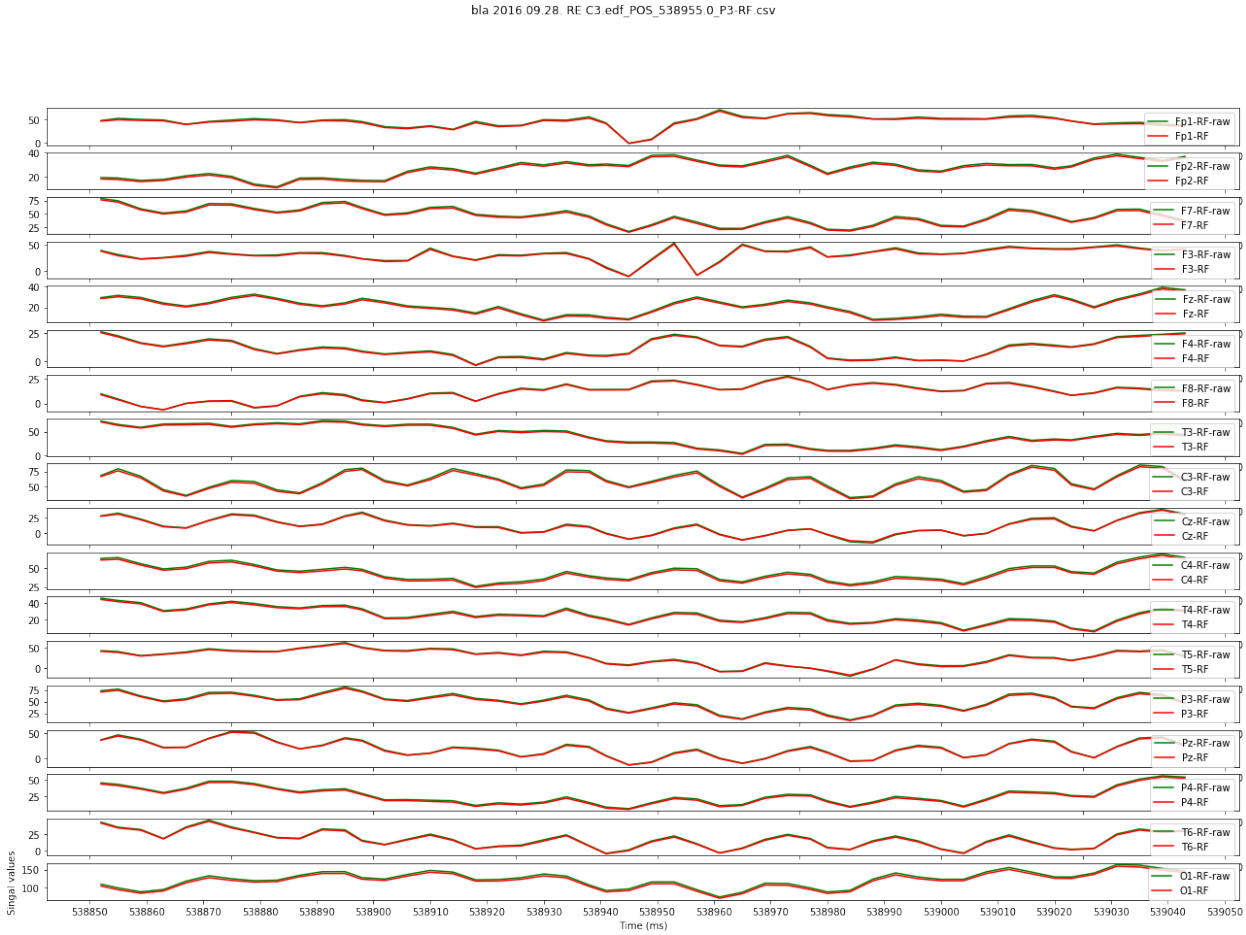
17 pav. 200ms. trukmės EEG iškarpa apdorota DB20 su 0.05 slenksčiu.



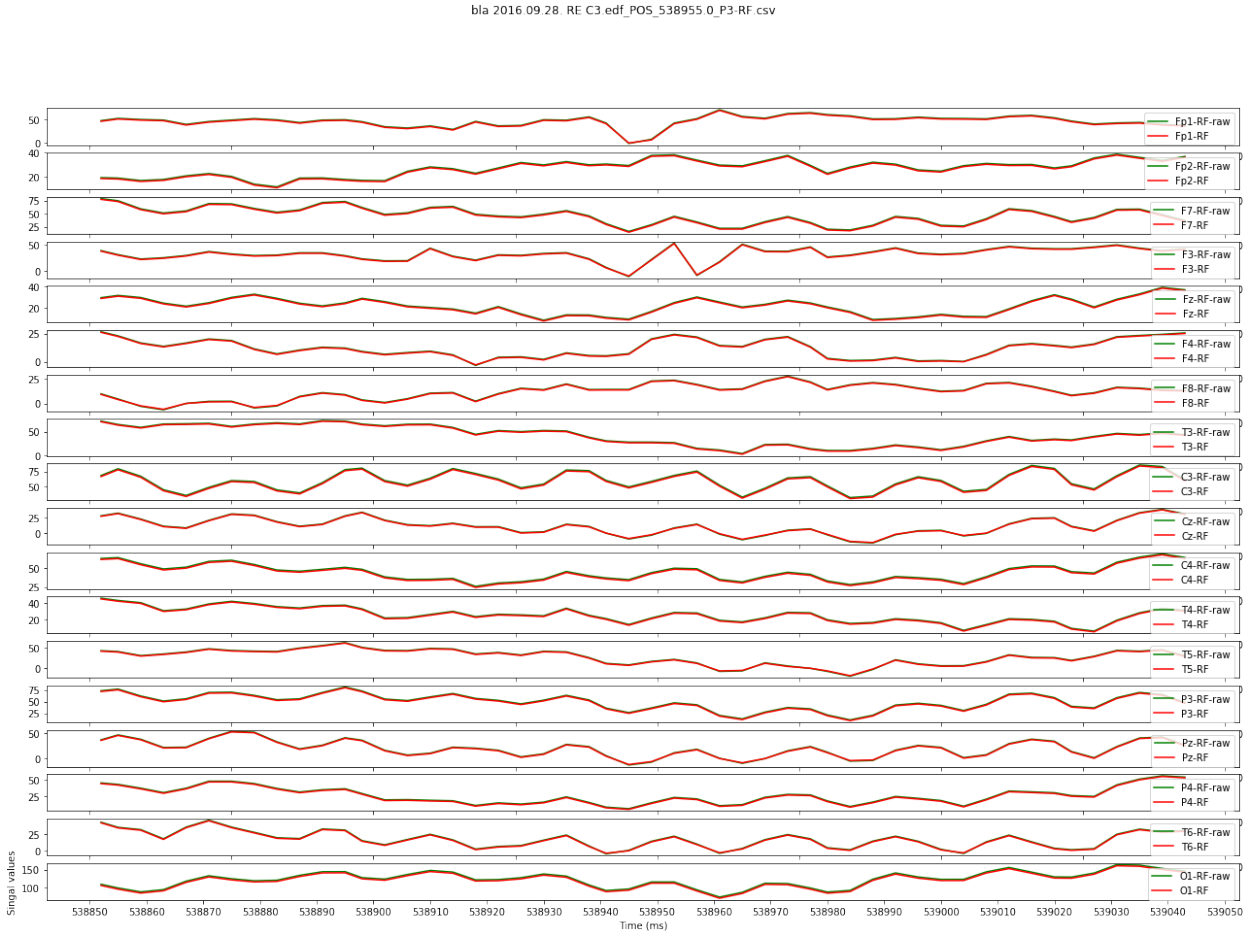
18 pav. 200ms. trukmės EEG iškarpa apdorota DB4 su 0.05 slenksčiu.



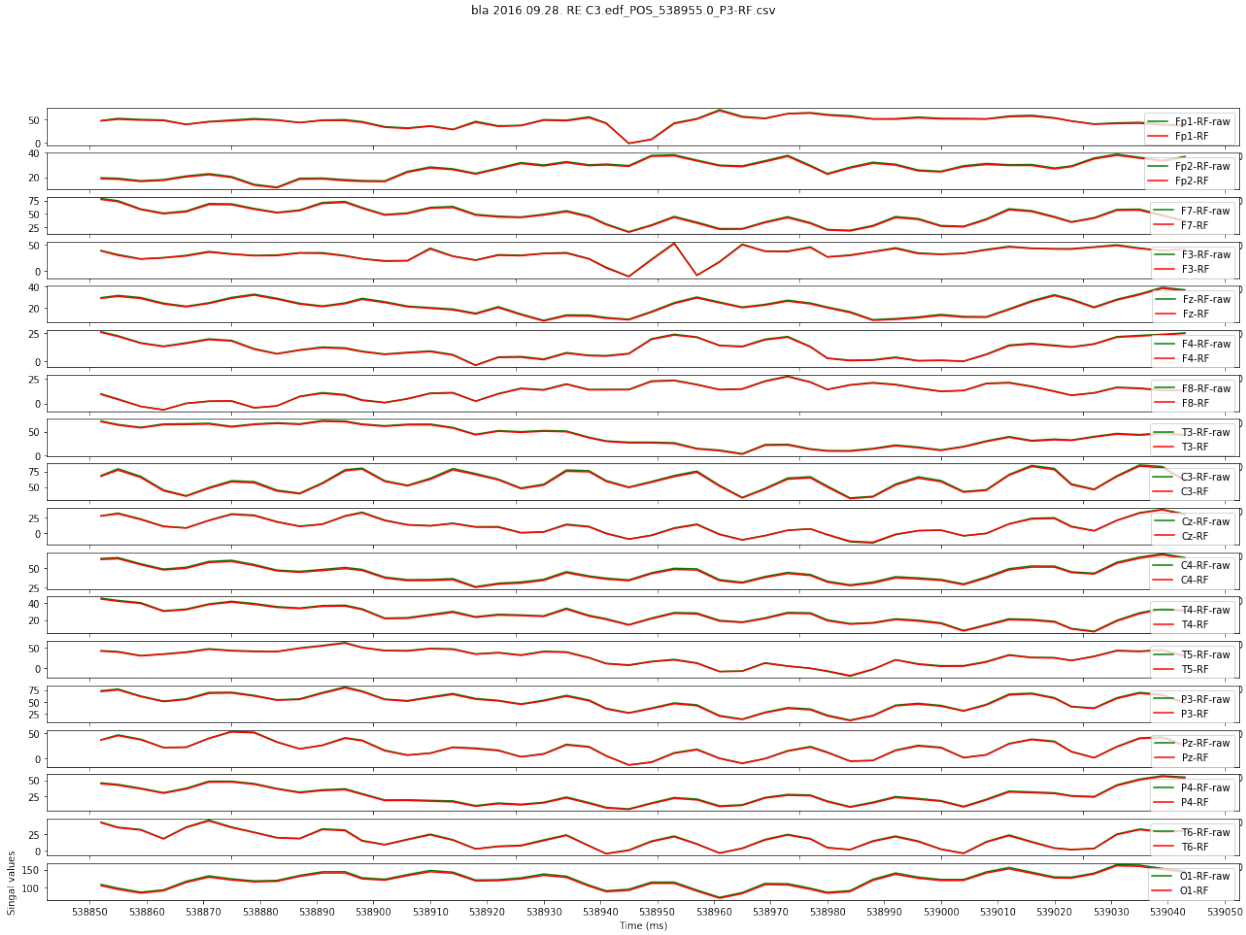
19 pav. 200ms. trukmės EEG iškarpa apdorota DB20 su 0.03 slenksčiu.



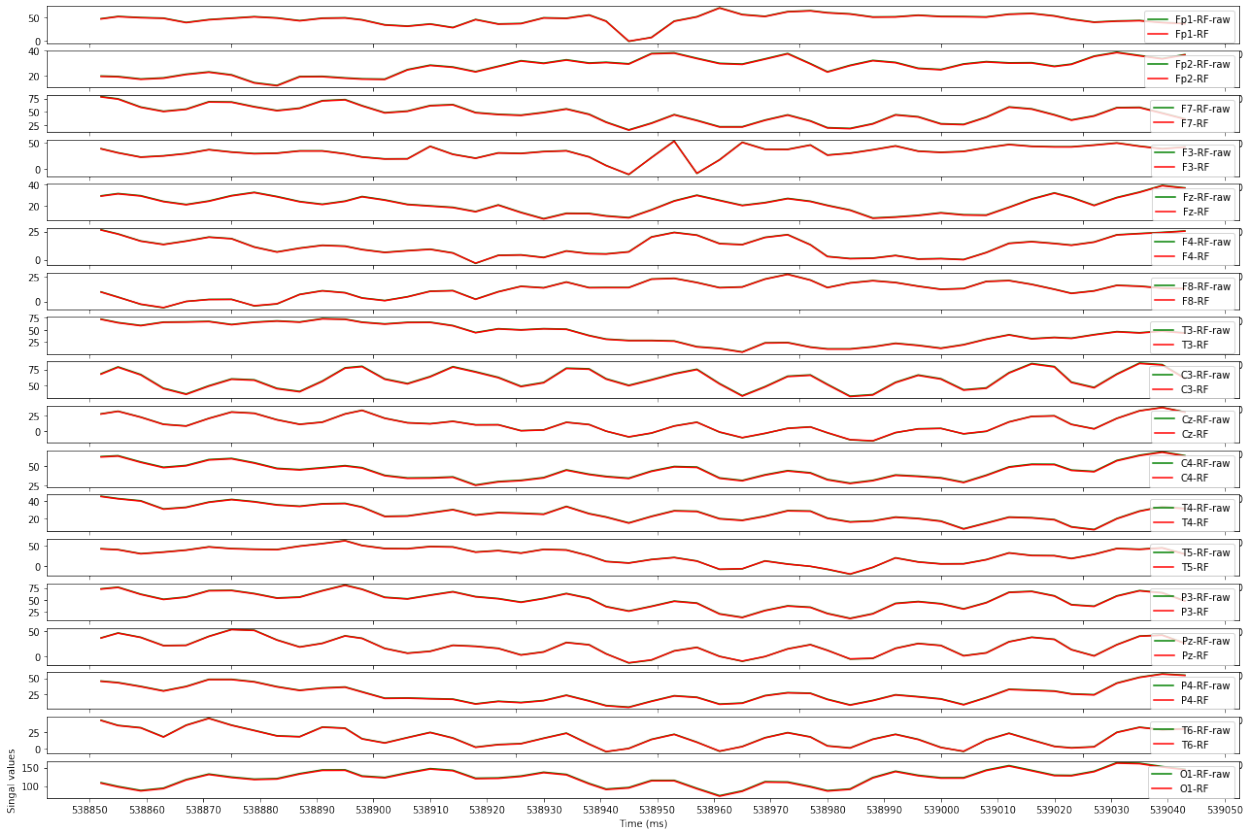
20 pav. 200ms. trukmės EEG iškarpa apdorota DB4 su 0.03 slenksčiu.



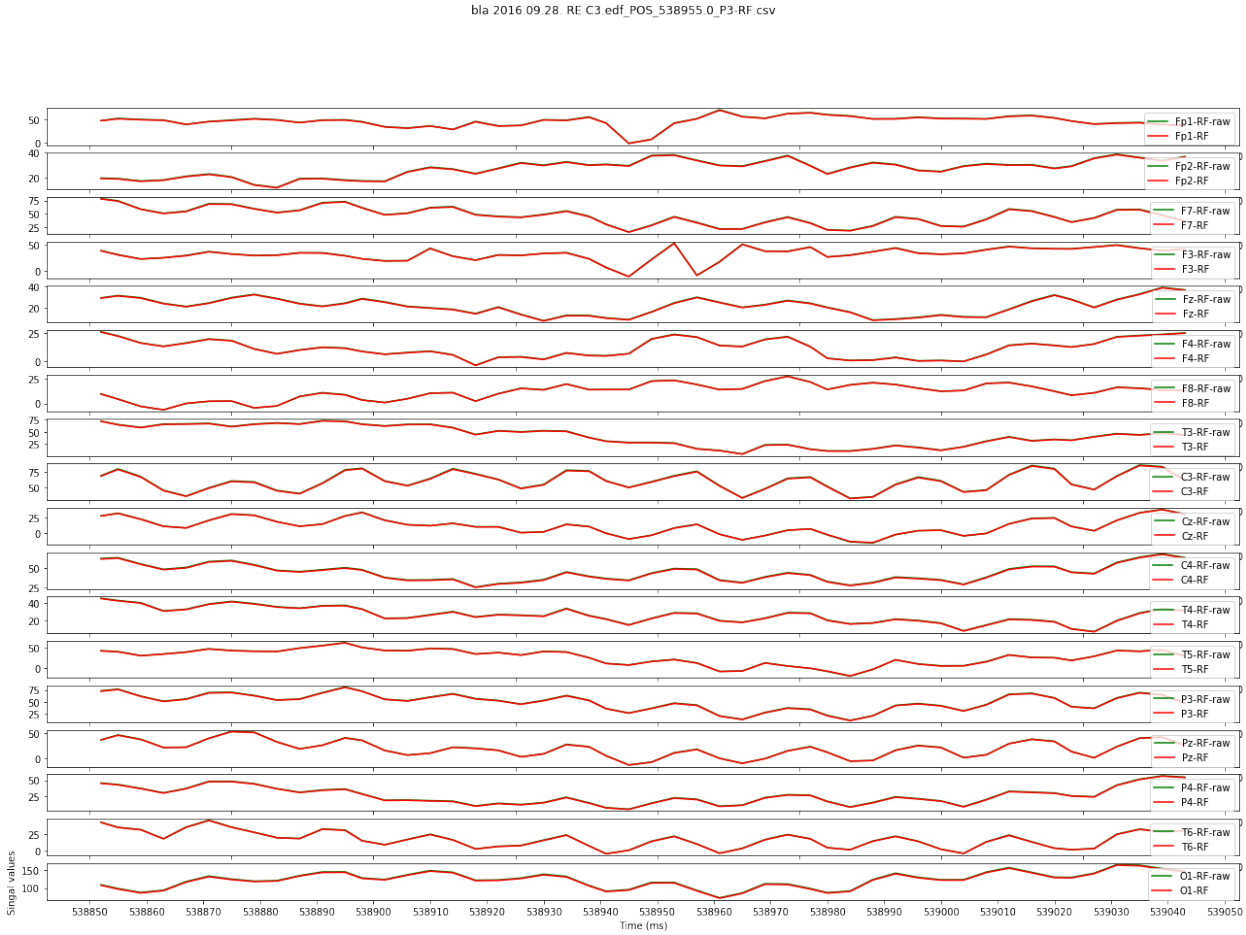
21 pav. 200ms. trukmės EEG iškarpa apdorota DB20 su 0.02 slenksčiu.



22 pav. 200ms. trukmės EEG iškarpa apdorota DB4 su 0.02 slenksčiu.



23 pav. 200ms. trukmės EEG iškarpa apdorota DB20 su 0.01 slenksčiu.



24 pav. 200ms. trukmės EEG iškarpa apdorota DB4 su 0.01 slenksčiu.

B. Tarpinės pirmojo eksperimento konvoliucinių neuroninio tinklo modelių architektūros

18 lentelė. Pradinio modelio architektūra

Sluoksnio tipas	Išeities forma	Parametrų skaičius
conv2d_1 (Conv2D)	(None, 49, 17, 16)	80
max_pooling2d_1 (MaxPooling2D)	(None, 24, 8, 16)	0
conv2d_2 (Conv2D)	(None, 23, 7, 16)	1040
max_pooling2d_2 (MaxPooling2D)	(None, 11, 3, 16)	0
flatten_1 (Flatten)	(None, 528)	0
dense_1 (Dense)	(None, 100)	52900
dense_2 (Dense)	(None, 2)	202

19 lentelė. CNN modelio su 32 branduoliais architektūra

Sluoksnio tipas	Išeities forma	Parametrų skaičius
conv2d_1 (Conv2D)	(None, 49, 17, 32)	160
max_pooling2d_1 (MaxPooling2D)	(None, 24, 8, 32)	0
conv2d_2 (Conv2D)	(None, 23, 7, 32)	4128
max_pooling2d_2 (MaxPooling2D)	(None, 11, 3, 32)	0
flatten_1 (Flatten)	(None, 528)	0
dense_1 (Dense)	((None, 1056)	105700
dense_2 (Dense)	(None, 2)	202

20 lentelė. CNN modelio su 64 branduoliais architektūra

Sluoksnio tipas	Išeities forma	Parametrų skaičius
conv2d_1 (Conv2D)	(None, 49, 17, 64)	320
max_pooling2d_1 (MaxPooling2D)	(None, 24, 8, 64)	0
conv2d_2 (Conv2D)	(None, 23, 7, 64)	16448
max_pooling2d_2 (MaxPooling2D)	(None, 11, 3, 64)	0
flatten_1 (Flatten)	(None, 2112)	0
dense_1 (Dense)	(None, 100)	211300
dense_2 (Dense)	(None, 2)	202

21 lentelė. CNN modelio su 256 branduoliais architektūra

Sluoksnio tipas	**Išeities forma**	**Parametrų skaičius**
conv2d_1 (Conv2D)	(None, 49, 17, 256)	1280
max_pooling2d_1 (MaxPooling2D)	(None, 24, 8, 256)	0
conv2d_2 (Conv2D)	(None, 23, 7, 256)	262400
max_pooling2d_2 (MaxPooling2D)	(None, 11, 3, 256)	0
flatten_1 (Flatten)	(None, 8448)	0
dense_1 (Dense)	(None, 100)	844900
dense_2 (Dense)	(None, 2)	202