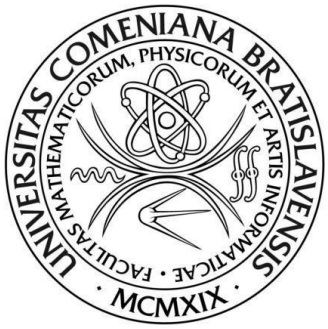
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



AUDIO PREHRÁVAČ OVLÁDANÝ MYSĽOU

Bakalárska práca

2019 **Daniel Kisel**

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

AUDIO PREHRÁVAČ OVLÁDANÝ MYSĽOU

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Mgr. Peter Náther, PhD.

Bratislava, 2019
Daniel Kisel

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Daniel Kisel
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Audio prehraváč ovládaný mysľou
Audio player controlled by mind

Anotácia: .

Cieľ: Cieľom práce je vytvoriť audio prehrávač, ktorý bude možné ovládať pomocou mysle s využitím zariadenia Mindwave Mobile

Literatúra: http://developer.neurosky.com

Vedúci: Mgr. Peter Náther, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 05.10.2017

Dátum schválenia: 10.10.2018 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

student vedúci práce

ČESTNÉ VYHLÁSENIE

Vyhlasujem, že som záverečnú prácu vypracoval samostatne a uviedol som všetku použitú literatúru.

……

POĎAKOVANIE

Ďakujem môjmu vedúcemu práce Petrovi Nátherovi za ochotu, cenné rady a usmernenie pri písaní záverečnej práce.

ABSTRAKT

Práca sa zaoberá tvorbou mobilnej aplikácie zvukového prehrávača pre platformu Android, ktorú je z hlavnej časti možné ovládať mysľou. Na ovládanie aplikácie pomocou mysle, používame zariadenie NeuroSky MindWave Mobile. Okrem použitia algoritmov, ktoré sú k tomuto zariadeniu voľne dostupné, sme sa zamerali na zozbieranie surových dát mozgových vĺn, na ktorých sme natrénovali modely strojového učenia, ktoré následne používame na rozpoznávanie určitých stavov mozgovej aktivity. Navodením týchto stavov mozgovej aktivity, potom ovládame niektoré procesy v aplikácii.

Kľúčové slová: zvukový prehrávač, NeuroSky MindWave Mobile, Android, mozgové vlny, Elektroencefalografia, strojové učenie.

ABSTRACT

The thesis is focusing on creation of mobile application of an audio player for the Android platform, where the main part of this application can be controlled by mind. To control the application we use NeuroSky MindWave Mobile device. In addition to using the algorithms that are freely available for this device, we focused on collecting raw brain wave data that we used for training the machine learning models to recognize certain states of brain activity. By inducing these states of brain activity we then control some processes inside the application.

Key words: audio player, NeuroSky MindWave Mobile, Android, brain waves, Electroencephalography, machine learning.

Obsah

Úvod

Ovládanie vecí vôkol nás pomocou mysle, je myšlienka, ktorá ľudí zaujíma už od nepamäti. Dnešná doba prinášajúca veľké pokroky v oblasti neurovedy a informačných technológií, nám pomáha túto myšlienku pretaviť v realitu. Ľudský mozog je fascinujúci orgán tvorený miliardami vzájomne prepojených neurónov. Technológia, dostupná širokej verejnosti, ktorá by dokázala rozlúštiť jednotlivé vzorce správania mozgu, pomocou ktorej by bol človek schopný ovládať určité procesy svojho každodenného života, bez použitia fyzickej sily, využívajúc iba svoj mozog, by mohla mať veľký dopad na život ľudí so zdravotným postihnutím, či na efektivitu práce ľudstva ako takého.
Z týchto a ďalších dôvodov vznikla téma práce, ktorej cieľom je vytvoriť audio prehrávač, ktorý bude možné ovládať pomocou mysle, s využitím zariadenia MindWave Mobile. Na rozdiel od podobných prác využívame zariadenie, umožňujúce získavanie dát mozgových vĺn niekoľkých frekvenčných pásiem, ktoré je kompaktné a cenovo dostupné. Audio prehrávač predstavuje mobilnú aplikáciu pre platformu Android, čo flexibilitu ovládania aplikácie, pomocou tohto zariadenia, zvyšuje. Človek tak nemusí sedieť pred počítačom, s množstvom káblov pripojených k senzorom na povrchu lebky, ale môže byť kdekoľvek.

V práci sa najskôr budeme zaoberať skúmaním zariadenia MindWave Mobile. Po preskúmaní zariadenia, zozbierame dáta rôznych stavov mozgovej aktivity, na ktorých natrénujeme modely strojového učenia. Natrénované modely, spolu s algoritmami dostupnými k tomuto zariadeniu, použijeme na ovládanie hlavnej časti zvukového prehrávača.

Prvá kapitola obsahuje teoretické poznatky, týkajúce sa problematiky skúmania mozgovej aktivity. V prvej kapitole sa taktiež zaoberáme popisom zariadenia MindWave Mobile, skúmaním jeho vlastností a možností, plus algoritmami, ktoré nám poskytuje. Záver kapitoly obsahuje popis metód strojového učenia, ktoré v práci používame na vytváranie modelov.

V druhej kapitole sa zaoberáme návrhom mobilnej aplikácie. Návrh práce sa zaoberá popisom grafické používateľského prostredia, funkcionality, aj algoritmov aplikácie.

V tretej kapitole riešime implementáciu aplikácie. Popisujeme vývoj aplikácie, zber troch rôznych typov dát mozgovej aktivity a ich následnú analýzu.

Na týchto dátach ďalej trénujeme modely strojového učenia, ktoré sa snažíme naučiť rozpoznať typy dát. Na záver kapitoly, pre zakomponovanie modelov do mobilnej aplikácie, riešime konverziu jazyka kódu modelov.

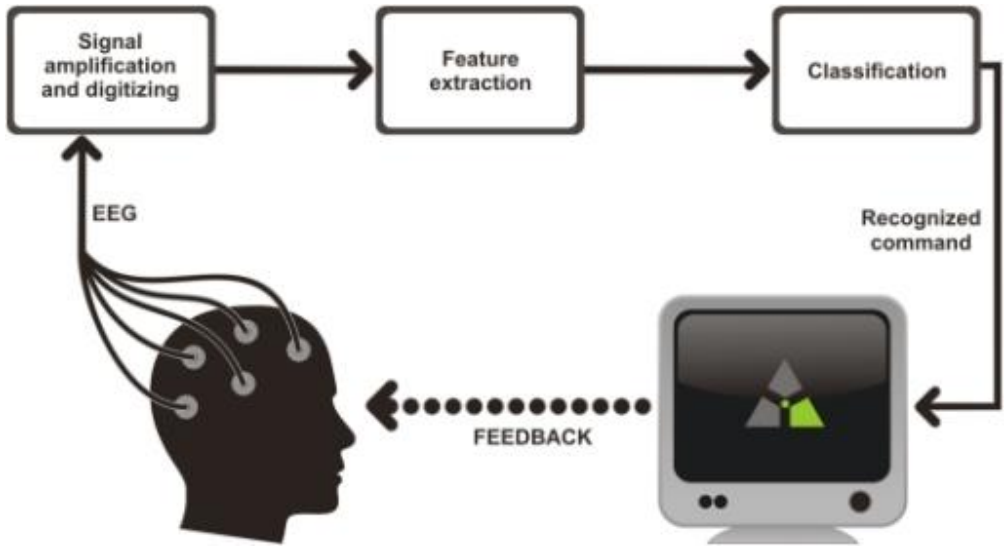
V poslednej kapitole zobrazujeme výsledky práce. Kapitola obsahuje tabuľky presností rozpoznávania typov dát, pomocou natrénovaných modelov, testovanie aplikácie a zhrnutie práce.

1 Východiská

V tejto kapitole sa oboznámime s niektorými základnými metódami používanými na pozorovanie mozgovej aktivity, popíšeme si vlastnosti a možnosti zariadenia, ktoré na tento účel použijeme, zistíme niektoré jeho plusy a mínusy, s ktorými sme sa stretli behom používania. Vysvetlíme si niekoľko pojmov, ktoré priamo, alebo nepriamo súvisia so získavaním a spracúvaním dát z mozgu, ako sú napríklad: BCI, EEG. Taktiež si priblížime niekoľko technológií a problémov s nimi spojenými, technológií, ktoré budeme potrebovať a použijeme na vytvorenie aplikácie pre mobilné zariadenie, ktorým je v našom prípade mobilný telefón s operačným systémom Android, komunikujúci s externým zariadením Neurosky MindWave Mobile, z ktorého získame potrebné dáta nameranej mozgovej aktivity. Popíšeme si niektoré metódy strojového učenia, ktoré použijeme na trénovanie modelov za účelom klasifikácie dát.
Medzi technológie ktoré budeme pri práci používať patria: Bluetooth, Android Studio, Jupyter Notebook, NeuroSky MindWave Mobile, Android Developer Tools 4.2.

1.1 Brain–computer interface (BCI)

Brain-computer interface, skrátene **BCI**, je priamym rozhraním medzi mozgom a externým zariadením, ktoré nám umožňuje získavať informácie o aktivite mozgu, ktoré následne môžeme použiť na komunikáciu s externým zariadením.
Typický BCI systém pozostáva zo senzora, zachytávajúceho mozgovú aktivitu, väčšinou taktiež zosilňovača signálu, počítača, respektíve zariadenia, ktoré signál spracuje a výstupu, čo môže byť ďalšie zariadenie, napr.: robotická ruka, alebo obrazovka.



Obr. 1. Brain-computer interface

Na to aby sme niečo označovali ako BCI, musí spĺňať nasledujúce podmienky:

- Mozgový signál je spracovávaný v reálnom čase
- Príkazy vykonávané externým zariadením musia byť vykonané na základe vôle používateľa
- Vstupné údaje získané pomocou zariadenia na skúmanie aktivity mozgu musia pochádzať priamo z mozgu
- Subjekt dokáže pozorovať reakciu, na základe ním vykonanej akcie

BCI nie je zariadenie na čítanie mysle, nedokáže čítať myšlienky, čo BCI robí je sledovanie špecifických vlastností signálu (mozgovej aktivity) a vykonávanie príslušných operácií.

Cieľom BCI by malo byť získanie informácie z mozgu na vykonanie príslušnej operácie a aj jej samotné vykonanie bez toho, aby musel človek nevyhnutne použiť fyzickú silu, respektíve nemusel použiť iný nástroj než svoj vlastný mozog. Dôvodom toho je nemožnosť vykonávania

určitého druhu pohybu u ľudí so zdravotným postihnutím. Kapitola 1.1 je spracovaná podľa zdroja [1]

1.2 **Elektroencefalografia (EEG)**

Elektroencefalografia, je jedna z najbežnejších neinvazívnych metód na zaznamenávanie elektrických signálov produkovaných neurónmi, že je daná metóda neinvazívna znamená, že mozgové signály sú merané zvonku, pasívne a do mozgu vôbec nezasahujeme.

Obr. 2. EEG čapica

Opakom neinvazívnych metód sú metódy invazívne, ktoré priamo zasahujú do mozgu subjektu a medzi najznámejšie invazívne metódy patrí ECoG (Elektrokortikografia). [2]
ECoG je schopné snímania širšieho frekvenčného pásma ako EEG. Strata informácií pri použití EEG je zapríčinená fyziologickými bariérami. Pravdepodobne neexistuje spôsob, ako snímať frekvencie vyššie ako 40Hz (Hertzov) s EEG elektródami. [1]
My sa však budeme zaoberať iba Elektroencefalografiou, pretože to je metóda, ktorú používa zariadenie, ktoré máme k dispozícii, o ktorom si povieme neskôr.

Pri používaní EEG senzora by mal subjekt sedieť vzpriamene, pokojne a nevykonávať žiadne pohyby tváre, pretože každý pohyb tváre je zaznamenaný pomocou EEG elektród, dokonca aj žmurknutie je zaznamenané na EEG grafe ako vrchol zaznamenávaných hodnôt a môže byť považovaný za narušenie elektromagnetického signálu, či šumom v zaznamenanom signáli [1], čo pri návrhu aplikácie v ďalšej kapitole práce použijeme ako jeden zo spôsobov spúšťania vybraného procesu v aplikácii.

1.3 Popis zariadenia

Zariadenie, ktoré máme k dispozícií je **Neurosky MindWave Mobile**.

Obr. 3. NeuroSky MindWave Mobile

Jedná sa o zariadenie (headset) poháňané jednou AAA batériou, ktoré nie je pripojené na pevno pomocou káblov ku zariadeniu (telefón, počítač, tablet), ktoré získané dáta spracúva ako je to u klasických EEG senzorov, ale pripája sa cez Bluetooth, čo znamená, že na použitie MindWave Mobile je podmienkou aby zariadenie, ktoré chceme spárovať s týmto EEG senzorom obsahovalo Bluetooth.
Headset pozostáva z nastaviteľnej čelenky, jednej elektródy a sponky na ucho.
Pomocou elektródy, MindWave Mobile v reálnom čase zachytáva mozgovú aktivitu, ktorú potom cez Bluetooth prenáša na spracovanie do spárovaného zariadenia.

1.3.1 Uvedenie zariadenia do prevádzky

Headset sa zapne tlačidlom na boku, ktoré krátko na to začne blikať na modro, čím signalizuje, že je pripravené na spárovanie s ďalším zariadením, ktoré bude spracúvať jeho výstupy. Po spárovaní sa MindWave nasadí na hlavu subjektu a dióda sa nastaví tak, aby sa dotýkala čela. Na ucho sa pripne sponka na umiestnená na spodku headsetu. Medzi sponkou a uchom, ani diódou a čelom by nemali byť vlasy ani iné objekty, ktoré by narušovali kvalitu signálu, po chvíli by všetko malo byť pripravené na používanie.

1.3.2 Prečo MindWave Mobile ?

Neurosky MindWave Mobile, na základe ktorého neskôr vznikla téma práce, som si vybral z dôvodu, že myšlienka ovládania aplikácie pomocou mysle znela zaujímavo a chcel som preskúmať možnosti a vlastnosti tohto zariadenia. Celkovo ma zaujali výhody MindWave Mobile, ktoré sú uvedené nižšie.

Výhody NeuroSky MindWave Mobile:

- Cenovo dostupné riešenie pre bežného spotrebiteľa, ktorý nevyžaduje kvality profesionálneho senzora, cena za NeuroSky MindWave Mobile 2, čo je aktuálna verzia headsetu sa dňa 6.2.2019 na internete pohybovala okolo hodnoty sto eur [3], pričom cena za profesionálne EEG zariadenia sa pohybovala v rovnakom čase v rozmedzí od niekoľko tisíc eur až po niekoľko desiatok tisíc eur, čo je výrazný rozdiel.
- Mobilné riešenie, nie je potrebné sedieť za počítačom, MindWave Mobile je malé ľahké zariadenie a vďaka Bluetooth pripojeniu je naozaj flexibilné.
- Množstvo existujúcich riešení, aplikácií vyrobených špeciálne pre MindWave Mobile, o niektorých z nich si povieme neskôr.
- **Developer Tools** – API (Application programming interface alebo rozhranie pre programovanie aplikácií), ktoré za nás rieši mnohé pomerne komplexné problémy ako napríklad: získavanie a prenos dát z MindWave do počítača, alebo mobilu, či pripojenie a komunikácia cez Bluetooth.

  API je možné integrovať do vlastného projektu, pričom ho môžeme, z internetovej stránky výrobcu, bezplatne stiahnuť a použiť, a to pre rôzne platformy. Existuje Developer Tools pre: Windows, Android, IOS aj Macintosh.

1.3.3 Nedostatky MindWave Mobile

Okrem výhod spojených s používaním MindWave Mobile však narážame aj na jeho nedostatky a možné problémy pri práci s ním.

Medzi **nedostatky** Neurosky MindWave Mobile patria:

- Nižšia presnosť merania aktivity mozgu ako u drahších modelov, MindWave Mobile má len jednu elektródu, ktorá je umiestnená na čele subjektu [4], avšak pri drahších modeloch sa bežne stretávame s oveľa väčším počtom elektród, ktoré sú rozmiestnené po celom povrchu lebky.

- Zariadenie nie je nabíjateľné a v balení sa nenachádza priamo pribalená AAA batéria. Lepšie riešenie by zrejme bolo použiť namiesto AAA batérie malú Li-Ion (Lítium-Iontovú) batériu, ktorá by sa jednoducho po vybití pomocou kábla do elektrickej siete opäť nabila, aj keď toto je skôr osobný názor než skutočná nevýhoda zariadenia. Výdrž zariadenia v prevádzke, pri použití jednej AAA batérie je výrobcom udávaná na 8 hodín, čo je pomerne prijateľné.
- Nepohodlie pri dlhšom používaní, zhruba od pätnástich minút nosenia na hlave, začne headset citeľne tlačiť na ucho a čelo.
- Slabšia podpora zo strany Neurosky, na stránke výrobcu sa nachádza, ako sme spomenuli pri výhodách zariadenia API ku MindWave Mobile - Developer Tools, v našom prípade pre platformu Android je to: Android Developer Tools 4.2, čo je posledná vydaná verzia, ktorá bola vydaná dňa: 19.9.2016. [5]

To, že posledná verzia je viac ako dva roky stará, pravdepodobne nie je skutočný problém, no problém nastal, hneď po úspešnom skompilovaní vzorového projektu pribaleného v Android Developer Tools 4.2 a nainštalovaní aplikácie, vychádzajúcej z daného projektu, na telefón s operačným systémom Android 8.0 (Oreo). Pri pokuse spustiť aplikáciu vzorového projektu na telefóne, aplikácia vyhodila chybovú hlášku miesto očakávaného výstupu a automaticky sa ukončila. Ukážkový projekt by pritom mal slúžiť na to, aby sa používateľ lepšie zorientoval v danej problematike a oboznámil s používaním Android Developer Tools, keďže projekt predstavuje ukážku toho, čo je možné s daným zariadením urobiť a obsahuje konkrétne kusy kódu.

Našťastie sa problém podarilo vyriešiť odstránením, nefunkčných a pre našu aplikáciu nepotrebných častí kódu. Po zdĺhavom testovaní a miernom pre usporiadaní kódu sa aplikácia na telefóne konečne spustila, s očakávaným výstupom a funkcionalitou.

Treba však mať na zreteli, že zariadenie, ktoré používame, nie je najnovší model od spoločnosti Neurosky. Aktuálny model headsetu je: MindWave Mobile 2, ktorý snáď zlepšuje vyššie popísané nedostatky, no v prípade Android Developer tools 4.2 zrejme problém pretrváva aj pri aktuálnom modeli MindWave Mobile, pretože sa jedná o chybný kód a nie o chybu zariadenia.

1.4 Existujúce riešenia

Na stránke Neurosky sa nachádza množstvo existujúcich riešení, ktoré sú vyrobené priamo pre zariadenie MindWave Mobile. Používateľ si tak môže vybrať z ponuky od aplikácií vizualizujúcich dáta, ktoré headset sníma, cez aplikácie, ktoré sľubujú zlepšenie kognitívnych schopností až po hry, ktoré sú ovládané mysľou.
Niektoré aplikácie sú zdarma, iné sú platené. Medzi existujúcimi riešeniami sa nachádza aj spomínaný Android Developer Tools 4.2.
Existujúce riešenia aplikácií pre platformu Android, ktoré nám pomôžu ako inšpirácia pri návrhu našej aplikácie, sú: MyndPlayer, MindWriter, Brainwave Visualizer a Blink Camera.[6]

MyndPlayer je aplikácia video prehrávača ovládaného mysľou pomocou MindWave Mobile. Aplikácia umožňuje používateľovi pomocou mysle pôsobiť na smerovanie videa, alebo na výsledný výstup.

MindWriter je aplikácia, ktorá umožňuje písať na mobilnom zariadení pomocou mysle. Aplikácia používa pred trénovanú neurónovú sieť, ktorá umožňuje rozoznať vzorce vo vlnách spojených s písmenami abecedy.

Brainwave Visualizer je aplikácia, ktorá nám umožňuje sledovať vizualizácie dát nasnímaných pomocou headsetu, počas počúvania používateľom zvolenej hudby. Taktiež umožňuje nahrávanie údajov reakcie mysle na konkrétne hudobné skladby.

Blink Camera je aplikácia, ktorá slúži na ovládanie kamery na našom zariadení pomocou žmurknutia, čo umožňuje fotiť na väčšie vzdialenosti bez nutnosti nastavenia automatickej uzávierky kamery, či bez použitia rúk.

1.5 Android Developer Tools 4.2

Ako sme už vyššie spomenuli Android Developer Tools 4.2 je API, ktoré sa stará o prepojenie aplikácie s MindWave Mobile.

Medzi hlavné zložky Developer Tools pre Android patria:

- Stream SDK for Android – SDK, ktoré sa používa na pripojenie Androidovej aplikácie ku NeuroSky headsetu cez Bluetooth, aj na získanie dát z headsetu.[7]
- EEG Algorithm SDK for Android – SDK, ktoré sa používa na analýzu a interpretáciu EEG dát z NeuroSky headsetu.[7]

1.5.1 Algoritmy z Android Developer Tools 4.2

Nasledujúca podkapitola je spracovaná podľa zdroja [8].
Medzi dáta ktoré môžeme získať zo zariadenia pomocou existujúcich algoritmov z Android Developer Tools 4.2 patria:

- **Attention (Att)** – algoritmus Attention meria intenzitu „sústredenia“, alebo „pozornosti“ s hodnotami od 0 po 100, v reálnom čase, pričom nové dáta sa zobrazujú na výstupe každú sekundu. Čím väčšia intenzita sústredenia je u subjektu zaznamenaná, tým vyššia hodnota je algoritmom Attention zobrazená na výstupe.
- **Meditation (Med)** – algoritmus Meditation meria intenzitu „pokoja“, alebo mentálnej „relaxácie“ s hodnotami od 0 po 100, taktiež s aktualizáciou údajov na výstupe prebiehajúcou každú sekundu a opäť vyššia hodnota na výstupe znamená vyšší stupeň mentálnej relaxácie.
- **BandPower (BP)** algoritmus BandPower meria hustotu spektra výkonu piatich frekvenčných pásiem:
    - Pásmo Delta: < 4 Hz (Hertzov)
    - Pásmo Theta: 4 ~ 8 Hz
    - Pásmo Alpha: 8 ~ 13 Hz
    - Pásmo Beta: 13 ~ 30 Hz
    - Pásmo Gamma: > 30 Hz

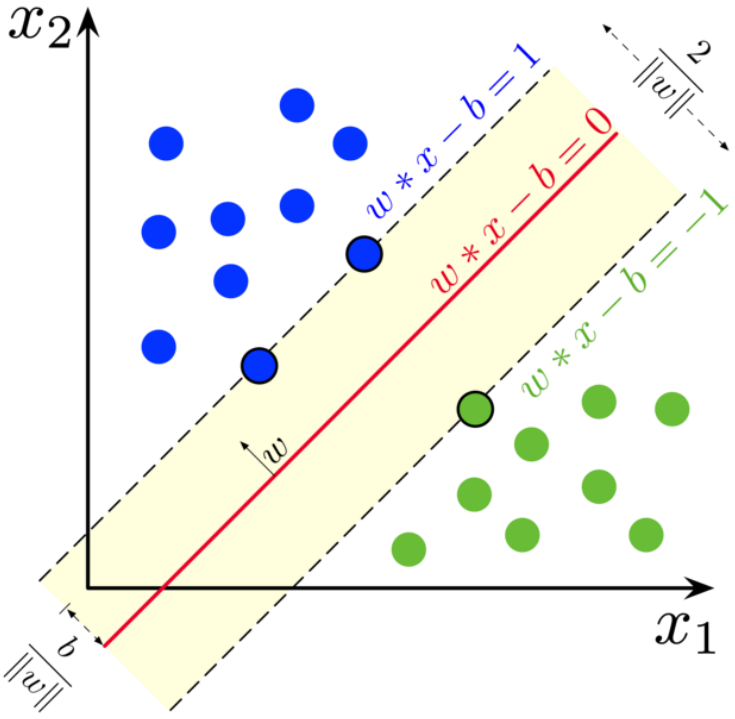
  Výstupnými hodnotami sú hodnoty z vyššie uvedených piatich frekvenčných pásiem v **dB** (decibeloch), pričom nové dáta z MindWave Mobile sú na výstupe zobrazované každú sekundu.
- **Eye Blink Detection (Blink) –** algoritmus Eye Blink Detection zaznamenáva „žmurknutie“ subjektu, pričom na výstupe zobrazuje intenzitu žmurknutia, v podobe čísla s kladnou hodnotou. Čím je žmurknutie intenzívnejšie, tým je táto hodnota vyššia.

1.6 Metódy strojového učenia

Po preskúmaní algoritmov, ktoré nám priložené SDK k MindWave Mobile ponúka, zisťujeme, že pre ovládanie zvukového prehrávača nie sú dostatočné a preto sme sa rozhodli pustiť do zbierania surových dát, ktoré nám toto zariadenie vďaka algoritmu BandPower poskytuje. Na týchto dátach následne natrénujeme modely pomocou strojového učenia (angl. Machine learning), ktorými sa pokúsime rozšíriť možnosti ovládania aplikácie, pomocou mysle. Pre získanie čo najlepších výsledkov rozpoznávania mozgovej aktivity, vyskúšame päť metód strojového učenia, ktoré použijeme na natrénovanie modelov pre klasifikáciu dát. V nasledujúcich podkapitolách si ukážeme, aké metódy sme sa rozhodli vyskúšať a priblížime si v stručnosti, ako fungujú.

1.6.1 Support Vector Classification (SVC)

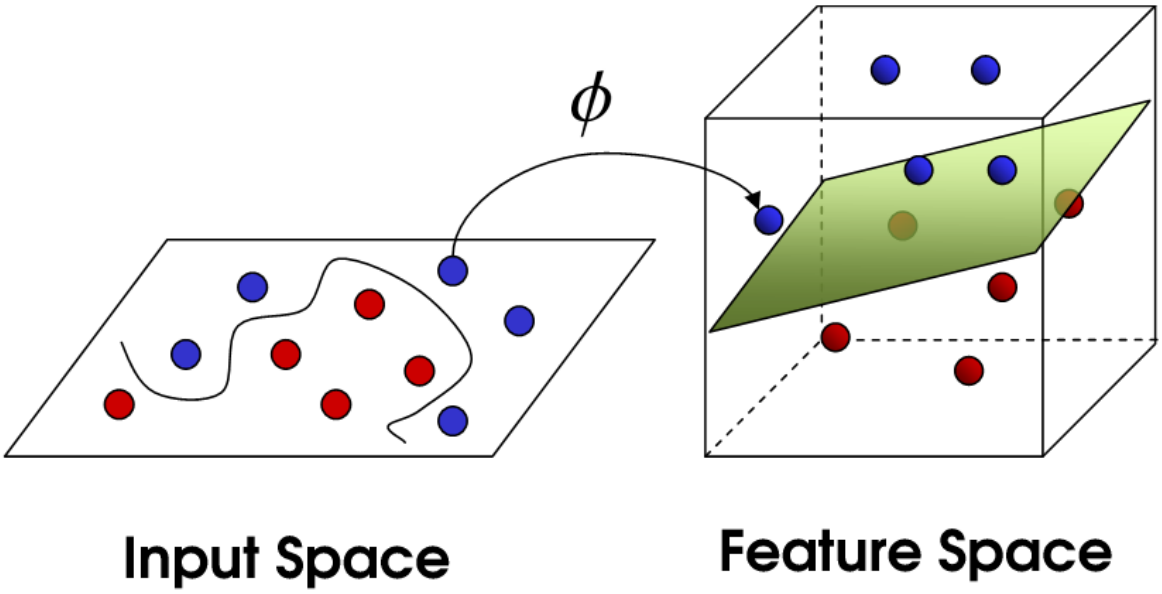
SVC je metóda „učenia s učiteľom“ (angl. Supervised learning), ktorá vychádza z metódy zvanej: Support Vector Machines (SVM), používanej predovšetkým na klasifikáciu dát. SVM dáta oddeľuje tak, že sa snaží nájsť maximálnu vzdialenosť okrajov dátových bodov (angl. Maximal margin), patriacich do rôznych kategórií a zároveň minimalizovať nepresnosti klasifikácie.[9]



Obr. 4. Maximalizácia vzdialenosti od okrajov

Na obrázku 4. vidíme dátové body dvoch rôznych kategórií oddelené priamkou, prechádzajúcou stredom priestoru, ktorý na obrázku vznikol vytvorením prerušovaných čiar, ktoré prechádzajú bodmi, predstavujúcimi okrajové body týchto dvoch kategórií.

Dáta na obrázku sú lineárne rozdeliteľné, my však v našom prípade budeme zbierať dáta, ktoré pravdepodobne nie sú dobre lineárne rozdeliteľné. Čo v takomto prípade môžeme urobiť je použiť takzvaný „jadrový trik“ (angl. kernel trick), vďaka ktorému dokážeme transformovať nelineárne rozdeliteľný priestor na lineárne rozdeliteľný tak, že priestor v ktorom sú naše dáta nelineárne rozdeliteľné, transformujeme pomocou „funkcie jadra“ (angl. kernel function), na viac-rozmerný priestor, ako je pôvodný, v ktorom už dáta budú lineárne rozdeliteľné.[10]

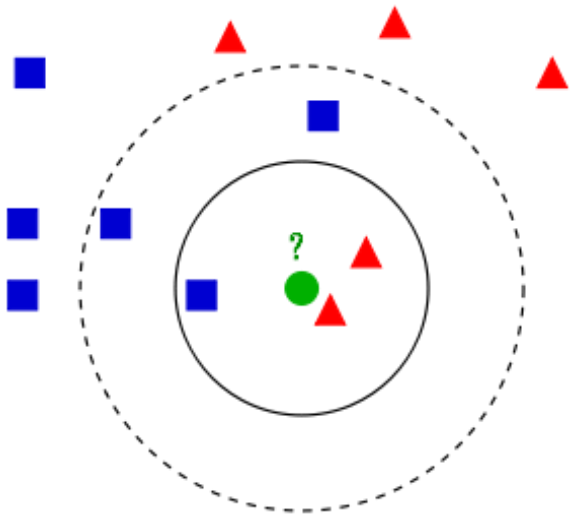


Obr. 5. Kernel trick

Na obrázku 5. je ilustrovaná transformácia z nelineárne rozdeliteľného vstupného priestoru na priestor, v ktorom už dáta dokážeme pomocou nadroviny (angl. Hyperplane) lineárne oddeliť. Tento trik sa používa aj pri niektorých iných metódach strojového učenia.

1.6.2 K Nearest Neighbours

Metóda „K najbližších susedov“ (angl. K nearest neighbours), je klasifikačná metóda, založená na jednoduchom princípe. Algoritmus sa rozhoduje, do ktorej kategórie zaradiť nový dátový bod, na základe počtu najbližších susedov v okolí tohto bodu, pričom tento bod zaradíme do tej kategórie, ktorej počet susedných bodov je najvyšší. Počet susedov, na ktorých berieme ohľad pri rozhodovaní o klasifikovaní tohto bodu, určuje parameter 'k'.



Obr. 6. KNN klasifikácia

Na obrázku KNN klasifikácie vidíme bod, ktorý sa snažíme klasifikovať, reprezentovaný zeleným kruhom. Okolo tohto bodu vidíme objekty patriace do dvoch rôznych kategórií v podobe modrého štvorca a červeného trojuholníka. Z obrázku môžeme vyvodiť, že parameter 'k' je nastavený na hodnotu 3. Zistili sme to tak, že v priestore vo vnútri kružnice s neprerušovanou čiarou sa okrem bodu, ktorý sa snažíme klasifikovať nachádzajú ešte tri ďalšie body. V tomto momente spočítame koľko najbližších susedov patrí do kategórie červených trojuholníkov a koľko do modrých štvorcov. Vidíme že dvaja susedia sú červený a jeden modrý, takže by sme v tomto prípade bod, ktorý sa snažíme klasifikovať, zaradili do kategórie červených trojuholníkov.

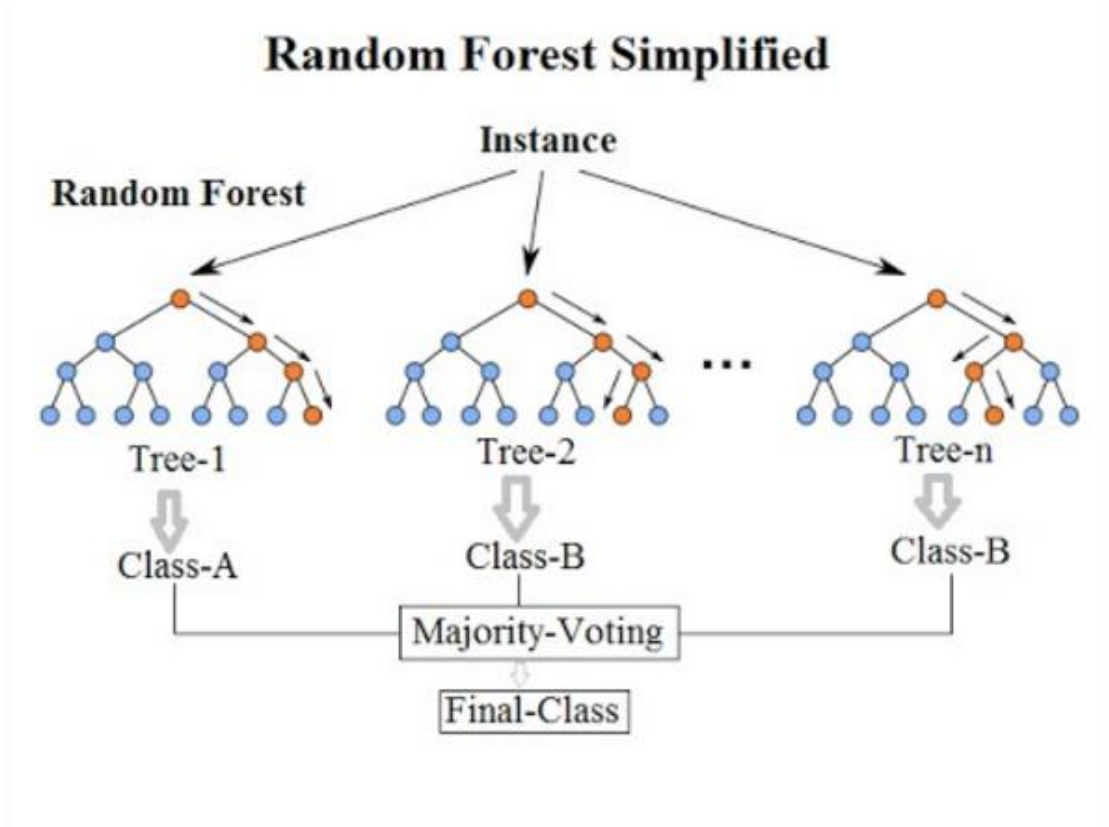
Pokiaľ by parameter 'k' bol nastavený na hodnotu 5, čo predstavuje priestor vo vnútri kružnice s prerušovanou čiarou, by bol výsledok opačný. Bod, ktorý sa snažíme klasifikovať by sme zaradili do kategórie modrých štvorcov, pretože by v danom priestore prevažovali v pomere troch ku dvom. Pri zvolení párnej hodnoty parametra 'k', môže dôjsť ku remíze pri počítaní okolitých bodov patriacich do rôznych kategórií, kde potom treba zvážiť či daný bod klasifikujeme náhodne, alebo nejakým iným spôsobom. Ak sa chceme prípadných remízam jednoducho vyhnúť, je zrejme vhodné Parameter 'k' nastaviť na nepárne číslo.

1.6.3 Random Forest

„Náhodný les“ (angl. Random forest) je metóda, využívajúca spôsob „učenia súborom modelov“ (angl. Enseble learning), kde sa použije viacero takzvaných „slabých klasifikátorov“ (angl. weak classifiers), ktorých kombinácia vytvorí jeden „silný klasifikátor“ (angl. strong classifier). Môže byť použitá na problémy klasifikácie, aj regresie. Slabé klasifikátory, z ktorých Náhodný les pozostáva, sa nazývajú „Rozhodovacie stromy“ (angl. Decision trees).[11]

1.6.3.1 Rozhodovací strom

Rozhodovací strom je klasifikátor so stromovou štruktúrou. Koreňovým uzlom stromu je rozhodovací uzol, ktorý definuje otázku, ktorá sa má vyhodnotiť. Dostupné odpovede na otázku rozdeľujú strom na vetvy. Počet možných odpovedí zodpovedá počtu vytvorených vetiev. Počet vytvorených vetiev testovacieho uzla je závislý na použitom algoritme. Ak obsahuje strom vždy len dve vetvy nazývame ho binárny. Každá vetva stromu vedie buď k ďalšiemu rozhodovaciemu uzlu, alebo k listovému uzlu, ktorý je posledný na aktuálnej ceste stromu. Každý rozhodovací uzol vyhodnocuje údaje zo vstupu, ktoré sú definované pre riešenie daného problému.[12] Listový uzol obsahuje výstupnú hodnotu stromu.



Obr. 7. Náhodný les

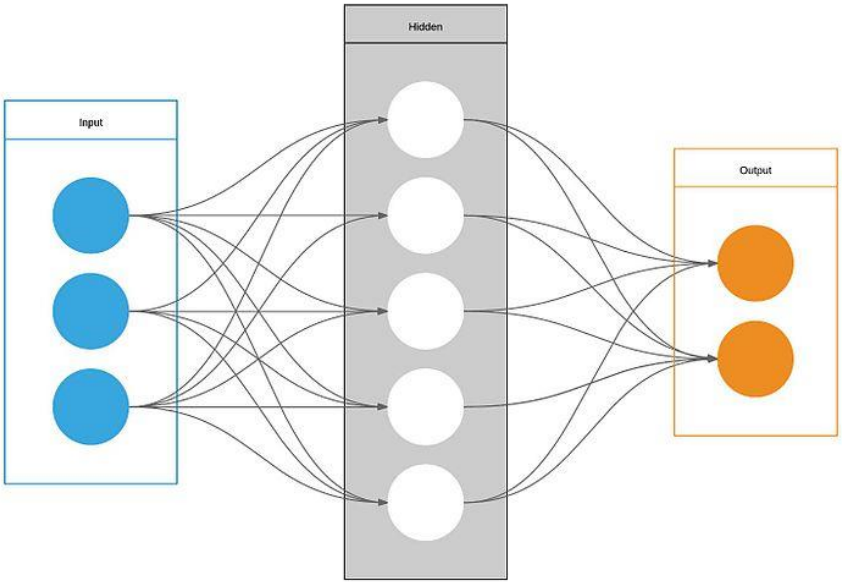
Metóda trénovania modelu náhodného lesa spočíva v tom, že si zvolíme počet stromov, nech je ich n. Pre každý rozhodovací strom sa vyberie náhodná podmnožina dát zo vstupnej tréningovej množiny dát, časť dát zo vstupnej množiny sa môže vynechať, alebo aj opakovať. Z množiny parametrov (angl. features), vyberieme náhodný počet parametrov. Na základe týchto podmnožín dát a parametrov sa potom vytvorí n stromov, pričom každý strom je vytvorený na jednej podmnožine a podmnožiny sú navzájom rôzne. Týmto zabezpečíme, že výsledky jednotlivých rozhodovacích stromov sa budú navzájom do istej miery odlišovať. Pri klasifikácii nového dátového bodu potom od každého rozhodovacieho stromu získame výsledok predikcie kategórie, do ktorej by daný bod zaradil. Spočítame výsledky predikcií do jednotlivých kategórií, pre ktoré sa rozhodovacie stromy rozhodli, tieto výsledky spriemerujeme a následne dátový bod, zaradíme do tej kategórie, ktorá získala najväčší počet hlasov.[13] Výhodou oproti použitiu jediného rozhodovacieho stromu je okrem iného vyššia presnosť predikcie a nižšia šanca pretrénovania (angl. overfitting) modelu. Vo všeobecnosti platí, že viac stromov nám poskytne lepšiu presnosť klasifikácie, no nevýhodou je vyššia komplexita modelu a nižšia rýchlosť predikcie, pretože musíme prejsť cez všetky rozhodovacie stromy.[11]

1.6.4 Extremely Randomized Trees

Táto metóda je veľmi podobná metóde náhodného lesa, odlišuje sa tým, ako rieši výber dát zo vstupnej tréningovej množiny dát, a použitie parametrov dát. Využíva viac náhodnosti pri vytváraní rozhodovacích stromov. Pre jednotlivé vrcholy stromu vyberie náhodnú vzorku parametrov dát a pre každý parameter zvolí náhodný prah vetvenia stromu.[14]

1.6.5 Multilayer Perceptron (MLP)

„Viacvrstvový perceptrón“ (angl. Multilayer perceptron), označovaný aj ako „Umelá neurónová sieť“(angl. Artificial neural network), je metóda strojového učenia, používaná pri klasifikácii aj regresii. Označenie neurónovej siete má pre to, že táto metóda vznikla na základe inšpirácie zo skúmania fungovania ľudského mozgu, konkrétne spôsobu akým sa mozog učí. Neurónová sieť je v počítači reprezentovaná vo forme orientovaného grafu.
Jednotlivé vrcholy grafu sú takzvané „neuróny“, ktoré spracúvajú a posielajú informáciu ďalej smerom k výstupným neurónom. [15]



Obr. 8. Neurónová sieť

Na obrázku 8. vidíme príklad jednoduchej neurónovej siete. Táto sieť sa skladá zo vstupnej vrstvy, v podobe modrých vrcholov grafu, za ňou nasledujúcou takzvanou „skrytou vrstvou“ (angl. hidden layer) , za ktorou nasleduje výstupná vrstva, reprezentovaná oranžovými vrcholmi grafu. Neurónová sieť môže obsahovať aj viacero skrytých vrstiev, kedy hovoríme, že ide o takzvanú metódu „hlbokého učenia“ (angl. deep learning).
Vstupná vrstva obsahuje informácie o vstupných dátach a výstupná vrstva výslednú hodnotu.
Pri klasifikácii dát je výsledná vrstva v podobe kategórií, do ktorých vstupné dáta zaraďujeme.

Neuróny jednotlivých vrstiev sú medzi sebou poprepájané takzvanými „synapsami“, pričom v synapse sa môže hodnota, ktorá predstavuje „silu aktivácie neurónu“, ktorú každý neurón obsahuje, putujúca k ďalšiemu neurónu zosilniť, alebo zoslabiť. Silu pôsobenia synapsy určuje „váha“ (angl. wieght). Výsledný signál na vrchole dostaneme sčítaním všetkých signálov, ktoré doňho smerujú. Takto dostaneme lineárnu väzbu medzi neurónmi. Pre nelineárnu väzbu signál prenásobený váhou synapsy dosadíme do „aktivačnej funkcie“. Takéto spojenie nazývame „Aktivačné spojenie“. V prípade, že na vstup dostaneme čisto nulový signál, používame „pseudo input“, nazývaný bias. Predstavuje synaptický spoj vstupu, ktorý je stále daný hodnotou +1.[15]

Neurónové siete predstavujú skutočne mocnú metódu strojového učenia, vďaka ktorej sa pri použití dostatočne výkonného stroja, podarilo nájsť spojitosti v dátach, aj tam kde ich ľudská myseľ nevidí. Z dôvodu nedostatku priestoru nebudeme zachádzať do hĺbky a preto pre lepšie pochopenie ich fungovania, sa odporúča čitateľovi hlbšie naštudovanie tejto problematiky.

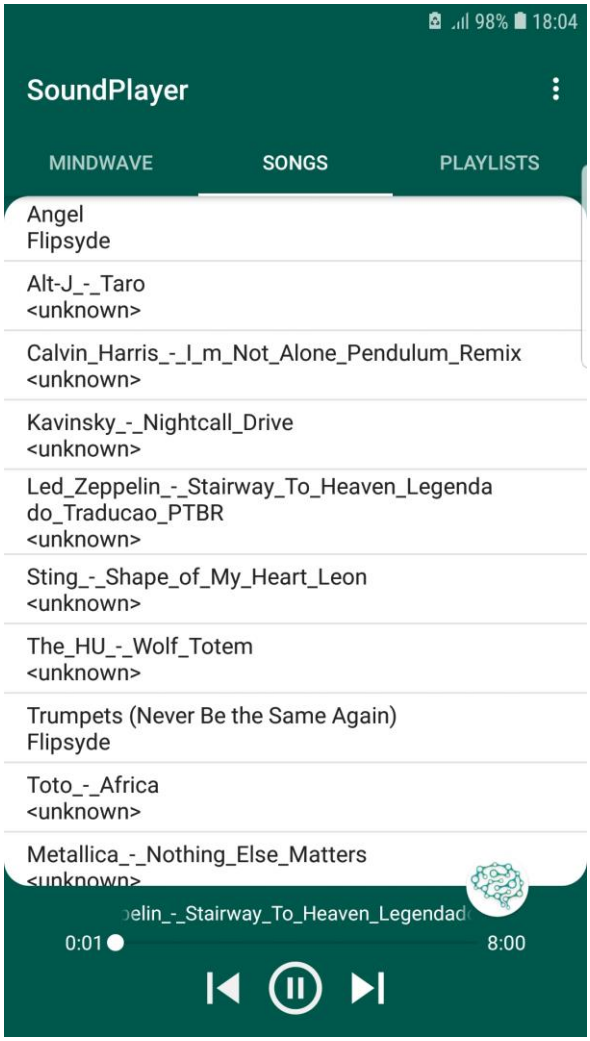
2 Návrh

Cieľom práce je vytvorenie aplikácie zvukového prehrávača, ktorého hlavnú časť bude možné ovládať pomocou MindWave Mobile, okrem ovládania rukou. Po zhodnotení vlastností a možností zariadenia, nasleduje návrh samotnej aplikácie zvukového prehrávača, do ktorej bude zakomponované SDK zariadenia. V tejto kapitole sa budeme venovať návrhu aplikácie zvukového prehrávača a popíšeme si funkcionalitu jednotlivých častí aplikácie.

2.1 Aplikácia zvukového prehrávača

Aplikácia je vytvorená v prostredí Android Studia[16] a skladá sa z dvoch hlavných fragmentov: Songs a MindWave. Každý fragment predstavuje logickú časť na pozadí jednej „obrazovky“, preto ich tak, pre lepšiu predstavu, aj budeme ďalej nazývať, hoci v skutočnosti fragmentov môže jedna „obrazovka“ obsahovať aj viacero.

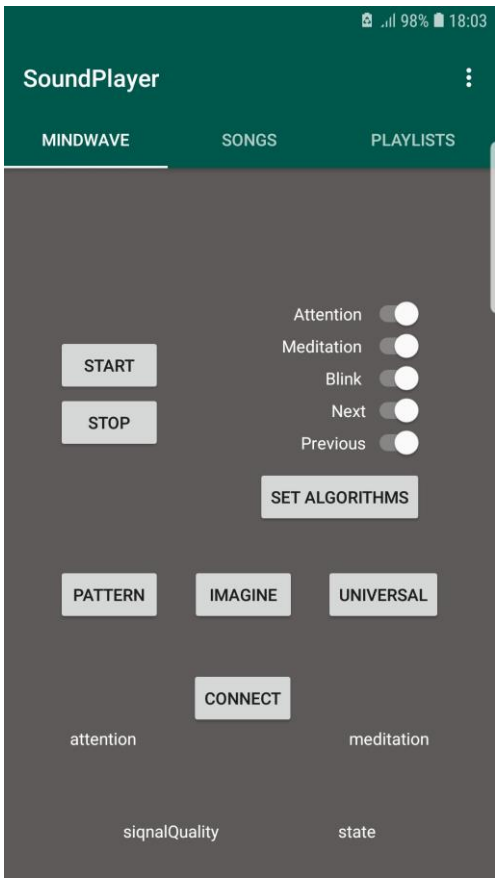
2.1.1 Obrazovka Songs



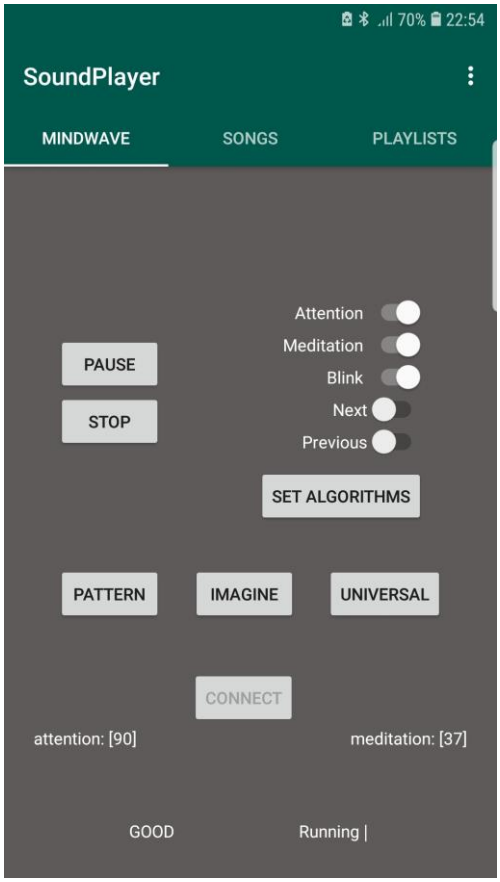
Obr. 9. Songs

Obrazovka Songs predstavuje hlavnú obrazovku zvukového prehrávača, na ktorej sa zobrazuje zoznam zvukových stôp, prípadne aj s menom autora stôp, pokiaľ ho zvukový súbor uložený v pamäti telefónu obsahuje. Obrazovka Songs je plne ovládateľná rukou, používateľ môže kliknúť na názov ľubovoľnej stopy v zozname, pričom sa následne táto stopa spustí a jej názov sa zobrazí pod zoznamom zvukových stôp. Tlačidlá na paneli pod zoznamom stôp fungujú intuitívne, teda ak stlačíme tlačidlo Next (v zmysle „ďalší“), ktoré predstavuje šípka vpravo, tak sa zastaví prehrávanie aktuálnej zvukovej stopy a spustí sa stopa nasledujúca. Podobne je to aj pri stlačení tlačidla Previous (v zmysle „predchádzajúci“), ktoré predstavuje šípka vľavo, kde sa spustí predchádzajúca zvuková stopa. Samotné tlačidlo Play (v zmysle „spustiť“) v strede panela slúži na spustenie, respektíve pozastavenie aktuálne zvolenej zvukovej stopy. Pretáčanie zvukovej stopy sa realizuje pomocou seekbaru (indikátora v podobe bielej guľôčky), presunutím na konkrétne miesto od ktorého chceme zvukovú stopu spustiť.

2.1.2 Obrazovka MindWave



Obr. 10. MindWave po spustení aplikácie



Obr. 11. MindWave po zapnutí algoritmov

Obrazovka MindWave predstavuje panel informácií a možností využitia MindWave Mobile v aplikácii. Aplikácia sa najskôr tlačidlom Connect pripojí ku zariadeniu, kde si používateľ následne môže zvoliť z piatich algoritmov, pomocou ktorých môžeme ovládať hlavnú časť zvukového prehrávača. Výber algoritmov sa potvrdí tlačidlom Set algorithms. Tlačidlá Start

a Stop slúžia na spustenie, respektíve zastavenie jednotlivých algoritmov v aplikácii, aj prenosu surových dát EEG zo zariadenia, pričom tlačidlo Stop zároveň preruší spojenie so zariadením.

2.1.2.1 Algoritmy obrazovky MindWave

- **Attention** - Jedná sa o už spomínaný algoritmus, ktorý nám ponúka priamo API zariadenia. V ľavom dolnom rohu obrazovky na obrázku 11. môžeme vidieť výstupnú hodnotu algoritmu Attention, ktorá nám ukazuje stupeň sústredenia, respektíve či sa subjekt práve sústredí. Pokiaľ táto hodnota vystúpi na, alebo nad stanovený prah, ktorý je, pre čo najlepšiu presnosť v našom prípade, nastavený na hodnotu 100, čiže maximálnu hodnotu algoritmu, odošle sa odozva v podobe požiadavky na proces, ktorý sa má vykonať, čo je v našom prípade zvýšenie hlasitosti prehrávania zvukových stôp.

- **Meditation** - Algoritmus Meditation sa správa veľmi podobne ako algoritmus Attention. Keď vystúpi výstupná hodnota Meditation na hodnotu 100, odošle sa odozva v podobe stíšenia hlasitosti prehrávania zvukových stôp.

- **Blink** - Posledný algoritmus z API k MindWave. Algoritmus Blink deteguje žmurknutie okom, pričom v našej aplikácií zvukového prehrávača je nastavený tak, že keď používateľ žmurkne dva krát po sebe v intervale kratšom ako je 450 milisekúnd, tak sa odošle odozva pre spustenie, respektíve pozastavenie prehrávania aktuálne zvolenej zvukovej stopy. Hodnota 450 milisekúnd bola zvolená ako vhodná, na základe experimentovania.

- **Next** - Algoritmus Next (v zmysle „ďalší“) sa nenachádza v priloženom SDK zariadenia. Jeho názov je odvodený od funkcie, ktorá je s ním spojená. Keď natrénované modely rozpoznajú po dobu aspoň troch sekúnd, že aktuálne surové dáta, ktoré získame z MindWave Mobile, patria do kategórie dát, s týmto algoritmom spojenej, odošle sa požiadavka na spustenie nasledujúcej zvukovej stopy zo zoznamu stôp. Doba troch sekúnd bola opäť zvolená experimentovaním za účelom minimalizovania zle rozpoznanej mozgovej aktivity.

- **Previous** - Algoritmus Previous (v zmysle „predchádzajúci“) funguje podobne ako algoritmus Next. V prípade zaradenia aktuálnych surových dát mozgových vĺn, zosnímaných zariadením, do kategórie spojenej s týmto algoritmom, sa spustí predchádzajúca zvuková stopa v zozname stôp.

  Na obrazovke MindWave sa nachádzajú ešte tri tlačidlá: Pattern, Imagine a Universal. Funkciou týchto tlačidiel je zber dát do troch rôznych kategórií dát, pričom tieto dáta uložíme do súboru, na ktorom naše modely pre algoritmy Next a Previous natrénujeme.

3 Implementácia

V tejto kapitole si ukážeme, ako sme riešili implementáciu aplikácie, zber dát potrebných na natrénovanie modelov strojového učenia, aj samotné trénovanie modelov, ktoré použijeme na rozšírenie ovládania zvukového prehrávača, pomocou MindWave Mobile.

3.1 Grafické prostredie aplikácie

Na vytvorenie grafického používateľského rozhrania (GUI) aplikácie používame Layout Editor, ktorý je súčasťou Android Studia. Dizajn jednotlivých obrazoviek, spolu so všetkými elementami daných obrazoviek, ktoré predstavujú všetky tlačidlá, zoznam zvukových stôp, ovládací panel, text, atď. vytvárame pomocou jazyka XML. Kód GUI je oddelený od logickej časti aplikácie, uložený v samostatných súboroch priložených k projektu. Na prepojenie logickej časti aplikácie s grafickým prostredím používame pri každom XML elemente unikátne id, napr. názov, podľa ktorého ich vieme jednoznačne navzájom odlíšiť.

3.2 Logická časť aplikácie

Logickú časť aplikácie opäť riešime v prostredí Android Studia, použitím jazyka Java. Na pozadí fragmentov („obrazoviek“) beží hlavná aktivita aplikácie (MainActivity), ktorá jednotlivé fragmenty vytvára a zároveň slúži aj ako prostredník v komunikácií medzi fragmentami. Fragmenty o existencii ostatných fragmentov nevedia a preto aby spolu mohli komunikovať, hlavná aktivita implementuje rozhranie FragmentCommunicator, ktoré obsahuje metódu respond (v zmysle odozva). Pomocou metódy respond si posielajú jednotlivé fragmenty svoju odozvu. Poslednou úlohou hlavnej aktivity je získanie povolenia na prístup k priečinkom telefónu a nájdenie zvukových stôp v pamäti telefónu, ktoré sa zobrazujú vo fragmente Songs.

Funkcionalitu Zvukového prehrávača vo fragmente Songs sme implementovali pomocou štandardnej triedy MediaPlayer[17]. Triedu MediaPlayer používame na pozadí tlačidiel a zoznamu stôp fragmentu Songs, na spustenie, alebo zastavenie aktuálne zvolenej zvukovej stopy. Pre zvyšovanie, alebo znižovanie hlasitosti používame triedu AudioManager[18].

Na pripojenie zariadenia k telefónu cez Bluetooth, používame štandardnú triedu BluetoothAdapter[19]. Fragment MindWave v sebe zahŕňa aj SDK zariadenia. Na správu spojenia používame triedy z SDK: TgStreamReader a TgStreamHandler. Funkcionalitu algoritmov z SDK rieši trieda NskAlgoSdk, ktorej príklady použitia sú v kapitole 3.2.1.

3.2.1 Ukážky použitia SDK

Nasledujúca podkapitola obsahuje niektoré časti kódu, ktoré používame vo svojom projekte, dostupné v SDK.

- Ukážka kódu spracovania EEG dát [20]:

```
public void onDataReceived(int datatype, int data, Object obj) {
    switch (datatype) {
        case
            MindDataType.CODE_ATTENTION:
            short attValue[] =
            {(short)data};
            NskAlgoDataStream(NskAlgoDataType.NSK_ALGO_DATA_TYPE_ATT.value,
            attValue, 1); break;
        case MindDataType.CODE_MEDITATION:
            short medValue[] = {(short)data};
            NskAlgoDataStream(NskAlgoDataType.NSK_ALGO_DATA_TYPE_MED.value,
            medValue, 1); break;
        case MindDataType.CODE_POOR_SIGNAL:
            short pqValue[] = {(short)data};
            NskAlgoDataStream(NskAlgoDataType.NSK_ALGO_DATA_TYPE_PQ.value,
            pqValue, 1); break;
        case MindDataType.CODE_RAW:
            raw_data[raw_data_index++] = (short)data;
            if (raw_data_index == 512) {
                NskAlgoDataStream(NskAlgoDataType.NSK_ALGO_DATA_TYPE_EEG.value,
raw_data, raw_data_index);
                    raw_data_index = 0;
            }
            break;
        default:
            break;
    }
}
```

- SDK Listener Metódy[11]:

setOnAttAlgoIndexListener

Attention Algorithm index notification listener method

```
public void setOnAttAlgoIndexListener(OnAttAlgoIndexListener listener);
```

Example

```
NskAlgoSdk nskAlgoSdk = new NskAlgoSdk();
nskAlgoSdk.setOnAttAlgoIndexListener(new NskAlgoSdk.OnAttAlgoIndexListener() {
    @Override
    public void onAttAlgoIndex(int value) {

       Log.i(TAG, "NskAlgoAttAlgoIndexListener: Attention:" + value);
       runOnUiThread(new Runnable() {
           @Override
           public void run() {
           }
       });
     }
});
```

setOnMedAlgoIndexListener

Meditation Algorithm index notification listener method

```
public void setOnMedAlgoIndexListener(OnMedAlgoIndexListener listener);
```

Example

```
NskAlgoSdk nskAlgoSdk = new NskAlgoSdk();

nskAlgoSdk.setOnMedAlgoIndexListener(new NskAlgoSdk.OnMedAlgoIndexListener() {
     @Override
    public void onMedAlgoIndex(int value) {
        Log.i(TAG, "NskAlgoMedAlgoIndexListener: Meditation:" + value);
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
            }
        });
    }
});
```

setOnBPAlgoIndexListener

EEG Bandpower Algorithm index notification listener method

```
public void setOnBPAlgoIndexListener(OnBPAlgoIndexListener listener);
```

Example

```
NskAlgoSdk nskAlgoSdk = new NskAlgoSdk();

nskAlgoSdk.setOnBPAlgoIndexListener(new
    NskAlgoSdk.OnBPAlgoIndexListener() {
    @Override
    public void onBPAlgoIndex(float delta, float theta, float alpha, float beta, float gamma) {
        Log.i(TAG, "NskAlgoBPAlgoIndexListener: BP: D[" + delta + " dB] T[" + theta + " dB] A["
        + alpha + " dB] B[" + beta + " dB] G[" + gamma + "]");
runOnUiThread(new Runnable() {
            @Override
            public void run() {
            }
        });
    }
});
```

setOnEyeBlinkDetectionListener

```
public void setOnEyeBlinkDetectionListener(OnEyeBlinkDetectionListener listener);
```

Example

```
NskAlgoSdk nskAlgoSdk = new NskAlgoSdk();

nskAlgoSdk.setOnEyeBlinkDetectionListener(new
    NskAlgoSdk.OnEyeBlinkDetectionListener() { @Override
    public void onEyeBlinkDetection(int value) {
        Log.i(TAG, "NskAlgoEyeBlinkDetectionListener: Eye blink
        detected");
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
            }
        });
    }
});
```

3.3 Zber dát

Pre to aby sme rozšírili možnosti ovládania aplikácie pomocou MindWave Mobile, sú pre nás kľúčové dáta. ktoré potrebujeme na natrénovanie modelov, pomocou strojového učenia. Dôležitým faktorom však nie je len dostatok dát, ale predovšetkým kvalita dát.
Pre dosiahnutie čo najväčšej úspešnosti rozpoznávania jednotlivých stavov mozgovej aktivity je dôležité, aby sme zachytili stavy, ktoré sú od seba dostatočne odlišné. Čím budú dáta jednotlivých kategórií navzájom odlišnejšie od dát v ostatných kategóriách, tým ich bude ľahšie možné jednoznačne oddeliť a s o to väčšou presnosťou ich naše modely budú vedieť klasifikovať.

Pri zbere surových dát je kľúčový práve algoritmus BandPower, ktorý sme si už spomenuli v kapitole 1.5.1. BandPower umožňuje zbierať dáta mozgových vĺn v piatich hladinách a to ako vektor piatich hodnôt aktualizovaný jeden krát za sekundu. Headset má žiaľ len jednu elektródu, ktorá je umiestnená na čele, čo nás obmedzuje na pozorovanie mozgovej aktivity iba prednej časti mozgu. Po zhodnotení viacerých faktorov ovplyvňujúcich výber mozgovej aktivity, ktorú potrebujeme opakovane vyvolať a následne zachytiť, sme sa rozhodli zbierať dáta troch rôznych kategórií. Tieto kategórie predstavujú práve tlačidlá Pattern, Imagine a Universal, umiestnené na obrazovke MindWave. Dáta, ktoré zbierame, sú zapisované do spoločného súboru Train.csv.

1	Label	Alpha	Beta	Gamma	Delta	Theta	Date
2	2	6.97739	4.59047	0.64881	2.4854	10.4861	41:39.0
3	2	7.45245	3.02531	0.70754	4.00576	10.3423	41:39.8
4	2	7.45612	3.22599	-0.01704	5.16649	10.5778	41:41.0
5	2	8.57257	3.89224	0.21428	7.87883	11.8593	41:41.9
6	2	9.07193	4.58566	1.32877	7.52626	12.1723	41:43.0
7	2	9.0066	4.59135	1.29479	2.9622	9.95064	41:54.8
8	2	7.46214	5.39631	3.92709	1.74107	8.75086	41:55.8
9	2	7.51244	5.37783	5.09529	0.56008	8.63347	41:56.9
10	2	6.94034	6.14662	6.09572	2.23409	8.91387	41:57.9
11	2	7.7663	6.74273	6.83834	5.46848	10.2215	41:58.9
12	2	8.29036	6.51725	6.45497	6.52967	11.2036	41:59.9
13	2	7.56484	5.41227	5.04351	6.32667	11.4638	42:00.8
14	2	8.2113	4.55697	3.56014	6.24798	12.1195	42:01.8
15	2	8.77124	4.13892	1.41258	6.16776	12.2914	42:02.9
16	2	8.70318	4.30266	1.18196	4.29992	11.8567	42:03.9
17	2	8.67739	4.50278	0.81008	6.52291	12.8076	42:04.8
18	2	8.39055	4.95506	1.16783	4.53791	11.5089	42:15.6
19	2	7.36624	4.34539	0.67151	4.34253	11.0883	42:16.7
20	2	6.46301	3.74698	-0.05129	3.90299	11.7126	42:17.7
21	2	6.50042	3.13779	-0.47737	3.34996	10.5882	42:18.7
22	2	5.49875	3.37104	-0.29804	0.36374	9.87512	42:19.8
23	2	6.77801	3.25069	-0.15235	0.04445	9.93975	42:20.8
24	2	7.83538	2.65408	0.22978	4.22797	9.77776	42:21.7
25	2	8.17442	1.8526	-0.0986	3.91607	10.8284	42:22.7

Obr. 12. Súbor Train.csv

Na obrázku 12. vidíme prvých 25 riadkov spoločného súboru, do ktorého sú dáta počas zbierania zapisované. Každý riadok obsahuje Label („označenie“), ktorý obsahuje hodnotu v podobe celého čísla z intervalu 0 až 2, pričom dáta typu Pattern majú Label pevne nastavený na hodnotu 0, dáta typu Imagine na hodnotu 1 a dáta typu Universal na hodnotu 2. Táto hodnota predstavuje informáciu o tom do ktorej kategórie dát daný riadok patrí, čo je nutné vedieť pri spôsobe trénovania modelov, ktorý v našej aplikácii používame.

Okrem Labelu každý riadok obsahuje hodnoty piatich hladín mozgových vĺn. Poslednou informáciou každého riadka je políčko Date, ktoré predstavuje dátum, kedy bol daný riadok zapísaný do súboru. Pre naše modely samotný dátum nehrá žiadnu rolu, je to len informácia navyše, ktorá môže byť užitočná napríklad pri analýze zozbieraných dát, prípadne pri použití na indexovanie dát. Pre lepšie pochopenie čo predstavujú tlačidlá: Pattern, Imagine a Universal a ich Label 0,1 a 2 si v nasledujúcich podkapitolách popíšeme spôsob zbierania týchto dát.

3.3.1 Dáta typu Pattern

Pod pojmom „pattern“ máme na mysli určitý vzorec správania v mozgu, ktorý sa snažíme opakovane napodobniť. Dáta typu Pattern predstavujú jednu kategóriu dát, ktorej dáta zbierame tak, že subjekt má nasadené zapnuté zariadenie na hlave a elektródu umiestnenú čele. Po stlačení tlačidla Pattern sa začne odpočítavať čas jedného cyklu zbierania dát. Tento čas je nastavený na trinásť sekúnd. Subjekt má počas týchto trinástich sekúnd zatvorené oči a sústredí sa svojím vnútorným zrakom aj zatvorenými očami smerom ku elektróde na čele.

Po odpočítaní času stanovenom na jeden cyklus zbierania, sa zbieranie ukončí a používateľ je o ukončení upozornený prehraním zvukovej stopy pripomínajúcej jemný gong. Po ukončení zbierania môže subjekt opäť otvoriť oči. Keďže algoritmus BandPower nám vie poskytnúť nové dáta mozgových vĺn každú sekundu znamená, že jeden trinásť sekundový cyklus zbierania rozšíri náš súbor o trinásť nových riadkov, pričom tieto riadky budú mať Label s hodnotou 0.

Čas trinástich sekúnd pre jeden cyklus zbierania dát typu Pattern bol nastavený experimentovaním, z dôvodu optimalizácie prijateľnej dĺžky čakania na odozvu od našej aplikácie a množstva nazbieraných dát počas tohto cyklu. Človek sa zrejme nechce sústrediť na určitý vzorec správania mozgu pridlho, kým získa odozvu od aplikácie a preto sme nechceli ani zbierať dáta, keď je subjekt niekoľko minút sústredený so zatvorenými očami na jeden bod na čele, kde by sa namerané hodnoty mohli výrazne líšiť od tých v prvých desiatich sekundách.

Pre zozbieranie dostatočného množstva dát typu Pattern, tento cyklus dookola opakujeme. Pre zozbieranie čo najlepších dát typu Pattern, sme mali medzi jednotlivými cyklami zbierania krátku pauzu. Tieto dáta sú priamo spojené s algoritmom Next, čiže po klasifikovaní surových dát natrénovanými modelmi do kategórie s Labelom s hodnotou 0, sa odošle odozva v podobe spustenia nasledujúcej zvukovej stopy.

3.3.2 Dáta typu Imagine

Pod pojmom „imagine“ máme na mysli predstavovanie si určitej akcie v mysli, konkrétne ide o predstavovanie si stláčania tlačidla, ktoré spúšťa predošlú zvukovú stopu zo zoznamu. Dáta zbierame tak, že máme opäť nasadené zapnuté zariadenie pripojené ku aplikácií, s elektródou umiestnenou na čele. Po celú dobu jedného cyklu zbierania sa snažíme sústrediť, tento krát s otvorenými očami, na tlačidlo previous, respektíve „predchádzajúci“, ktoré predstavuje šípka vľavo na obrazovke Songs, prípadne si môžeme predstavovať, ako ho svojím prstom stláčame.

Jeden Cyklus zbierania trvá sedem sekúnd, pričom hodnoty z prvých troch sekúnd nie sú do súboru zapisované, z dôvodu zaznamenania optimálnejších hodnôt sústredenia. Po jednom cykle teda v súbore pribudnú štyri nové riadky, ktoré budú mať Label nastavený na hodnotu 1. V prípade dát typu Imagine nemôže jeden cyklus zbierania dát trvať príliš dlho kvôli domnienke, že myseľ má po krátkej dobe tendenciu strácať koncentráciu a ubiehať iným smerom. Opäť hodnota siedmich sekúnd bola zvolená experimentovaním.

Podobne ako pri zbieraní dát typu Pattern nás na ukončenie zapisovania dát do súboru upozorní rovnaká zvuková stopa gongu, čo nám v tomto prípade neprekáža, pretože v jednom momente môžeme zbierať dáta iba jednej kategórie dát, nemôže sa teda stať, že tie isté dáta by boli označené súčasne ako dáta typu Pattern aj Imagine. Opäť cyklus zbierania dát typu Imagine opakujeme, kým nemáme dostatočný počet dát.

3.3.3 Dáta typu Universal

Posledným typom dát, ktoré naša aplikácia umožňuje zbierať sú dáta typu Universal. Pod pojmom „universal“ máme na mysli dáta univerzálne, všeobecné. Pri zbieraní dát typu Universal sa teda na nič konkrétne nesústredíme, nenapodobňujeme žiaden konkrétny vzorec správania mozgu. Tieto dáta predstavujú akýsi „neutrálny“ stav mozgovej aktivity, ktorý má za úlohu pomôcť zistiť, či sa práve používateľ snaží napodobniť istý typ správania mozgu, alebo nie. Pri týchto dátach nemáme stanovené časové obmedzenie cyklu zbierania dát, používateľ spúšťa aj

zastavuje zapisovanie dát do súboru sám. Pre zapnutie zapisovania zbieraných dát jednoducho používateľ, s elektródou umiestnenou na čele a zariadením pripojeným ku aplikácii, stlačí tlačidlo Universal, pričom zbieranie dát zastavujeme manuálne, stlačením tohto tlačidla druhý krát. Keďže zvuk gongu v tomto prípade nemá zmysel, pretože cyklus zbierania spúšťame aj ukončujeme manuálne, je používateľ o zapnutom zapisovaní dát do súboru informovaný vizuálne, stmavením pozadia tlačidla. Ostatné tlačidlá na zbieranie dát zobrazujú taktiež okrem zvukovej aj túto istú vizuálnu informáciu, ktorá ukazuje či sú práve aktívne. Pri zapnutom zbieraní dát typu Universal pribudne každú sekundu do súboru nový riadok dát piatich hladín mozgových vĺn, ktorý má nastavený Label na hodnotu 2.

3.4 Analýza dát

Po zozbieraní dostatočného množstva dát je vhodné, pre lepšiu predstavu o kvalite dát a pochopenia správania mozgu v stavoch zaznamenávaných pomocou MindWave, dáta analyzovať. Na analýzu dát použijeme prostredie Jupyter Notebook[21] a jazyk Python, s použitím knižníc: NumPy[22], Pandas[23], Matplotlib[24] a Seaborn[25].

```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
```

Obr. 13. Import knižníc

Z dôvodu nedostatku priestoru na popis všetkých knižníc, sa v prípade záujmu, čitateľovi odporúča ich podrobnejšie preštudovanie, pretože predstavujú akýsi dnešný štandard pre prácu s dátami v jazyku Python.

Ako prvé si dáta zo súboru uložíme do tabuľky.

```
df = pd.read_csv('train.csv', parse_dates=['Date']).set_index('Date')
```

Obr. 14. Načítanie dát zo súboru

Dáta máme teraz uložené v premennej df, ktorá predstavuje tabuľku, s ktorou budeme ďalej pracovať.

```
print('Labels count = '+str(df['Label'].value_counts().sum())+':')
df['Label'].value_counts()
```

```
Labels count = 5247:

2    3246
0    1311
1     690
Name: Label, dtype: int64
```

Obr. 15. Počet riadkov súboru

Na obrázku 15. zobrazujeme počet riadkov pre jednotlivé kategórie dát.
Ako ďalšie sa môžeme pozrieť na priemerné hodnoty hladín v jednotlivých kategóriách dát.

```
print('Feature means by label:')
df.groupby('Label').mean()
```

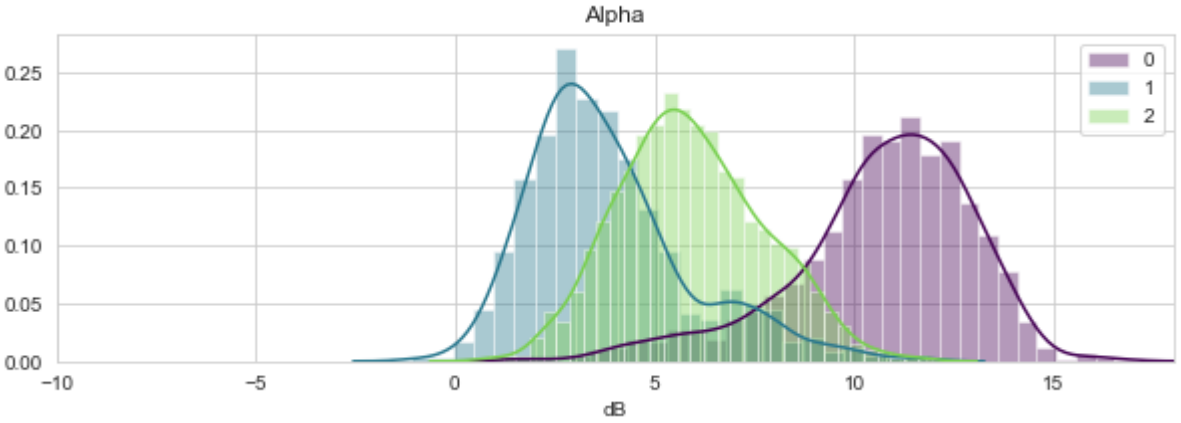
```
Feature means by label:
```

Label	Alpha	Beta	Gamma	Delta	Theta
0	10.697966	4.557121	1.971904	1.802673	10.180265
1	3.822989	1.482477	-1.116319	-1.255681	7.175921
2	5.917053	3.257057	0.765227	1.060584	8.963430

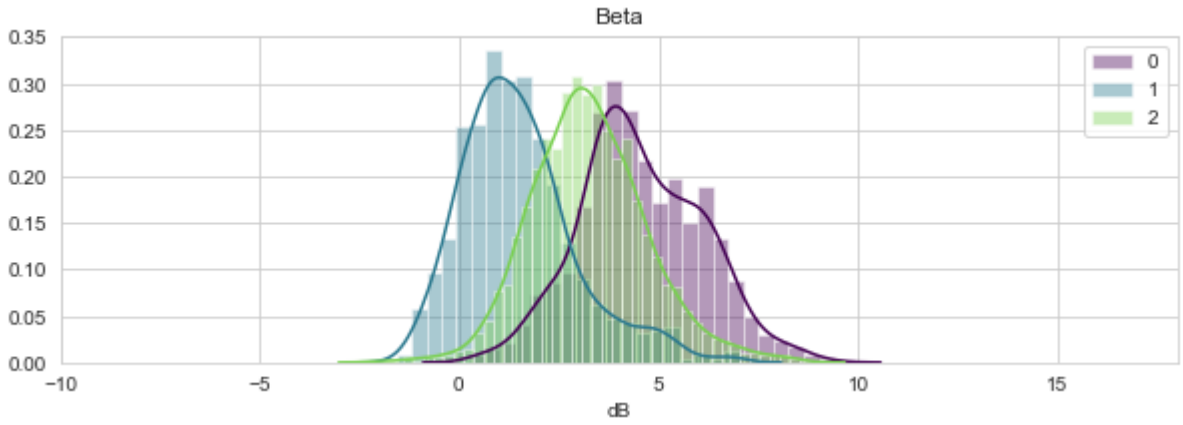
Tab. 1. Priemerné hodnoty dát

V tabuľke 1. môžeme vidieť, že rozdiel v priemerných hodnotách kategórií 0,1 a 2 v hladine Alpha je výraznejší, ako napríklad v hladine Theta, čo nám hovorí, že dáta v hladine Alpha sú zrejme ľahšie rozdeliteľné do týchto troch kategórií, ako v hladine Theta. Analogicky by sme teda mohli tvrdiť, že pokiaľ by naše dáta mali byť vo všeobecnosti dobre rozdeliteľné do kategórií, mali by sa jednotlivé kategórie, pri dostatočnom množstve dát, navzájom dostatočne líšiť v priemerných hodnotách. Toto tvrdenie však nemusí platiť vždy.

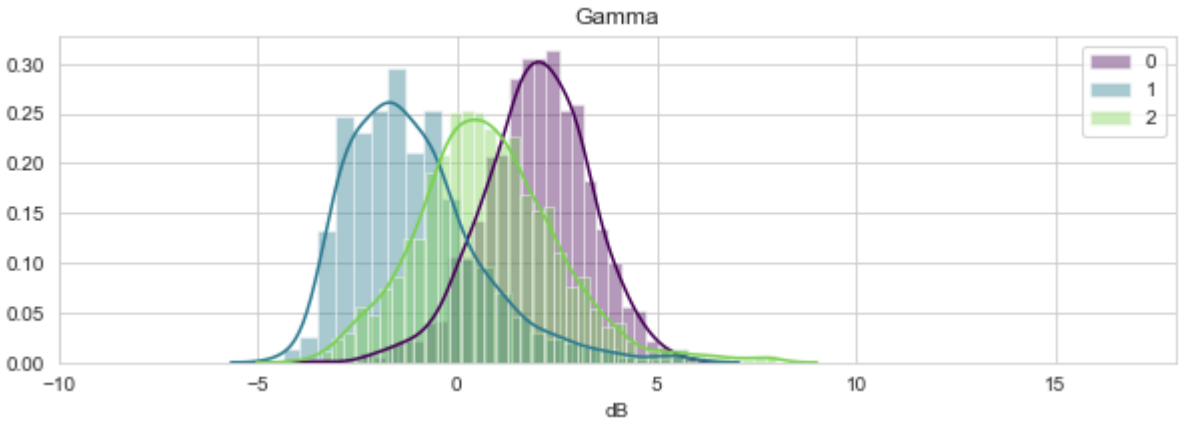
Pre lepšiu predstavu o dátach si vykreslíme grafy jednotlivých hladín, kde budú dáta rôzne ofarbené, podľa kategórie do ktorej patria.



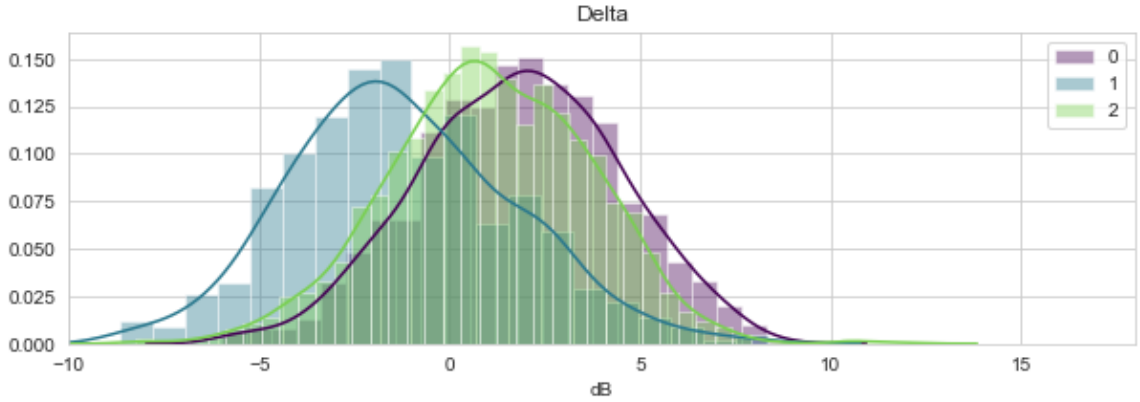
Obr. 16. Graf hladiny Alpha



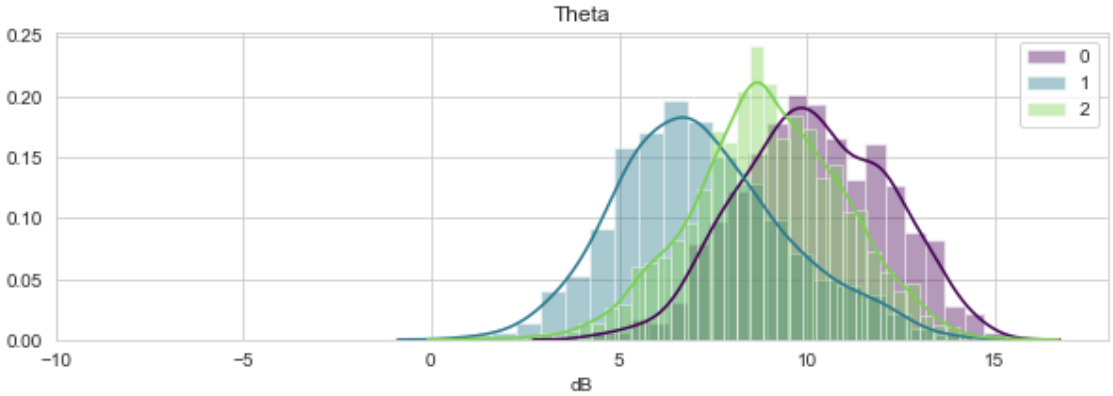
Obr. 17. Graf hladiny Beta



Obr. 18. Graf hladiny Gamma



Obr. 19. Graf hladiny Delta



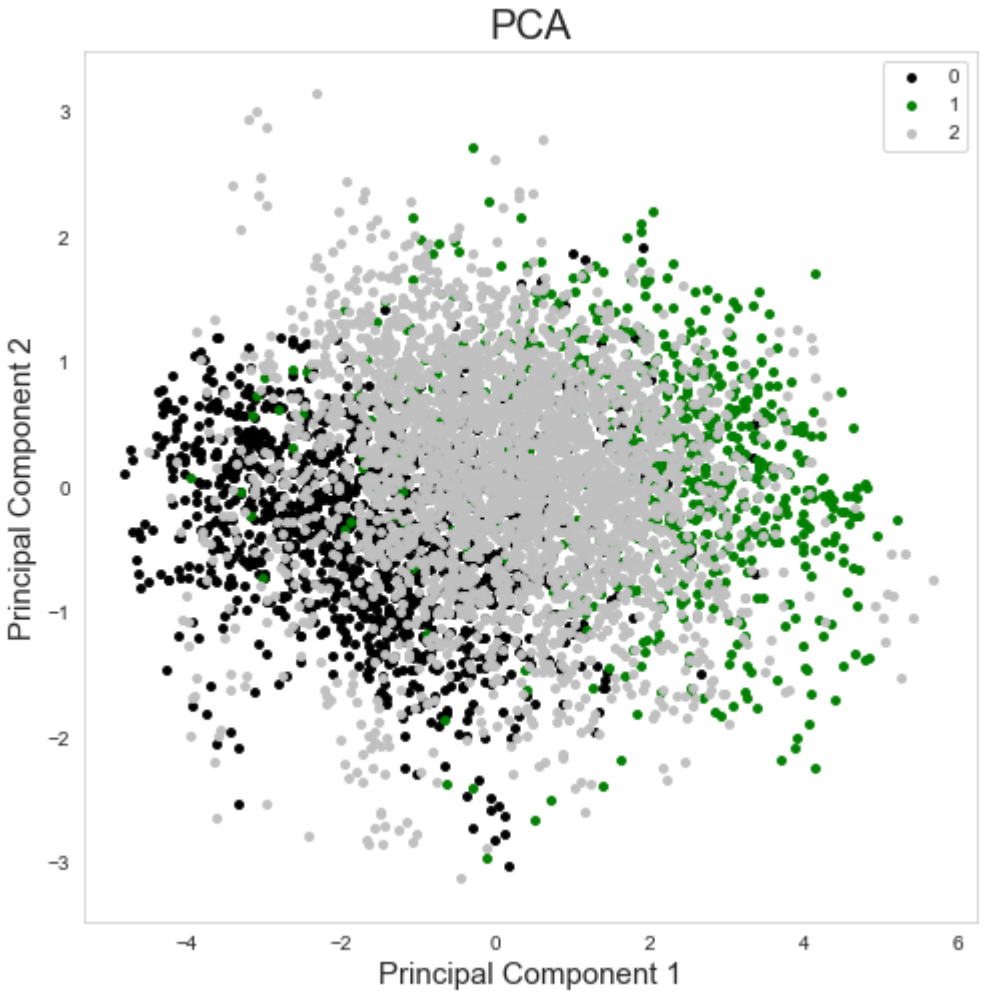
Obr. 20. Graf hladiny Theta

Pre každú hladinu vidíme histogramy, ktoré zobrazujú distribúciu dát jednotlivých kategórií. Tam kde je histogram „vyšší“, sa nachádza viac riadkov tabuľky s rovnakým labelom, ktoré obsahujú rovnakú, alebo podobnú hodnotu danej hladiny. Graf hladiny Alpha potvrdzuje pomerne dobrú rozdeliteľnosť dát, čo vidíme podľa toho, že dáta z rôznych kategórií sa navzájom na grafe príliš neprelínajú. Na grafe hladiny Delta môžeme pozorovať výraznejšie vzájomné prelínanie dát z rôznych kategórií, čo môže mať za následok vyššiu náročnosť jednoznačného rozdelenia dát tejto hladiny.

Niektoré histogramy jednotlivých hladín sú relatívne „úzke“, čo je dobre.

Znamená to, že v našich dátach nemáme priveľa extrémnych hodnôt, dáta vyzerajú akoby pochádzali z normálneho rozdelenia.

Na vizualizáciu rozptylu dát, všetkých mozgových vĺn, použijeme: „Analýzu hlavných komponentov“[26], (angl. Principal Component Analysis, skrátene PCA), pomocou ktorej zredukujeme dimenzionalitu pôvodných korelovaných dát z piatich rozmerov na dva rozmery nekorelovaných, reškálovaných dát. Pre tieto transformované dáta si vykreslíme si graf, s bodmi ofarbenými podľa kategórií, do ktorých patria.



Obr. 21. Graf PCA

```
print(pca.explained_variance_ratio_)
```

```
[0.69122026 0.14869451 0.08263205 0.04741054 0.03004264]
```

Obr. 22. Pomer rozptylu

Na obrázku 22. vidíme, že percentuálne najväčší rozptyl zhruba 69% zobrazuje prvý komponent, ktorý predstavuje x-ovú os grafu PCA. Rozptyl na tejto osi je najvýznamnejší. Druhý najväčší rozptyl dát zhruba 15% zobrazuje komponent na osi y. Zvyšné komponenty predstavujú percentuálne pomerne malý pomer rozptylu a preto je možné ich zanedbať. PCA nám taktiež pomáha odhaliť extrémne hodnoty (angl. outliers), ktoré môžu predstavovať nežiadúci šum pri zbere dát. Takéto hodnoty sa žiaľ v našich dátach nachádzajú, ale v pomerne prijateľnej miere. Nechceme ich však odstrániť úplne, aby sme modely nenatrénovali na príliš konkrétny typ dát.

Analýzou dát zisťujeme, že naše dáta žiaľ vo všeobecnosti nie sú príliš dobre rozdeliteľné, no zároveň nie sú ani príliš komplexné a preto použitím možno aj jednoduchších metód strojového učenia, by sme tak mohli dostať dobré výsledky pri klasifikácii dát do kategórií.

3.5 Vytváranie modelov

V tejto kapitole natrénujeme modely použitím metód strojového učenia ktoré sme popísali v kapitole 1.6.

Ako sme už spomenuli, spôsob trénovania modelov, ktorý použijeme, je takzvané „učenie s učiteľom“ (angl. Supervised learning), čo znamená trénovanie, pri ktorom vopred poznáme vstupné hodnoty dát (hodnoty mozgových vĺn), no zároveň aj výstupnú hodnotu dát v podobe kategórie (labelu), do ktorej patria.

Na trénovanie modelov opäť používame prostredie Jupyter Notebook a jazyk Python, pričom potrebujeme importovať ďalšie knižnice, obsahujúce algoritmy strojového učenia. Knižnica, ktorú na to použijeme sa nazýva Scikit-learn[27].

```
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
```

Obr. 23. Import knižníc zo Scikit-learn.

Medzi algoritmy, ktoré používame na trénovanie modelov patria:

```
estimators = [
    SVC(kernel='rbf', C=1.0, gamma='auto'),
    KNeighborsClassifier(7),
    RandomForestClassifier(30),
    ExtraTreesClassifier(30),
    MLPClassifier()
]
```

Obr. 24. Algoritmy strojového učenia

Každý algoritmus na obrázku 24. obsahuje viacero parametrov, ktorým sa dá optimalizovať pre konkrétne dáta. My väčšinou používame predvolené, alebo mierne upravené hodnoty parametrov, ktoré po otestovaní, fungovali vo všeobecnosti dobre.

Parametre, ktorých hodnoty sme zvolili odlišné ako predvolené, sú:
Počet najbližších susedov pri algoritme „K najbližších susedov“ na hodnotu 7 miesto hodnoty 5. Počet rozhodovacích stromov pri algoritmoch „Náhodného lesa“ a „Extrémne náhodných stromov“ na hodnotu 30 miesto hodnoty 10.

Na trénovanie modelov sme vytvorili funkciu train.

```
def train(df, estimator):
    X = df.drop('Label', 1)
    y = df['Label']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=9)
    estimator.fit(X_train, y_train)

    pred_train = estimator.predict(X_train)
    pred_test = estimator.predict(X_test)

    error_train = 1 - accuracy_score(y_train, pred_train)
    error_test = 1 - accuracy_score(y_test, pred_test)

    accuracy_train = 100*accuracy_score(y_train, pred_train)
    accuracy_test = 100*accuracy_score(y_test, pred_test)

    return accuracy_train, accuracy_test, error_train, error_test


def Model_results(df, estimators):
    results = pd.DataFrame(index=['train_accuracy', 'test_accuracy','train_error', 'test_error', ])
    for estimator in estimators:
        name = str(estimator).split('(')[0]
        results[name] = train(df, estimator)
    return results.transpose()
```

Obr. 25. Trénovanie modelov

Funkcia train, očakáva ako vstupné parametre dáta na trénovanie a algoritmus, ktorý na trénovanie modelu použijeme. Do premennej X vkladáme celú vstupnú množinu dát, okrem hodnôt v stĺpci Label a do y vkladáme naopak len hodnoty stĺpca Label.

Funkcia train_test_split, slúži na rozdelenie vstupnej množiny na množinu tréningovú a testovaciu. Parameter test_size hovorí o tom, aká časť dát bude určená na trénovanie a aká na testovanie modelu. V našom prípade je pre testovaciu množinu vyhradených 20% všetkých dát a pre tréningovú množinu zvyšných 80% dát.

Samotné natrénovanie modelu vykonáva metóda fit, ktorá má na vstupe už rozdelenú tréningovú množinu označených dát. Na záver po úspešnom natrénovaní modelu, používame metódu predict, ktorá slúži na otestovanie modelu v predikcii kategórie vstupných dát.

Funkcia train vráti štyri hodnoty v podobe výslednej presnosti a chybovosti modelu testovaného na testovacej aj tréningovej množine dát, ktoré sme získali z funkcie accuracy_score.

Funkcia Model_results, používa vstupné pole algoritmov na natrénovanie modelov, na záver vráti tabuľku obsahujúcu úspešnosť klasifikácie dát jednotlivých modelov.

3.6 Konverzia modelov

Modely sme natrénovali v jazyku Python, pričom samotnú aplikáciu zvukového prehrávača sme vytvorili v jazyku Java. Na to aby sme mohli natrénované modely použiť v aplikácii, potrebujeme ich prekonvertovať.

Na konverziu modelov používame knižnicu Sklearn-porter[28]. Knižnica nám umožňuje preložiť niektoré algoritmy strojového učenia knižnice Scikit-learn z jazyka Python do iných jazykov ako napríklad: C, Java, JavaScript, atď.

```
def trainModel(df, estimator):
    train(df, estimator)
    return estimator
```

Obr. 26. Získanie natrénovaného modelu

```
for estimator in estimators:
    porter = Porter(trainModel(df, estimator), language='java')
    output = porter.export(embed_data=True)

    name = str(estimator).split('(')[0]
    with open(name + '.java', 'w') as out:
        out.write(output)
```

Obr. 27. Konverzia modelov

Konverziu vykonávame tak, že každý natrénovaný model vložíme do Portera, ktorý preloží zdrojový kód modelu do zvoleného jazyka, v našom prípade jazyka Java. Následne si získame výstup, exportovaním tohto modelu, ktorý si uložíme do premennej output.

Pre uloženie výstupu do počítača, si vytvoríme súbor, ktorého názov prispôsobíme názvu metódy strojového učenia, ktorou bol model natrénovaný. Výstup zapíšeme do súboru, vo forme textu zdrojového kódu daného modelu v jazyku Java.

Súbor obsahujúci zdrojový kód vložíme do projektu zvukového prehrávača v Android Studiu, kde následne využívame metódu predikt, ktorú daný model obsahuje. Táto metóda nám na surových dátach piatich hladín mozgových vĺn, ktoré chceme pomocou modelu klasifikovať, vráti celočíselnú hodnotu kategórie, do ktorej by ich model zaradil.

Týmto máme natrénované modely v jazyku Java, pripravené na použitie pri algoritmoch Next a Previous, ktoré sme si popísali v kapitole 2.1.2.1.

4 Výsledky

Po zozbieraní celkom vyše 5000 riadkov dát zapísaných v súbore Train.csv a následnom natrénovaní modelov, pomocou strojového učenia, na týchto dátach, sa nám podarilo dosiahnuť nasledujúcu presnosť klasifikácie dát:

	train_accuracy	test_accuracy	train_error	test_error
SVC	86.990708	85.809524	0.130093	0.141905
KNeighborsClassifier	86.752442	84.857143	0.132476	0.151429
RandomForestClassifier	99.857041	84.761905	0.001430	0.152381
ExtraTreesClassifier	100.000000	85.523810	0.000000	0.144762
MLPClassifier	85.299023	85.047619	0.147010	0.149524

Tab. 2. Výsledky modelov pôvodných dát

Výsledná presnosť klasifikácie dát do kategórií sa podľa tabuľky 2. pohybuje okolo hranice 85%. Pokus o zlepšenie presnosti bolo použitie transformovaných dát získaných pomocou metódy PCA, na natrénovanie modelov.

Presnosť modelov natrénovaných na dátach z PCA:

	train_accuracy	test_accuracy	train_error	test_error
SVC	73.600191	73.809524	0.263998	0.261905
KNeighborsClassifier	77.507744	70.476190	0.224923	0.295238
RandomForestClassifier	99.690255	69.809524	0.003097	0.301905
ExtraTreesClassifier	100.000000	68.761905	0.000000	0.312381
MLPClassifier	73.719323	73.904762	0.262807	0.260952

Tab. 3. Výsledky modelov dát z PCA

Žiaľ presnosť klasifikácie sa dátami z PCA nepodarilo zvýšiť, naopak, ako vidíme podľa tabuľky 3., presnosť klesla zhruba o 9%.

Posledným pokusom o zlepšenie presnosti bolo použitie podobnej metódy ako je PCA, konkrétne metódy: Lineárne diskriminačnej analýzy (LDA). LDA na rozdiel od PCA predstavuje metódu, ktorá pri transformácií dát berie ohľad aj na jednotlivé kategórie dát, ktorých dátové body sa snaží od seba čo najlepšie oddeliť.[29]

Presnosť modelov natrénovaných na dátach z LDA:

	train_accuracy	test_accuracy	train_error	test_error
SVC	83.678818	84.285714	0.163212	0.157143
KNeighborsClassifier	85.084584	82.952381	0.149154	0.170476
RandomForestClassifier	99.690255	82.761905	0.003097	0.172381
ExtraTreesClassifier	100.000000	82.095238	0.000000	0.179048
MLPClassifier	83.583512	84.571429	0.164165	0.154286

Tab. 4. Výsledky modelov dát z LDA

Ani dátami z LDA sa nám žiaľ presnosť klasifikácie nepodarilo zvýšiť a preto sme v aplikácií použili modely natrénované na pôvodných dátach.

Pomocou knižnice Sklearn-porter sa nám nakoniec podarilo prekonvertovať kód 4 z 5 modelov, ktoré sme následne použili v aplikácii zvukového prehrávača. Konverziou jazyka modelov vznikli rozsiahle súbory, ktorých použitie v projekte Android Studia, bolo problematické.

Výsledkom práce je aplikácia zvukového prehrávača pre platformu Android, ktorá v sebe zahŕňa algoritmy z SDK MindWave Mobile, plus dva ďalšie algoritmy (Next a Previous) reagujúce na dva rôzne stavy mozgovej aktivity, vďaka ktorým sme rozšírili ovládanie zvukového prehrávača, pomocou mysle, použitím zariadenia NeuroSky MindWave Mobile.

Na záver sme algoritmy Next a Previous, popísané v kapitole 2.1.2.1, otestovali.
V stručnosti, algoritmy spúšťajú nasledujúcu (Next), alebo predošlú (Previous) zvukovú stopu zo zoznamu stôp v telefóne.

Test algoritmov v aplikácií prebiehal nasledovne:

Dva subjekty si nasadili na hlavu zariadenie MindWave Mobile, pripojené ku telefónu.
Desať krát za sebou, s drobnými prestávkami medzi pokusmi, sa pokúsili vyvolať mozgovú aktivitu, odpovedajúcu dátam typu Pattern (kapitola 3.3.1), ktoré sú spojené s algoritmom Next. Pokiaľ modely strojového učenia správne rozpoznali túto mozgovú aktivitu a algoritmus Next, poslal odozvu na spustenie nasledujúcej zvukovej stopy, daný pokus sa vyhodnotil ako úspešný. Inak sa pokus vyhodnotil ako neúspešný.

Na každý z desiatich pokusov vyvolania akcie daného algoritmu mal subjekt čas 30 sekúnd.

Následne mali desať pokusov vyvolať mozgovú aktivitu, odpovedajúcu dátam typu Imagine (kapitola 3.3.2), spojenými s algoritmom Previous. Opäť na každý pokus bol limit 30 sekúnd.

Pokiaľ sa počas 30 sekúnd merania stalo, že istý typ mozgovej aktivity, ktorý sme sa pokúšali vyvolať, vyhodnotili natrénované modely zle a spustil sa iný proces ako sa mal (spustenie predošlej zvukovej stopy, miesto nasledujúcej a naopak), zaznamenali sme to ako chybne vyhodnotenú mozgovú aktivitu, alebo chybne klasifikované vstupné dáta z MindWave Mobile.

	Úspešnosť algoritmu Next	Úspešnosť algoritmu Previous	Chybne vyhodnotená mozgová aktivita
Subjekt č.1	100%	80%	10%
Subjekt č.2	100%	70%	0%

Tab. 5. Test algoritmov

V tabuľke 5. už vidíme výsledky testovania subjektov.
Výsledky Subjektu č.1 predstavujú moje výsledky testovania. Ja aj subjekt č. 2 sme dosiahli pri teste algoritmu Next, pozitívny výsledok pri 10 z 10 pokusov.

Pri teste rozpoznávania mozgovej aktivity, spojenej s algoritmom Previous sa mne podarilo získať, v časovom limite 30 sekúnd, pozitívny výsledok pri 8 z 10 pokusov. Subjektu č. 2 sa podarilo získať pozitívny výsledok pri 7 z 10 pokusov.

Pri jednom zo svojich pokusov vyvolať typ mozgovej aktivity spojený s algoritmom Previous, modely zle vyhodnotili túto aktivitu a spustili opačný proces, ktorý je spojený s algoritmom Next a iným, typom dát. Subjektu č. 2 sa nič podobné počas merania nestalo.

Záver

Cieľom práce bolo vytvoriť audio prehrávač, ktorý bude možné ovládať pomocou mysle s využitím zariadenia MindWave Mobile. Myslím si, že cieľ práce sa nám splniť podarilo. Na ovládanie audio prehrávača, sme využili všetky dostupné algoritmy zariadenia, ktoré sme rozšírili o dva vlastné algoritmy. Vďaka týmto algoritmom je teda skutočne možné hlavnú časť zvukového prehrávača ovládať pomocou MindWave Mobile.

Výsledky testu ukazujú, že aj napriek tomu, že surové dáta mozgových vĺn, použitých na trénovanie modelov strojového učenia, boli zbierané výhradne mnou, tieto modely boli schopné pomerne dobre rozpoznávať surové dáta mozgových vĺn, aj ďalšieho subjektu, ktorý o fungovaní aplikácie predom nič nevedel. Výsledná presnosť klasifikácie surových dát sa tak skutočne pohybovala okolo hodnoty zhruba 85%. Treba však poznamenať, že na každý pokus vyvolať určitý typ mozgovej aktivity bol v teste určený časový limit 30 sekúnd. Niekedy sa podarilo rozpoznať danú mozgovú aktivitu už do pár sekúnd, niekedy až koncom časového limitu. Ovládanie rukou, preto zatiaľ stále predstavuje oveľa rýchlejší spôsob ovládania aplikácie.
Ovládanie mysľou, je však veľmi zaujímavou alternatívou bežného ovládania rukou a dúfam, že aj táto práca prispeje k posunu vpred práve týmto smerom a možno pomôže v budúcom vývoji ovládania zariadení vôkol nás.

Za hlavný prínos práce považujem posun vpred v oblasti spracovania surových dát mozgových vĺn a ich využitia na trénovanie modelov strojového učenia, schopných rozpoznávať určité stavy mozgovej aktivity, a následného prepojenia modelov s ovládaním aplikácie. Osobným prínosom práce je okrem iného lepšie zorientovanie sa v problematike fungovania ľudského mozgu, aj v lepšom zorientovaní sa v niektorých podoblastiach umelej inteligencie.

Aj napriek rýchlemu tempu vývoja technológií dnes ešte mobilné telefóny nie sú dostatočne výkonné na trénovanie niektorých modelov strojového učenia a preto sme modely trénovali na počítači. Natrénované modely potom vkladáme do aplikácie, čo nie je najefektívnejší spôsob aktualizácie modelov. V budúcom smerovaní práce by bolo vhodné lepšie riešenie aktualizácie modelov. Príkladom by mohlo byť cloudové riešenie, kam by si používatelia cez internet posielali svoje zozbierané dáta mozgových vĺn, či pred trénovaný model na veľkom počte dát, ktorý by si používateľ iba doladil na svoju množinu dát, podľa vlastných potrieb.

Stavy mozgovej aktivity, ktoré sme sa pri zbere dát snažili zachytiť, zrejme nie sú jediné, ktoré by podobné zariadenia dokázali rozlišovať. Zariadenie MindWave Mobile však má svoje limity. Použitím podobného zariadenia s väčším počtom, presnejších elektród, by zrejme tiež pomohlo rozšíriť možnosti ovládania aplikácie pomocou mysle, či zlepšiť presnosť modelov. Verím, že táto práca čitateľa inšpiruje k zlepšeniu spomínaných nedostatkov.

Použitá literatúra

1 ZAJÍČEK, Dominik. *Introduction to brain-computer interface*. Bakalárska práca. Bratislava : Fakulta Matematiky, Fyziky a Informatiky. 2012.

2 KOKOŠKA, Martin. *Využitie rozhrania mozog-počítač na vykonávanie akcií*. Diplomová práca. Bratislava : Fakulta Matematiky, Fyziky a Informatiky. 2013.

3 NeuroSky. c2015. Dostupné on-line: *https://store.neurosky.com/*. Citované: 6.2.2019.

4 LARSEN, Erik, Andreas. *Classification of EEG Signals in a Brain-Computer Interface System*. Diplomová práca. Norwegian University of Science and Technology. 2011. s. 7.

5 NeuroSky. c2015. Dostupné on-line: *https://store.neurosky.com/products/android-developer-tools-4*. Citované: 6.2.2019.

6 NeuroSky. c2015. Dostupné on-line: *https://store.neurosky.com/collections/apps*. Citované: 7.2.2019.

7 NeuroSky. 19.2.2016. *MDT for Android Instruction*. s. 5. Citované: 6.2.2019.

8 NeuroSky. 22.2.2016. *Eeg algorithm Descriptions*. s. 5. Citované: 6.2.2019.

9 CIBULA, Miroslav. *Support Vector Machines*. 2017. Dostupné on-line: *http://smnd.sk/mcibula/alg/SVM.html*. Citované: 19.5.2019.

10 KANDAN, Harish. *Understanding the kernel trick*. Dostupné on-line: *https://towardsdatascience.com/understanding-the-kernel-trick-e0bc6112ef78*. Citované: 20.5.2019.

11 NAVLANI, Avinash. *Understanding Random Forest Classifiers in Python*. Dostupné on-line: *https://www.datacamp.com/community/tutorials/random-forests-classifier-python#algorithm*. Citované: 20.5.2019.

12 POTANČOK, Ivan. *Dolovanie v dátach na webe - učenie sa s učiteľom*. Bratislava : Fakulta informatiky a informačných technológií. Dostupné on-line: *http://www2.fiit.stuba.sk/~kapustik/ZS/Clanky0910/potancok/index.html#Sekcia4*. Citované: 20.5.2019.

13 GÁBIK, Jozef. *Kreditný skóring pomocou náhodných lesov*. Diplomová práca. Bratislava: Fakulta Matematiky, Fyziky a Informatiky. 2010. s. 14.

14 GEURTS, Pierre, ERNST, Damien, WEHENKEL, Louis. *Extremely randomized trees*. 2006.

15 FEDÁKOVÁ, Dominika. *Aplikované využitie neurónových sietí*. Bakalárska práca. Bratislava : Fakulta Matematiky, Fyziky a Informatiky. 2011. s. 3-6.

16 Android Studio. *Meet Android Studio.* Dostupné on-line: *https://developer.android.com/studio/intro/.* Citované: 7.2.2019.

17 MediaPlayer. Dostupné on-line: *https://developer.android.com/reference/android/media/MediaPlayer*. Citované: 8.5.2019.

18 AudioManager. Dostupné on-line: *https://developer.android.com/reference/android/media/AudioManager*. Citované: 8.5.2019.

19 BluetoothManager. Dostupné on-line: *https://developer.android.com/reference/android/bluetooth/BluetoothManager*. Citované: 8.5.2019.

20 NeuroSky. 2.2.2016. *EEG algorithm SDK for Android: Development Guide*. s. 8-14. Citované: 13.2.2019.

21 Jupyter Notebook. Dostupné on-line: *https://jupyter.org/*. c2019. Citované: 16.5.2019.

22 NumPy. Dostupné on-line: *https://www.numpy.org/*. c2019. Citované: 16.5.2019.

23 Pandas. Dostupné on-line: *https://pandas.pydata.org/*. Citované: 16.5.2019.

24 Matplotlib. Dostupné on-line: *https://matplotlib.org/*. c2002 Citované: 16.5.2019.

25 Seaborn. Dostupné on-line: *https://seaborn.pydata.org/*. c2012. Citované: 16.5.2019.

26 DUBOVÁ, Mária. *Metoda hlavních komponent a její aplikace.* Bakalárska práca. Praha : Matematicko-fyzikální fakulta. 2011.

27 Scikit-learn. Dostupné on-line: *https://scikit-learn.org/stable/*. Citované: 19.5.2019.

28 Sklearn-porter. Dostupné on-line: *https://github.com/nok/sklearn-porter*. Citované: 24.5.2019

29 MARTÍNEZ, M. Aleix, KAK, C. Avinash. *PCA versus LDA.* School of Electrical and Computer Engineering. 2001. IN 47907-1285. Dostupné on-line: *https://cse.buffalo.edu/~jcorso/t/555pdf/pca-versus-lda.pdf*. Citované: 26.5.2019

Zoznam obrázkov

Zoznam tabuliek