POLITECHNIKA WROCŁAWSKA
WYDZIAŁ PODSTAWOWYCH PROBLEMÓW TECHNIKI

KIERUNEK: INŻYNIERIA BIOMEDYCZNA
SPECJALNOŚĆ: INFORMATYKA MEDYCZNA

PRACA DYPLOMOWA INŻYNIERSKA

Kategoryzacja słuchowych odpowiedzi wywołanych mózgu za pomocą uczenia maszynowego

Categorization of auditory evoked potentials using machine learning

AUTOR:
Tomasz Hawro

PROWADZĄCY PRACĘ:
dr hab. inż. Cezary Sielużycki

WROCŁAW, 2020

Spis treści

Wprowadzenie

Uczenie maszynowe wielokrotnie było stosowane przez liczne zespoły badawcze w celu klasyfikacji pozyskanych danych. W niniejszej pracy przedstawiono próbę sklasyfikowania słuchowych odpowiedzi wywołanych mózgu pod kątem słuchania aktywnego *vs* pasywnego. Sygnały, które analizowano, pochodzą z badania 148-kanałowym magnetoencefalografem. Klasa *aktywne* odnosi się do słuchania, w którym osoba badana miała za zadanie zasygnalizować (klikając przycisk) skategoryzowanie bodźca dźwiękowego, którym był liniowo modulowany sygnał świergotowy, z uwagi na kierunek modulacji częstotliwościowej (w górę *vs* w dół) [1, 2, 3]. Klasa *pasywne* odnosi się do słuchania, w którym osoba badana nie miała w żaden sposób skupiać się na słyszanych dźwiękach.

W niniejszej pracy klasyfikacja[1] omawianych sygnałów magnetoencefalograficznych (MEG) odbywa się za pomocą uczenia maszynowego, a dokładniej modeli takich, jak:

- regresja logistyczna,
- drzewo decyzyjne,
- las losowy,
- k-najbliższych sąsiadów,
- naiwny klasyfikator bayesowski,
- maszyna wektorów nośnych.

Każdą z tych metod sprawdzono w następujących wariantach klasyfikacyjnych:

- klasyfikacja rodzaju słuchania (aktywne *vs* pasywne) dla lewej półkuli mózgu,
- analogiczna klasyfikacja rodzaju słuchania (aktywne *vs* pasywne) dla prawej półkuli,
- klasyfikacja źródła sygnału (lewa *vs* prawa półkula mózgu) dla słuchania aktywnego,
- analogiczna klasyfikacja dla słuchania pasywnego.

Celem niniejszej pracy jest próba odpowiedzi na następujące pytania:

- Jakie cechy badanego sygnału MEG mają wpływ na dokładność klasyfikacji[2] przy użyciu wymienionych modeli uczenia maszynowego?
- Która półkula mózgu zaangażowana jest w kategoryzację sygnałów świergotowych z uwagi na kierunek ich modulacji częstotliwościowej?

[1] Terminy *klasyfikacja* oraz *kategoryzacja* używane są w tej pracy zamiennie.

[2] Zamiast frazy *dokładność klasyfikacji* dla uproszczenia często używa się terminu *dokładność*.

Rozdział 1

Wstęp teoretyczny

1.1. Magnetoencefalografia

1.1.1. Czym jest magnetoencefalografia?

Badanie magnetoencefalograficzne (MEG) polega na pomiarze pola magnetycznego wytwarzanego przez neuronalną aktywność elektryczną mózgu [4]. Pole magnetyczne generowane przez neurony w mózgu jest bardzo słabe. Jego wartość rzadko kiedy przekracza rząd setek femtotesli (fT). Dla porównania pole magnetyczne ziemi zawiera się pomiędzy 10^{-4} a 10^{-5} T. Urządzenie służące do rejestracji pola magnetycznego mózgu to magnetoencefalograf. Znaczną przewagą MEG nad elektroencefalografią (EEG) [5] jest fakt, iż pole magnetyczne przechodzi przez tkanki ludzkie praktycznie bez żadnych zniekształceń (przepuszczalność magnetyczna tkanek biologicznych jest mocno zbliżona do przepuszczalności magnetycznej próżni), co zapewnia wysoką jakość badania. Spośród wielu metod obrazowania aktywności mózgu to właśnie MEG zapewnia wysoką rozdzielczość zarówno przestrzenną, jak i czasową. Dodatkowy atut stanowi nieinwazyjność tego badania [6].

1.1.2. Generacja pola magnetycznego w mózgu

Wypadkowy sygnał MEG stanowi przede wszystkim uśrednienie aktywności wszystkich neuronów w pobliżu danego kanału. Podczas reakcji na konkretny bodziec bierze udział wiele neuronów. Jednakże nie wszystkie neurony w danym obszarze skupiają się na bodźcu, który podajemy badanemu obiektowi. Może to powodować pozyskanie przez nas niepożądanych informacji, które w kontekście analizy określonej aktywności mózgu traktuje się jako szum. Pojedyncze neurony generują sygnał o zbyt niskim stosunku sygnału do szumu (ang. signal-to-noise ratio, SNR), w przeciwieństwie do zespołów neuronów, w przypadku których SNR jest znacznie większy. Wyróżniamy wiele typów neuronów, m.in. neurony piramidalne i neurony hamujące. Neurony piramidalne występują głównie w korze mózgowej i ich aktywność wpływa na rejestrowany sygnał MEG. Neurony hamujące posiadają krótsze dendryty i symetryczną strukturę dendrytyczną, przez co tworzą zamknięte pole magnetyczne, które nie ma znacznego wpływu na badanie MEG. Przemieszczanie jonów Ca^{2+} oraz Na^{+} w neuronach skutkuje powstaniem prądu elektrycznego. Prąd ten jest zawsze związany z polem magnetycznym, które jest prostopadłe

do kierunku jego przepływu (reguła prawej ręki). Ruch jonów zatem powoduje powstanie pola magnetycznego, które następnie mierzone jest w badaniu za pomocą magnetoencefalografu [7].

1.1.3. Utrudnienia związane z pomiarem pola magnetycznego

Pomiar pola magnetycznego związany jest z licznymi utrudnieniami związanymi głównie ze słabą siłą tego pola (nie przekracza setek femtotesli). Pole magnetyczne ziemi jest znacznie silniejsze od pola magnetycznego generowanego przez mózg, przez co koniecznym jest wykonywanie badania MEG w specjalnie do tego przystosowanych pomieszczeniach. Pomieszczenia te są ekranowane, dzięki czemu pole magnetyczne z zewnątrz nie wpływa (albo wpływa minimalnie) na badanie, które odbywa się w środku. Kolejnym utrudnieniem jest mała liczba urządzeń, które byłyby w stanie wykonywać pomiar tak słabego pola magnetycznego.

1.1.4. Przebieg badania MEG

Podczas badania pól magnetycznych mózgu osoba badana musi przez cały czas siedzieć nieruchomo, a jej głowa umieszczona jest w hełmie magnetoencefalografu. Badanie to niesie za sobą wiele komplikacji. Konieczność unikania ruchu podczas badania znacznie utrudnia pomiary pacjentów w wieku dziecięcym (jakikolwiek ruch powoduje znaczne zniekształcenie otrzymanego sygnału, przez co sygnał pochodzący od mózgu staje się niezauważalny). Rozmiar hełmu magnetoencefalografu także utrudnia badania osób młodszych, których głowa jest naturalnie mniejsza od głowy osoby dorosłej, przez co konieczne jest opracowywanie specjalnych adapterów stabilizujących głowę względem hełmu.

1.2. Słuchowe odpowiedzi wywołane mózgu

1.2.1. Cechy charakterystyczne odpowiedzi wywołanych

Korowe słuchowe odpowiedzi wywołane to sygnały elektryczne/magnetyczne odzwierciedlające aktywność kory słuchowej w odpowiedzi na bodziec dźwiękowy. Przebieg czasowy reprezentujący sygnał będący odpowiedzią mózgu posiada trzy charakterystyczne załamki. Są to załamki P50, N100 (w MEG nazywany także M100 albo N100m) oraz P200. Załamek P50 osiąga swoją wartość szczytową po $\sim$50 ms od wystąpienia bodźca, N100 po $\sim$100 ms, a P200 po $\sim$200 ms.

1.2.2. Odpowiedzi analizowane w niniejszej pracy

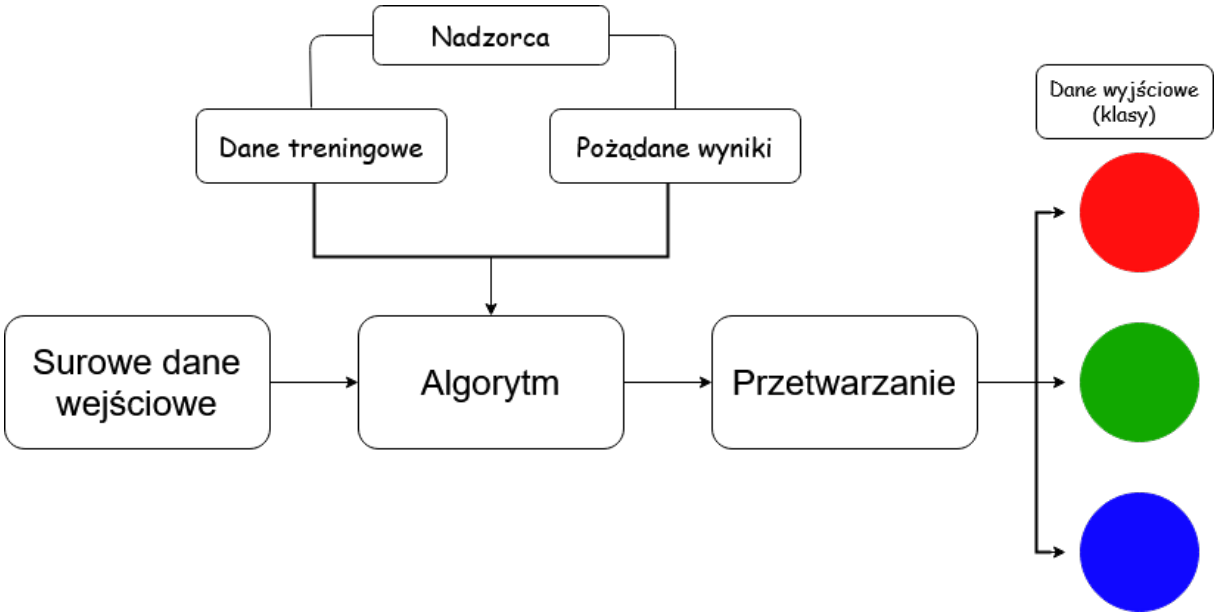
Dane eksperymentalne analizowane w niniejszej pracy dyplomowej pochodzą z badań, w których bodziec słuchowy był sygnałem świergotowym (ang. *chirp*) o częstotliwości modulowanej liniowo. Zbadanie jednej osoby polegało na wielokrotnym powtórzeniu pomiaru dla słuchania aktywnego oraz słuchania pasywnego. W przypadku, gdy najpierw odbyło się słuchanie aktywne, a następnie pasywne, osoba badana mogła nieświadomie podejść do wariantu pasywnego w sposób aktywny. Z tego powodu kolejność rodzajów słuchania nie była taka sama w przypadku każdego badanego. Umożliwiło to częściową eliminację błędu związanego z kolejnością badań.

1.3. Python

Python jest obecnie jednym z najczęściej stosowanych języków programowania w środowisku naukowym. Jest to język, który cały czas się rozwija i zdobywa nowych użytkowników. Dzieje się tak ze względu na jego prostotę, duży obszar zastosowań oraz możliwość kompilacji wydzielonych części kodu. Prostota języka Python polega na braku konieczności precyzowania typów zmiennych, dużo większej niż w innych językach elastyczności kolekcji oraz sposobie formułowania komend, które brzmią jak proste zdania w języku angielskim. Duży obszar zastosowań języka Python jest zapewniony przez duże grono użytkowników, a także biblioteki przez nich tworzone, które są udostępniane innym użytkownikom. Biblioteki te dają takie możliwości, jak np. wczytywanie danych, ich wizualizacja i analiza statystyczna oraz przetwarzanie obrazów. Możliwość kompilacji wydzielonych części kodu to bardzo pomocne narzędzie, które umożliwia przetestowanie każdego fragmentu kodu oddzielnie. Umożliwia to znacznie szybsze znalezienie ewentualnych błędów i naprawienie ich. Narzędzie, które to oferuje, to *JupyterLab*. Wszystkie fragmenty kodu przedstawione w tej sekcji zostały stworzone w JupyterLab. W zastosowaniach naukowych korzystanie z języka Python to głównie używanie niektórych jego bibliotek, a mianowicie *NumPy*, *SciPy*, *scikit-learn*, *Matplotlib*, *pandas* oraz *TensorFlow*. Biblioteka NumPy zawiera wiele funkcjonalności matematycznych. Niektóre z nich to możliwość operacji na wielowymiarowych macierzach, wykorzystanie w obliczeniach algebry liniowej, licznych transformat oraz generatorów liczb pseudolosowych. Biblioteka SciPy również zawiera wiele podstawowych funkcjonalności matematycznych, lecz obok nich są tam także takie elementy, jak możliwość przetwarzania sygnałów oraz optymalizacja funkcji. Biblioteka Matplotlib jest główną biblioteką służącą do rysowania wykresów. Połączenie biblioteki Matplotlib z funkcjonalnościami, jakie daje JupyterLab, umożliwia wizualizację danych w czasie rzeczywistym w oknie przeglądarki. Biblioteka pandas umożliwia manipulowanie danymi w formacie przypominającym dane tabelaryczne z programu Excel. W szczególności umożliwia tworzenie zapytań na wzór języka SQL oraz łączenie tabel. Inne biblioteki również oferują podobne funkcjonalności, lecz tym, co wyróżnia bibliotekę pandas, jest fakt, iż umożliwia ona stosowanie różnych typów danych w jednej tabeli. Kolejnym atutem stosowania tej biblioteki jest to, iż akceptuje ona dane w różnych formatach (np. SQL, Excel, CSV) [8]. Biblioteka scikit-learn zapewnia dostęp do licznych modeli uczenia maszynowego oraz umożliwia korzystanie z funkcji służących do wstępnego przetwarzania danych (np. standaryzacji). Biblioteki TensorFlow często używa się w zastosowaniach związanych z uczeniem głębokim. Jest to związane z faktem, iż umożliwia ona znacznie lepszą kontrolę obliczeń, np. poprzez realizację operacji równoległych przy użyciu karty graficznej komputera. Biblioteka ta zwiększa również (względem innych zastosowań) wydajność samego procesu uczenia [9]. Python jest jednym z najbardziej rozpowszechnionych języków programowania w środowisku zajmującym się analizą dużych ilości danych. Pomimo faktu, iż wydajność tego języka w porównaniu z językami niskopoziomowymi jest niska, istnieją biblioteki takie jak NumPy i SciPy, które zapewniają poprawę tej wydajności. Biblioteki te zostały stworzone korzystając z języka Fortran oraz C, dzięki czemu możliwe jest wykonywanie szybkich operacji wektorowych na wielowymiarowych macierzach.

1.4. Uczenie maszynowe

Metody uczenia maszynowego, z których korzystano w tej pracy, należą do grupy uczenia nadzorowanego. Uczenie nadzorowane polega na wprowadzeniu danych wejściowych, tzw. zestawu danych uczących oraz danych wyjściowych, czyli pożądanych przez nadzorcę odpowiedzi [10]. Danymi uczącymi może być np. wektor, a danymi wyjściowymi konkretna wartość liczbowa. Gdy udostępnimy systemowi wystarczającą liczbę par klucz-wartość, to w pewnym momencie zacznie on dostrzegać pewne wzorce, które odpowiadają danemu problemowi. Gdy posiadamy klarowny zbiór danych, które moglibyśmy wprowadzić na wejście systemu, to zaleca się korzystanie z uczenia nadzorowanego. Wykazano, iż za pomocą uczenia nadzorowanego można sklasyfikować dane pochodzenia biomedycznego [11]. Dane te pochodziły z badania kardiotokograficznego (KTG, badanie polegające na monitorowaniu akcji serca płodu wraz z zapisem czynności skurczowej mięśnia macicy). Na podstawie uzyskanych danych stwierdzono, iż skuteczność analizy KTG znacznie wzrosła dzięki wykorzystaniu uczenia maszynowego. Klasyfikacja w tym badaniu polegała na przyporządkowaniu danych do trzech grup: normalnej, podejrzanej oraz patologicznej. Dla każdej z grup wyniki były zadowalające [11]. Schemat uczenia nadzorowanego przedstawia rysunek 1.1.



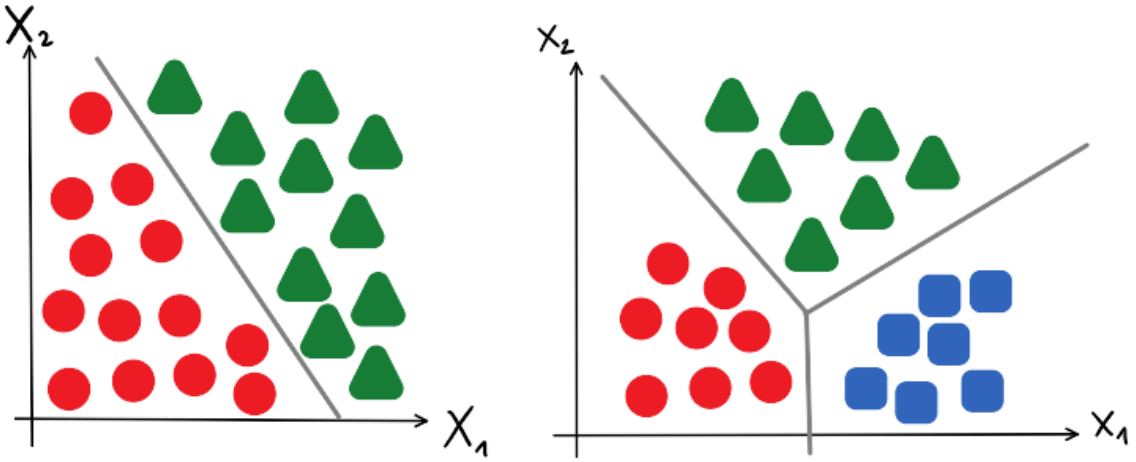
Rys. 1.1: Schemat uczenia nadzorowanego. Rysunek własny.

Jednym z wielu zastosowań uczenia maszynowego jest klasyfikacja danych. Klasyfikacja jest często przypisywana do uczenia nadzorowanego. Jest to proces polegający na przewidywaniu klasy danych wejściowych na podstawie wcześniej wyuczonych wzorców. Typy klasyfikacji można podzielić ze względu liczebność klas oraz równowagę pomiędzy klasami w zestawie danych. Podział ze względu na liczebność klas wyróżnia trzy typy klasyfikacji: binarna, wieloklasowa oraz wieloetykietowa [12]. Podział ze względu na równowagę pomiędzy klasami wyróżnia klasyfikację zrównoważoną oraz niezrównoważoną.

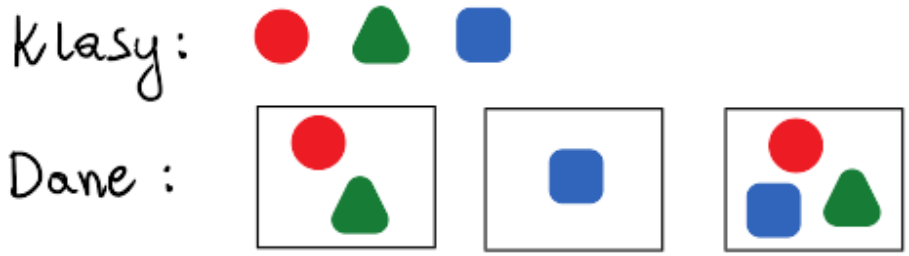
Klasyfikacja binarna jest używana w wypadku, gdy dane potrzeba podzielić na dwie klasy, czyli np. detekcja spamu, czy podział na słuchanie aktywne i pasywne. Zazwyczaj w klasyfikacji binarnej mamy podział na klasę „prawidłową” oraz „nieprawidłową”. Schemat klasyfikacji

binarnej przedstawia rysunek 1.2. Z klasyfikacji wieloklasowej korzysta się, gdy dane należy podzielić na więcej niż dwie klasy, np. klasyfikacja rysów twarzy, gatunków zwierząt, czy znaków drogowych. W tym wypadku sprawdzane jest, czy zbiór wchodzących danych (pojedynczy obiekt) przynależy do jednej z klas. W tego rodzaju klasyfikacji, w celu przydzielenia do klas korzysta się z uogólnionego rozkładu Bernoulliego. Schemat klasyfikacji wieloklasowej przedstawia rysunek 1.2. Klasyfikacja wieloetykietowa jest wykorzystywana w przypadku, gdy dane należy podzielić na więcej niż dwie klasy, ale przyjmuje się możliwość, że do jednego obiektu można przypisać więcej niż jedną klasę. Doskonałym przykładem jest klasyfikacja zdjęć. Na jednym zdjęciu może być wiele przedmiotów i w przypadku tej klasyfikacji skutkuje to przypisaniem wielu klas (każdy rozpoznany przedmiot odpowiada konkretnej klasie) do jednego zdjęcia. Schemat klasyfikacji wieloetykietowej przedstawia rysunek 1.3.

Klasyfikacja zrównoważona dotyczy przypadku, w którym liczebność każdej z klas w zestawie danych jest do siebie zbliżona. Klasyfikacja niezrównoważona występuje w momencie, gdy w zbiorze danych liczebność klas jest rozłożona nierównomiernie.



Rys. 1.2: Schemat klasyfikacji binarnej (lewa) i wieloklasowej (prawa). Rysunek własny.



Rys. 1.3: Schemat klasyfikacji wieloetykietowej. Rysunek własny.

Algorytmy uczenia maszynowego można podzielić na *parametryczne* (gorliwe) oraz *nieparametryczne* (leniwe). Uczenie leniwe cechuje się krótkim czasem uczenia oraz długim czasem klasyfikacji. Z kolei uczenie gorliwe charakteryzuje się krótkim czasem klasyfikacji oraz długim czasem uczenia [13].

Modele parametryczne klasyfikują dane testowe za pomocą funkcji, która powstała na podstawie danych treningowych, lecz dane te przy klasyfikacji potrzebne nie są. Przykładem

modelu parametrycznego jest regresja logistyczna, która klasyfikuje dane korzystając z funkcji sigmoidalnej.

Modele nieparametryczne klasyfikują dane testowe na podstawie całego zbioru danych treningowych. Nie posiadają one funkcji utworzonej na podstawie parametrów wydobytych z danych treningowych, a ponadto zbiór parametrów, które decydują o klasyfikacji, rośnie wraz ze wzrostem zestawu treningowego. Przykładem modeli nieparametrycznych są drzewa decyzyjne oraz lasy losowe [14]. Podczas analizy danych w tej pracy korzystano z metod uczenia nadzorowanego, za pomocą których dokonano binarnej klasyfikacji sygnałów MEG. Modele, które będą podlegać analizie, to regresja logistyczna, drzewa decyzyjne, lasy losowe, k-najbliższych sąsiadów, naiwny klasyfikator Bayesowski oraz maszyna wektorów nośnych.

1.4.1. Regresja logistyczna

Regresja logistyczna (ang. *logistic regression*, LR) jest modelem, który wyjątkowo dobrze radzi sobie w wypadku, gdy klasy są liniowo separowalne. Nadaje się ona zarówno do klasyfikacji binarnej, jak i wieloklasowej (w przypadku wieloklasowej jest to wtedy wielomianowa regresja logistyczna). W regresji logistycznej stosuje się specyficzną wielkość zwaną *szansą* (ang. *odds*). Szansa jest stosunkiem prawdopodobieństwa sukcesu do prawdopodobieństwa porażki [14]:

$$\text{szansa} = \frac{p}{1-p}, \tag{1.1}$$

gdzie p jest prawdopodobieństwem zajścia oczekiwanego zdarzenia. Logarytmując szansę otrzymamy funkcję logit:

$$\text{logit} = \ln \frac{p}{1-p}. \tag{1.2}$$

Istotny jest fakt, iż funkcja logit przyjmuje wartości z zakresu $(0, 1)$, a zwraca wartości z zakresu $(-\infty, +\infty)$, co umożliwia pokazanie zależności pomiędzy wektorem analizowanych parametrów (x) a logarytmem szans:

$$\text{logit}(p(y = Y|x)) = w_0x_0 + w_1x_1 + \ldots + w_nx_n = \sum_{i=0}^{n} w_ix_i = w^{\mathrm{T}}x, \tag{1.3}$$

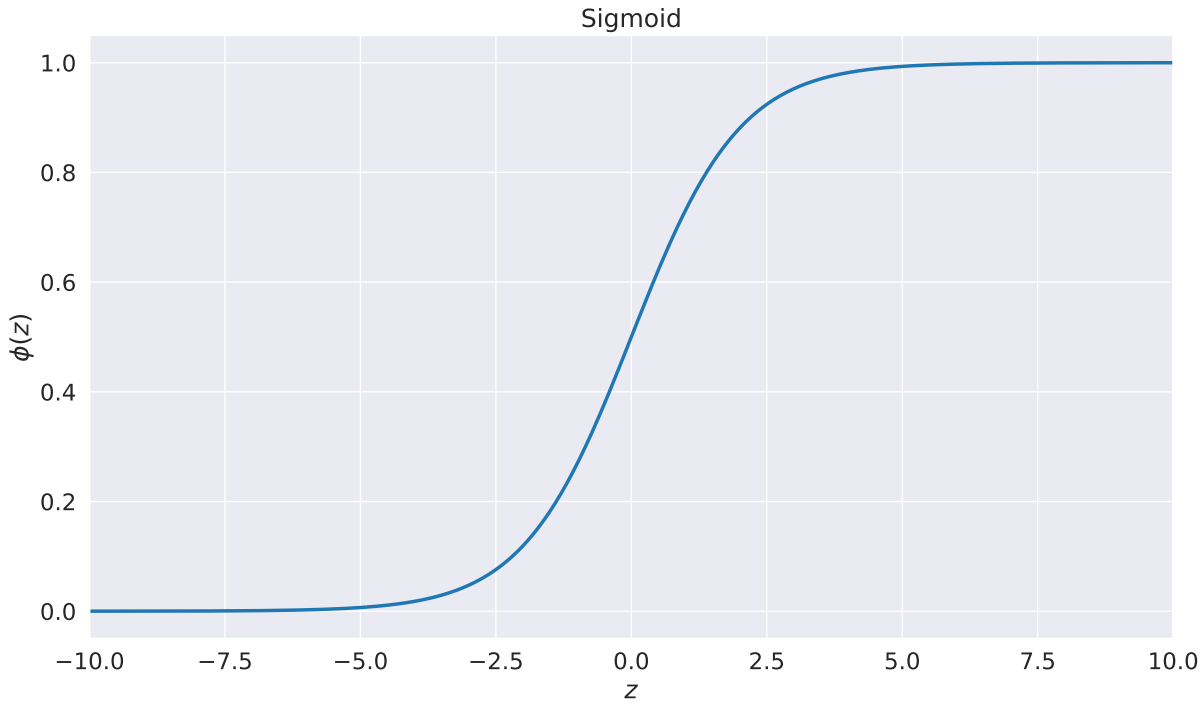
gdzie pierwszy element sumy odnosi się do stronniczości metody, w_0 to obciążenie, a x_0 jest równe 1. Argumentem funkcji logit jest prawdopodobieństwo przynależności zestawu danych do klasy Y, biorąc pod uwagę, że zestaw ten jest przedstawiany jako wektor parametrów x. Symbol w oznacza wagi przypisane do danych parametrów x. Za pomocą funkcji odwrotnej do funkcji logit jesteśmy w stanie wyliczyć prawdopodobieństwo, że dany zbiór parametrów przynależy do którejś z klas. Funkcja, która nam to umożliwi, to funkcja sigmoidalna (1.4):

$$\phi(z) = \frac{1}{1+e^{-z}}, \tag{1.4}$$

gdzie z oznacza *wejście*, czyli iloczyn parametrów oraz wag do nich przypisanych:

$$z \ = w^{\mathrm{T}}x = w_0x_0 + w_1x_1 + \ldots + w_nx_n. \tag{1.5}$$

Funkcję sigmoidalną przedstawia rysunek 1.4.



Rys. 1.4: Funkcja sigmoidalna. Rysunek własny.

Z wykresu bardzo łatwo można wyczytać, że gdy z dąży do ∞, to wartości funkcji dążą do 1, a gdy z dąży do $-\infty$, to wartości funkcji dążą do 0. Istotny jest również fakt, iż funkcja ta jako argumenty przyjmuje wszystkie wartości z przedziału $(-\infty, +\infty)$, a zwraca wartości z przedziału $(0, 1)$. Oznacza to, że na jej wejście możemy zadać jakąkolwiek kombinację sum iloczynów wag oraz parametrów, a na wyjściu otrzymamy prawdopodobieństwo przynależności tej kombinacji do klasy Y. Dużą zaletą regresji logistycznej jest możliwość otrzymania prawdopodobieństwa przynależności do konkretnej klasy, a nie jedynie etykiety tej klasy. Granica przynależności do konkretnej klasy jest wyznaczana na podstawie argumentu, dla którego funkcja sigmoidalna przyjmuje wartość 0,5, jest to zatem argument $z = 0$. Na podstawie tej informacji można stworzyć funkcję, która zwróci klasę zbioru danych:

$$y = \begin{cases} 1 & \text{dla } \phi(z) \geq 0.5 \\ 0 & \text{dla } \phi(z) < 0.5 \end{cases} \tag{1.6}$$

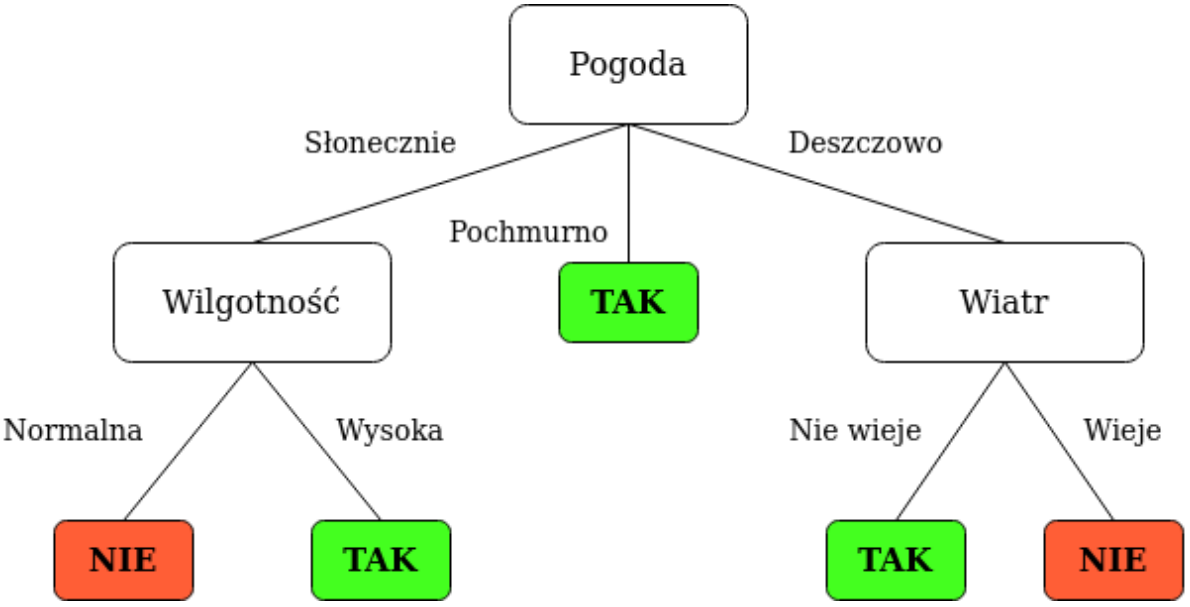
Biorąc pod uwagę postać funkcji sigmoidalnej wyrażenie (1.6) można uprościć:

$$y = \begin{cases} 1 & \text{dla } z \geq 0 \\ 0 & \text{dla } z < 0 \end{cases} \tag{1.7}$$

1.4.2. Drzewo decyzyjne

Drzewo decyzyjne (ang. *decision tree*, DT) jest modelem wyjątkowo prostym w interpretacji oraz nie wymagającym skalowania parametrów. Tworzenie drzewa polega na rozłożeniu danych na czynniki pierwsze za pomocą serii odpowiednich pytań. Przykład drzewa decyzyjnego

przedstawia rysunek 1.5. Dane, które są klasyfikowane za pomocą drzewa decyzyjnego, mogą być zarówno danymi etykietowanymi, jak i numerycznymi. W przypadku parametrów przyjmujących ciągłe wartości numeryczne o dużym zakresie odpowiednim podejściem jest sprawdzanie, czy dany parametr jest większy/mniejszy od zadanego progu. Budowanie drzewa decyzyjnego rozpoczynamy od korzenia, od którego następnie rozdzielamy dane biorąc pod uwagę parametr, który daje największy przyrost informacji (ang. *information gain*, IG). Powtarzamy proces dzielenia danych na kolejne podzbiory, do momentu, gdy wszystkie dane w podzielonym zbiorze nie przynależą do tej samej klasy (zbiór po podziale posiada dane etykietowane tylko jedną klasą). Stosowanie tej strategii może skutkować powstaniem drzew decyzyjnych z wieloma poziomami, co prowadzi do nadmiernego dopasowania modelu. Aby temu zapobiec, stosuje się technikę zwaną *przycinaniem* [14]. Przycinanie polega na likwidacji węzła, którego usunięcie poprawi klasyfikację zbioru testowego. Przycinamy drzewo, dopóki dokładność klasyfikacji na zbiorze testowym się zwiększa.



Rys. 1.5: Przykład drzewa decyzyjnego, które przypisuje klasy oznaczające informację odnośnie decyzji wyjścia z domu. Rysunek własny.

Najważniejsza w tworzeniu drzewa decyzyjnego jest maksymalizacja przyrostu informacji:

$$\arg\max_f \ IG(D_p, f) = \arg\max_f \ I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j), \tag{1.8}$$

gdzie f to parametr, względem którego dokonujemy podziału, D_p i D_j to zbiory danych węzła *rodzicielskiego* oraz węzła *dziecka*, I jest miarą nieczystości informacji, N_p jest całkowitą ilością danych węzła rodzicielskiego, zaś N_j jest liczbą danych węzła dziecka. Na podstawie wzoru (1.8) widać, iż przyrost informacji (IG) jest różnicą pomiędzy nieczystością węzła rodzica oraz sumą nieczystości węzłów dzieci. Prowadzi to do wniosku, iż im mniejsza nieczystość węzłów dzieci, tym większy przyrost informacji. Ze względu na możliwą złożoność niektórych drzew decyzyjnych większość bibliotek korzysta z binarnego podziału węzłów. Po uwzględnieniu tego faktu wzór na przyrost informacji wygląda następująco:

$$IG(D_p, f) = I(D_p) - \frac{N_{\text{left}}}{N_p} I(D_{\text{left}}) - \frac{N_{\text{right}}}{N_p} I(D_{\text{right}}), \tag{1.9}$$

gdzie left oraz right to etykiety, odpowiednio, lewego oraz prawego węzła. Do oceny nieczystości lub kryteriów podziału węzłów binarnych drzew decyzyjnych najczęściej stosuje się indeks Giniego (I_G), entropię (I_H) oraz błąd klasyfikacji (I_E). Błąd klasyfikacji jest najsłabszym kryterium z wymienionych, a entropia oraz indeks Giniego dają bardzo zbliżone rezultaty [14]. Z tego powodu omówione zostanie jedynie kryterium entropii. Entropia danego węzła t przedstawia się następująco:

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) \log_2 p(i|t), \tag{1.10}$$

gdzie $p(i|t)$ jest prawdopodobieństwem, że w danym węźle t dane są przypisane do klasy i. Entropia osiąga wartość minimalną równą zero, gdy wszystkie dane w zbiorze przynależą do tej samej klasy, a wartość maksymalną równą 1, gdy rozkład klas jest równomierny. Przy wyborze najczystszego węzła ważne jest, aby wartość I_H była jak najmniejsza.

1.4.3. Las losowy

Las losowy (ang. *random forest*, RF) jest przykładem łączenia wielu algorytmów uczenia maszynowego w jeden. Jest to tzw. uczenie zespołowe (ang. ensemble learning). W tym wypadku zespół algorytmów składa się z wielu drzew decyzyjnych. Połączenie wielu drzew decyzyjnych w jeden model eliminuje problem wysokiej wariancji pojedynczego drzewa i zazwyczaj zapewnia lepszą kategoryzację danych przy mniejszej podatności na nadmierne dopasowanie. Dodatkowo zazwyczaj nie musimy się martwić *przycinaniem* drzew, ponieważ model zespołowy jest odporny na nieprawidłowości wynikające z pojedynczych drzew. Najważniejszym parametrem jest liczba drzew k, z której będzie się składał las losowy. Wraz ze wzrostem liczby drzew rośnie jakość modelu lasu losowego, lecz wzrasta złożoność obliczeniowa. Algorytm tworzenia lasów losowych może być przedstawiony w czterech krokach [14]:

1. Stworzenie próbki bootstrapowej [15] rozmiaru n (losowo wybierając ze zwracaniem n przykładów z zestawu treningowego). Wybór rozmiaru n pozwala nam kontrolować kompromis między obciążeniem, a wariancją. Zazwyczaj najlepszy kompromis zapewnia rozmiar próbki bootstrapowej, który jest równy rozmiarowi danych treningowych.
2. Stworzenie drzewa decyzyjnego z bootstrapowej próbki. Dla każdego węzła:
   - Losowy wybór d parametrów bez zwracania. Liczba d musi być mniejsza od liczby wszystkich parametrów zbioru danych. Zazwyczaj liczba ta jest zadana równaniem $d = \sqrt{m}$, gdzie m jest liczbą parametrów w zbiorze danych treningowych.
   - Podział węzła używając parametru, który najlepiej spełnia kryterium podziału (maksymalizuje przyrost informacji).
3. Powtórzenie kroków 1–2 k razy.
4. Uwzględniając predykcję każdego drzewa przypisanie klasy danych za pomocą *głosu decydującego*.

1.4.4. K-najbliższych sąsiadów

Model k-najbliższych sąsiadów (ang. *k-nearest neighbours*, KNN) nie tworzy funkcji na podstawie danych treningowych, przez co należy do algorytmów leniwych. KNN jest algorytmem należącym do podkategorii modeli nieparametrycznych, opisywanych jako modele, których uczenie jest oparte na przykładach (ang. *instance-based learning*). Modele te *zapamiętują* dane treningowe i na ich podstawie kategoryzują dane testowe. Algorytm KNN może być przedstawiony w trzech krokach [14]:

1. Wybór wartości k oraz *przestrzeni metrycznej*.
2. Znalezienie k-najbliższych sąsiadów zestawu parametrów, który podlega klasyfikacji.
3. Przypisanie klasy stosując zasadę większości głosów (wybrana zostaje klasa, która przynależała do największej liczby sąsiadów). W przypadku remisu (taka sama liczba sąsiadów dla różnych klas) wiele algorytmów faworyzuje sąsiadów znajdujących się bliżej analizowanego punktu. Jeżeli wciąż jest remis (odległości się równoważą), algorytm wybierze klasę, która jest pierwsza w zbiorze danych treningowych.

Zaletą KNN jest fakt, iż wprowadzenie do modelu nowych danych treningowych nie skutkuje tworzeniem nowych funkcji, więc model ten szybko przystosowuje się do rosnącego zbioru treningowego. Wady tego modelu to rosnąca złożoność obliczeniowa klasyfikacji wraz ze zwiększaniem ilości danych treningowych oraz tzw. *przekleństwo wymiarowości* (ang. *the curse of dimensionality*). Przekleństwo wymiarowości powoduje wykładniczy wzrost niezbędnych danych treningowych towarzyszący rosnącej liczbie parametrów (a zatem zwiększaniu wymiarów analizowanego zbioru). W celu rozwiązania problemu stosuje się techniki *redukowania wymiarowości* (ang. *dimensionality reduction*). Najważniejszym parametrem, który wpływa na jakość klasyfikacji, jest liczba k sąsiadów. Prawidłowy wybór tej wartości zapewnia równowagę między nadmiernym dopasowaniem a niedopasowaniem. Ważnym parametrem jest również przestrzeń metryczna, która powinna być dostosowana do typu danych. Zazwyczaj wystarczająca jest metryka euklidesowa, która najlepiej działa przy ustandaryzowanych danych (dane są standaryzowane, aby ich wkład w dystans był równomierny).

1.4.5. Naiwny klasyfikator bayesowski

Naiwny klasyfikator bayesowski (ang. *naive Bayes classifier*, NB) należy do grupy algorytmów korzystających z uczenia nadzorowanego. Jest modelem, który często stosuje się przy klasyfikacji dokumentów oraz diagnostyce chorób. Model NB opiera się na twierdzeniu Bayesa i zakłada, że parametry w zbiorze są wzajemnie niezależne (stąd *naiwny*). Często założenie to jest nieprawidłowe, lecz pomimo tego klasyfikator jest wysoce efektywny. Założenie niezależności umożliwia efektywną naukę modelu dla wielowymiarowej przestrzeni parametrów nawet przy małej ilości danych. Dzięki temu w przypadku małych zestawów danych naiwny klasyfikator bayesowski może osiągnąć lepsze wyniki niż inne algorytmy [16]. Wzór stanowiący fundament tego modelu wygląda następująco [17]:

$$P(c_j|x_i) = \frac{P(x_i|c_j)P(c_j)}{P(x_i)}, \tag{1.11}$$

gdzie x_i to wektor parametrów próbki danych i, c_j jest etykietą przypisaną klasie o indeksie j, $P(x_i|c_j)$ to prawdopodobieństwo zaobserwowania próbki x_i należącej do klasy c_j, $P(c_j)$ to prawdopodobieństwo napotkania klasy c_j w analizowanym zbiorze danych, a $P(x_i)$ to prawdopodobieństwo wystąpienia danych x_i w zbiorze. Dzięki założeniu o wzajemnej niezależności parametrów wzór wyrażający prawdopodobieństwo zaobserwowania próbki x_i należącej do klasy c_j wygląda następująco:

$$P(x|c_j) = P(x_1|c_j)P(x_2|c_j)\dots P(x_d|c_j) = \prod_{k=1}^{d} P(x_k|c_j), \tag{1.12}$$

gdzie $P(x|c_j)$ oznacza prawdopodobieństwo tego, że dany zbiór parametrów x przynależy do klasy c_j. Prawdopodobieństwa pojedynczych parametrów są wyznaczane na podstawie estymatora największej wiarygodności, który w przypadku danych kategorycznych jest częstotliwością występowania parametru w danej klasie:

$$\hat{P}(x_i|c_j) = \frac{N_{x_i,c_j}}{N_{c_j}}, \quad i = 1, \dots, n, \tag{1.13}$$

gdzie N_{x_i,c_j} oznacza liczbę wystąpień parametru x_i przypisanego do klasy c_j, a N_{c_j} całkowitą liczbę parametrów przypisanych do klasy c_j. $P(c_j)$ z równania (1.11) oznacza prawdopodobieństwo wystąpienia klasy c_j, estymowane jako:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_T}, \tag{1.14}$$

gdzie N_{c_j} oznacza liczbę próbek przypisanych do klasy c_j, a N_T całkowitą liczbę próbek. $P(x_i)$ z równania (1.11), będące prawdopodobieństwem wystąpienia danych x_i w zbiorze, można wyznaczyć z równania:

$$P(x_i) = P(x_i|c_j)P(c_j) + P(x_i|c'_j)P(c'_j), \tag{1.15}$$

gdzie c'_j oznacza *dopełnienie*, czyli przypadek nieprzynależności do klasy c_j. Wyrażenie $P(x_i)$ ma znaczenie, gdy z równania (1.11) chcemy policzyć dokładną wartość $P(c_j|x_i)$, jednakże przy klasyfikacji możemy ten czynnik pominąć. Mamy taką możliwość, gdyż interesuje nas jedynie to, czy $P(c_j|x_i)$ osiąga wartość największą dla klasy c_j. W przypadku klasyfikacji binarnej do zestawu parametrów x_i zostanie przypisana klasa 1, gdy spełniona zostanie nierówność:

$$\frac{P(x_i|c_1)P(c_1)}{P(x_i)} > \frac{P(x_i|c_2)P(c_2)}{P(x_i)} \tag{1.16}$$

Z równania (1.16) wynika, iż wyrażenie $P(x_i)$ w (1.15) służy jedynie jako czynnik skalujący. Zatem nie wnosi on istotnych informacji przy podziale na klasy.

1.4.6. Maszyna wektorów nośnych

Algorytm maszyny wektorów nośnych (ang. *support vector machine*, SVM) jest znacznie bardziej złożony od algorytmów omówionych dotychczas. Model ten opiera się na maksymalizacji marginesu oddzielającego poszczególne klasy. Jego zaletą jest to, iż analizuje on najtrudniejsze punkty przestrzeni (czyli te, które leżą blisko siebie pomimo faktu, iż należą do różnych

klas). Załóżmy, że zbiór danych uczących to (x_i, y_i) dla $i = 1, \ldots, n$, gdzie x_i jest wektorem parametrów, a y_i jest klasą próbki i. Dla uproszczenia omówiona zostanie klasyfikacja binarna, więc $y_i \in \{-1; +1\}$, gdzie $y_i = +1$ oznacza klasę *pozytywną*, a $y_i = -1$ klasę *negatywną*. Gdy zbiór danych jest liniowo separowalny, istnieje hiperpłaszczyzna separująca te klasy:

$$w^{\mathrm{T}} x + b = 0, \tag{1.17}$$

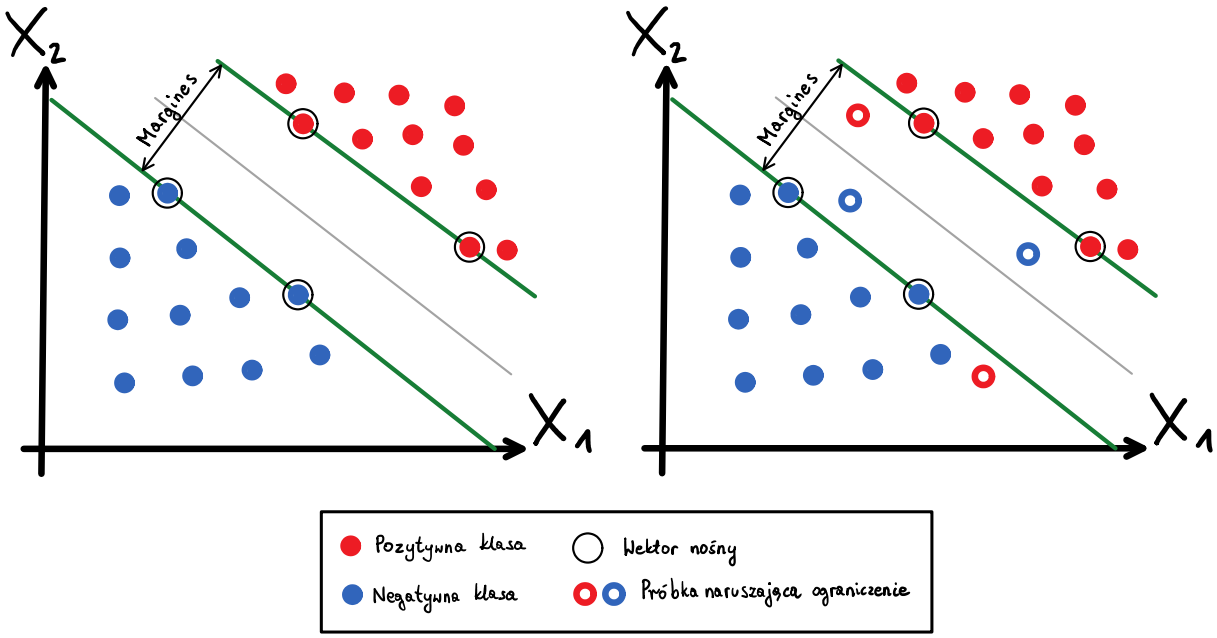
gdzie w oznacza prostopadły do hiperpłaszczyzny wektor wag, x jest wektorem parametrów, a b jest offsetem. Na podstawie równania hiperpłaszczyzny rozdzielającej klasy można sformułować regułę decyzyjną:

$$y_i = \begin{cases} +1, & \text{gdy} \quad w^{\mathrm{T}} x + b \geq 0 \\ -1, & \text{gdy} \quad w^{\mathrm{T}} x + b \leq 0 \end{cases} \tag{1.18}$$

którą można skrócić do prostej nierówności

$$y_i (w^{\mathrm{T}} x + b) \geq 1. \tag{1.19}$$

Nierówność 1.19 wyznacza wektory nośne (ang. *support vectors*), czyli pary punktów (x_i, y_i) określające położenie hiperpłaszczyzny oraz szerokość marginesu separującego klasy. Równania (1.18)–(1.19) dotyczą przypadku *twardego marginesu* (ang. *hard-margin*), czyli sytuacji, gdy dane są liniowo separowalne i istnieje możliwość stworzenia marginesu, który całkowicie rozdzieli dwie klasy. Klasy jednak często nie są liniowo separowalne i należy zastosować tzw. *miękki margines* (ang. *soft-margin*), który uwzględnia sytuację, w której niektóre wektory parametrów leżą po niewłaściwej stronie hiperpłaszczyzny. Margines miękki oraz twardy przedstawiono na rysunku 1.6.



Rys. 1.6: Margines twardy (lewa) oraz margines miękki (prawa). Rysunek własny na podstawie [18].

Nierówność uwzględniająca przypadek marginesu miękkiego wygląda następująco [19]:

$$y_i(w^{\mathrm{T}}x + b) \geq 1 - \delta_i, \tag{1.20}$$

gdzie $\delta_i \geq 0$ służy do zredukowania marginesu, zatem dąży się do jego zminimalizowania. Istnieją trzy możliwe zakresy wartości δ_i:

- $0 \leq \delta_i < 1$ — wtedy (x_i, y_i) jest prawidłowo sklasyfikowany,
- $\delta_i = 1$ — wtedy (x_i, y_i) leży na hiperpłaszczyźnie i klasyfikacja jest nieokreślona
- $1 < \delta_i$ — wtedy (x_i, y_i) jest nieprawidłowo sklasyfikowany.

Hiperpłaszczyzny będące równoległe do marginesu separacji przedstawiają się równaniami [14]:

$$w^{\mathrm{T}}x_+ + b = 1 \tag{1.21a}$$

$$w^{\mathrm{T}}x_- + b = -1 \tag{1.21b}$$

Gdy od równania (1.21a) odejmiemy równanie (1.21b), otrzymamy:

$$w^{\mathrm{T}}(x_+ - x_-) = 2. \tag{1.22}$$

Równanie (1.22) można znormalizować używając długości wektora w:

$$\frac{w^{\mathrm{T}}(x_+ - x_-)}{\|w\|} = (x_+ - x_-)\frac{w^{\mathrm{T}}}{\|w\|} = \frac{2}{\|w\|}. \tag{1.23}$$

Otrzymane wyrażenie (1.23) może być interpretowane jako szerokość marginesu separacji, którą należy zmaksymalizować. Interpretacje powyższych równań (1.20)–(1.23) przedstawia rysunek 1.7.

Głównym zadaniem modelu SVM jest maksymalizacja szerokości marginesu separacji, czyli maksymalizacja wyrażenia $\frac{2}{\|w\|}$. Oznacza to, iż dążymy do minimalizacji $\|w\|$. W praktyce minimalizacja $\|w\|$ polega na minimalizacji wyrażenia [21]

$$\frac{1}{2}\|w\|^2 \quad : \quad y_i(w^{\mathrm{T}}x_i + b) \geq 1, \quad i = 1, \ldots, n \tag{1.24}$$

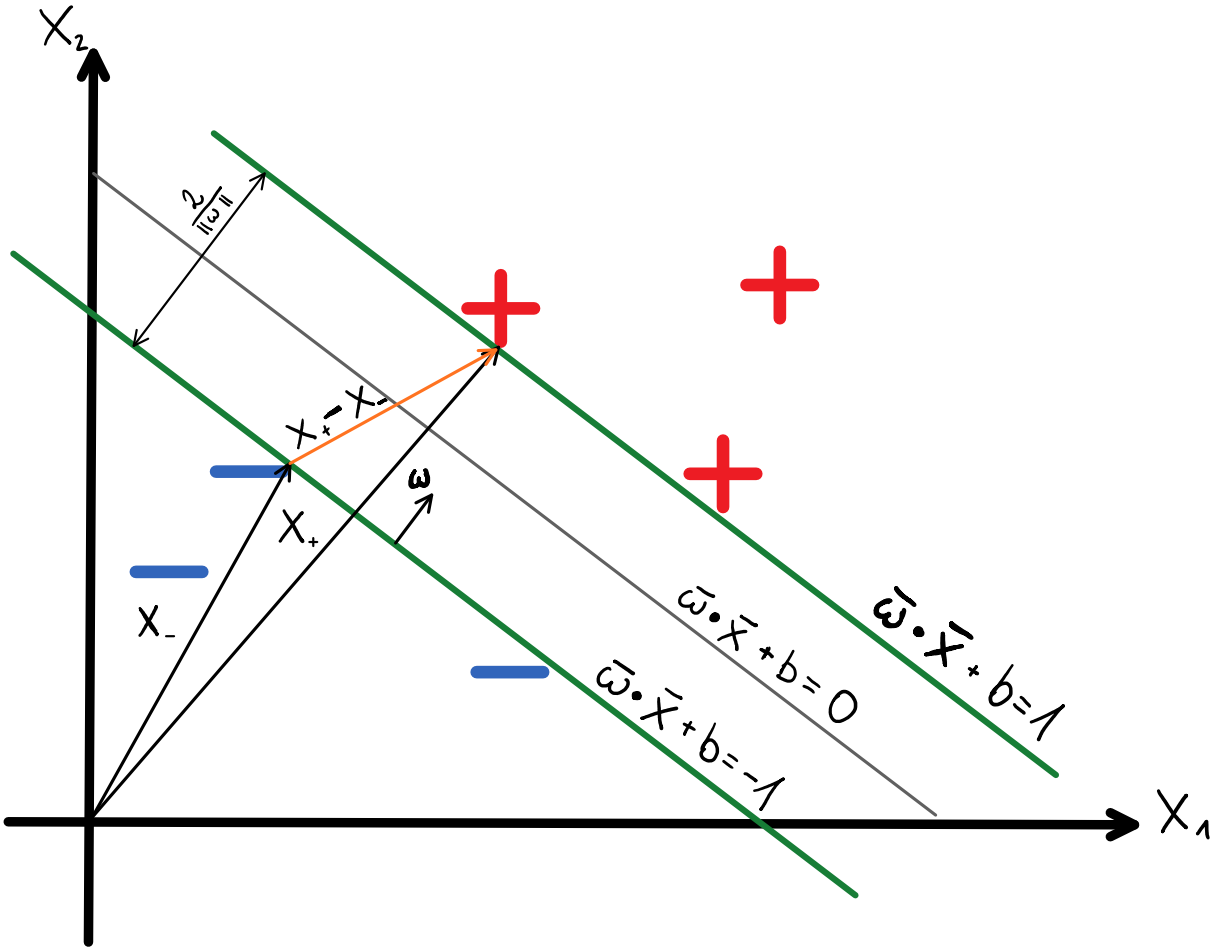
w przypadku marginesu twardego, tudzież wyrażenia

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\delta_i \quad : \quad \delta_i \geq 0, \quad y_i(w^{\mathrm{T}}x_i + b) \geq 1 - \delta_i, \quad i = 1, \ldots, n \tag{1.25}$$

w przypadku marginesu miękkiego (C jest parametrem służącym do karania źle sklasyfikowanych przypadków). Optymalizacji tych dokonuje się za pomocą *programowania kwadratowego*. Z każdym zadaniem programowania kwadratowego można powiązać funkcję Lagrange'a:

$$L(x, \alpha) = f(x) + \sum_{i=1}^{m}\alpha_i g_i(x), \tag{1.26}$$

gdzie α_i to mnożniki Lagrange'a. Funkcja Lagrange'a dla modelu SVM z marginesem miękkim wygląda następująco [22]:



Rys. 1.7: Margines separacji wraz z wektorami niezbędnymi do wyznaczenia szerokości marginesu. Rysunek własny na podstawie [20].

$$L(w,b,\delta,\alpha,\beta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\delta_i - \sum_{i=1}^{m}\alpha_i[y_i(w^T x_i + b) - (1-\delta_i)] - \sum_{i=1}^{n}\beta_i\delta_i. \tag{1.27}$$

Aby znaleźć ekstremum funkcji $L(w,b,\delta,\alpha,\beta)$, należy uwzględnić warunki Karusha–Kuhna–Tuckera (KKT):

$$\begin{aligned} \tfrac{\partial L}{\partial w} &= 0 \\ \tfrac{\partial L}{\partial b} &= 0 \\ \tfrac{\partial L}{\partial \delta} &= 0 \end{aligned} \tag{1.28}$$

Uwzględniając warunki (1.28) otrzymujemy:

$$\begin{aligned} \tfrac{\partial L}{\partial w} &= w - \textstyle\sum_{i=1}^{m}\alpha_i y_i x_i \quad \Rightarrow \quad w = \textstyle\sum_{i=1}^{m}\alpha_i y_i x_i \\ \tfrac{\partial L}{\partial b} &= -\textstyle\sum_{i=1}^{m}\alpha_i y_i \quad \Rightarrow \quad \textstyle\sum_{i=1}^{m}\alpha_i y_i = 0 \\ \tfrac{\partial L}{\partial \delta_i} &= C - \alpha_i - \beta_i \quad \Rightarrow \quad 0 \geq \alpha_i \geq C \end{aligned} \tag{1.29}$$

Podstawiając wynikające z warunków KKT własności (1.29) do funkcji Lagrange'a otrzymujemy funkcję postaci [20]:

$$\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i}^{n}\sum_{i}^{n} y_i y_j \alpha_i \alpha_j (x_i^{\mathrm{T}} x_j), \tag{1.30}$$

którą należy zmaksymalizować względem α przy zachowaniu następujących warunków:

$$\begin{aligned} &\textstyle\sum_{i=1}^{m} \alpha_i y_i = 0 \\ &0 \geq \alpha_i \geq C \end{aligned} \tag{1.31}$$

Na podstawie równań (1.31) jesteśmy w stanie wyznaczyć wartości *mnożników Lagrange'a*, α_i. Znając α_i jesteśmy w stanie wyznaczyć wektor wag w:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i. \tag{1.32}$$

Znając wektor wag $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ możemy ponownie zapisać równanie hiperpłaszczyzny:

$$w^{\mathrm{T}} x + b = \sum_{i=1}^{n} \alpha_i y_i x_i^{\mathrm{T}} x + b = 0. \tag{1.33}$$

Za pomocą równania (1.33), na podstawie zbioru uczącego (x_i, y_i), gdzie $i = 1, \ldots, n$, dokonywana będzie klasyfikacja nieznanego zestawu parametrów x. Podczas wyznaczania równania płaszczyzny należy pamiętać, iż większość mnożników Lagrange'a przyjmie wartość zero, a jedynie elementy sumy odnoszące się do wektorów nośnych przyjmą niezerowe wartości α_i. Powodem, dla którego jedynie wektory nośne będą miały wpływ na margines separacji, jest fakt, iż minimalizacja względem w jest równoważna maksymalizacji względem α [23]. Podstawiając powyższe elementy do równania na funkcję Lagrange'a otrzymujemy:

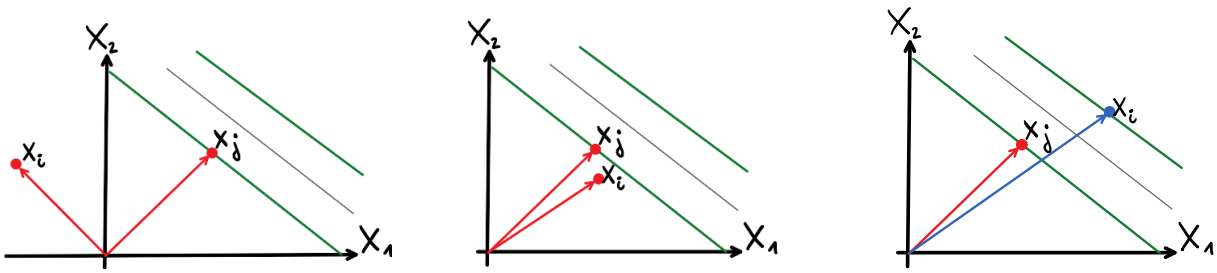
$$L(w, b, \alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^{\mathrm{T}} x_j. \tag{1.34}$$

Maksymalizacji funkcji (1.34) polega na przypisywaniu do α_i wartości niezerowych odnoszących się do wektorów nośnych (x_i, y_i) oraz (x_j, y_j), a więc tych, które odgrywają największą rolę przy poszukiwaniu największego marginesu separacji. Możliwe są trzy przypadki:

1. Wektory parametrów x_i oraz x_j są zupełnie niepodobne, więc ich iloczyn skalarny jest równy 0, co skutkuje brakiem wpływu na wartość funkcji Lagrange'a. α_i w tym wypadku ma wartość 0.
2. Wektory parametrów x_i oraz x_j są podobne, a ich klasy y_i oraz y_j są takie same, więc iloczyny $y_i y_j$ oraz $x_i^{\mathrm{T}} x_j$ zwracają wartości pozytywne, przez co iloczyn tych dwóch iloczynów również jest pozytywny. Oznacza to, iż obniżają one wartość funkcji Lagrange'a, co nie jest pożądane. α_i w tym wypadku ma wartość 0.
3. Wektory parametrów x_i oraz x_j są podobne, lecz ich klasy y_i oraz y_j są różne, więc iloczyn $x_i^{\mathrm{T}} x_j$ zwraca wartość pozytywną, a iloczyn $y_i y_j$ negatywną, przez co iloczyn tych dwóch iloczynów jest negatywny. Oznacza to, iż podwyższają one wartość funkcji Lagrange'a, co jest pożądane. To te wektory są wektorami nośnymi. α_i w tym wypadku ma wartość większą od zera.

Powyższe przypadki zostały przedstawione na rysunku 1.8.

Z równania hiperpłaszczyzny (1.33) wynika, iż klasyfikacja nowego wektora parametrów x zależy od iloczynu skalarnego x_i oraz x, gdzie x_i to wektory nośne. Całe powyższe rozumowanie dotyczyło danych liniowo separowalnych, jednak często zdarza się, iż analizowany zbiór danych ciężko jest odseparować liniowo. W tym wypadku stosuje się metodę zwaną *sztuczką jądra* (ang.



Rys. 1.8: Trzy możliwości ustawienia wektorów. Kolejność analogiczna do wspomnianej w tekście. Rysunek własny.

kernel trick) polegającą na rzutowaniu danych w inną przestrzeń funkcyjną, gdzie klasy będą separowalne liniowo. Przekształcenie to jest wyrażane za pomocą funkcji $\varphi(x)$. Rzutowanie polega na przekształceniu każdego wektora parametrów x_i z jego n-wymiarowej postaci do k-wymiarowej postaci $\varphi_j(x)$, gdzie $j = 1, \ldots, k$. Przekształcenie to skutkuje zmianą postaci maksymalizowanej względem α funkcji Lagrange'a

$$L(w, b, \alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j), \tag{1.35}$$

gdzie K jest *funkcją jądra* (ang. *kernel function*), która wyraża się wzorem:

$$K(x_i, x_j) = \varphi^{\mathrm{T}}(x_i)\varphi(x_j). \tag{1.36}$$

Rozwiązanie tej optymalizacji polega na wyznaczeniu wektora w oraz offsetu b:

$$\begin{aligned} w &= \textstyle\sum_{i=1}^{n} \alpha_i y_i \varphi(x_i) \\ b &= \pm 1 - w^{\mathrm{T}} \varphi(x_i) \end{aligned} \tag{1.37}$$

Uwzględniając założenia (1.36)–(1.37), równanie hiperpłaszczyzny zmienia postać na:

$$y(x) = w^{\mathrm{T}} \varphi(x) + b = \sum_{j=1}^{k} \alpha_i y_i K(x_i, x) + b = 0. \tag{1.38}$$

Istnieje wiele funkcji jądra K, które można stosować w algorytmie SVM. Najczęściej stosowane to:

- jądro wielomianowe (ang. *polynomial kernel*)

$$K(x_i, x) = (x^{\mathrm{T}} x_i + \gamma)^n, \tag{1.39}$$

- radialna funkcja bazowa (ang. *radial basis function*)

$$K(x_i, x) = e^{-\frac{\|x - x_i\|^2}{2\sigma^2}} = e^{-\gamma \|x - x_i\|^2}, \tag{1.40}$$

- jądro liniowe (ang. *linear kernel*)

$$K(x_i, x) = x^{\mathrm{T}} x_i + \gamma, \tag{1.41}$$

- jądro sigmoidalne (ang. *sigmoid kernel*)

$$K(x_i, x) = \mathrm{tgh}(\beta x^{\mathrm{T}} x_i + \gamma). \tag{1.42}$$

W równaniach (1.39)–(1.42) γ oraz β to parametry wolne, a n to stopień wielomianu.

Rozdział 2

Technologie

2.1. JupyterLab

JupyterLab jest jednym z najnowszych osiągnięć projektu *Project Jupyter*. Jest to narzędzie programistyczne umożliwiające korzystanie z wielu funkcjonalności niezbędnych w nauce o danych (ang. *data science*) za pomocą pojedynczej karty przeglądarki. Funkcjonalności te, to m.in. notatnik, terminal, edytor tekstu oraz przeglądarka plików. JupyterLab umożliwia tworzenie rozwiązań naukowych korzystając z wielu języków programowania, m.in. *Julia*, *R*, *Ruby* i oczywiście *Python*, dzięki czemu w krótkim czasie zyskał ogromną popularność. Jest to idealny interfejs użytkownika w sytuacjach, gdy potrzebujemy wykonywać wiele różnych wizualizacji danych jednocześnie. Zapewnia również możliwość uruchamiania wybranych fragmentów kodu, co jest niezmiernie przydatne w momencie, gdy potrzebujemy zmienić część kodu, a nie chcemy, aby wszystkie obliczenia były wykonywane od zera [24].

2.2. NumPy

NumPy to jedna z wielu naukowych bibliotek języka programowania Python. Umożliwia ona wydajną pracę z tablicami oraz wielowymiarowymi macierzami dużych rozmiarów. NumPy zawiera zarówno funkcje służące do tworzenia nowych, wypełnionych macierzy i tablic (np. `numpy.arange(start, end, step)`, jak i funkcje wykonujące na nich operacje. Przykładami takich funkcji wykorzystanych w niniejszej pracy są `numpy.mean(Array)` oraz `numpy.std(Array)` służące, odpowiednio, do obliczenia średniej oraz odchylenia standardowego zestawu danych. Przykład użycia przedstawiono w listingu 2.1, a efekt działania programu w listingu 2.2.

List. 2.1: Przykład użycia biblioteki NumPy.

```
1 import numpy as np
2 A = np.arange(1, 2, 30)
3 mean = np.mean(A)
4 standard_deviation = np.std(A)
5 print("Array A:", A)
6 print("Mean: ", mean)
7 print("Standard deviation: ", standard_deviation)
```

List. 2.2: Efekt działania kodu z listingu 2.1.

```
Array A: [ 1  3  5  7  9 11 13 15 17 19 21 23 25 27 29]
Mean: 15.0
Standard deviation: 8.640987597877148
```

2.3. SciPy

SciPy to biblioteka języka Python korzystająca z funkcjonalności oferowanych przez NumPy (tablice oraz funkcje z nimi związane) w celu stworzenia środowiska umożliwiającego rozwiązywanie naukowych problemów za pomocą programowania. Środowisko to udostępnia funkcjonalności służące do rozwiązywania zadań z zakresu algebry liniowej, analizy matematycznej, równań różniczkowych oraz przetwarzania sygnałów. Przykładem użycia SciPy w tej pracy są funkcje służące do znajdowania wszystkich ekstremów oraz filtrowania sygnału. Przykład użycia tej biblioteki przedstawiono na listingu 2.3

List. 2.3: Przykład użycia biblioteki SciPy.

```
from scipy.signal import butter, filtfilt
import math
import numpy as np

def butter_lowpass_filter(data, cutoff, fs, order):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    y = filtfilt(b, a, data)
    return y

t = np.arange(-2, 2, 0.01) # wektor czasu
f1 = 4 # częstotliwość pierwszego sygnału
f2 = 2 # częstotliwość drugiego sygnału
y1 = np.sin(2*math.pi*f1*t) # pierwszy sygnał
y2 = np.sin(2*math.pi*f2*t) # drugi sygnał
y3 = y1 + y2 # trzeci sygnał będący sumą y1 i~y2
cutoff = 3 # częstotliwość graniczna
fs = 100 # częstotliwość próbkowania filtrowanego sygnału
order = 5 # rząd filtru
y3_LPF = butter_lowpass_filter(y3, cutoff, fs, order) # y3 po filtrowaniu
```

W przykładzie tym doskonale widać, jak dobrze współgrają ze sobą biblioteki NumPy oraz SciPy oraz w jaki sposób efektywnie z nich korzystać. Bibliotekę NumPy wykorzystano do stworzenia wektora czasu t, sygnałów sinusoidalnych $y_1 = \sin(8\pi t)$, $y_2 = \sin(4\pi t)$ oraz $y_3 = y_1 + y_2$, a SciPy do wykonania operacji filtrowania dolnoprzepustowego na sygnale y_3. Zmienna `y3_LPF` przechowuje wynik filtrowania sygnału y_3.

2.4. Seaborn

Seaborn jest biblioteką działającą na podstawie biblioteki *Matplotlib*. Służy do wizualizacji danych w postaci różnego rodzaju wykresów, m.in. liniowych, pudełkowych oraz kropkowych. Jest to bardzo intuicyjne narzędzie ułatwiające pracę przy tworzeniu estetycznej graficznej prezentacji danych. Przykład użycia biblioteki Seaborn przedstawiono na listingu 2.4, a wynik działania kodu na rysunku 2.1. W przykładzie tym korzystano z wcześniej (2.3) utworzonych za pomocą bibliotek NumPy oraz SciPy sygnałów. Przykładem użycia Seaborn w tej pracy są liczne wykresy ilustrujące przebiegi sygnałów magnetoencefalograficznych oraz wykresy przedstawiające porównanie metod uczenia maszynowego.

List. 2.4: Przykład użycia biblioteki Seaborn.

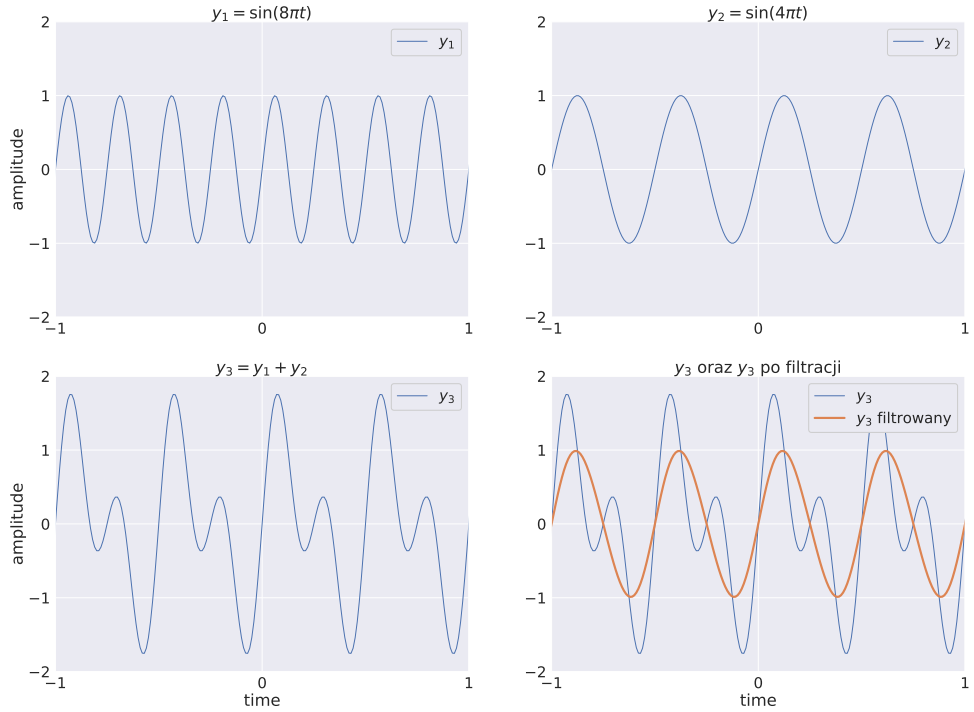
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(font_scale=2.5) # ustawienie rozmiaru czcionki
plt.figure(figsize=(30, 22)) # ustawienie rozmiaru wykresów
plt.subplot(2, 2, 1) # umiejscowienie wykresu na siatce 2x2
g1 = sns.lineplot(x=t, y=y1) # użycie Seaborn do narysowania
                             # wykresu liniowego
g1.set_title(r"$y_1=sin(8\pi t)$") # ustawienie tytułu wykresu
g1.legend([r"$y_1$"]) # przypisanie legendy do wykresu
g1.set_ylabel(r'amplitude') # ustawienie etykiety osi Y
g1.set_ylim([-2, 2]) # ustawienie zakresu wartości dla osi Y

plt.subplot(2, 2, 2)
g2 = sns.lineplot(x=t, y=y2)
g2.set_title(r"$y_2=sin(4\pi t)$")
g2.legend([r"$y_2$"])
g2.set_ylim([-2, 2])

plt.subplot(2, 2, 3)
g3 = sns.lineplot(x=t, y=y3)
g3.set_title(r"$y_3=y_1 + y_2$")
g3.legend([r"$y_3$"])
g3.set_xlabel(r'time') # ustawienie etykiety dla osi X
g3.set_ylabel(r'amplitude')
g3.set_ylim([-2, 2])

plt.subplot(2, 2, 4)
g4_1 = sns.lineplot(x=t, y=y3)
g4_1.set_title(r"$y_3$ oraz $y_3$ po filtracji")
g4_2 = sns.lineplot(x=t, y=y_LPF, linewidth = 4)
g4_1.legend([r"$y_3$", r"$y_3$ filtrowany"])
g4_1.set_xlabel(r'time')
g4_1.set_ylim([-2, 2])
```



Rys. 2.1: Przykład użycia biblioteki Seaborn.

2.5. Pandas

Pandas jest biblioteką opartą na NumPy służącą do manipulacji zbiorami danych. Umożliwia importowanie oraz eksportowanie danych z różnych formatów, m.in. CSV, Excel oraz SQL, dzięki czemu stanowi uniwersalne narzędzie stosowane w wielu środowiskach naukowych. Podstawową strukturą używaną w pandas jest *ramka danych* (`DataFrame`). Ramki danych mogą przechowywać zarówno dane kategoryczne, jak i numeryczne. Biblioteka ta dostarcza zestaw funkcji, które umożliwiają m.in. indeksowanie, łączenie, selekcję oraz operacje matematyczne na całych wierszach/ kolumnach ramki. Korzystanie z pandas jest bardzo proste dzięki kompatybilności tej biblioteki z podstawowymi strukturami danych języka Python. Przykład wykorzystania tej biblioteki przedstawiono na rysunku 2.5. Dane, z których korzystano, to popularny zbiór dotyczący katastrofy Titanica [25]. Dużym atutem pandas jest również sposób prezentacji ramek danych w interfejsie JupyterLab, przedstawiony w tabeli 2.1. W tej pracy wykorzystanie biblioteki pandas polegało na manipulacji sygnałami MEG oraz wynikami modeli uczenia maszynowego.

List. 2.5: Przykład użycia biblioteki pandas.

```
import pandas as pd

titanic_df = pd.read_csv("titanic.csv") # importowanie danych z~pliku
# wyświetlenie pierwszych pięciu rekordów za pomocą funkcji render_table
render_table(
    titanic_df[10:15].reset_index().drop(['index'], axis=1),
    header_columns=0, col_width=2, row_height=0.55)
```

Tab. 2.1: Przykład użycia biblioteki pandas w połączeniu z interfejsem JupyterLab.

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
11	1	3	Sandstrom	female	4.0	1	1	PP 9549	16.7	G6	S
12	1	1	Bonnell	female	58.0	0	0	113783	26.55	C103	S
13	0	3	Saundercock	male	20.0	0	0	A/5. 2151	8.05	nan	S
14	0	3	Andersson	male	39.0	1	5	347082	31.275	nan	S
15	0	3	Vestrom	female	14.0	0	0	350406	7.8542	nan	S

2.6. Scikit-learn

Scikit-learn jest zbudowaną na bazie NumPy, SciPy oraz Matplotlib biblioteką stosowaną do projektów związanych z uczeniem maszynowym. Zawiera w sobie wiele gotowych do użycia modeli uczenia maszynowego, do których wystarczy jedynie dostarczyć zbiory danych. Scikit-learn umożliwia jednak zmiany parametrów określonych modeli w celu zwiększenia ich dokładności. Oprócz gotowych modeli oferuje również liczne funkcje służące do wstępnej obróbki danych (ang. *data preprocessing*) oraz do analizy otrzymanych z modeli wyników. Korzystanie z modeli oferowanych przez tę bibliotekę jest wyjątkowo proste, ponieważ wystarczy jedynie odpowiednio wybrać parametry charakteryzujące konkretną metodę. Przykład kodu użytego do stworzenia modelu maszyny wektorów nośnych przedstawiono na listingu 2.6. W kodzie tym wykorzystano model k-najbliższych sąsiadów ustawiając parametr k na 10, a przestrzeń metryczną jako euklidesową. Dane, które klasyfikowano, pochodzą z popularnego zestawu *Iris flower data set*. Zestaw danych najpierw został zaciągnięty z biblioteki scikit-learn, następnie został podzielony na dane treningowe i testowe, a na koniec został znormalizowany. Za pomocą takiego zestawu wyuczony został model KNN, a następnie sklasyfikowano dane testowe. Końcową dokładność oraz macierz konfuzji tego modelu przedstawiono na listingu 2.7.

List. 2.6: Przykład użycia biblioteki scikit-learn.

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.preprocessing import StandardScaler
import pandas as pd

iris_dataset = datasets.load_iris() # załadowanie zestawu danych Iris

# Stworzenie obiektu dictionary
iris_dict = {
    "sepal_length": iris_dataset['data'][:, 0],
    "sepal_width": iris_dataset['data'][:, 1],
    "petal_length": iris_dataset['data'][:, 2],
    "petal_width": iris_dataset['data'][:, 3],
    "type": iris_dataset['target']
}
iris_df = pd.DataFrame.from_dict(iris_dict) # stworzenie ramki danych

# Tworzenie treningowego i~testowego zestawu danych
X = iris_df[['sepal_length', 'sepal_width',
```

```
    'petal_length', 'petal_width']].values
y = iris_df['type'].values
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.33, random_state=42)

# Tworzenie modelu k najbliższych sąsiadów z k=10 i metryką euklidesową
KNN = KNeighborsClassifier(n_neighbors = 10, metric = 'minkowski', p = 2)

# Skalowanie danych treningowych i testowych
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Dopasowanie modelu do danych treningowych
KNN.fit(X_train, y_train)
# Przetestowanie modelu na danych testowych
y_pred = KNN.predict(X_test)

# Wypisanie dokładności oraz macierzy konfuzji
print('Accuracy: ', accuracy_score(y_test, y_pred))
print('Confusion matrix: \n', confusion_matrix(y_test, y_pred))
```

List. 2.7: Dokładność oraz macierz konfuzji wcześniej stworzonego modelu.

```
Accuracy:  0.98
Confusion matrix:
 [[19  0  0]
 [ 0 15  0]
 [ 0  1 15]]
```

Rozdział 3

Dane MEG

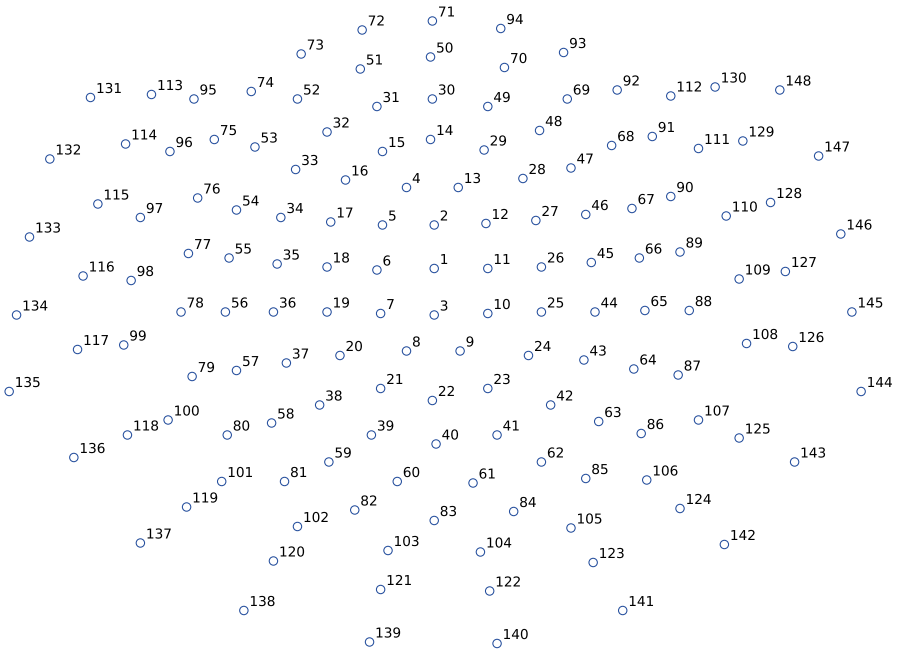
Dane, z których korzystano w pracy, to sygnały czasowe pochodzące z badania magnetoencefalograficznego.

3.1. Geneza

Pomiary wykonano za pomocą 148-kanałowego magnetoencefalografu. Trójwymiarową strukturę przestrzenną kanałów po zrzutowaniu na płaszczyznę przedstawiono na rysunku 3.1. Badanie polegało na podawaniu modulowanego częstotliwościowo sygnału dźwiękowego (ang. *chirp*) do układu słuchowego osoby badanej. Pomiary były przeprowadzane w dwóch trybach, *aktywnym* i *pasywnym*. Tryb aktywny polegał na nasłuchiwaniu przez osobę badaną podawanego sygnału dźwiękowego i wykonaniu konkretnej czynności w momencie, gdy ten sygnał usłyszy. Czynnością tą było kliknięcie przycisku sygnalizujące wynik kategoryzacji przez osobę badaną bodźca z uwagi na kierunek jego modulacji częstotliwościowej. Tryb pasywny polegał na biernym słuchaniu bodźców dźwiękowych. Dla każdego z trybów wynik badania jednej osoby był średnią arytmetyczną z wielu powtórzonych pomiarów (prób, ang. *trials*). W badaniach MEG obliczanie tego typu średnich jest zalecane ze względu na słaby stosunek sygnału do szumu dla pojedynczego pomiaru. Pomimo słabego SNR sygnał MEG posiada cechy charakterystyczne, więc po uśrednieniu po wielu próbach cechy te pozostaną w uśrednionym sygnale, a szum (który z założenia jest procesem o zerowej wartości oczekiwanej) zostanie zredukowany. W badaniu brało udział 16 osób. Dla każdej z osób wykonano pomiary w trybie aktywnym oraz pasywnym. Kolejność wykonywania trybów badań była zmieniana, aby wyeliminować jej możliwy wpływ na wyniki. Wszystkie osoby zostały zbadane według omówionego wyżej schematu oraz za pomocą tego samego sprzętu.

3.2. Charakterystyka

W tej sekcji zostanie omówiona charakterystyka danych pochodzących od jednej osoby badanej, dla pomiaru aktywnego oraz pasywnego. Ze względu na cechy charakterystyczne sygnałów, charakterystykę tę można uogólnić dla wszystkich osób badanych. Do pomiarów użyto 148-kanałowego magnetoencefalografu, czas pojedynczej próby był równy $\sim$1300 ms,



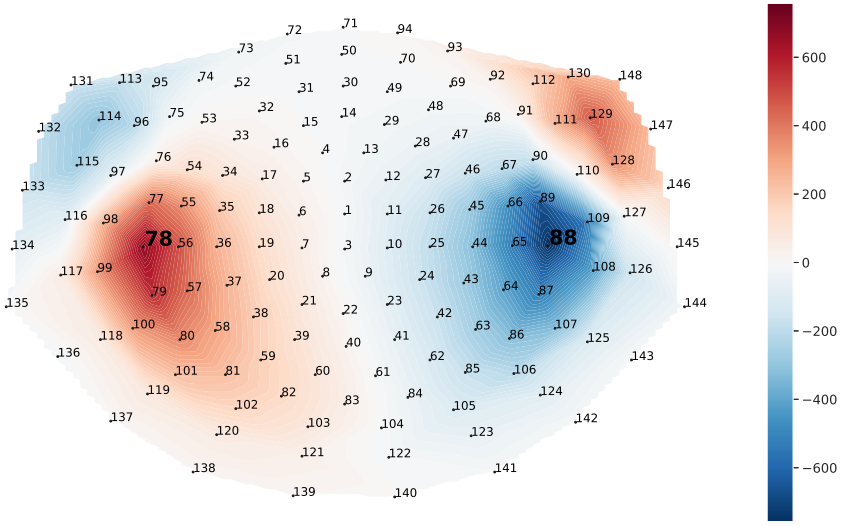
Rys. 3.1: Struktura przestrzenna kanałów magnetoencefalografu po zrzutowaniu na płaszczyznę. Widoczne liczby odpowiadają etykietom 148 kanałów. Rysunek własny.

a częstotliwość próbkowania po wynosiła $f_s = 254{,}3\,\text{Hz}$. Dane są dwuwymiarową macierzą o rozmiarach 148×331, gdzie 148 to liczba kanałów, a 331 to liczba próbek.

Tematem pracy jest kategoryzacja słuchowych odpowiedzi wywołanych mózgu, zatem uwagę skupiono na kanałach znajdujących się w pobliżu kory słuchowej. Mapę cieplną aktywności mózgu będącej odpowiedzią na bodziec dźwiękowy przedstawiono na rysunku 3.2. Widoczne na rysunku pola dipolowe odzwierciedlają przepływ prądów neuronalnych w korze słuchowej. Różnice w znakach (a zarazem kolorach) wartości przedstawionych na mapie cieplnej wynikają z różnego zwrotu pola magnetycznego mierzonego przez kanały. Różne zwroty pola magnetycznego wynikają z reguły prawej ręki. Płynący prąd elektryczny będący wynikiem przesyłania sygnałów w neuronach indukuje pole magnetyczne, które zgodnie z regułą prawej ręki w jednym kanale posiada zwrot w dół, w innym zaś do góry. Podział ten jest wyraźny na rysunku 3.2.

W celu pokazania możliwych do zaobserwowania różnic między odpowiedzią aktywną a pasywną na rysunku 3.3 przedstawiono przebiegi będące średnią z zarejestrowanych odpowiedzi wszystkich osób badanych, osobno dla słuchania aktywnego oraz pasywnego. Poczynione obserwacje są następujące:

- Przebieg aktywny załamków P50, N100 oraz P200 dla lewej półkuli pokrywa się niemalże całkowicie z przebiegiem pasywnym.
- Amplituda załamka P50 w trybie słuchania pasywnego jest mniejsza niż w trybie słuchania aktywnego dla obu półkul.

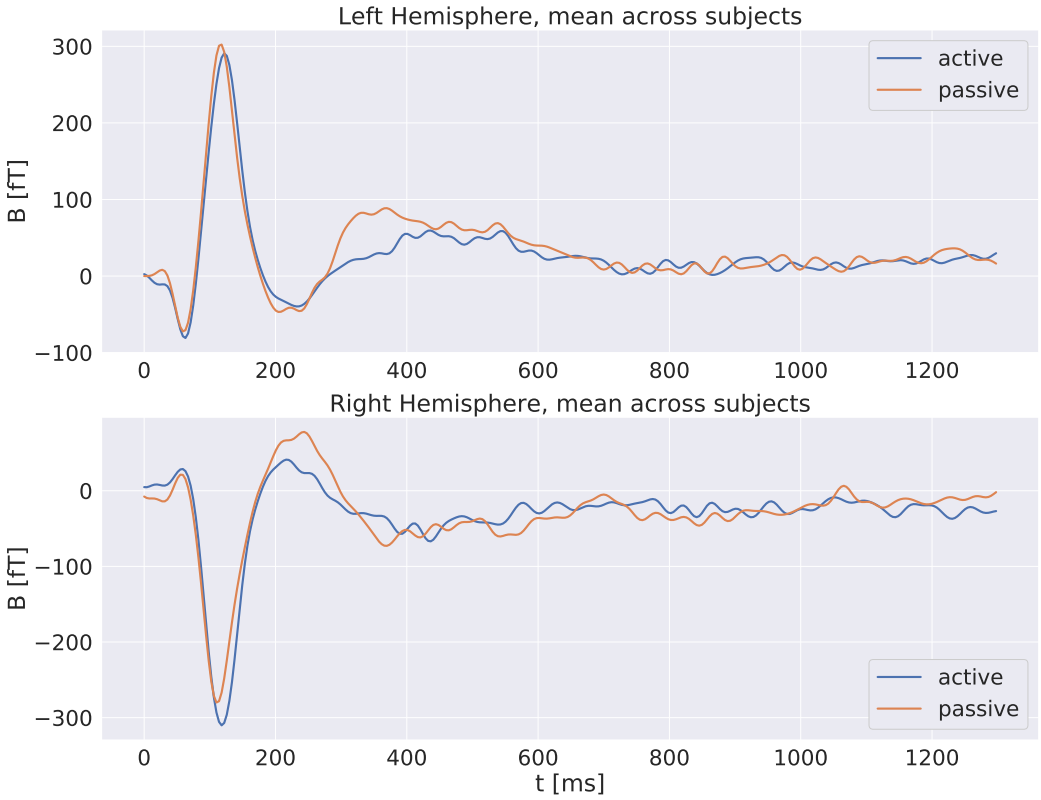


Rys. 3.2: Mapa cieplna (w femtoteslach) przedstawiająca aktywność mózgu będącą odpowiedzią na bodziec dźwiękowy wraz z naniesioną strukturą przestrzenną kanałów. Mapa pokazuje chwilę czasu, w której występuje wartość szczytowa załamka N100. Rysunek własny.

- Czas wystąpienia wartości szczytowej (latencja) załamka N100 w trybie słuchania pasywnego jest mniejszy niż w trybie słuchania aktywnego dla obu półkul.
- Amplituda załamka P200 w trybie słuchania pasywnego jest większa niż w trybie słuchania aktywnego dla obu półkul.
- Przebieg czasowy po załamku P200 zdaje się przybierać charakterystyczny kształt przypominający kolejny załamek. Jednak ze względu na późny czas wystąpienia tego zjawiska nie uwzględnia się go w analizie (może to być wpływ aktywności innej części mózgu, związanej z kliknięciem przycisku podczas zadania kategoryzacji słuchowej).

Ze względu na fakt, iż przebiegi przedstawione na rysunku 3.3 to średnie po wszystkich osobach badanych, powyższe spostrzeżenia posłużyły jedynie jako wskazówki do późniejszego postępowania z danymi. Do dalszych analiz brane będą pod uwagę dla każdego trybu dwa przebiegi, jeden pochodzący z półkuli lewej, a drugi z półkuli prawej. Zatem dla jednej osoby badanej łącznie będą to cztery przebiegi, dwa dla słuchania aktywnego oraz dwa dla słuchania pasywnego. Przykładowe przebiegi (dla osoby, której dane stanowią ilustrację do rozważań w niniejszej sekcji) przedstawiono na rysunku 3.4. Wybór kanałów polega na odnalezieniu tych, które posiadają najsilniejszą odpowiedź na bodziec dźwiękowy. W przypadku omawianej osoby są to kanały 78 (lewy) oraz 88 (prawy); ich etykiety przestawiono na rysunku 3.2 pogrubioną czcionką.

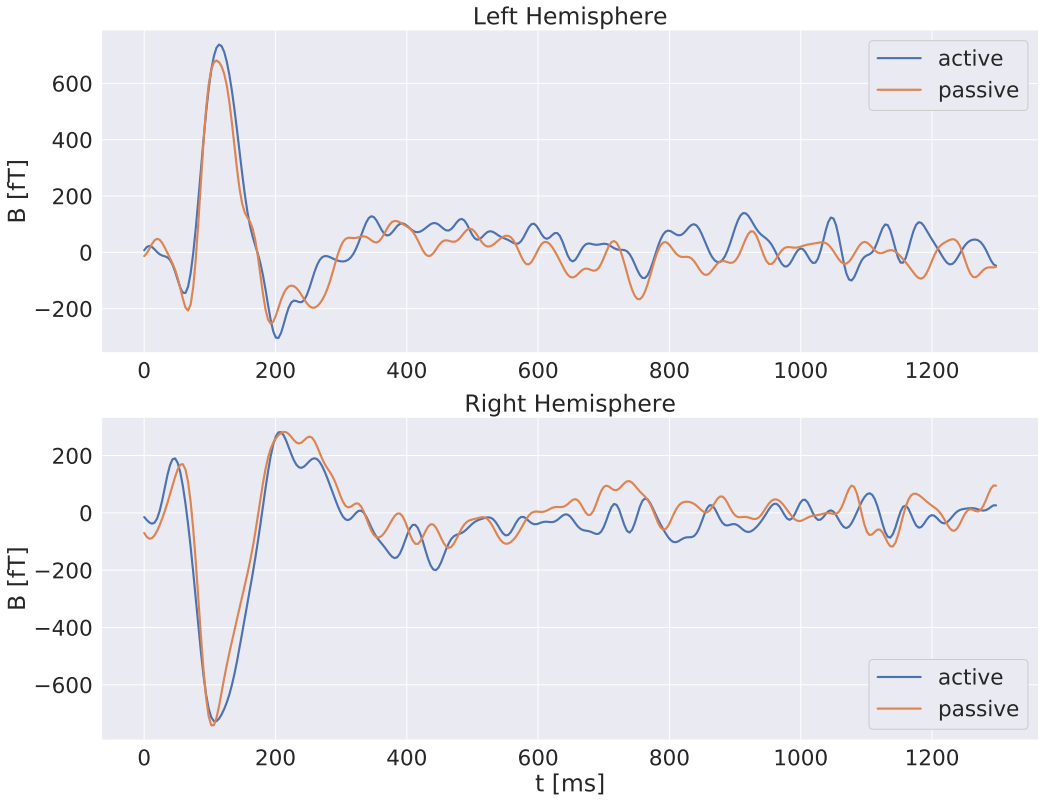
Sygnały na rysunku 3.4 wyraźnie różnią się od tych przedstawionych na rysunku 3.3. Różnica ta wynika m.in. z faktu, że sygnały przedstawione na rysunku 3.4 pochodzą od wybranej osoby badanej, przez co są zaszumione bardziej niż sygnały uśrednione po wszystkich osobach badanych. Na podstawie analizy licznych przebiegów czasowych sygnału MEG będącego odpowiedzią kory słuchowej na bodziec dźwiękowy stworzono schemat przykładowych odpowiedzi, pozytywnej oraz negatywnej. Schematy te (przedstawione na rysunkach 3.5 oraz 3.6) w dalszych częściach pracy posłużą jako wzorce do wyznaczania parametrów oryginalnych sygnałów. W obu



Rys. 3.3: Przebieg czasowy przedstawiającą średnią po wszystkich badanych dla dwu półkul, osobno dla słuchania aktywnego i pasywnego. Górny wykres przedstawia odpowiedź pozytywną (kanały z lewej półkuli), a dolny odpowiedź negatywną (kanały z prawej półkuli).

schematach zaznaczono charakterystyczne miejsca sygnałów. Są to niektóre z parametrów opisujących załamki P50, N100 oraz P200. Wartość szczytowa załamka P50 występuje przeważnie w okolicach 50 ms. W przypadku półkuli lewej jest ona lokalnym minimum, a w przypadku półkuli prawej lokalnym maksimum. Latencja załamka N100 to zwykle około 100 ms. W przypadku półkuli lewej wartość szczytowa jest lokalnym maksimum, a w przypadku półkuli prawej lokalnym minimum. Wartość szczytowa załamka P200 występuje w okolicach 200 ms i dla półkuli lewej jest lokalnym minimum, a dla prawej lokalnym maksimum. Miejsca zaznaczone na schematach to:

1. czas początku załamka P50,
2. latencja załamka P50,
3. wartość szczytowa załamka P50,
4. czas końca załamka P50 równoznaczny z czasem rozpoczęcia załamka N100,
5. latencja załamka N100,
6. wartość szczytowa załamka N100,
7. czas końca załamka N100 równoznaczny z czasem rozpoczęcia załamka P200,
8. latencja załamka P200,
9. wartość szczytowa załamka P200,
10. czas końca załamka P200.



Rys. 3.4: Przebieg czasowy wybranej osoby badanej. Górny wykres przedstawia odpowiedź pozytywną (kanał z lewej półkuli), a dolny odpowiedź negatywną (kanał z prawej półkuli).

Załamki te w rzeczywistości zwykle nie przyjmują tak regularnej formy. Ponadto, prawdopodobnie nie wszystkie parametry mają potencjał dyskryminacyjny w kontekście słuchania aktywnego *vs* pasywnego. Czas wystąpienia bodźca dźwiękowego w przedstawionych sygnałach to $t = 0$. Zakłada się, iż odpowiedź kory mózgowej na ten bodziec to załamki P50, N100 oraz P200. Sygnał występujący po załamku P200 może zawierać elementy świadczące o tym, iż osoba badana myślała o kliknięciu przycisku (nawet w przypadku badania pasywnego mogło być to podświadome). Biorąc to pod uwagę sygnał występujący po załamku P200 nie będzie podlegał analizie dotyczącej odpowiedzi kory słuchowej. Wykazano, iż najistotniejszym załamkiem jest N100, a w szczególności jego latencja [3]. Parametry, jakie będą podlegać analizie w przypadku każdego z załamków, to (X oznacza dany załamek, tj. P50, N100, P200):

- X_start — czas rozpoczęcia załamka,
- X_lat — latencja załamka (czas wystąpienia wartości szczytowej),
- X_amp — amplituda załamka (wartość szczytowa),
- X_end — czas końca załamka,
- X_onset_slope — zbocze narastające załamka, liczone jako:

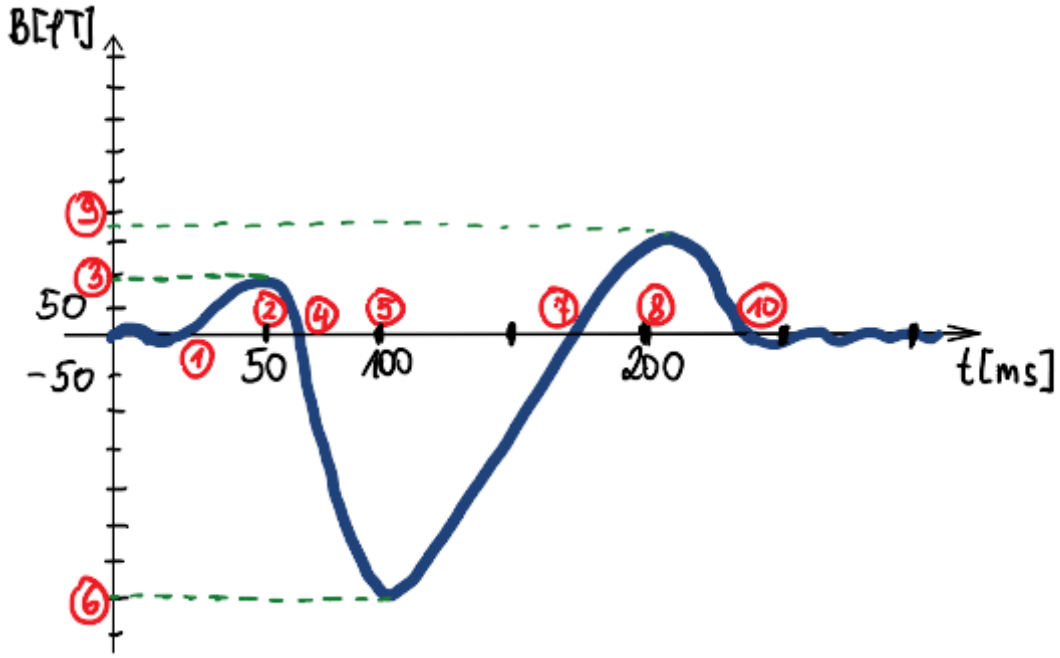
$$\text{X\_onset\_slope} = \frac{\text{X\_amp} - \text{X\_start\_value}}{\text{X\_lat} - \text{X\_start}}, \tag{3.1}$$

gdzie X_start_value to wartość występująca dla czasu X_start,

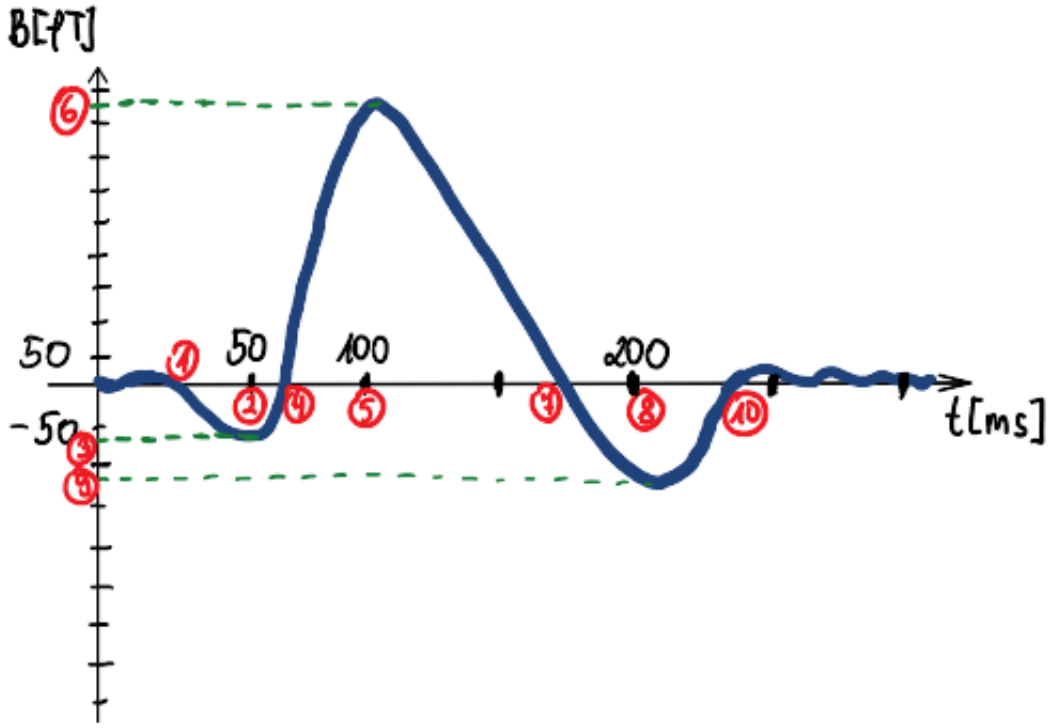
- X_offset_slope — zbocze opadające załamka, liczone jako:

$$\text{X\_offset\_slope} = \frac{\text{X\_end\_value} - \text{X\_amp}}{\text{X\_end} - \text{X\_lat}}, \tag{3.2}$$

  gdzie X_end_value to wartość występująca dla czasu X_end,
- X_surface — pole powierzchni pod załamkiem.



Rys. 3.5: Przykładowa odpowiedź negatywna. Na czerwono zaznaczono parametry, które są wydobywane z sygnału (omówione w tekście).



Rys. 3.6: Przykładowa odpowiedź pozytywna. Na czerwono zaznaczono parametry, które są wydobywane z sygnału (omówione w tekście).

Rozdział 4

Implementacja

Najważniejszym elementem nauki modeli uczenia maszynowego jest dostarczenie im odpowiednich danych treningowych. Dane te zostały pozyskane z sygnałów omówionych we wcześniejszym rozdziale. Wydobycie parametrów z sygnałów odbywało się za pomocą stworzonego do tego celu algorytmu. Następnie wybierając różne podzbiory parametrów uczono modele i oceniano, które cechy sygnału zapewniły najlepsze rezultaty. Przedstawione poniżej metody służą do klasyfikowania danych pod kątem słuchania aktywnego *vs* pasywnego dla danych pochodzących z konkretnej półkuli (lewa albo prawa). W pracy dokonano również klasyfikacji pod kątem pochodzenia sygnału (lewa półkula *vs* prawa półkula) dla konkretnego trybu słuchania (aktywnego albo pasywnego). Metody służące do obu typów klasyfikacji są analogiczne. Pełna lista metod wraz z ich implementacją znajduje się na płycie CD dołączonej do pracy.

4.1. Wydobycie parametrów z sygnału

Dla każdego załamka jest siedem parametrów, zatem uwzględniając trzy załamki, będzie łącznie 21 cech. Czas końca załamka P50 jest równy czasowi początku załamka N100, a czas końca załamka N100 jest równy czasowi początku załamka P200, przez co usunięto z listy $\mathrm{P50\_end}$ oraz $\mathrm{P200\_start}$ (w przyjętej nomenklaturze skupiono się na załamku N100 ze względu na jego rolę w analizie odpowiedzi kory słuchowej). Po redukcji dane treningowe przyjmują postać wektora o rozmiarze 19×1. Wektory tego typu będą służyły jako dane wejściowe służące do uczenia modeli. W celu wydobycia parametrów napisano algorytm, osobno dla odpowiedzi pozytywnej oraz negatywnej. Pseudokod uproszczonego algorytmu dotyczącego odpowiedzi negatywnej przedstawia listing 4.1. Pełny kod uwzględniający możliwość wystąpienia mniej przewidywalnych sygnałów wgrano na płytę CD będącą dodatkiem do pracy. Uproszczenie algorytmu w prezentowanym poniżej listingu polega na pominięciu sytuacji występujących rzadko; np. wystąpienie załamka P200 znajdującego się w całości pod osią czasu (czyli o wartościach wyłącznie ujemnych na całym przedziale trwania). Zestawienie parametrów wyznaczonych za pomocą algorytmu 4.1 dla odpowiedzi negatywnej z wykresu 3.4 przedstawia tabela 4.1.

List. 4.1: Pseudokod uproszczonego algorytmu służącego do wydobycia parametrów z sygnału pochodzącego z kanału dla prawej półkuli (odpowiedź negatywna).

```
Zadeklaruj funkcję get_features przyjmującą jako parametr przebieg sygnału
Zadeklaruj zmienne p50_start, p50_start_value, p50_lat, p50_amp, p50_end,
p50_end_value, p50_onset_slope, p50_offset_slope, p50_surface, n100_start,
n100_start_value, n100_lat, n100_amp, n100_end, n100_end_value,
n100_onset_slope, n100_offset_slope, n100_surface, p200_start,
p200_start_value, p200_lat, p200_amp, p200_end, p200_end_value,
p200_onset_slope, p200_offset_slope, p200_surface,

Znajdź wszystkie ekstrema sygnału i przypisz je do zmiennej.

Znajdź minimum globalne i zainicjalizuj zmienne n100_lat oraz n100_amp.

Przypisz do zmiennych p50_lat oraz p50_amp wartości maksimum lokalnego
występującego przed minimum globalnym.

Przypisz do zmiennych p200_lat oraz p200_amp wartości maksimum lokalnego
występującego po minimum globalnym.

Przypisz do zmiennych p50_start oraz p50_start_value wartości minimum
lokalnego występującego przed wartością p50_lat.

Dla każdej wartości sygnału na lewo od p50_lat (krok = -1):
    Jeśli wartości są malejące oraz zmieniają znak:
            Przypisz do zmiennych p50_start oraz p50_start_value wartości
            sygnału, dla których nastąpiła zmiana znaku i wyjdź z pętli.

Dla każdej wartości sygnału na prawo od p50_lat (krok = 1):
    Jeśli wartości są malejące oraz zmieniają znak:
            Przypisz do zmiennych p50_end oraz p50_end_value wartości
            sygnału, dla których nastąpiła zmiana znaku i wyjdź z pętli.

Przypisz do zmiennych p50_onset_slope, p50_offset_slope, p50_surface
wartości wyliczone na podstawie wcześniej wspomnianych wzorów.

Przypisz do zmiennych n100_start oraz n100_start_value wartości p50_end
oraz p50_end_value.

Dla każdej wartości sygnału na prawo od n100_lat (krok = 1):
    Jeśli wartości są rosnące oraz zmieniają znak:
        Przypisz do zmiennych n100_end oraz n100_end_value wartości
        sygnału, dla których nastąpiła zmiana znaku i wyjdź z pętli.

Przypisz do zmiennych n100_onset_slope, n100_offset_slope, n100_surface
wartości wyliczone na podstawie wcześniej wspomnianych wzorów.

Przypisz do zmiennych p200_start oraz p200_start_value wartości n100_end
oraz n100_end_value.

Dla każdej wartości sygnału na prawo od p200_lat (krok = 1):
```

```
    Jeśli wartości są malejące oraz zmieniają znak:
            Przypisz do zmiennych p200_end oraz p200_end_value wartości
            sygnału, dla których nastąpiła zmiana znaku i wyjdź z pętli.

Przypisz do zmiennych p200_onset_slope, p200_offset_slope, p200_surface
wartości wyliczone na podstawie wcześniej wspomnianych wzorów.

Zwróć wszystkie parametry w postaci obiektu Dictionary.
```

Tab. 4.1: Przykładowe parametry sygnału wyznaczone za pomocą opracowanego algorytmu.

parameter	value	parameter	value	parameter	value
p50_start	19.66	n100_start	62.91	p200_start	180.88
p50_lat	47.19	n100_lat	110.1	p200_lat	208.4
p50_amp	190.18	n100_amp	-727.21	p200_amp	281.69
p50_end	62.91	n100_end	180.88	p200_end	239.86
p50_onset_slope	7.64	n100_onset_slope	-16.1	p200_onset_slope	9.44
p50_offset_slope	-10.04	n100_offset_slope	10.58	p200_offset_slope	-3.96
p50_surface	4884.36	n100_surface	51375.55	p200_surface	11735.06

4.2. Tworzenie modeli

Modele, które będą podlegać dalszej analizie, to drzewa decyzyjne (DT), lasy losowe (RF), regresja logistyczna (LR), naiwny klasyfikator bayesowski (NB), k-najbliższych sąsiadów (KNN), maszyna wektorów nośnych (SVM) z jądrem RBF oraz SVM z jądrem wielomianowym (SVCP). Danymi treningowymi będą wydobyte wcześniej parametry sygnałów wraz z odpowiadającą sygnałowi klasą (*active* albo *passive*). Stworzenie, wyuczenie oraz sprawdzenie pojedynczego modelu można przedstawić w kilku krokach:

1. Wydobycie parametrów z sygnału.
2. Przypisanie odpowiedniej klasy do zbioru parametrów.
3. Podział danych na treningowe i testowe.
4. Odseparowanie parametrów od klas w zestawie treningowym i testowym.
5. Stworzenie obiektu standaryzującego dane i dopasowanie go do danych treningowych.
6. Standaryzacja parametrów treningowych i testowych.
7. Stworzenie obiektu modelu uczenia maszynowego.
8. Dopasowanie modelu do danych treningowych (parametrów oraz klas).
9. Sprawdzenie jakości modelu na danych testowych.

Z uwagi na fakt, iż w celu rzetelnego porównania wszystkie modele będą wyuczane tymi samymi danymi treningowymi, kroki 1–6 zostały wykonane jednokrotnie, dla wszystkich klasyfikatorów. Kod służący do stworzenia i oceny wszystkich analizowanych modeli przedstawiono na listingu 4.2.

List. 4.2: Kod przedstawiający tworzenie modeli uczenia maszynowego.

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

#parameters - lista zawierająca parametry sygnału, które będą
# brane pod uwagę przy trenowaniu modeli
#active - sygnały pochodzące ze słuchania aktywnego
#passive - sygnały pochodzące ze słuchania pasywnego
#indexes - aktualnie analizowane indeksy osób badanych (omówione poniżej)
def check_models(parameters, active, passive, indexes):

    train = [] #lista zawierająca dane treningowe
               # (w postaci przebiegów czasowych)
    test = [] #lista zawierająca dane testowe
              # (w postaci przebiegów czasowych)

    #Podział danych na treningowe i testowe
    for index in indexes['train']:
        train.append(active[index])
        train.append(passive[index])
    for index in indexes['test']:
        test.append(active[index])
        test.append(passive[index])

    #Tworzenie treningowych i testowych ramek danych za pomocą funkcji,
    # która wydobywa z sygnału zadane w liście parametry oraz przypisuje
    # zestawowi danych odpowiednią klasę.
    train_features = generate_df_with_feat(parameters, train)
    test_features = generate_df_with_feat(parameters, test)

    #Odseparowanie klas.
    X_train = train_features.iloc[:, :-1].values
    y_train = train_features.iloc[:, -1].values
    X_test = test_features.iloc[:, :-1].values
    y_test = test_features.iloc[:, -1].values

    #Standaryzacja danych.
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)

    #Tworzenie obiektów modeli uczenia maszynowego.
    SVM = SVC(kernel='rbf', random_state=0, gamma='scale', C=1)
    RF = RandomForestClassifier(n_estimators=50, criterion='entropy',
        random_state=0)
    LR = LogisticRegression(random_state=0)
    KNN = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2)
    SVC_P = SVC(kernel='poly', random_state=0)
    NB = GaussianNB()
    DT = DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
56
57      #Dopasowanie modeli do danych treningowych.
58      SVM.fit(X_train, y_train)
59      RF.fit(X_train, y_train)
60      LR.fit(X_train, y_train)
61      KNN.fit(X_train, y_train)
62      SVC_P.fit(X_train, y_train)
63      NB.fit(X_train, y_train)
64      DT.fit(X_train, y_train)
65
66      #Stworzenie list zawierających sklasyfikowane przez modele dane testowe.
67      y_pred_svm = SVM.predict(X_test)
68      y_pred_rf = RF.predict(X_test)
69      y_pred_lr = LR.predict(X_test)
70      y_pred_knn = KNN.predict(X_test)
71      y_pred_svcp = SVC_P.predict(X_test)
72      y_pred_nb = NB.predict(X_test)
73      y_pred_dt = DT.predict(X_test)
74
75      #Sprawdzenie dokładności modeli na podstawie porównania klas
76      # przydzielonych przez modele z rzeczywistymi klasami.
77      accuracy_svm = round(accuracy_score(y_test, y_pred_svm), 2)
78      accuracy_rf = round(accuracy_score(y_test, y_pred_rf), 2)
79      accuracy_lr = round(accuracy_score(y_test, y_pred_lr), 2)
80      accuracy_knn = round(accuracy_score(y_test, y_pred_knn), 2)
81      accuracy_svcp = round(accuracy_score(y_test, y_pred_svcp), 2)
82      accuracy_nb = round(accuracy_score(y_test, y_pred_nb), 2)
83      accuracy_dt = round(accuracy_score(y_test, y_pred_dt), 2)
84
85      #Funkcja zwraca dokładności modeli w formie obiektu Dictionary.
86      return {"SVM": accuracy_svm, "RF": accuracy_rf, "LR": accuracy_lr,
87      "KNN": accuracy_knn, "SVCP": accuracy_svcp, "NB": accuracy_nb,
88      "DT": accuracy_dt}
```

W zmiennych `accuracy_svm`, `accuracy_rf`, `accuracy_lr`, `accuracy_knn`, `accuracy_svcp`, `accuracy_nb`, `accuracy_dt` przechowywane są miary dokładności każdego z modeli, liczone jako stosunek poprawnie przypisanych klas do całkowitego rozmiaru danych testowych. Przykładowe wyniki działania metody `check_models` przedstawia tabela 4.2.

Tab. 4.2: Wyniki pochodzące z uruchomienia metody `check_models` dla wszystkich parametrów wyznaczonych z załamków P50, N100 oraz P200.

model	accuracy
SVM	1.0
RF	1.0
LR	1.0
KNN	0.5
SVCP	1.0
NB	0.5
DT	0.5

4.3. Ocena dokładności kategoryzacji

W celu oceny dokładności modeli zastosowano *sprawdzian krzyżowy* (ang. *cross-validation*), a dokładniej metodę *leave-one-out*. Jest to odmiana metody k*-krotnej walidacji*, w której k jest równe 1. Metoda leave-one-out jest zalecana przy małych zbiorach danych i polega na przypisaniu do zbioru testowego tylko jednego zestawu parametrów. W przypadku tej pracy będzie to polegało na przypisaniu do zbioru treningowego 15 osób badanych, a do zbioru testowego jednej osoby badanej. Stosując tą metodę otrzymano 16 różnych wariantów danych uczących oraz treningowych. Warianty te przedstawiono w tabeli 4.3, gdzie liczby w nawiasach odpowiadają numerom osób badanych. Jako dokładność całkowitą pojedynczego modelu uznaje się średnią arytmetyczną z tych 16 wariantów. Dokładność pojedynczego wariantu wyznaczana jest za pomocą funkcji przedstawionej na listingu 4.2, gdzie zmienne `train` oraz `test` przechowują aktualnie analizowane sygnały (wybierane za pomocą metody leave-one-out). Średnia arytmetyczna z 16 podejść będzie służyła do oceny jakości klasyfikatora dla danego zestawu parametrów. Funkcję odpowiedzialną za wyznaczenie całkowitych dokładności modeli przedstawiono na listingu 4.3.

Tab. 4.3: Możliwe warianty podziału na dane treningowe i testowe, wyznaczone za pomocą metody leave-one-out. Etykiety liczbowe odpowiadają poszczególnym osobom badanym.

train	test
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]	[1]
[1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]	[2]
[1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]	[3]
[1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]	[4]
[1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]	[5]
[1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]	[6]
[1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16]	[7]
[1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16]	[8]
[1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16]	[9]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16]	[10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16]	[11]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16]	[12]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16]	[13]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16]	[14]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16]	[15]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]	[16]

List. 4.3: Zastosowanie metody leave-one-out do oceny dokładności modeli.

```
def compare_mode(
    k=1,
    parameters=['p50_start', 'p50_lat', 'p50_amp', 'p50_onset_slope',
    'p50_offset_slope', 'p50_surface', 'n100_start', 'n100_lat', 'n100_amp',
    'n100_end', 'n100_onset_slope', 'n100_offset_slope', 'n100_surface'],
    active=[],
    passive=[]):

```

```
    #Inicjalizacja list, które będą przechowywać dokładności dla zadanych
    # podziałów wynikających z metody Leave-one-out.
    accuracy_svm_list = []
    accuracy_rf_list = []
    accuracy_lr_list = []
    accuracy_knn_list = []
    accuracy_svcp_list = []
    accuracy_nb_list = []
    accuracy_dt_list = []

    #16 wariantów indeksów wygenerowanych przez funkcję all_subsets,
    # która dla zadanej listy oraz długości k generuje wszystkie możliwe
    # podzbiory.
    all_indexes = all_subsets([i for i in range(len(active))], k)

    #Zapełnienie list wynikami. W każdym obiegu pętli wywoływana jest
    # funkcja check_models, która zwraca obiekt Dictionary zawierający
    # w sobie dokładności wszystkich modeli dla zadanego podziału
    # oraz parametrów.
    for indexes in all_indexes:
        accuracies = check_models(parameters=parameters, active=active,
        passive=passive, indexes=indexes)
        accuracy_svm_list.append(accuracies['SVM'])
        accuracy_rf_list.append(accuracies['RF'])
        accuracy_lr_list.append(accuracies['LR'])
        accuracy_knn_list.append(accuracies['KNN'])
        accuracy_svcp_list.append(accuracies['SVCP'])
        accuracy_nb_list.append(accuracies['NB'])
        accuracy_dt_list.append(accuracies['DT'])

    #Wyznaczenie za pomocą biblioteki NumPy odchyleń standardowych
    # oraz średnich z list dokładności.
    accuracy_svm = round(np.mean(accuracy_svm_list), 2)
    accuracy_svm_std = round(np.std(accuracy_svm_list), 2)

    accuracy_rf = round(np.mean(accuracy_rf_list), 2)
    accuracy_rf_std = round(np.std(accuracy_rf_list), 2)

    accuracy_lr = round(np.mean(accuracy_lr_list), 2)
    accuracy_lr_std = round(np.std(accuracy_lr_list), 2)

    accuracy_knn = round(np.mean(accuracy_knn_list), 2)
    accuracy_knn_std = round(np.std(accuracy_knn_list), 2)

    accuracy_svcp = round(np.mean(accuracy_svcp_list), 2)
    accuracy_svcp_std = round(np.std(accuracy_svcp_list), 2)

    accuracy_nb = round(np.mean(accuracy_nb_list), 2)
    accuracy_nb_std = round(np.std(accuracy_nb_list), 2)

    accuracy_dt = round(np.mean(accuracy_dt_list), 2)
    accuracy_dt_std = round(np.std(accuracy_dt_list), 2)
```

```
61
62      #Funkcja zwraca dla każdego modelu średnią dokładność, odchylenie
63      # standardowe oraz pełną listę dokładności.
64      return {
65          "SVM": accuracy_svm, "SVM_std": accuracy_svm_std,
66          "SVM_list": accuracy_svm_list, "RF": accuracy_rf,
67          "RF_std": accuracy_rf_std, "RF_list": accuracy_rf_list,
68          "LR": accuracy_lr, "LR_std": accuracy_lr_std,
69          "LR_list": accuracy_lr_list, "KNN": accuracy_knn,
70          "KNN_std": accuracy_knn_std, "KNN_list": accuracy_knn_list,
71          "SVCP": accuracy_svcp, "SVCP_std": accuracy_svcp_std,
72          "SVCP_list": accuracy_svcp_list, "NB": accuracy_nb,
73          "NB_std": accuracy_nb_std, "NB_list": accuracy_nb_list,
74          "DT": accuracy_dt, "DT_std": accuracy_dt_std,
75          "DT_list": accuracy_dt_list,
76      }
```

Zwracane przez metodę `compare_mode` odchylenie standardowe oraz pełna lista wyników dla każdego modelu posłużą do analizy statystycznej otrzymanych wyników. Przykładowe wyniki uruchomienia metody `compare_mode` przedstawia tabela 4.4.

Tab. 4.4: Przykładowe wyniki pochodzące z uruchomienia metody `compare_mode` dla wszystkich parametrów wyznaczonych z załamków P50, N100 oraz P200.

model	mean accuracy	standard deviation
SVM	0.62	0.22
RF	0.62	0.28
LR	0.69	0.3
KNN	0.56	0.3
SVCP	0.56	0.24
NB	0.53	0.12
DT	0.62	0.28

4.4. Porównanie parametrów

W poprzednim rozdziale przedstawiono wynik działania metody `compare_mode` uruchomionej dla wszystkich 19 parametrów. Część parametrów może wnosić mało istotne informacje do klasyfikacji. Z tego powodu opracowano metodę `compare_parameters` (listing 4.4), która zbiera wyniki działania metody `compare_mode` dla wszystkich możliwych podzbiorów listy parametrów. Lista ta zadana jest jako jeden z argumentów wejściowych metody `compare_parameters`. Pierwszych pięć wyników uzyskanych za pomocą tej metody (biorąc pod uwagę jedynie załamki P50 oraz N100) przedstawiono w tabeli 4.5. Ze względu na przejrzystość tabeli pominięto pełne listy oraz odchylenia standardowe, a pokazano jedynie średnią dokładność każdego z modeli.

List. 4.4: Kod przedstawiający metodę służącą do porównania parametrów.

```
1 #signal_dataframes - obiekt Dictionary zawierający listy z sygnałami
2 # pochodzącymi ze słuchania aktywnego oraz pasywnego
3 #params - lista parametrów, której podzbiory będą analizowane. Domyślnie
4 # wszystkie parametry załamka N100
```

```
#mode - tryb określający pochodzenie danych. 'negative' dla prawej półkuli,
# 'positive' dla lewej. Domyślnie 'negative'
def compare_parameters(
    signal_dataframes,
    params=['n100_peak', 'n100_value', 'n100_end', 'n100_rise_slope',
    'n100_fall_slope', 'n100_surface'],
    mode='negative'):

    results = [] #lista, w której przechowywane będą wyniki

    #Wygenerowana za pomocą metody all_subsets lista
    # zawierająca wszystkie kombinacje parametrów.
    all_parameters = all_subsets(params)

    #Stworzenie list active oraz passive będących źródłami danych
    # użytych do uczenia modeli.
    df_act = signal_dataframes['act']
    df_pass = signal_dataframes['pass']
    active = []
    passive = []

    for df in df_act:
        features = get_feat(df, mode)
        features['active'] = 1
        active.append(features)
    for df in df_pass:
        features = get_feat(df, mode)
        features['active'] = 0
        passive.append(features)

    #Wypełnianie listy results wynikami pochodzącymi z metody
    # compare_mode dla kolejnych zestawów parametrów.
    for parameters in all_parameters:
        accuracies = compare_mode(1, parameters, active, passive)
        results.append({"accuracies":accuracies, "parameters":parameters})

    #Metoda zwraca listę zawierającą obiekty Dictionary z polami
    # odpowiadającymi zestawom parametrów oraz wynikami ich dotyczącymi.
    return results
```

Tab. 4.5: Pierwszych pięć wyników pochodzących z metody `compare_parameters` uruchomionej dla wszystkich parametrów z załamków P50 oraz N100.

params	SVM	RF	LR	KNN	SVCP	NB	DT
['p50_start']	0.5	0.5	0.5	0.56	0.5	0.41	0.47
['p50_lat']	0.38	0.44	0.5	0.69	0.5	0.41	0.47
['p50_amp']	0.38	0.38	0.47	0.41	0.56	0.47	0.38
['p50_onset_slope']	0.5	0.72	0.44	0.56	0.53	0.56	0.72
['p50_offset_slope']	0.28	0.62	0.31	0.34	0.5	0.47	0.62

Rozdział 5

Dane treningowe

Metoda `compare_parameters` posiada wysoki koszt obliczeniowy. Jest to spowodowane faktem, iż jest ona uruchamiana dla wszystkich podzbiorów parametrów z zadanej listy. Gdy weźmiemy pod uwagę wszystkie 19 parametrów, liczba m możliwych podzbiorów to 524 287 (patrz (5.1), gdzie wyłączono zbiór pusty).

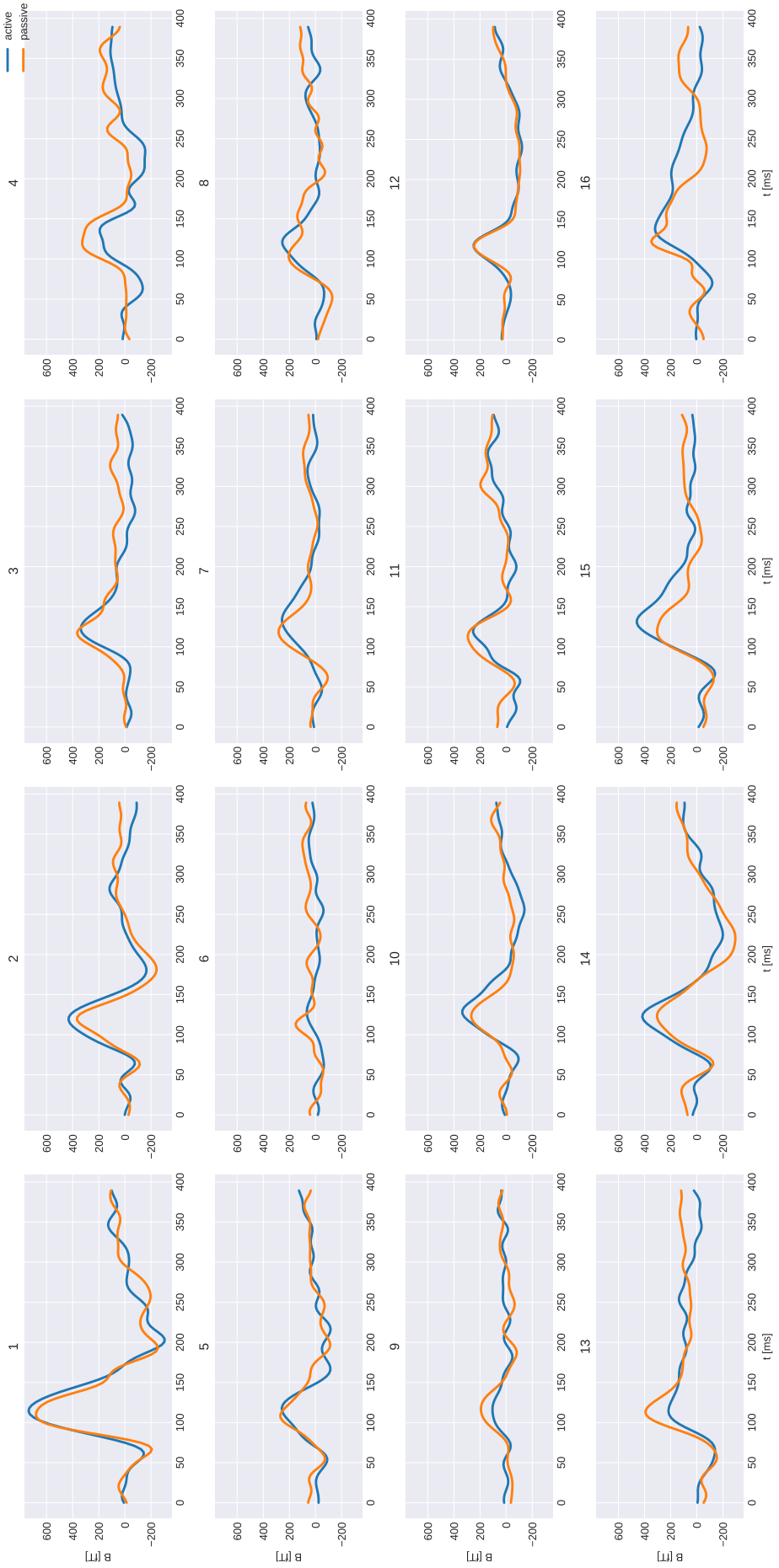
$$m = 2^n - 1 = 2^{19} - 1 = 524\,288 - 1 = 524\,287. \tag{5.1}$$

Aby sprawdzić dokładność w każdym możliwym wariancie, konieczne byłoby wykonanie 524 287 sekwencji uczenia modeli. Czas na wykonanie tylu sekwencji jest zbyt duży. Postanowiono zatem zredukować liczbę analizowanych parametrów. Ze względu na istotność pików P50 oraz N100 przy analizie odpowiedzi korowych mózgu, to właśnie na tych załamkach się skupiono. Odrzucono więc wszystkie parametry załamka P200, dzięki czemu liczba możliwości zmniejszyła się do $2^{13} - 1 = 8\,191$.

5.1. Sygnały i parametry

W celu jak najdokładniejszego wglądu do danych, z których korzystano w procesie uczenia modeli, w podrozdziale tym przedstawiono wszystkie warianty klasyfikacyjne wraz z sygnałami ich dotyczącymi. Analizie będą podlegać cztery warianty klasyfikacji:

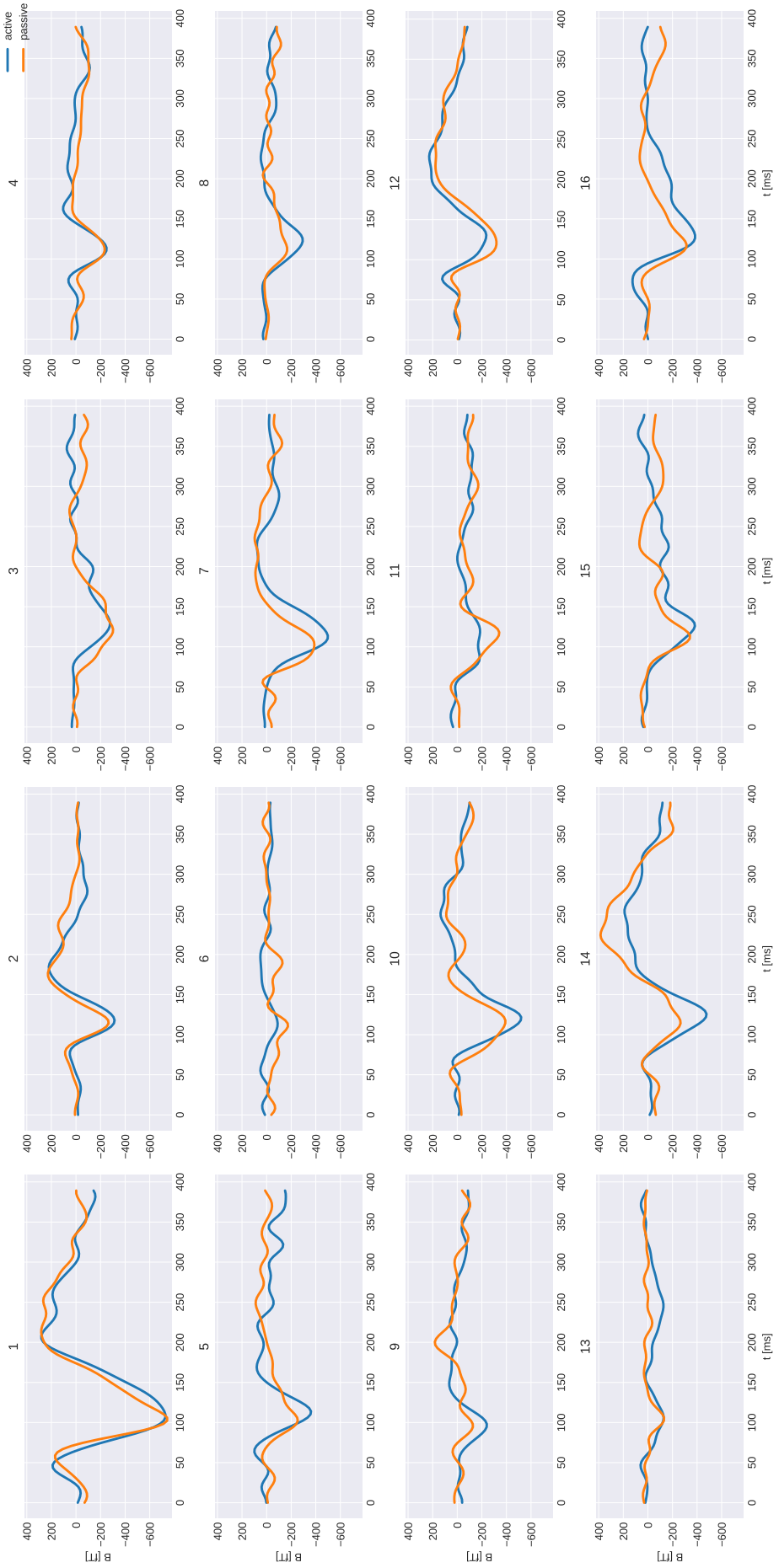
1. Klasyfikacja pod kątem słuchania aktywnego *vs* pasywnego dla lewej półkuli mózgu. Sygnały dla tego wariantu przedstawia rysunek 5.1, a parametry tabela 5.1.
2. Klasyfikacja pod kątem słuchania aktywnego *vs* pasywnego dla prawej półkuli mózgu. Sygnały dla tego wariantu przedstawia rysunek 5.2, a parametry tabela 5.2.
3. Klasyfikacja pod kątem półkuli lewej *vs* prawej dla słuchania aktywnego. Sygnały dla tego wariantu przedstawia rysunek 5.3, a parametry tabela 5.3.
4. Klasyfikacja pod kątem półkuli lewej *vs* prawej dla słuchania pasywnego. Sygnały dla tego wariantu przedstawia rysunek 5.4, a parametry tabela 5.4.



Rys. 5.1: Sygnały dla słuchania aktywnego (niebieski) oraz pasywnego (pomarańczowy) pochodzące z lewej półkuli mózgu.

Tab. 5.1: Zbiór parametrów sygnałów z lewej półkuli mózgu.

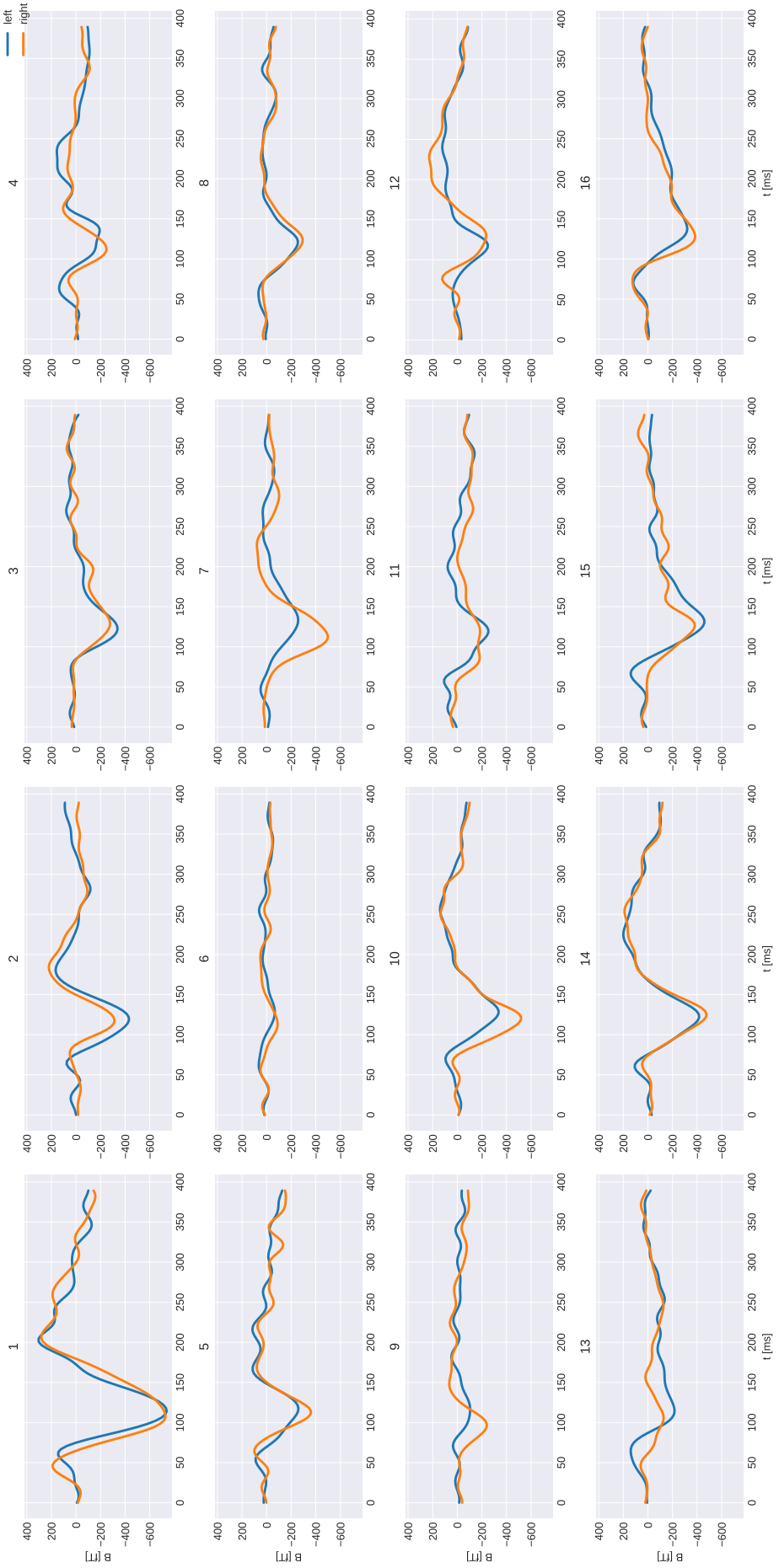
p50_start	p50_lat	p50_amp	p50_onset_slope	p50_offset_slope	p50_surface	n100_start	n100_lat	n100_amp	n100_end	n100_onset_slope	n100_offset_slope	n100_surface	active
19.66	62.91	-144.32	-3.36	11.93	3532.55	74.71	114.03	737.71	173.01	18.85	-12.57	40421.64	1.0
31.46	66.85	-206.82	-6.31	15.65	4865.47	78.64	110.1	681.33	173.01	22.37	-11.01	35426.33	0.0
47.19	62.91	-75.74	-6.22	5.29	1248.96	74.71	117.96	431.13	157.29	10.28	-11.27	21162.51	1.0
43.25	62.91	-114.7	-7.09	6.13	2156.39	74.71	117.96	365.73	149.42	9.44	-11.99	14792.45	0.0
39.32	70.78	-42.62	-1.06	2.36	1182.83	82.58	121.9	338.0	196.61	8.97	-3.65	18806.05	1.0
15.73	62.91	4.88	0.05	5.64	2277.98	94.37	117.96	365.45	208.4	7.76	-3.23	20822.36	0.0
39.32	62.91	-137.34	-5.84	4.5	4595.77	90.44	133.69	192.57	157.29	4.77	-8.54	8660.32	1.0
39.32	51.12	-11.6	-0.15	0.48	307.35	74.71	117.96	326.74	173.01	7.56	-6.07	18013.29	0.0
27.53	55.05	-89.64	-3.11	5.46	2035.46	66.85	117.97	258.61	149.42	5.55	-8.58	12408.8	1.0
39.32	55.05	-73.12	-5.37	4.12	1297.4	66.85	110.1	268.5	176.95	6.78	-4.16	13294.96	0.0
35.39	62.92	-63.84	-2.67	1.51	2565.9	98.3	129.76	66.47	173.02	2.45	-1.55	2526.77	1.0
15.73	55.05	-60.12	-1.6	3.27	2202.26	70.78	114.03	151.84	153.36	3.71	-3.11	4446.86	0.0
27.52	47.19	-49.38	-2.68	2.47	1165.71	66.85	133.69	256.93	224.13	3.86	-2.85	18632.65	1.0
35.39	62.91	-93.24	-3.86	5.29	2362.89	78.64	117.96	283.44	196.61	7.47	-2.87	15526.1	0.0
23.59	55.05	-65.96	-2.14	2.76	2200.59	74.71	121.9	254.77	173.02	5.65	-5.2	12807.28	1.0
0.0	51.12	-128.62	-2.13	4.76	5732.04	70.78	102.24	206.04	153.36	7.66	-1.3	10757.23	0.0
55.05	70.78	-35.03	-2.8	2.2	507.16	78.64	117.97	104.63	169.09	3.11	-2.3	5244.2	1.0
55.05	66.85	-17.18	-0.6	1.46	275.41	74.71	117.97	193.26	161.22	4.6	-4.66	9355.72	0.0
27.53	70.78	-95.47	-2.31	5.32	2582.68	82.58	129.76	336.26	184.81	7.82	-6.36	17998.19	1.0
39.32	55.05	-44.33	-3.3	3.02	799.6	66.85	121.9	267.06	165.15	5.01	-6.22	12878.32	0.0
39.32	58.98	-106.48	-2.58	7.57	2557.81	70.78	117.97	253.05	161.22	5.73	-6.02	11967.08	1.0
35.39	55.05	-66.24	-4.5	4.8	1280.49	66.85	114.03	293.53	149.42	6.42	-8.54	14637.71	0.0
28.51	56.03	-37.13	-1.6	1.74	1081.44	75.69	118.95	248.85	150.41	5.82	-8.56	9274.1	1.0
59.97	75.69	-36.2	-2.84	2.09	501.17	83.56	118.95	245.05	146.47	7.48	-9.87	8226.51	0.0
15.73	62.92	-139.45	-2.78	5.62	5903.0	86.51	114.03	216.86	212.34	8.13	-1.16	16788.84	1.0
27.53	55.05	-153.58	-4.33	4.9	5313.47	78.64	114.03	393.38	228.07	12.19	-2.95	24170.52	0.0
39.32	58.98	-107.22	-6.03	6.24	2242.86	74.71	121.9	417.15	169.08	9.03	-9.06	21556.45	1.0
47.19	62.91	-126.51	-8.63	7.13	2540.15	78.64	121.9	304.69	169.08	7.38	-6.66	15966.1	0.0
35.39	66.85	-140.12	-3.97	6.62	3860.19	82.58	129.76	459.22	271.32	10.5	-2.69	36050.88	1.0
31.46	58.98	-129.74	-2.74	4.91	4733.78	82.58	117.96	303.86	192.67	8.98	-3.14	18663.18	0.0
15.73	70.78	-119.26	-2.18	4.9	3738.11	94.37	137.62	320.05	204.47	7.49	-1.87	22988.43	1.0
47.19	58.98	-61.01	-5.23	4.21	918.84	70.78	121.9	349.18	153.35	7.05	-3.66	14382.54	0.0



Rys. 5.2: Sygnały dla słuchania aktywnego (niebieski) oraz pasywnego (pomarańczowy) pochodzące z prawej półkuli mózgu.

Tab. 5.2: Zbiór parametrów sygnałów z prawej półkuli mózgu.

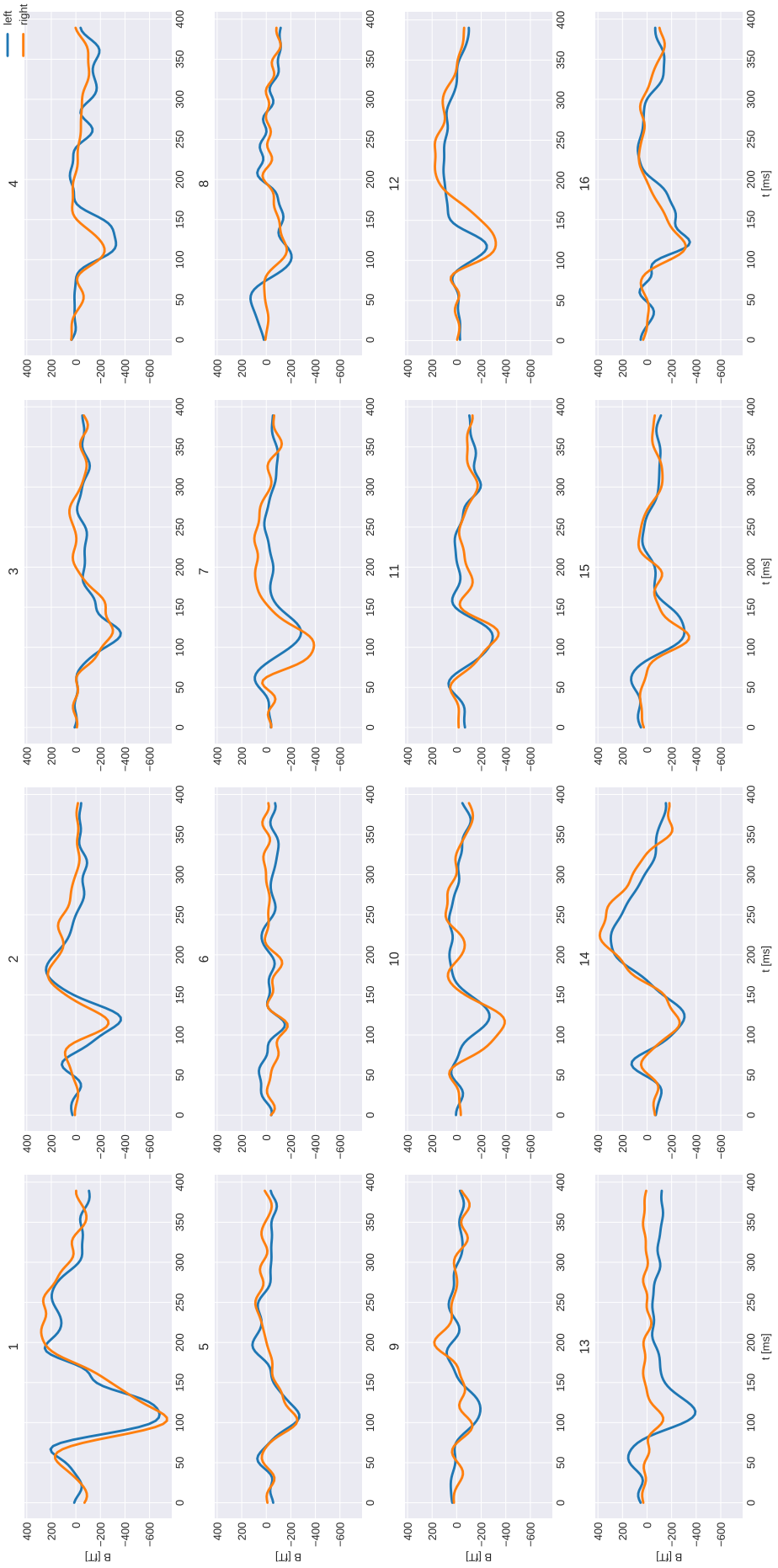
p50_start	p50_lat	p50_amp	p50_onset_slope	p50_offset_slope	p50_surface	n100_start	n100_lat	n100_amp	n100_end	n100_onset_slope	n100_offset_slope	n100_surface	active
19.66	47.19	190.18	7.64	-10.04	4884.36	62.91	110.1	-727.21	180.88	-16.1	10.58	51375.55	1.0
27.52	58.98	169.91	6.02	-11.12	4312.35	70.78	102.24	-741.69	176.95	-24.81	10.54	42016.06	0.0
51.12	74.71	49.77	2.17	-3.19	1050.06	86.51	117.96	-313.95	153.35	-10.37	10.19	11846.3	1.0
35.39	78.64	88.17	2.21	-6.32	2614.16	90.44	114.03	-265.46	145.49	-11.83	9.62	8065.01	0.0
55.05	70.78	24.48	0.52	-1.56	551.72	82.58	129.76	-277.18	196.61	-6.0	2.04	18370.11	1.0
11.8	58.98	-2.3	0.02	-5.46	2701.68	90.44	121.9	-301.72	149.42	-4.06	2.13	14778.79	0.0
55.05	74.71	61.15	3.3	-4.98	1036.31	82.58	114.03	-248.22	145.49	-8.59	8.15	9158.38	1.0
55.05	74.71	-9.61	2.71	-1.29	846.44	78.64	110.1	-233.41	153.35	-6.95	5.63	9975.97	0.0
43.25	62.92	99.37	5.41	-4.92	2157.09	78.64	114.03	-361.06	153.36	-10.82	9.8	14141.46	1.0
43.25	55.05	36.11	4.08	-1.98	586.66	66.85	102.24	-253.39	173.02	-7.52	2.87	13155.77	0.0
39.32	55.05	50.39	3.27	-1.83	1216.62	82.58	114.03	-89.75	149.42	-2.86	2.74	3295.43	1.0
78.64	90.44	-85.26	1.25	-0.21	2241.53	102.24	110.1	-173.26	157.29	-10.89	2.47	4675.79	0.0
3.93	23.59	24.34	0.52	-0.93	773.98	47.19	114.03	-497.71	176.95	-7.48	7.98	29990.17	1.0
35.39	55.05	32.67	2.29	-0.31	1787.65	74.71	102.24	-388.82	157.29	-15.09	7.37	20057.55	0.0
27.53	62.92	33.46	0.88	-1.69	974.9	74.71	121.9	-293.39	180.88	-6.51	4.99	15710.47	1.0
39.32	66.85	17.12	0.79	-1.16	438.4	78.64	110.1	-166.77	176.95	-5.41	1.57	9903.14	0.0
39.32	51.12	-14.72	0.9	-1.76	465.23	62.92	98.3	-240.09	129.76	-5.78	7.87	9396.36	1.0
51.12	62.92	37.77	3.5	-2.1	483.1	70.78	94.37	-127.05	141.56	-6.29	1.24	4377.37	0.0
51.12	66.85	37.87	2.97	-3.03	439.68	70.78	121.9	-519.69	184.81	-10.67	8.33	28399.76	1.0
35.39	51.12	60.61	3.88	-4.86	1016.21	62.92	117.97	-391.13	161.22	-7.16	9.66	22217.85	0.0
39.32	47.19	16.59	0.86	-1.01	219.51	55.05	117.97	-185.91	169.09	-3.09	2.25	14673.55	1.0
31.46	51.12	51.8	2.78	-3.93	891.4	58.98	117.97	-341.57	180.88	-6.15	3.36	19317.4	0.0
56.03	75.69	122.91	6.55	-6.23	2342.23	87.49	130.74	-236.17	166.13	-6.6	7.22	10967.5	1.0
63.9	75.69	48.08	4.4	-2.9	587.28	83.56	118.95	-318.62	177.93	-9.72	5.81	17918.6	0.0
19.66	47.19	59.55	2.02	-3.79	1305.86	58.98	106.17	-128.42	149.42	-3.04	3.17	6262.74	1.0
66.85	78.64	-6.14	1.02	-0.15	359.17	90.44	106.17	-131.29	141.56	-7.85	3.83	3583.31	0.0
47.19	62.91	47.22	3.39	-2.97	688.77	70.78	125.83	-476.86	169.08	-9.1	11.06	24465.67	1.0
51.12	62.91	49.05	4.66	-3.65	751.07	74.71	114.03	-265.58	165.15	-6.91	5.4	13940.51	0.0
39.32	51.12	8.83	0.22	-0.6	176.3	62.91	125.83	-382.38	176.95	-6.11	4.21	22678.6	1.0
0.0	39.32	57.91	0.81	-1.92	2722.06	66.85	114.03	-341.98	192.67	-7.35	2.8	17920.78	0.0
31.46	70.78	124.9	3.17	-5.17	4718.48	94.37	125.83	-383.59	188.74	-12.29	3.02	24003.87	1.0
47.19	70.78	50.99	2.45	-3.29	1069.4	82.58	117.96	-314.33	200.54	-9.23	3.93	17658.1	0.0



Rys. 5.3: Sygnały dla słuchania aktywnego pochodzące z lewej oraz z prawej półkuli mózgu. Wartości pochodzące z półkuli lewej zostały przemnożone przez -1 w celu lepszego porównania sygnałów.

Tab. 5.3: Zbiór parametrów sygnałów dla słuchania aktywnego.

p50_start	p50_lat	p50_amp	p50_onset_slope	p50_offset_slope	p50_surface	n100_start	n100_lat	n100_amp	n100_end	n100_onset_slope	n100_offset_slope	n100_surface	left
19.66	62.91	144.32	3.36	-11.93	3532.55	74.71	114.03	-737.71	173.01	-18.85	12.57	40421.64	1.0
19.66	47.19	190.18	7.64	-10.04	4884.36	62.91	110.1	-727.21	180.88	-16.1	10.58	51375.55	0.0
47.19	62.91	75.74	6.22	-5.29	1248.96	74.71	117.96	-431.13	157.29	-10.28	11.27	21162.51	1.0
51.12	74.71	49.77	2.17	-3.19	1050.06	86.51	117.96	-313.95	153.35	-10.37	10.19	11846.3	0.0
39.32	70.78	42.62	1.06	-2.36	1182.83	82.58	121.9	-338.0	196.61	-8.97	3.65	18806.05	1.0
55.05	70.78	24.48	0.52	-1.56	551.72	82.58	129.76	-277.18	196.61	-6.0	2.04	18370.11	0.0
39.32	62.91	137.34	5.84	-4.5	4595.77	90.44	133.69	-192.57	157.29	-4.77	8.54	8660.32	1.0
55.05	74.71	61.15	3.3	-4.98	1036.31	82.58	114.03	-248.22	145.49	-8.59	8.15	9158.38	0.0
27.53	55.05	89.64	3.11	-5.46	2035.46	66.85	117.97	-258.61	149.42	-5.55	8.58	12408.8	1.0
43.25	62.92	99.37	5.41	-4.92	2157.09	78.64	114.03	-361.06	153.36	-10.82	9.8	14141.46	0.0
35.39	62.92	63.84	2.67	-1.51	2565.9	98.3	129.76	-66.47	173.02	-2.45	1.55	2526.77	1.0
39.32	55.05	50.39	3.27	-1.83	1216.62	82.58	114.03	-89.75	149.42	-2.86	2.74	3295.43	0.0
27.52	47.19	49.38	2.68	-2.47	1165.71	66.85	133.69	-256.93	224.13	-3.86	2.85	18632.65	1.0
3.93	23.59	24.34	0.52	-0.93	773.98	47.19	114.03	-497.71	176.95	-7.48	7.98	29990.17	0.0
23.59	55.05	65.96	2.14	-2.76	2200.59	74.71	121.9	-254.77	173.02	-5.65	5.2	12807.28	1.0
27.53	62.92	33.46	0.88	-1.69	974.9	74.71	121.9	-293.39	180.88	-6.51	4.99	15710.47	0.0
55.05	70.78	35.03	2.8	-2.2	507.16	78.64	117.97	-104.63	169.09	-3.11	2.3	5244.2	1.0
39.32	51.12	14.72	0.9	-1.76	465.23	62.92	98.3	-240.09	129.76	-5.78	7.87	9396.36	0.0
27.53	70.78	95.47	2.31	-5.32	2582.68	82.58	129.76	-336.26	184.81	-7.82	6.36	17998.19	1.0
51.12	66.85	37.87	2.97	-3.03	439.68	70.78	121.9	-519.69	184.81	-10.67	8.33	28399.76	0.0
39.32	58.98	106.48	2.58	-7.57	2557.81	70.78	117.97	-253.05	161.22	-5.73	6.02	11967.08	1.0
39.32	47.19	16.59	0.86	-1.01	219.51	55.05	117.97	-185.91	169.09	-3.09	2.25	14673.55	0.0
28.51	56.03	37.13	1.6	-1.74	1081.44	75.69	118.95	-248.85	150.41	-5.82	8.56	9274.1	1.0
56.03	75.69	122.91	6.55	-6.23	2342.23	87.49	130.74	-236.17	166.13	-6.6	7.22	10967.5	0.0
15.73	62.92	139.45	2.78	-5.62	5903.0	86.51	114.03	-216.86	212.34	-8.13	1.16	16788.84	1.0
19.66	47.19	59.55	2.02	-3.79	1305.86	58.98	106.17	-128.42	149.42	-3.04	3.17	6262.74	0.0
39.32	58.98	107.22	6.03	-6.24	2242.86	74.71	121.9	-417.15	169.08	-9.03	9.06	21556.45	1.0
47.19	62.91	47.22	3.39	-2.97	688.77	70.78	125.83	-476.86	169.08	-9.1	11.06	24465.67	0.0
35.39	66.85	140.12	3.97	-6.62	3860.19	82.58	129.76	-459.22	271.32	-10.5	2.69	36050.88	1.0
39.32	51.12	8.83	0.22	-0.6	176.3	62.91	125.83	-382.38	176.95	-6.11	4.21	22678.6	0.0
15.73	70.78	119.26	2.18	-4.9	3738.11	94.37	137.62	-320.05	204.47	-7.49	1.87	22988.43	1.0
31.46	70.78	124.9	3.17	-5.17	4718.48	94.37	125.83	-383.59	188.74	-12.29	3.02	24003.87	0.0



Rys. 5.4: Sygnały dla słuchania pasywnego pochodzące z lewej oraz z prawej półkuli. Wartości pochodzące z półkuli lewej zostały przemnożone przez -1 w celu lepszego porównania sygnałów.

Tab. 5.4: Zbiór parametrów sygnałów dla słuchania pasywnego.

p50_start	p50_lat	p50_amp	p50_onset_slope	p50_offset_slope	p50_surface	n100_start	n100_lat	n100_amp	n100_end	n100_onset_slope	n100_offset_slope	n100_surface	left
31.46	66.85	206.82	6.31	-15.65	4865.47	78.64	110.1	-681.33	173.01	-22.37	11.01	35426.33	1.0
27.52	58.98	169.91	6.02	-11.12	4312.35	70.78	102.24	-741.69	176.95	-24.81	10.54	42016.06	0.0
43.25	62.91	114.7	7.09	-6.13	2156.39	74.71	117.96	-365.73	149.42	-9.44	11.99	14792.45	1.0
35.39	78.64	88.17	2.21	-6.32	2614.16	90.44	114.03	-265.46	145.49	-11.83	9.62	8065.01	0.0
15.73	62.91	4.88	0.05	-5.64	2277.98	94.37	117.96	-365.45	208.4	-7.76	3.23	20822.36	1.0
11.8	58.98	2.3	0.02	-5.46	2701.68	90.44	121.9	-301.72	149.42	-4.06	2.13	14778.79	0.0
39.32	51.12	11.6	0.15	-0.48	307.35	74.71	117.96	-326.74	173.01	-7.56	6.07	18013.29	1.0
55.05	74.71	9.61	2.71	-1.29	846.44	78.64	110.1	-233.41	153.35	-6.95	5.63	9975.97	0.0
39.32	55.05	73.12	5.37	-4.12	1297.4	66.85	110.1	-268.5	176.95	-6.78	4.16	13294.96	1.0
43.25	55.05	36.11	4.08	-1.98	586.66	66.85	102.24	-253.39	173.02	-7.52	2.87	13155.77	0.0
15.73	55.05	60.12	1.6	-3.27	2202.26	70.78	114.03	-151.84	153.36	-3.71	3.11	4446.86	1.0
78.64	90.44	85.26	1.25	-0.21	2241.53	102.24	110.1	-173.26	157.29	-10.89	2.47	4675.79	0.0
35.39	62.91	93.24	3.86	-5.29	2362.89	78.64	117.96	-283.44	196.61	-7.47	2.87	15526.1	1.0
35.39	55.05	32.67	2.29	-0.31	1787.65	74.71	102.24	-388.82	157.29	-15.09	7.37	20057.55	0.0
0.0	51.12	128.62	2.13	-4.76	5732.04	70.78	102.24	-206.04	153.36	-7.66	1.3	10757.23	1.0
39.32	66.85	17.12	0.79	-1.16	438.4	78.64	110.1	-166.77	176.95	-5.41	1.57	9903.14	0.0
55.05	66.85	17.18	0.6	-1.46	275.41	74.71	117.97	-193.26	161.22	-4.6	4.66	9355.72	1.0
51.12	62.92	37.77	3.5	-2.1	483.1	70.78	94.37	-127.05	141.56	-6.29	1.24	4377.37	0.0
39.32	55.05	44.33	3.3	-3.02	799.6	66.85	121.9	-267.06	165.15	-5.01	6.22	12878.32	1.0
35.39	51.12	60.61	3.88	-4.86	1016.21	62.92	117.97	-391.13	161.22	-7.16	9.66	22217.85	0.0
35.39	55.05	66.24	4.5	-4.8	1280.49	66.85	114.03	-293.53	149.42	-6.42	8.54	14637.71	1.0
31.46	51.12	51.8	2.78	-3.93	891.4	58.98	117.97	-341.57	180.88	-6.15	3.36	19317.4	0.0
59.97	75.69	36.2	2.84	-2.09	501.17	83.56	118.95	-245.05	146.47	-7.48	9.87	8226.51	1.0
63.9	75.69	48.08	4.4	-2.9	587.28	83.56	118.95	-318.62	177.93	-9.72	5.81	17918.6	0.0
27.53	55.05	153.58	4.33	-4.9	5313.47	78.64	114.03	-393.38	228.07	-12.19	2.95	24170.52	1.0
66.85	78.64	6.14	1.02	-0.15	359.17	90.44	106.17	-131.29	141.56	-7.85	3.83	3583.31	0.0
47.19	62.91	126.51	8.63	-7.13	2540.15	78.64	121.9	-304.69	169.08	-7.38	6.66	15966.1	1.0
51.12	62.91	49.05	4.66	-3.65	751.07	74.71	114.03	-265.58	165.15	-6.91	5.4	13940.51	0.0
31.46	58.98	129.74	2.74	-4.91	4733.78	82.58	117.96	-303.86	192.67	-8.98	3.14	18663.18	1.0
0.0	39.32	57.91	0.81	-1.92	2722.06	66.85	114.03	-341.98	192.67	-7.35	2.8	17920.78	0.0
47.19	58.98	61.01	5.23	-4.21	918.84	70.78	121.9	-349.18	153.35	-7.05	3.66	14382.54	1.0
47.19	70.78	50.99	2.45	-3.29	1069.4	82.58	117.96	-314.33	200.54	-9.23	3.93	17658.1	0.0

Rozdział 6

Wyniki

Wyniki, które będą analizowane, to dokładność modeli przy kategoryzacji polegającej na:

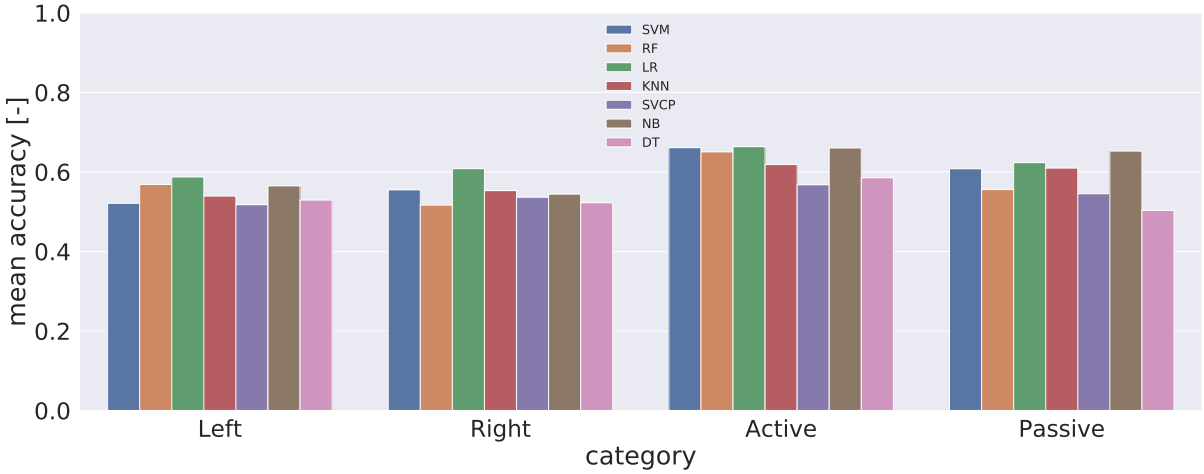
1. Przypisaniu klasy rodzaju słuchania, czyli aktywne *vs* pasywne, osobno dla sygnałów pochodzących z lewej oraz z prawej półkuli mózgu.
2. Przypisaniu klasy źródła sygnału, czyli lewa *vs* prawa półkula mózgu, osobno dla słuchania aktywnego oraz pasywnego.

Porównanie dokładności uzyskanej przez modele w tych czterech wariantach klasyfikacyjnych pomoże odpowiedzieć na pytania przedstawione we Wprowadzeniu. Analizę wyników podzielono na porównanie ogólne i szczególne. W porównaniu ogólnym pod uwagę brane są wszystkie wyniki uzyskane przez modele. W porównaniu szczególnym pod uwagę brane są wszystkie modele z osobna, dla zestawów parametrów, które najlepiej odwzorowują klasyfikację słuchania (aktywne *vs* pasywne) oraz półkuli mózgu (lewa *vs* prawa). Uzyskanie wyników przedstawionych w tym rozdziale wymagało zastosowania wielu różnych podejść. Niektóre z nich dawały dobre rezultaty (te przedstawione w rozdziale Implementacja). Inne zaś powodowały zmniejszenie dokładności modeli albo były niepoprawne merytorycznie. Najbardziej interesujące z nich to zastosowanie techniki bootstrap w celu zwiększenia ilości danych treningowych oraz próba wydobycia parametrów z części sygnału występującej po końcu załamka P200. Zastosowanie techniki bootstrap polegało na powielaniu danych treningowych poprzez uśrednianie sygnałów wylosowanych ze zwracaniem z dostępnej puli badań (osobno dla słuchania aktywnego oraz pasywnego). Technika ta okazała się jednak niekorzystna dla dokładności modeli uczenia maszynowego (niższa wartość średniej dokładności klasyfikacji), przez co ją odrzucono. Wydobycie parametrów z części sygnału występującej po załamku P200 wymagało stworzenia osobnego algorytmu, który obejmował m.in. przefiltrowanie sygnału filtrem dolnoprzepustowym celem wygładzenia. Wyniki uzyskane przez modele wyuczone za pomocą dodatkowych parametrów były bardzo obiecujące (dokładność $\sim$84%, a odchylenie standardowe $\sim$3%), lecz podejście to również należało odrzucić, gdyż część sygnału, z której wydobyto parametry, prawdopodobnie była powiązana z reakcją mózgu na myśl o kliknięciu przycisku.

6.1. Porównanie ogólne

W poniższych podsekcjach, odpowiadających czterem analizowanym wariantom klasyfikacyjnym, przedstawiono tabele ze średnimi dokładnościami modeli klasyfikacyjnych dla tych podzbiorów parametrów, dla których wartości średnich dokładności klasyfikacji okazały się najwyższe (dla co najmniej jednego z siedmiu analizowanych modeli klasyfikacyjnych). W każdej z tabel, w każdym wierszu najwyższe wartości zaznaczono na zielono. Dla porównania, przedstawiono również wyniki otrzymane z pozostałych modeli klasyfikacyjnych.

Następnie za pomocą wykresów pudełkowych pokazano wpływ poszczególnych parametrów sygnału na dokładność klasyfikacji. Przy tworzeniu wykresów pudełkowych dla konkretnych parametrów sygnału brane były pod uwagę wszystkie podzbiory, w których występował dany parametr. Liczba takich podzbiorów to, jak wiadomo, 2^{n-1}. W rozważanym kontekście $n = 13$, zatem liczba wszystkich podzbiorów, w których występuje dany parametr, to $2^{12} = 4\,096$. Wyniki przedstawione na tych wykresach pochodzą z modelu regresji logistycznej. Wybór tego modelu został poparty analizą wykresów pokazanych na rysunku 6.1, który przedstawia średnią dokładność każdego z modeli (liczoną dla wszystkich analizowanych parametrów). Dla trzech z czterech wariantów najlepszą średnią dokładność posiada model regresji logistycznej (słupki zielone), zatem to dla niego opracowano wykresy pudełkowe.

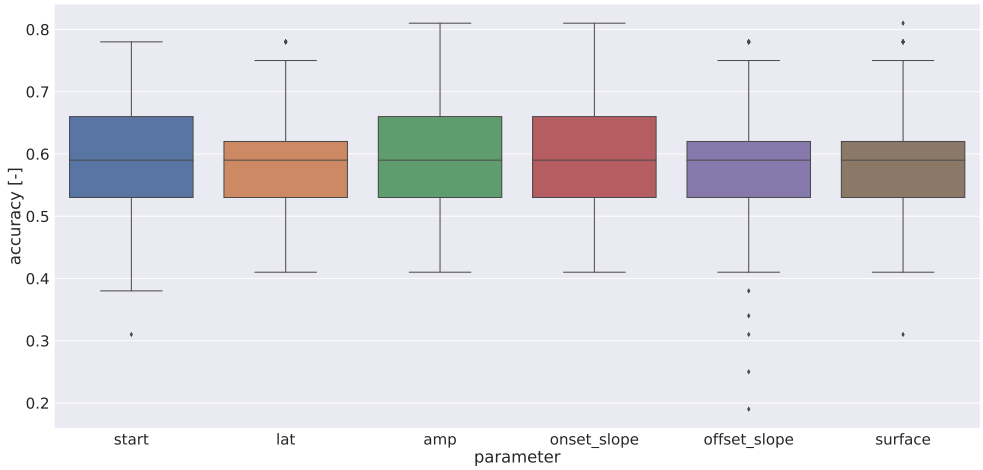


Rys. 6.1: Średnia dokładność każdego z modeli dla wszystkich wariantów.

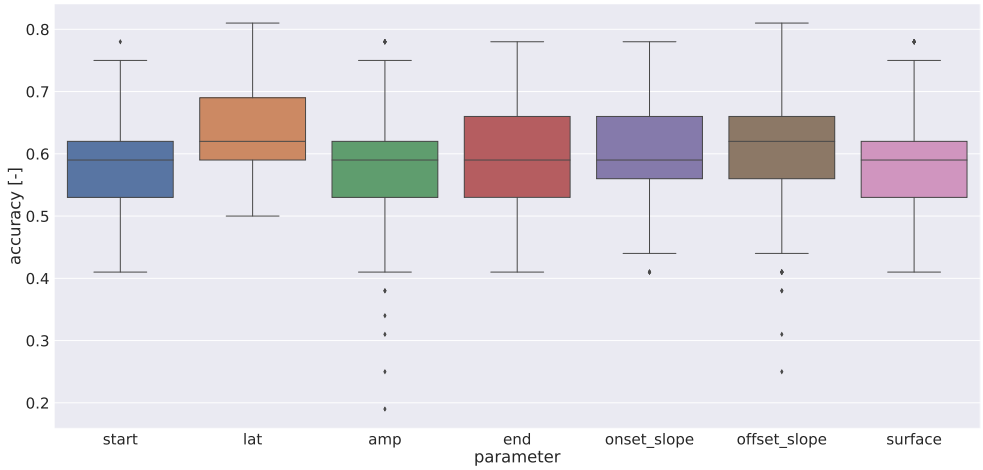
6.1.1. Lewa półkula, klasyfikacja rodzaju słuchania

Tab. 6.1: Dziesięć najlepszych wyników klasyfikacji rodzaju słuchania dla lewej półkuli mózgu.

parameters	SVM	RF	LR	KNN	SVCP	NB	DT
p50_amp, p50_onset_slope, n100_lat, n100_offset_slope	0.66	0.75	**0.81**	0.69	0.72	0.62	0.53
p50_start, p50_lat, p50_onset_slope, p50_surface, n100_lat, n100_amp	0.44	**0.81**	0.62	0.59	0.53	0.59	0.56
p50_start, n100_start, n100_lat, n100_amp, n100_offset_slope	0.59	0.69	0.66	**0.81**	0.53	0.66	0.62
p50_start, p50_amp, p50_surface, n100_lat, n100_onset_slope, n100_offset_slope	0.69	0.69	0.66	**0.81**	0.59	0.62	0.59
p50_amp, n100_lat, n100_amp, n100_end, n100_onset_slope	0.56	**0.81**	0.66	0.62	0.62	0.62	0.69
p50_amp, p50_onset_slope, p50_surface, n100_lat, n100_offset_slope	0.66	0.62	**0.81**	0.66	0.62	0.66	0.62
p50_amp, n100_start, n100_lat, n100_amp, n100_onset_slope, n100_offset_slope	0.66	**0.81**	0.62	0.59	0.56	0.59	0.5
p50_surface, n100_lat	0.69	**0.81**	0.66	0.53	0.69	0.66	0.69
p50_start, p50_lat, p50_surface, n100_lat, n100_amp, n100_offset_slope, n100_surface	0.53	**0.81**	0.66	0.66	0.59	0.69	0.62
p50_amp, p50_onset_slope, n100_lat, n100_end, n100_offset_slope	0.53	0.59	**0.78**	0.53	0.53	0.59	0.53



Rys. 6.2: Wykresy pudełkowe dokładności klasyfikacji dla parametrów załamka P50, dla sygnałów pochodzących z lewej półkuli.

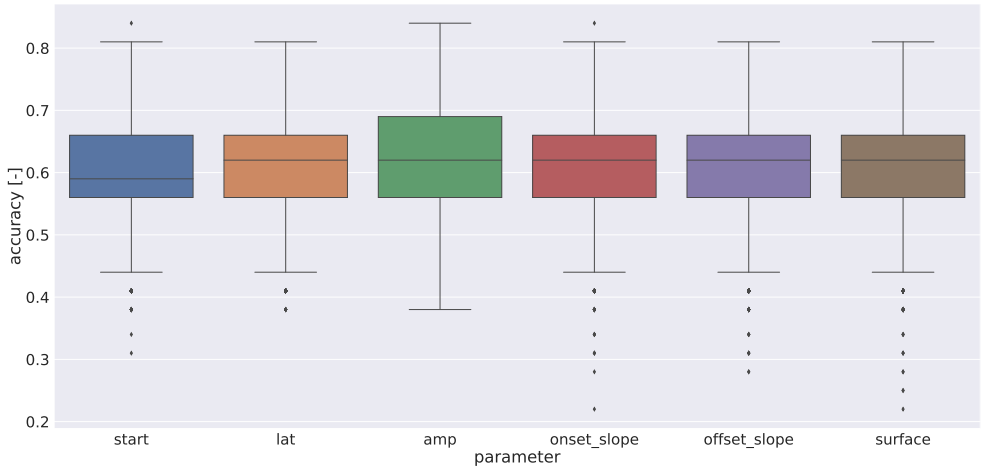


Rys. 6.3: Wykresy pudełkowe dokładności klasyfikacji dla parametrów załamka N100, dla sygnałów pochodzących z lewej półkuli.

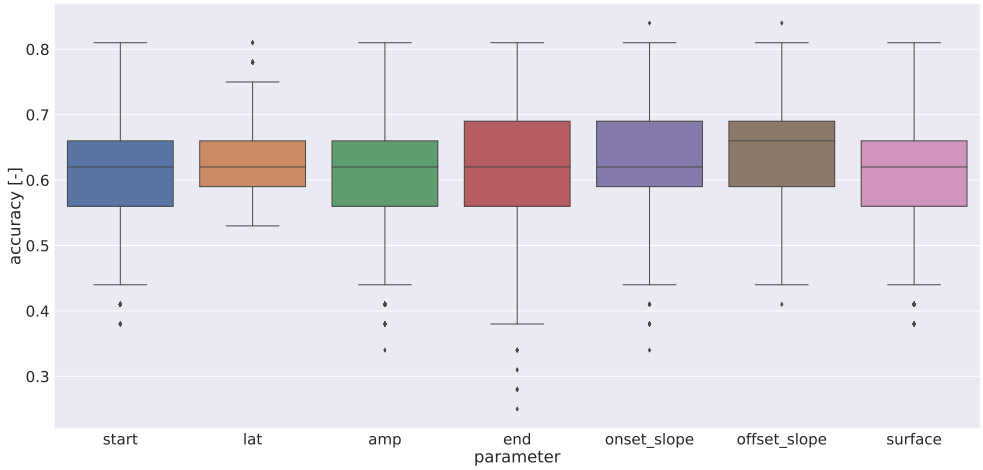
6.1.2. Prawa półkula, klasyfikacja rodzaju słuchania

Tab. 6.2: Dziesięć najlepszych wyników klasyfikacji rodzaju słuchania dla prawej półkuli mózgu.

parameters	SVM	RF	LR	KNN	SVCP	NB	DT
p50_start, p50_onset_slope, n100_start, n100_lat, n100_amp, n100_onset_slope, n100_offset_slope	0.59	0.66	0.62	0.59	0.66	0.59	**0.91**
p50_start, p50_amp, n100_start, n100_lat, n100_amp, n100_onset_slope, n100_offset_slope	0.53	0.56	0.66	0.62	0.5	0.62	**0.88**
p50_start, p50_onset_slope, p50_surface, n100_lat, n100_amp, n100_onset_slope, n100_offset_slope	0.62	0.62	0.62	0.56	0.66	0.56	**0.88**
p50_start, p50_surface, n100_start, n100_lat, n100_end, n100_onset_slope, n100_offset_slope	0.59	0.59	0.62	0.53	0.56	0.53	**0.88**
p50_start, p50_amp, p50_onset_slope, p50_offset_slope, p50_surface, n100_lat, n100_amp, n100_onset_slope, n100_offset_slope, n100_surface	0.66	0.59	0.62	0.5	0.56	0.56	**0.88**
p50_start, p50_amp, p50_onset_slope, p50_offset_slope, n100_lat, n100_amp, n100_onset_slope, n100_offset_slope, n100_surface	0.59	0.56	0.59	0.53	0.56	0.53	**0.84**
p50_start, p50_offset_slope, n100_start, n100_lat, n100_amp, n100_onset_slope, n100_offset_slope	0.56	0.59	0.62	0.59	0.47	0.59	**0.84**
p50_start, p50_amp, p50_surface, n100_lat, n100_amp, n100_onset_slope, n100_offset_slope	0.56	0.59	0.62	0.5	0.59	0.56	**0.84**
p50_start, p50_onset_slope, p50_offset_slope, n100_start, n100_lat, n100_onset_slope, n100_offset_slope	0.66	0.5	0.62	0.62	0.56	0.59	**0.84**
p50_start, p50_offset_slope, p50_surface, n100_lat, n100_amp, n100_onset_slope, n100_offset_slope	0.53	0.62	0.66	0.53	0.56	0.53	**0.84**



Rys. 6.4: Wykresy pudełkowe dokładności klasyfikacji dla parametrów załamka P50, dla sygnałów pochodzących z prawej półkuli.

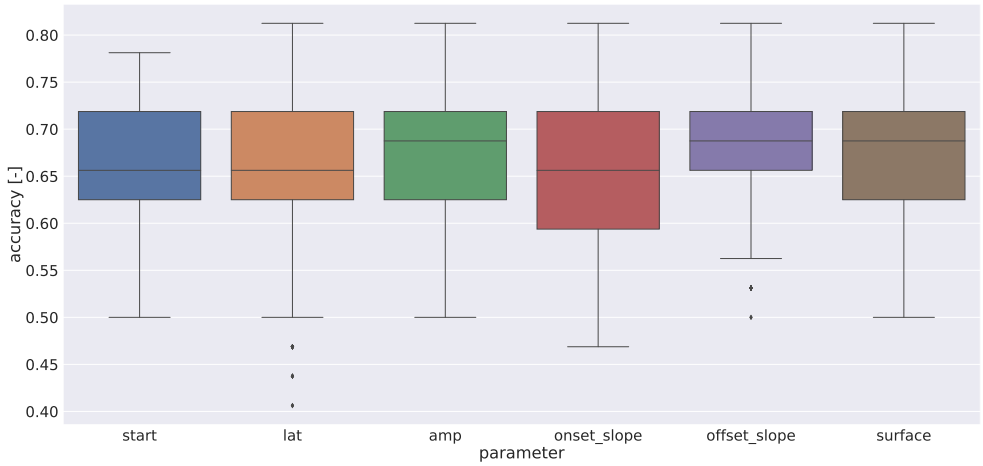


Rys. 6.5: Wykresy pudełkowe dokładności klasyfikacji dla parametrów załamka N100, dla sygnałów pochodzących z prawej półkuli.

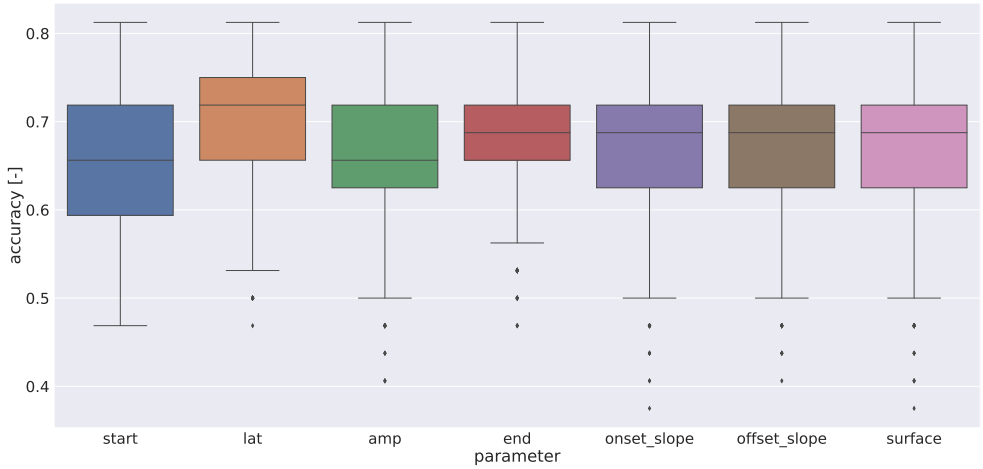
6.1.3. Słuchanie aktywne, klasyfikacja półkuli mózgu

Tab. 6.3: Dziesięć najlepszych wyników klasyfikacji półkuli mózgu dla słuchania aktywnego.

parameters	SVM	RF	LR	KNN	SVCP	NB	DT
p50_amp, n100_end, n100_onset_slope	0.69	**0.84**	0.69	0.59	0.62	0.59	0.53
p50_amp, p50_offset_slope, n100_start, n100_lat, n100_end, n100_onset_slope, n100_surface	0.66	**0.84**	0.69	0.59	0.59	0.69	0.44
p50_start, p50_lat, p50_amp, p50_surface, n100_lat, n100_amp, n100_end, n100_onset_slope, n100_offset_slope	0.69	0.72	0.69	0.53	0.5	**0.84**	0.5
p50_start, p50_lat, p50_onset_slope, p50_offset_slope, p50_surface, n100_lat, n100_end, n100_surface	0.72	0.69	0.75	0.53	0.5	**0.84**	0.56
p50_lat, p50_offset_slope, n100_start, n100_lat, n100_amp, n100_end, n100_surface	0.62	0.66	0.78	0.62	0.66	0.62	**0.84**
p50_lat, p50_offset_slope, n100_start, n100_lat, n100_amp, n100_offset_slope, n100_surface	0.66	0.62	0.69	0.72	0.59	0.56	**0.84**
p50_start, p50_lat, p50_amp, n100_lat, n100_end, n100_onset_slope, n100_surface	0.62	0.69	0.69	0.53	0.56	**0.81**	0.53
p50_start, p50_lat, p50_offset_slope, p50_surface, n100_amp, n100_end, n100_offset_slope	0.72	0.72	0.69	0.62	0.69	**0.81**	0.56
p50_start, p50_lat, p50_offset_slope, p50_surface, n100_lat, n100_end, n100_surface	0.75	0.69	0.75	0.69	0.53	**0.81**	0.53
p50_lat, n100_start, n100_onset_slope	0.53	0.66	0.56	0.5	0.5	0.59	**0.81**



Rys. 6.6: Wykresy pudełkowe dokładności klasyfikacji dla parametrów załamka P50, dla sygnałów słuchania aktywnego.

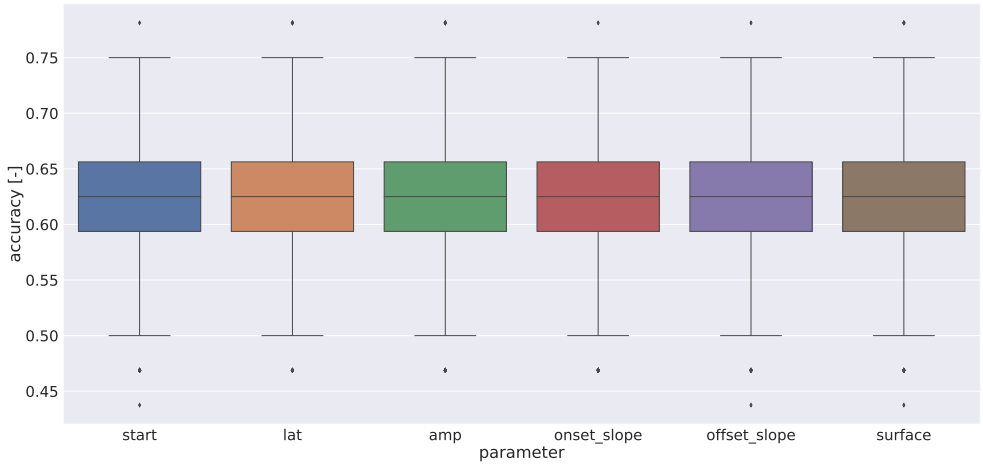


Rys. 6.7: Wykresy pudełkowe dokładności klasyfikacji dla parametrów z załamka N100 dla sygnałów słuchania aktywnego.

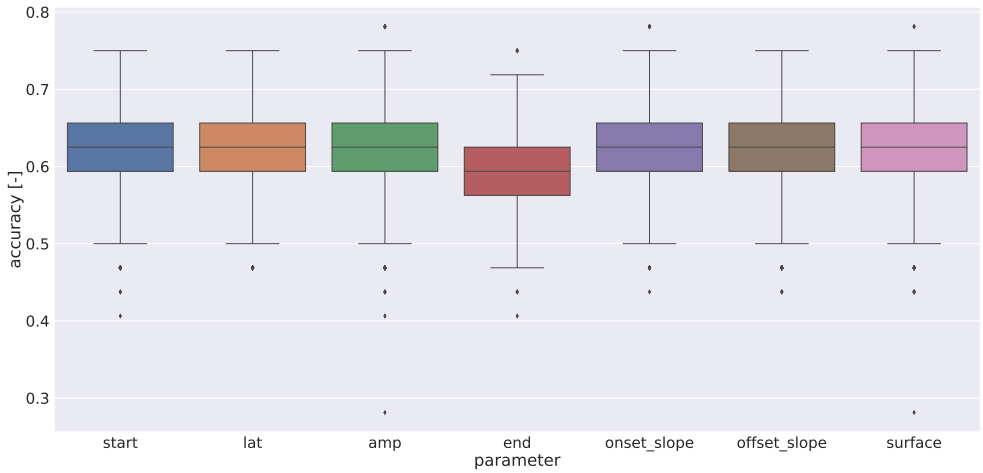
6.1.4. Słuchanie pasywne, klasyfikacja półkuli mózgu

Tab. 6.4: Dziesięć najlepszych wyników klasyfikacji półkuli mózgu dla słuchania pasywnego.

parameters	SVM	RF	LR	KNN	SVCP	NB	DT
p50_lat, p50_amp, n100_start, n100_lat, n100_onset_slope, n100_offset_slope	0.75	0.66	0.62	0.66	0.47	**0.84**	0.47
p50_amp, p50_offset_slope, p50_surface, n100_lat, n100_amp, n100_surface	0.62	0.69	0.62	0.62	**0.84**	0.59	0.47
p50_lat, p50_onset_slope, p50_offset_slope, n100_start, n100_lat, n100_onset_slope	0.69	0.62	0.72	0.75	0.59	**0.81**	0.44
p50_amp, p50_surface, n100_lat, n100_surface	0.59	0.62	0.59	0.69	**0.81**	0.59	0.62
p50_lat, p50_amp, n100_start, n100_lat	0.69	0.66	0.62	0.59	0.56	**0.81**	0.56
p50_start, p50_lat, p50_amp, p50_offset_slope, p50_surface, n100_start, n100_onset_slope	0.69	0.62	0.69	0.69	0.5	**0.81**	0.56
p50_lat, p50_amp, n100_start, n100_lat, n100_onset_slope	0.72	0.62	0.66	0.62	0.53	**0.81**	0.69
p50_lat, p50_amp, p50_onset_slope, n100_start, n100_lat, n100_onset_slope	0.62	0.62	0.66	0.72	0.53	**0.81**	0.56
p50_lat, p50_onset_slope, p50_surface, n100_start, n100_lat	0.59	0.62	0.66	0.69	0.5	**0.81**	0.5
p50_lat, p50_amp, p50_offset_slope, n100_start, n100_onset_slope	0.72	0.66	0.69	0.69	0.53	**0.81**	0.53



Rys. 6.8: Wykresy pudełkowe dokładności klasyfikacji dla parametrów załamka P50, dla sygnałów słuchania pasywnego.



Rys. 6.9: Wykresy pudełkowe dokładności klasyfikacji dla parametrów załamka N100, dla sygnałów słuchania pasywnego.

6.2. Porównanie szczególne

Wyniki przedstawione w tej części będą służyć ocenie, czy kategoryzacja sygnałów świergotowych z uwagi na kierunek ich modulacji częstotliwościowej zachodzi w jednej półkuli mózgu bardziej intensywnie niż w drugiej. W poniższych tabelach zestawiono wyniki klasyfikacji rodzaju słuchania (aktywne *vs* pasywne) dla obu półkul (tabela 6.5) oraz klasyfikacji półkuli (lewa *vs* prawa) dla obu rodzajów słuchania (tabela 6.6). Dla każdego z siedmiu modeli wybrane zostały te same zbiory parametrów, a mianowicie te, które zawierają w sobie wyłącznie parametry opisujące załamek N100, co zgodnie z równaniem (5.1) przekłada się na $2^7 - 1 = 127$ możliwości. Wybór tego załamka został podyktowany jego istotnością przy analizie sygnałów pochodzących z kory słuchowej mózgu.

Tab. 6.5: Porównanie klasyfikacji rodzaju słuchania dla obu półkul mózgu, dla załamka N100. Dla danego modelu kolorem zielonym zaznaczono większą z wartości dokładności klasyfikacji.

model	left	right
SVM	0.57	**0.6**
RF	**0.57**	**0.57**
LR	0.61	**0.64**
KNN	0.54	**0.58**
SVCP	0.56	**0.61**
NB	**0.61**	0.58
DT	0.52	**0.56**

Tab. 6.6: Porównanie klasyfikacji półkuli mózgu dla obu rodzajów słuchania, dla załamka N100. Dla danego modelu kolorem zielonym zaznaczono większą z wartości dokładności klasyfikacji.

model	active	passive
SVM	0.52	**0.54**
RF	**0.59**	0.45
LR	**0.57**	0.56
KNN	**0.52**	0.49
SVCP	**0.52**	0.49
NB	0.54	**0.59**
DT	**0.58**	0.43

Rozdział 7

Wnioski

Dane, które analizowano w pracy, to dane MEG. Są to sygnały pochodzenia neurologicznego, zatem cechują się wysoką trudnością pracy z nimi. Grupa badawcza obejmowała 16 osób, a badanie polegało na przeprowadzeniu próby słuchania aktywnego oraz pasywnego dla wszystkich badanych. Pojedyncze badanie rejestrowane było przez 148-kanałowy magnetoencefalograf. Do analizy tych sygnałów wykorzystano uczenie maszynowe. Jeden z problemów, jakie napotkano, to mały rozmiar grupy badawczej. Rozwiązanie tego problemu to wykorzystanie modeli nadających się do mniejszych zbiorów danych oraz skorzystanie z metody leave-one-out przy ocenie ich dokładności. Modele, z których skorzystano, to regresja logistyczna (LR), drzewa decyzyjne (DT), lasy losowe (RF), naiwny klasyfikator bayesowski (NB), k-najbliższych sąsiadów (KNN), maszyna wektorów nośnych (SVM) z jądrem RBF oraz maszyna wektorów nośnych z jądrem wielomianowym (SVCP).

Opracowano algorytm, który z każdego sygnału wydobywa zbiór skalarnych parametrów, których następnie używano do uczenia modeli. Dane klasyfikowano w czterech różnych wariantach. Pierwszy wariant to klasyfikacja rodzaju słuchania dla sygnałów pochodzących z lewej półkuli. Analizując wykresy 6.2 oraz 6.3 zauważono, iż parametr `n100_lat` znacząco odbiega od pozostałych (na plus). Na tej podstawie stwierdzono, iż główny wpływ na klasyfikację rodzaju słuchania w lewej półkuli ma latencja załamka N100. Oznacza to, iż parametr ten jest czynnikiem, który najlepiej różnicuje słuchanie aktywne *vs* pasywne dla lewej półkuli.

Drugim wariantem jest klasyfikacja rodzaju słuchania dla sygnałów pochodzących z prawej półkuli. Po analizie wykresów 6.4 oraz 6.5 stwierdzono, że żaden z parametrów nie odbiega znacząco od pozostałych. Mimo to, na podstawie porównania wykresów pudełkowych lewej oraz prawej półkuli stwierdzono, że to prawa półkula jest mocniej zaangażowana w słuchanie aktywne. Stwierdzenie to poparte jest faktem, iż dla niemalże wszystkich parametrów sygnału wyniki klasyfikacji dla prawej półkuli były wyższe niż wyniki klasyfikacji dla lewej półkuli. Potwierdzają to również dane przedstawione w tabelach 6.1 oraz 6.2, gdzie nawet najsłabszy (z dziesięciu najlepszych) model wyuczony za pomocą danych z prawej półkuli okazał się być dokładniejszy niż najlepszy model wyuczony za pomocą danych z lewej półkuli.

Kolejnym wariantem jest klasyfikacja źródła sygnału dla słuchania aktywnego. Analizując wykresy 6.6 oraz 6.9 zauważono, iż parametr `n100_lat` znacząco odbiega od pozostałych (na plus). Oznacza to, iż w przypadku słuchania aktywnego latencja załamka N100 dla jednej półkuli znacznie się różni od latencji załamka N100 drugiej półkuli.

Ostatnim wariantem jest klasyfikacja źródła sygnału w przypadku słuchania pasywnego. Wykresy pudełkowe niemalże wszystkich parametrów (6.8, 6.9) są mocno do siebie zbliżone. Jedynym odbiegającym parametrem jest `n100_end`, którego wartości są niższe od pozostałych. Oznacza to, iż czas końca załamka N100 jest mało wnoszącą informacją w tym przypadku klasyfikacji.

Porównując wykresy pudełkowe słuchania aktywnego oraz pasywnego zauważono, iż dla wszystkich parametrów lepsze wyniki są uzyskiwane dla danych przynależących do słuchania aktywnego. Oznacza to, że słuchanie aktywne skutkuje lepszymi rezultatami przy dyskryminacji źródła sygnału niż słuchanie pasywne. Można zatem stwierdzić, że jedna półkula jest bardziej odpowiedzialna za słuchanie aktywne niż druga.

W celu dokładniejszego porównania pomiędzy półkulami oraz rodzajami słuchania stworzono dwie tabele, a mianowicie 6.5 oraz 6.6. Dane przedstawione w pierwszej z nich potwierdzają, że sygnały pochodzące z prawej półkuli dają lepsze rezultaty przy klasyfikacji rodzaju słuchania niż sygnały pochodzące z lewej półkuli. Dane przedstawione w drugiej tabeli potwierdzają, iż słuchanie aktywne zapewnia lepsze rezultaty, gdy wymagana jest klasyfikacja źródła sygnału. Łącząc te fakty z doniesieniami z prac [1, 2, 3] stwierdzono, iż prawa półkula odpowiada za słuchanie aktywne mocniej niż lewa.

Dane znajdujące się w tabelach 6.1, 6.2, 6.3 oraz 6.4, przedstawiających najlepsze modele dla konkretnych wariantów klasyfikacyjnych, pokazują, że pomimo małych rozmiarów grupy badawczej, trudności związanych z analizą sygnałów MEG oraz zbliżonego charakteru klas możliwe było stworzenie modeli uczenia maszynowego kategoryzujących dane z wysoką dokładnością.

Dodatek A

Płyta CD

Dodatkiem do pracy jest płyta CD zawierająca implementację wszystkich metod, które zostały wykorzystane do analizy omawianych sygnałów. Metody te służą m.in. do wydobycia parametrów z sygnałów, wstępnego przetwarzania danych oraz tworzenia i wyuczania modeli uczenia maszynowego.

Literatura

[1] A. Brechmann and H. Scheich, "Hemispheric shifts of sound representation in auditory cortex with conceptual listening," *Cerebral Cortex*, vol. 15, no. 5, pp. 578–587, 2005. https://doi.org/10.1093/cercor/bhh159.

[2] F. W. Ohl and H. Scheich, "Learning-induced plasticity in animal and human auditory cortex," *Current Opinion in Neurobiology*, vol. 15, pp. 470–477, 2005. https://www.sciencedirect.com/science/article/pii/S0959438805001030.

[3] R. König, C. Sielużycki, C. Simserides, P. Heil, and H. Scheich, "Effects of the task of categorizing FM direction on auditory evoked magnetic fields in the human auditory cortex," *Brain Research*, vol. 1220, pp. 102–117, 2008. http://www.sciencedirect.com/science/article/pii/S0006899308005453.

[4] S. Supek and C. J. Aine, eds., *Magnetoencephalography: From Signals to Dynamic Cortical Networks*. Springer, 2014.

[5] D. L. Schomer and F. H. Lopes da Silva, eds., *Niedermeyer's Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. Oxford University Press, 2017.

[6] S. P. Singh, "Magnetoencephalography: Basic principles," *Annals of Indian Academy of Neurology*, vol. 17, no. 5, pp. 107–112, 2014. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4001219.

[7] A. Hajizadeh, A. Matysiak, P. J. C. May, and R. König, "Explaining event-related fields by a mechanistic model encapsulating the anatomical structure of auditory cortex," *Biological Cybernetics*, vol. 113, pp. 321–345, 2019. https://link.springer.com/article/10.1007/s00422-019-00795-9.

[8] K. Reitz, "Scientific Applications." https://docs.python-guide.org/scenarios/scientific.

[9] S. Raschka, "What is the main difference between TensorFlow and scikit-learn?." https://sebastianraschka.com/faq/docs/tensorflow-vs-scikitlearn.html.

[10] M. Mohammed, M. B. Khan, and E. B. M. Bashier, *Machine Learning: Algorithms and Applications*. 6000 Broken Sound Parkway NW, Suite 300: CRC Press, 2016. https://www.researchgate.net/publication/303806260_Machine_Learning_Algorithms_and_Applications.

[11] C. Sundar, M. Chitradevi, and G. Geetharamani, "Classification of cardiotocogram data using neural network based machine learning technique," *International Journal of Computer Applications*, vol. 47, pp. 19–25, 2012. https://www.researchgate.net/publication/258651652_Classification_of_Cardiotocogram_Data_using_Neural_Network_based_Machine_Learning_Technique.

[12] J. Brownlee, "4 Types of Classification Tasks in Machine Learning," 4 2020. https://machinelearningmastery.com/types-of-classification-in-machine-learning.

[13] S. Asiri, "Machine Learning Classifiers," 6 2018. https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623.

[14] S. Raschka and V. Mirjalili, *Python Machine Learning*. Birmingham, UK: Packt Publishing, 2019. https://www.packtpub.com/product/python-machine-learning-third-edition/9781789955750.

[15] A. M. Zoubir and D. R. Iskander, "Bootstrap methods and applications: A tutorial for the signal processing practitioner," *IEEE Signal Processing Magazine*, pp. 10–19, July 2007. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4286560.

[16] S. Fan, "Understanding the mathematics behind Naive Bayes," 6 2018. https://shuzhanfan.github.io/2018/06/understanding-mathematics-behind-naive-bayes.

[17] S. Raschka, "Naive Bayes and text classification I — Introduction and Theory," *arXiv e-prints*, 10 2014. https://ui.adsabs.harvard.edu/abs/2014arXiv1410.5329R.

[18] MLMath.io, "Math behind SVM (Support Vector Machine)," 2 2019. https://medium.com/@ankitnitjsr13/math-behind-svm-support-vector-machine-864e58977fdb.

[19] A. Horzyk, "Metody inżynierii wiedzy: Maszyna wektorów nośnych." http://home.agh.edu.pl/~horzyk/lectures/miw/MIW-SVM.pdf.

[20] P. Winston, "Artificial Intelligence: Support Vector Machines," 2010. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010.

[21] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, Massachusetts, London, England: The MIT Press, 2012.

[22] A. Smola and S. V. N. Vishwanathan, *Introduction to Machine Learning*. New York, NY, USA: Cambridge University Press, 2008.

[23] Y. Liang, "Machine learning basics, Lecture 5: SVM II," 2 2016. https://www.cs.princeton.edu/courses/archive/spring16/cos495.

[24] M. Mota, "JupyterLab is the data science UI we have been looking for," 10 2017. https://towardsdatascience.com/jupyterlab-you-should-try-this-data-science-ui-for-jupyter-right-now-a799f8914bb3.

[25] "Titanic: Machine Learning from Disaster." https://www.kaggle.com/c/titanic/data?select=train.csv.

Spis rysunków

Spis tabel

Spis listingów