Politechnika Wrocławska
Wydział Podstawowych Problemów Techniki

Jezyki programowania do zastosowań biomedycznych

Klasyfikator wypadków

Autorzy:
Zakia Shefa #264476

Prowadzacy: dr hab. inż. Witold Dyrka

Spis treści

1 Wstep

Celem pracy jest analiza, wybór oraz implementacja modelu klasyfikacji do wykrywania upadków u osób starszych z wykorzystaniem biosensorów wbudowanych w smartfony. Praca obejmuje przeglad metod uczenia maszynowego, przygotowanie zbioru danych na podstawie sygnałów z akcelerometru i żyroskopu, a nastepnie trenowanie modelu klasyfikacyjnego. Finalnym etapem jest integracja modelu z aplikacja mobilna działajaca na systemie Android oraz testowanie skuteczności detekcji upadków.

2 Zbiór danych

Do realizacji projektu wykorzystano zbiór danych dostepny pod adresem: https://www.kaggle.com/datasets/kmknation/mobifall-dataset-v20/data.

Zbiór Mobifall zawiera dane sensoryczne pozyskane z akcelerometru, żyroskopu oraz czujnika orientacji. Dane zostały zebrane od 24 uczestników w wieku od 30 do 60 lat. Każda osoba wykonała zestaw określonych czynności, obejmujacych zarówno codzienne aktywności, jak i symulowane upadki.

W zbiorze znajduja sie także dodatkowe informacje demograficzne, takie jak: wzrost, waga, płeć oraz wiek badanych.

Zbior jest podzielony na zestaw nastepujacych aktywnosci:

Codzienne aktywności (Activities of Daily Living)

Table 1: Zestaw codziennych aktywności

ID	Kod	Opis
1	STD	Stanie z delikatnymi ruchami
2	WAL	Normalne chodzenie
3	JOG	Bieganie (jogging)
4	JUM	Skakanie ciagłe
5	STU	Wchodzenie po schodach (10 stopni)
6	STN	Schodzenie ze schodów (10 stopni)
7	SCH	Siadanie na krześle
8	CSI	Wsiadanie do samochodu
9	CSO	Wysiadanie z samochodu

Upadki (Falls)

Table 2: Zestaw symulowanych upadków

ID	Kod	Opis
10	FOL	Upadek do przodu ze stania, z amortyzacja rekami
11	FKL	Upadek do przodu ze stania, pierwsze uderzenie kolanami
12	BSC	Upadek do tyłu podczas próby siadania na krześle
13	SDL	Upadek na bok ze stania, z ugieciem nóg

3 Wstepne przetwarzanie danych

Pierwszym krokiem w przetwarzaniu danych było ich odpowiednie przygotowanie do dalszej analizy. Ze wzgledu na to, że dane zebrano w oddzielnych plikach dla każdego sensora, konieczne było ich połaczenie w jeden spójny zbiór, co znaczaco ułatwiło dalsze etapy pracy.

W trakcie integracji danych uwzgledniono różnice w czestotliwości próbkowania pomiedzy akcelerometrem a żyroskopem. Akcelerometr (ACC) rejestrował dane z czestotliwościa 50 Hz, natomiast żyroskop (GYRO) – z dużo wyższa, bo aż 2000 Hz. W zwiazku z tym niektóre próbki żyroskopu zostały pominiete, a wartości z obu sensorów zostały połaczone na podstawie najbliższych w czasie znaczników (timestampów).

Table 3: Czestotliwość próbkowania sensorów

Sensor	Czestotliwość próbkowania
Akcelerometr (ACC)	0,02 s (20 ms) / 50 Hz
Żyroskop (GYRO)	0,0005 s (0,5 ms) / 2000 Hz

Na podstawie połaczonych danych wygenerowano spójne pliki CSV z nastepujacymi nagłówkami:

`timestamp, x_acc, y_acc, z_acc, x_gyro, y_gyro, z_gyro, fall, activity, subject_id, age, height, weight, gender`

4 Analiza danych

W tej cześci przedstawiono przykładowe wykresy sygnałów sensorycznych (akcelerometru i żyroskopu), zarejestrowanych podczas wybranych aktywności codziennych oraz upadków. Dane przedstawiono w postaci wektorów wartości w funkcji czasu. Każdy wykres odpowiada jednemu pacjentowi. Poniżej znajduja sie wizualizacje dla trzech różnych przypadków dla każdej aktywności i upadku.

4.1 Aktywności codzienne

CSO – Car Step Out (Wysiadanie z samochodu)

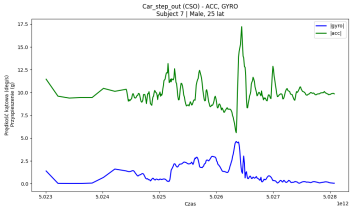

Figure 1: CSO – pacjent 7

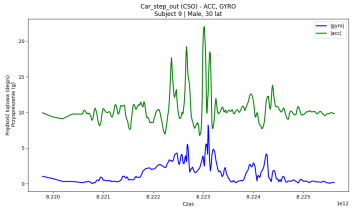

Figure 2: CSO – pacjent 9

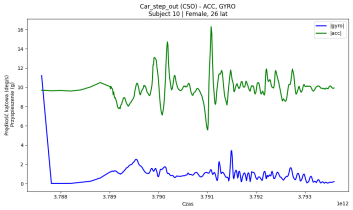

Figure 3: CSO – pacjent 10

JUM – Jumping (Skakanie)

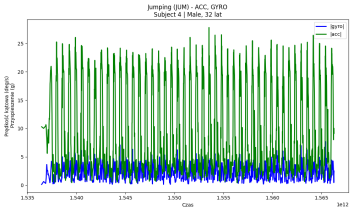

Figure 4: JUM – pacjent 4

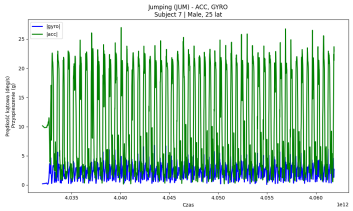

Figure 5: JUM – pacjent 7

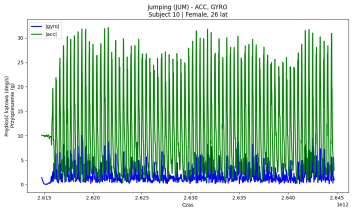

Figure 6: JUM – pacjent 10

SCH – Sit Chair (Siadanie na krześle)

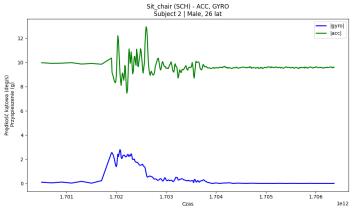

Figure 7: SCH – pacjent 2

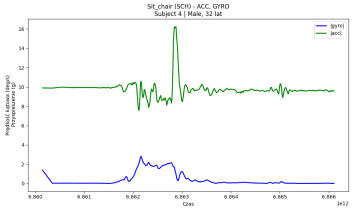

Figure 8: SCH – pacjent 4

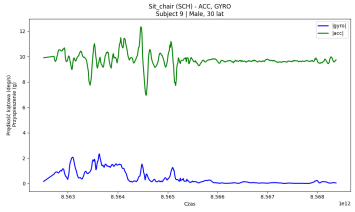

Figure 9: SCH – pacjent 9

Charakterystycznym w wynikach z akceleromtreu wynikach jest to ze nawet po najwiekszym ruchu sensora dane nie spadaja szybko do stabilnego ruchu tylko zazwuczaj do mniejszego dak jak np na Wykresach SCH 7 8 9. Podobnie dane zyroskopu sa chaotyczne i widocznie pokazujace ciag aktywnosci.

4.2 Upadki

FOL – Forward Lying (Upadek do przodu z amortyzacja rekami)

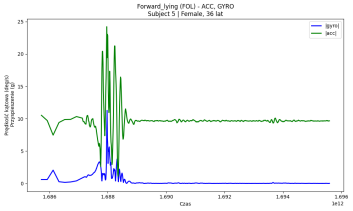

Figure 10: FOL – pacjent 5

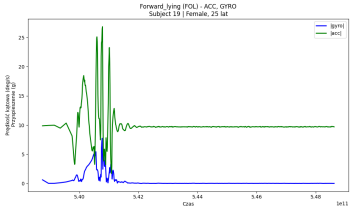

Figure 11: FOL – pacjent 19

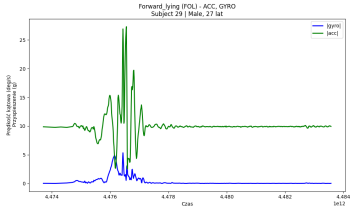

Figure 12: FOL – pacjent 29

FKL – Front Knees Lying (Upadek do przodu z pierwszym uderzeniem kolan)

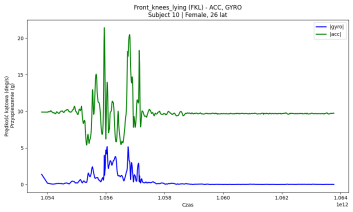

Figure 13: FKL – pacjent 10

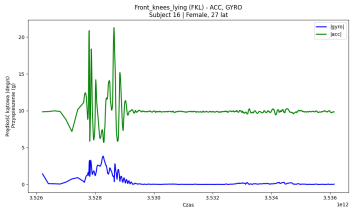

Figure 14: FKL – pacjent 16

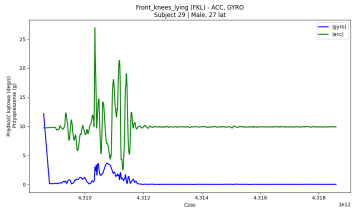

Figure 15: FKL – pacjent 29

SDL – Sideward Lying (Upadek na bok)

PRakrycznie wszystkie wyniki przedstawiaja jeden duzy skok a nastepnie ’ wyciszenie”. podobnie w Przypadku zyroskopu zauwazalne duze skok danych a nastepnie wyrownaie

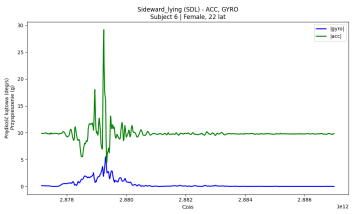

Figure 16: SDL – pacjent 6

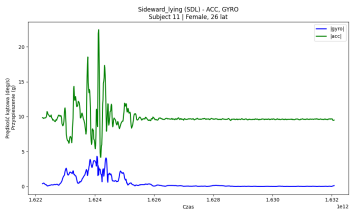

Figure 17: SDL – pacjent 11

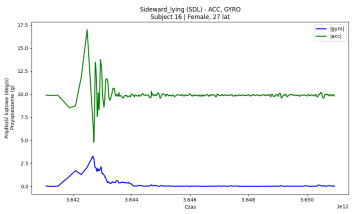

Figure 18: SDL – pacjent 16

5 Ekstrakcja cech

W celu umożliwienia skutecznej klasyfikacji zdarzeń, konieczne było przetworzenie surowych danych z czujników akcelerometru oraz żyroskopu na reprezentatywny zbiór cech numerycznych.Każda próbka zawierała pomiary przyspieszenia (`x_acc`, `y_acc`, `z_acc`) oraz predkości katowej (`x_gyro`, `y_gyro`, `z_gyro`).

5.1 Obliczanie wektorowych wielkości

Pierwszym krokiem było utworzenie pośrednich zmiennych reprezentujacych wielkość wektorowa przyspieszenia oraz predkości katowej:

- **Moduł przyspieszenia (acc_magnitude)**:

$$acc\_magnitude = \sqrt{x_{acc}^2 + y_{acc}^2 + z_{acc}^2}$$

  Określa ogólna siłe działania sił inercyjnych i odzwierciedla intensywność ruchu.

- **Moduł predkości katowej (gyro_magnitude)**:

$$gyro\_magnitude = \sqrt{x_{gyro}^2 + y_{gyro}^2 + z_{gyro}^2}$$

  Umożliwia ocene rotacji urzadzenia w przestrzeni 3D.

Nastepnie dla każdej z powyższych wielkości obliczono różnice pomiedzy kolejnymi próbkami, co pozwala uchwycić nagłe zmiany (cechy dynamiczne):

- `acc_diff` = różnica pomiedzy kolejnymi wartościami `acc_magnitude`,
- `gyro_diff` = różnica pomiedzy kolejnymi wartościami `gyro_magnitude`.

5.2 Okno czasowe i statystyki

W celu ujednolicenia długości sekwencji czasowych oraz umożliwienia zastosowania klasyfikatorów klasycznych, dane zostały podzielone na nachodzace na siebie okna czasowe (tzw. *sliding window*):

- długość okna: 100 próbek (co odpowiada około 2 sekundom),
- przesuniecie okna: 10 próbek (aby uzyskać płynność analizy).

Dla każdej sekwencji w oknie obliczane sa nastepujace cechy statystyczne:

- **Średnia wartość modułu przyspieszenia i predkości katowej** (`acc_magnitude_mean`, `gyro_magnitude_mean`) – uśredniona siła ruchu w danym oknie,
- **Średnia różnic przyspieszenia i predkości katowej** (`acc_diff_mean`, `gyro_diff_mean`) – zmienność sygnału,
- **Odchylenie standardowe modułów** (`acc_magnitude_std`, `gyro_magnitude_std`) – miara rozrzutu wartości, odzwierciedlajaca nieregularność ruchu.

Wprowadzono również atrybuty demograficzne:

- `age`, `height`, `weight` – wiek, wzrost i waga użytkownika (wartości maksymalne w oknie),
- `gender` – zakodowana liczbowo płeć (0 – mężczyzna, 1 – kobieta).

Cechy demograficzne te moga mieć istotne znaczenie przy różnicowaniu wzorców ruchowych różnych użytkowników, np. ze wzgledu na mase ciała lub sposób poruszania sie.

5.3 Zmienna celu

Jako zmienna decyzyjna (*target*) wykorzystano binarna etykiete `fall`, wskazujaca czy w danym oknie czasowym wystapił upadek (wartość maksymalna tej kolumny w oknie czasowym – 1 oznacza co najmniej jeden upadek, 0 – brak).

6 Podział na dane treningowe oraz testowe

W celu oceny skuteczności działania klasyfikatora, przeprowadzono testy z wykorzystaniem dwóch różnych podejść do podziału zbioru danych: podziału losowego oraz podziału na podstawie identyfikatorów osób badanych. Oba podejścia pozwalaja ocenić model w różnych kontekstach i weryfikuja jego zdolność do generalizacji.

6.1 Podział losowy (Random Split)

W pierwszym podejściu zastosowano klasyczny podział zbioru danych na cześć treningowa i testowa w stosunku 70% do 30%. Dane były losowo dzielone przy użyciu funkcji `train_test_split()` z biblioteki `scikit-learn`:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)
```

Wybór stałej wartości `random_state=42` gwarantuje powtarzalność wyników. To podejście jest korzystne do szybkiej oceny ogólnej skuteczności klasyfikatora na przekrojowym zbiorze danych.

Należy jednak pamietać, że dane nie sa równomiernie rozłożone pomiedzy osoby – nie każdy badany dostarczył tyle samo próbek. W zwiazku z tym, losowy podział może prowadzić do optymistycznych wyników, ponieważ przykłady z tej samej osoby moga trafić zarówno do zbioru treningowego, jak i testowego.

6.2 Podział na osoby (Split by Person)

Drugie podejście opiera sie na bardziej realistycznym scenariuszu: zakłada ono, że klasyfikator bedzie używany przez osobe, która wcześniej nie wystepowała w danych treningowych. W tym celu dokonano podziału danych na podstawie identyfikatora osoby (`id`).

Zbiór testowy zawierał dane pochodzace od wybranej podgrupy osób (np. pierwsze 30% unikalnych `id`), a pozostałe osoby trafiły do zbioru treningowego:

```
unique_subjects = features_df['subject_id'].unique()

# Mieszamy osoby losowo
np.random.shuffle(unique_subjects)

# Wyznaczamy ile osób ma być w zbiorze testowym
n_testowych = int(len(unique_subjects) * procent_testowych)

test_subjects = unique_subjects[:n_testowych]
train_subjects = unique_subjects[n_testowych:]

# Podzia␣l danych na podstawie subject_id
train_df = features_df[features_df['subject_id'].isin(train_subjects)]
test_df = features_df[features_df['subject_id'].isin(test_subjects)]
```

Takie podejście jest znacznie bardziej wymagajace dla modelu, ponieważ musi on uogólnić wiedze zdobywana na danych pewnych osób i zastosować ja do zupełnie nowych użytkowników. Testowanie z wykorzystaniem tego wariantu pozwala lepiej ocenić, jak model poradzi sobie w rzeczywistym zastosowaniu – np. jako cześć aplikacji mobilnej używanej przez nowych użytkowników.

7 Budowa Modelu

Do eksperymentów wybrano trzy algorytmy: **K-Nearest Neighbors (KNN)**, **Random Forest (RF)** oraz **SVC**. Wybór tych metod oparty był na ich skuteczności w problemach klasyfikacyjnych oraz zdolności do pracy z wielowymiarowymi danymi czasowymi.

7.1 K-Nearest Neighbors (KNN)

KNN to klasyfikator oparty na miarze odległości (np. euklidesowej) pomiedzy punktami w przestrzeni cech. Klasyfikacja nowej próbki odbywa sie poprzez analize k najbliższych sasiadów w zbiorze treningowym — próbka zostaje przypisana do klasy, która dominuje wśród tych sasiadów.

Główne cechy:

- Brak założeń o rozkładzie danych – metoda nieparametryczna.
- Naturalna obsługa danych wielocechowych – istotne dla zbioru danych zawierajacego cechy czasowe i demograficzne.
- Wysoka interpretowalność – klasyfikacja oparta na intuicyjnym założeniu „bliskości".

W eksperymentach stosowano różne wartości parametru k (liczby sasiadów), z których najlepsze wyniki osiagnieto dla $k = 3$ oraz $k = 5$. Dla poprawy jakości działania, dane zostały uprzednio standaryzowane (`StandardScaler` z biblioteki `scikit-learn`), co jest konieczne ze wzgledu na wrażliwość KNN na różne skale cech.

7.2 Random Forest (RF)

Random Forest to model zespołowy (ensemble learning), który składa sie z wielu drzew decyzyjnych tworzonych na losowych podzbiorach danych i cech. Końcowa predykcja modelu powstaje w wyniku głosowania wiekszościowego.

Główne cechy:

- Wysoka odporność na przeuczenie dzieki losowości i agregacji wyników.
- Skuteczność w pracy z dużymi zbiorami danych i duża liczba cech.
- Możliwość interpretacji modelu poprzez analize ważności cech.

Random Forest został wybrany jako bardziej zaawansowany klasyfikator do porównania z KNN, ze wzgledu na swoja zdolność do modelowania nieliniowych zależności oraz odporność na szum w danych.

7.3 Support Vector Machines (SVM)

Algorytm **Support Vector Machines (SVC)** to jedna z najpopularniejszych metod klasyfikacyjnych, szczególnie skuteczna w przypadku danych o wysokiej wymiarowości. Jego głównym celem jest znalezienie hiperpowierzchni (tzw. hiperpłaszczyzny), która najlepiej rozdziela próbki należace do różnych klas, maksymalizujac margines miedzy nimi.

Główne cechy:

- Skuteczność w problemach klasyfikacyjnych, zwłaszcza przy ograniczonej liczbie próbek i dużej liczbie cech.
- Możliwość stosowania funkcji jadra (kernel), co pozwala na modelowanie nieliniowych granic decyzyjnych.
- Wysoka ogólność i elastyczność w dostosowywaniu do różnych typów danych.

8 Eksperyment

8.1 Podział losowy

Podział losowy polegał na odseparowaniu przypadków aktywności do zbioru testowego niezależnie od osób. Testy przeprowadzono dla długości okna 150 oraz przesuniecia okna o 50 próbek.

8.1.1 KNN

	precision	recall	f1-score	support
0	0.96	0.98	0.97	4532
1	0.93	0.84	0.88	1264
accuracy			0.95	5796

Model KNN osiagnał bardzo dobre wyniki, zwłaszcza dla klasy codziennych aktywności (klasa 0), z wysoka precyzja i czułościa. Dla klasy upadków (klasa 1), choć recall był nieco niższy (0.84), model skutecznie identyfikował wiekszość przypadków.

8.1.2 RF

	precision	recall	f1-score	support
0	0.98	0.98	0.98	4532
1	0.94	0.92	0.93	1264
accuracy			0.97	5796

Random Forest okazał sie najlepszym modelem w tym scenariuszu, osiagajac najwyższa dokładność (97%). Wysokie wartości precyzji i czułości dla obu klas wskazuja na duża skuteczność i równowage w rozpoznawaniu zarówno upadków, jak i codziennych aktywności.

8.1.3 SVM

	precision	recall	f1-score	support
0	0.93	0.97	0.95	4532
1	0.88	0.75	0.81	1264
accuracy			0.92	5796

Model SVM uzyskał nieco niższa skuteczność niż KNN i RF, zwłaszcza w wykrywaniu klasy 1, gdzie recall wyniósł 0.75. Choć dokładność ogólna nadal była wysoka (92%), SVM gorzej radził sobie z detekcja rzadkich przypadków upadków.

8.2 Podział na osoby

W tym podejściu osoby testowe zostały całkowicie wyłaczone ze zbioru treningowego, co lepiej odwzorowuje realne zastosowanie modelu. Do testów wybrano 20% osób majacych zarówno aktywności codzienne, jak i upadki.

8.2.1 KNN

	precision	recall	f1-score	support
0	0.64	0.24	0.35	1726
1	0.24	0.63	0.35	646
accuracy			0.35	2372

Powyższy wynik pokazuje, że model KNN rozpoznaje tylko niewielka cześć klas. Czułość (recall) dla upadków (klasa 1) jest wzglednie wysoka (0.63), ale precyzja bardzo niska (0.24), co oznacza duża liczbe fałszywych alarmów.

8.2.2 RF

Najlepszy przypadek:

precision	recall	f1-score	support
0 0.94	0.89	0.91	1618
1 0.75	0.85	0.79	629
accuracy		0.88	2247

W najbardziej korzystnym przypadku RF osiagnał bardzo dobre wyniki – dokładność 88% i wysokie wartości f1-score dla obu klas. To pokazuje, że model może skutecznie generalizować, nawet w warunkach testowania na nowych osobach.

Gorszy przypadek:

precision	recall	f1-score	support
0 0.95	0.42	0.58	1694
1 0.39	0.94	0.55	654
accuracy		0.57	2348

W mniej korzystnym scenariuszu model nadal dobrze wykrywał upadki (recall = 0.94), ale miał bardzo niska skuteczność w rozpoznawaniu aktywności codziennych, co prowadziło do wielu fałszywych alarmów.

8.2.3 SVM

W celu poprawy detekcji rzadkiej klasy upadków, zastosowano zwiekszona wage tej klasy podczas trenowania modelu.

Wagi klasy 1: 5

precision	recall	f1-score	support
0 0.81	0.45	0.58	1726
1 0.33	0.72	0.45	646
accuracy		0.53	2372

Model lepiej identyfikował upadki (recall = 0.72), jednak kosztem licznych fałszywych alarmów (niska precyzja 0.33). Skuteczność dla klasy 0 również pozostała niezadowalajaca.

Wagi klasy 1: 10

precision	recall	f1-score	support
0 0.91	0.63	0.74	1694
1 0.47	0.85	0.60	654
accuracy		0.69	2348

Zwiekszenie wagi klasy upadków do 10 poprawiło wyniki modelu – recall dla upadków wzrósł do 0.85, a ogólna dokładność osiagneła 69%. Mimo to, precyzja nadal pozostaje umiarkowana (0.47), co może prowadzić do fałszywych powiadomień.

9 Podsumowanie

Podejście z podziałem na osoby znaczaco utrudniało zadanie klasyfikatorom – pokazujac rzeczywiste możliwości generalizacji. KNN całkowicie zawiódł w tym scenariuszu. SVM wykazał poprawe po odpowiednim dostrojeniu wag, ale nadal nie dorównał skuteczności RF. Random Forest, mimo zmienności w wynikach, wciaż osiagał najlepszy kompromis miedzy precyzja a czułościa. Dla podziału losowego wszystkie algorytmy osiagnęły bardzo wysoka dokładność (powyżej 90%). Klasa upadków została dobrze wykryta, jednak wyniki moga być zawyżone z powodu obecności danych tych samych osób w zbiorach treningowym i testowym. Może to prowadzić do przeuczenia na indywidualne cechy (np. wzrost, sposób chodzenia), co w praktycznym zastosowaniu może obniżać skuteczność.

Na podstawie przeprowadzonych eksperymentów, najlepsze rezultaty osiagnał model **Random Forest** — zarówno w scenariuszu podziału losowego, jak i realistyczniejszym podziale na osoby. RF okazał sie najstabilniejszy, a także najbardziej zrównoważony w wykrywaniu upadków i unikania fałszywych alarmów.

Biorac pod uwage końcowe zastosowanie modelu w aplikacji wspierajacej bezpieczeństwo osób starszych, kluczowe jest skuteczne wykrywanie upadków, nawet kosztem umiarkowanej liczby fałszywych alarmów. Random Forest, ze wzgledu na swoja elastyczność i wysoka dokładność, jest najbardziej obiecujacym kandydatem do implementacji.