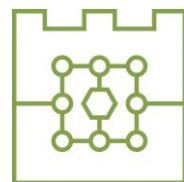




**Politechnika Krakowska  
im. Tadeusza Kościuszki**

Wydział Informatyki i Telekomunikacji



**Klaudia Kromołowska**

Numer albumu: 145692

**Zastosowanie metod uczenia maszynowego w  
diagnostyce cukrzycy w oparciu o nacisk i  
temperaturę podeszwowej powierzchni stopy**

**The application of machine learning methods in  
diagnosing diabetes based on foot plantar  
pressure and temperature**

**Praca magisterska  
na kierunku Informatyka  
specjalność Data Science**

Praca wykonana pod kierunkiem:  
**dr inż. Anny Plichty**

Kraków, 2023



*Serdecznie dziękuję Pani dr inż. Annie  
Plichcie oraz moim najbliższym za cenne  
wskazówki, sugestie i okazane wsparcie.*



# **Spis treści**

<b>1. Wstęp.....</b>	<b>7</b>
1.1. Cukrzyca i jej diagnostyka .....	7
1.2. Cel pracy.....	8
1.3. Struktura i założenia pracy .....	9
<b>2. Część teoretyczna .....</b>	<b>11</b>
2.1. Przegląd podobnych rozwiązań.....	11
2.2. Data Science, czyli nauka o danych .....	12
2.3. Uczenie maszynowe .....	14
2.3.1. Istotne pojęcia .....	14
2.3.2. Podział zbiorów uczących.....	15
2.3.3. Rodzaje uczenia maszynowego .....	15
2.4. Algorytmy uczące.....	16
2.5. Metody ewaluacji modelu .....	17
<b>3. Implementacja.....</b>	<b>19</b>
3.1. Wykorzystywane narzędzia .....	19
3.2. Zbiór danych.....	20
3.3. Przygotowanie danych.....	22
3.4. Analiza danych .....	22
3.4.1. Wykresy związane z rozkładem nacisku.....	23
3.4.2. Wykresy związane z rozkładem temperatury .....	31
3.5. Modele klasyfikacji .....	38
3.5.1. Linear Regression Classifier .....	38
3.5.2. Naive Bayes Classifier .....	39
3.5.3. Descision Tree Classifier .....	40
3.5.4. Random Forest Classifier .....	42
3.5.5. Support Vector Machine Classifier .....	43

3.5.6. Gradient Boosting Classifier .....	45
3.5.7. K-Nearest Neighbours Classifier .....	46
<b>4. Wyniki .....</b>	<b>49</b>
4.1. Korelacje danych .....	49
4.2. Ewaluacja dla rozkładu ciśnienia .....	52
4.3. Ewaluacja dla rozkładu temperatury .....	54
<b>5. Podsumowanie .....</b>	<b>57</b>
5.1. Osiągnięte cele .....	57
5.2. Możliwości rozwoju algorytmu.....	58
5.3. Napotkane problemy .....	58
<b>Bibliografia .....</b>	<b>60</b>
<b>Spis rysunków .....</b>	<b>65</b>
<b>Spis tabel .....</b>	<b>66</b>
<b>Załączniki .....</b>	<b>67</b>

# **1. Wstęp**

Już kilkadziesiąt lat przed naszą erą Wergiliusz przyznał, że *Największym bogactwem jest zdrowie* [1], poparł go później również Buddha mówiąc że *Zdrowie to największy dar* [2]. W obecnych czasach zdrowie wciąż pozostało bardzo ważnym tematem, nie tylko dla każdego przeciętnego obywatela, ale również dla firm i koncernów. Znaczący rozwój technologii umożliwił powstanie wielu firm z obszaru MedTech (ang. medical technology - technologie medyczne), a trend ten nasilił się jeszcze bardziej przez pandemię COVID-19. Podkreślano nie tylko istotę zdrowia psychicznego, zapewniając mobilne aplikacje służące jako wsparcie chorych na depresję, ale też fizycznego, przez rozwój telemedycyny oraz diagnostyki.

Niestety wraz z rozwojem społeczeństwa nasila się też występowanie chorób cywilizacyjnych, między innymi nowotworów złośliwych, alergii, ale także otyłości, miażdżycy czy cukrzycy [3]. Najliczniejszą grupę chorych stanowią alergicy (ok. 12 mln. Polek i Polaków), drugą co do wielkości są osoby otyłe (ok. 10 mln. osób) [4]. Jednak nie są to problemy zdrowotne, które bezpośrednio zagrażają życiu pacjenta. Kolejną stale rosnącą grupą osób są chorzy na cukrzycę. Szacuje się, że w 2022 roku liczba chorych na świecie wynosiła ok. 422 miliony osób [5], a każdego roku ok 1.5 mln. zgonów jest bezpośrednio związań z tą chorobą. W Polsce problem ten dotyczy ponad 3 milionów ludzi [6], a aż 1/3 z nich nawet o tym nie wie. Co więcej, źródła [7, 8] podają, że skraca ona życie średnio aż o 12 lat, co roku zabijając ponad 25 tysięcy osób w Polsce.

## **1.1. Cukrzyca i jej diagnostyka**

Cukrzyca jest chorobą, którą stosunkowo łatwo zdiagnozować - wystarczy doustny test tolerancji glukozy, zwany inaczej „testem obciążenia glukozą” i „krzywą cukrową”. Niestety badanie trwa aż dwie godziny, co prawdopodobnie jest jedną z przyczyn tego, że wiele osób nie bada się regularnie. Objawy cukrzycy również nie są zbyt charakterystyczne - wzmożone pragnienie, utrata masy ciała, osłabienie i senność łatwo przeoczyć lub przypisać przepracowaniu, czy stresującemu trybowi życia.

W uproszczeniu wyróżnia się dwa typy cukrzycy:

1. typ 1 dotyczący osób z uszkodzonymi komórkami trzustki, które zaprzestają produkcji insuliny,
2. typ 2 dotyczący osób dorosłych, towarzyszący otyłości lub wynikający z predyspozycji genetycznych, związany z insulinoopornością i niedoborem insuliny.

Konsekwencje cukrzycy mogą być bardzo poważne, należą do nich między innymi udary, demencja, niewydolność serca czy nerek, ślepotę i stopa cukrzycowa. Ostatnie ze wspomnianych dolegliwości związane jest ze spowodowanym cukrzycą uszkodzeniem nerwów, co może skutkować osłabieniem mięśni stóp. Po zaobserwowaniu tego faktu wiele naukowców postanowiło przyjrzeć się temu tematowi, szukając szansy na szerszą diagnostykę cukrzycy i kontrolowany przebieg choroby [9].

Zespół stopy cukrzycowej istotnie wpływa na jakość życia pacjenta [10], wiążąc się nie tylko z owrzodzeniami stopy [11], ale również zwiększoną śmiertelnością [12]. Co więcej, dane pokazują, że ryzyko amputacji stopy jest 17–40 razy większe u diabetyków [13], w porównaniu do osób bez tego schorzenia.

Badania wykazały, że dane baropodometryczne z podeszwowej części stopy pozwalają zróżnicować osoby zdrowe, chore lub zagrożone chorobą (ang. pre-diabetics). Pojawiła się więc szansa, że badanie rozkładu nacisku podeszwowego stóp mogłoby pomagać we wstępnym diagnozowaniu tej choroby, tym samym znacznie skracając czas, kiedy pacjent jest nieświadomy swoich problemów zdrowotnych.

## 1.2. Cel pracy

Celem pracy jest analiza porównawcza algorytmów wykorzystujących metody uczenia maszynowego do rozróżnienia osób zdrowych oraz chorych na cukrzycę. Algorytmy przygotowano w oparciu o dwa parametry – nacisk oraz temperaturę podeszbowej części stopy. W celu weryfikacji postępów prac badawczych postawiono kilka hipotez:

- Czy na podstawie nacisku lub temperatury podeszbowej części stopy z pomiaru (ok. 2 godziny) można odróżnić osoby zdrowe i chore na cukrzycę?
- Która z metod uczenia maszynowego sprawdzi się w tym celu najlepiej?
- W których częściach stopy najlepiej umieścić czujniki, by skutecznie diagnozować cukrzycę?
- Czy dla temperatury i nacisku będą to te same miejsca?

Podsumowując, w ramach pracy zostanie przeprowadzony eksperyment, polegający na odpowiednim przygotowaniu (dobraniu argumentów) i porównaniu skuteczności siedmiu wybranych algorytmów uczenia maszynowego, spośród których każdy zostanie zweryfikowany dla dwóch wspomnianych wcześniej parametrów (nacisku i temperatury).

### 1.3. Struktura i założenia pracy

Metodologię związaną z celami badawczymi, opisywaną w pracy, można podzielić na dwie istotne części:

- Analiza danych – aby lepiej zrozumieć dane, którymi posłużyono się w pracy, przeprowadzono ich wstępную analizę. Przyjrzano się podstawowym statystykom i przygotowano różne rodzaje wykresów poszczególnych parametrów.
- Implementacja modeli klasyfikacji – przedstawiono i porównano wybrane metody uczenia maszynowego (m.in. regresję logistyczną, drzewo decyzyjne, las losowy, drzewa wzmacniane gradientowo, k-najbliższych sąsiadów, maszynę wektorów nośnych).

Niniejsza praca składa się z pięciu rozdziałów. Pierwszy z nich zawiera wprowadzenie do tematu pracy i zarys problematyki, czyli informacje związane z cukrzycą, jej konsekwencjami i pojawiającymi się możliwościami diagnostycznymi. Ponadto opisano cel badawczy oraz zakładane hipotezy, a także strukturę pracy.

W drugim rozdziale opisano teoretyczny aspekt pracy. Zwrócono uwagę na istniejące już rozwiązania oparte o nacisk podeszwowej części stopy oraz systemy i technologie umożliwiające rozpoznawanie cukrzycy. Przedstawiono zastosowania uczenia maszynowego w tego typu badaniach, rodzaje oraz metody klasyfikacji, a także sposoby ewaluacji wspomnianych modeli.

Kolejny, trzeci rozdział, zawiera opis implementacji - informacje na temat wykorzystywanych narzędzi i technologii oraz szczegółowe informacje dotyczące użytego zbioru danych. Przedstawiono w nim również etap przygotowania i analizy danych, w tym statystyki i wykresy.

Czwarty rozdział jest najistotniejszą częścią pracy - zawiera wyniki przeprowadzonych eksperymentów, które pozwalają odpowieść na pytania postawione na początku pracy. Przedstawiono w nim informacje dotyczące korelacji między danymi z poszczególnych czujników, a stanem zdrowia pacjenta oraz zastosowane modele klasyfikacji wraz z wynikami ewaluacji tych modeli dla każdego z parametrów: nacisku i temperatury.

W ostatnim rozdziale zaprezentowano wnioski z przeprowadzonych prac badawczych. Zwrócono uwagę na osiągnięte cele, to w jaki sposób można rozwinąć badania w przyszłości, a także wspomniano o głównych trudnościach napotkanych w czasie realizacji niniejszej pracy dyplomowej. Praca obejmuje także bibliografię oraz spisy załączników, rysunków i tabel.

## **2. Część teoretyczna**

Pierwszy rozdział pracy to wprowadzenie w problematykę – opisano jak powszechną i poważną chorobą jest cukrzyca, a także oraz jak baropodometria umożliwia różnicowanie osoby zdrowe i diabetyków. Kolejnym krokiem jest więc sprawdzenie, w jakich celach wykorzystywane jest obecnie monitorowanie rozkładu nacisku podeszbowego stóp oraz jakie obecnie istnieją metody wspomagania diagnostyki cukrzycy.

### **2.1. Przegląd podobnych rozwiązań**

Obecnie jedyną szeroko stosowaną metodą diagnostyki cukrzycy jest badanie poziomu glukozy we krwi w teście obciążenia glukozą. Już kilka lat temu naukowcy ze Stanów Zjednoczonych zauważyli problem z diagnozowaniem cukrzycy i w ramach artykułu „A New Look at Screening and Diagnosing Diabetes Mellitus” [14] (pl. „Nowe spojrzenie na badania przesiewowe i diagnostykę cukrzycy”) opisali metodę HbA1c jako nowy, skuteczniejszy i bardziej innowacyjny sposób diagnozowania tej choroby.

Metoda ta opiera się na pomiarze hemoglobiny glikowanej, która wskazuje, czy poziom glukozy był prawidłowy w ciągu ostatnich trzech miesięcy. Została tzw. „złotym standardem” w diagnostyce cukrzycy, a jednak wciąż ponad 30% chorych nie wie, że to ona jest przyczyną ich problemów zdrowotnych. W literaturze naukowej, w tym najnowszych publikacjach nie widać niestety innych metod, które mogłyby służyć badaniom przesiewowym. Dlatego też, odkrycie powiązań między rozkładem nacisku stóp na podłożu, a chorobą daje nadzieję na skuteczniejszą diagnostykę – obecnie średni czas między pierwszymi objawami, a diagnozą to ok. 7 lat. Jeżeli badanie baropodometryczne lub specjalistyczne wkładki do butów wyposażone w czujniki nacisków [15], mogłyby diagnozować tak powszechną chorobę, jak cukrzyca, to czego jeszcze można byłoby się z nich dowiedzieć?

Z odpowiedzią na to pytanie przychodzi inna publikacja naukowa – „Review on plantar data analysis for disease diagnosis” [16], która zawiera przegląd możliwości diagnostyki chorób przy użyciu analizy danych podeszbowej części stopy.

Autorzy przywołują w niej kilka przykładów zastosowań danych z podeszwowej części stopy:

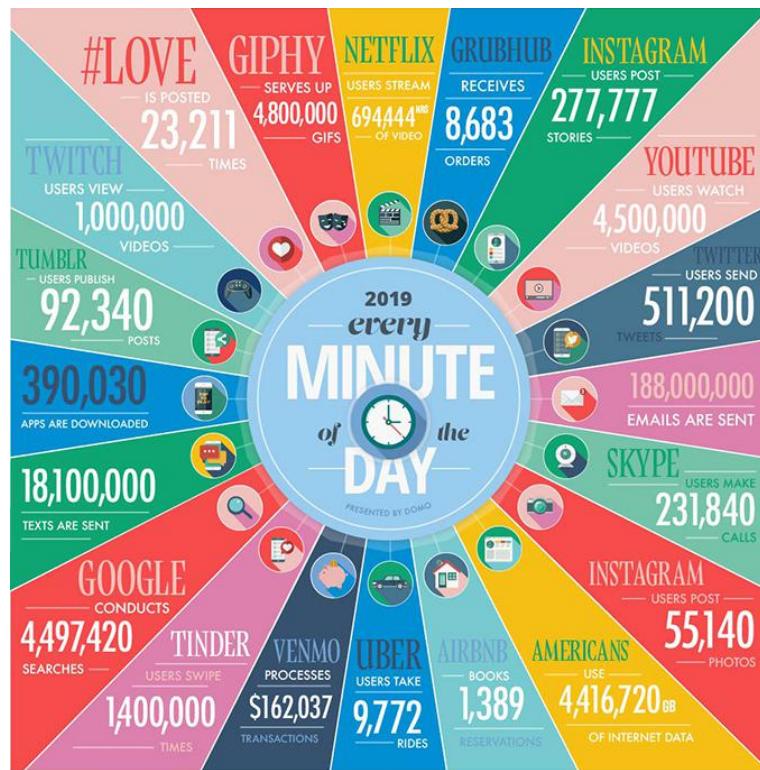
- diagnostyka cukrzycy,
- rozpoznawanie typów stopy, takich jak pronująca, supinująca, płaskostopie,
- rozpoznawanie koślawości palucha,
- rozpoznanie pacjentów z zrekonstruowanym lub usztywnionym stawem skokowym (alloplastyka i artrodeza stawu skokowego),
- rozpoznawanie pacjentów z różnicami w długości kończyn dolnych,
- rozpoznawanie pacjentów z zapaleniem kości i stawów,
- wykrywanie ryzyka upadku lub poślizgnięcia,
- różnicowanie pacjentów z chorobą Parkinsona lub niedowładem połowiczym w oparciu o długość i czas cyklu kroku.

Podsumowując, analiza danych, które otrzymywanych w wyniku monitorowania chodu oraz rozkładu nacisku podeszwowego stóp może mieć bardzo szerokie zastosowania. Dlatego bardzo ważnym i ciekawym problemem badawczym jest to, w jaki sposób można wspomniane dane analizować i jak wyciągać z nich wnioski.

## 2.2. Data Science, czyli nauka o danych

Data Science (pl. nauka o danych) to dziedzina, która łączy ze sobą wiedzę z zakresu matematyki i statystyki oraz umiejętności programistyczne. Choć termin ten w przedstawionym wyżej znaczeniu widnieje już od 2001 roku, do jego znacznego rozpowszechnienia doszło w 2012 roku, kiedy to Harvard Business Review ogłosił zawód Data Scientist (osoby zajmującej się dziedziną nauki o danych zawodowo) „najseksowniejszym zawodem XXI wieku” [17]. Dane są podstawą do różnego rodzaju badań i nowych teorii, jednak należy zauważyć do jak ogólnego wzrostu ich dostępności przyczynił się rozwój technologii ostatnich lat. Szacuje się, że aż ok. 90% danych na świecie powstało w ciągu ostatnich dwóch lat [18] – jeśli nikt ich nie przeanalizuje, pozostaną bezużyteczne – dlatego data science stało się tak ważną dziedziną.

Rysunek 2.1 przedstawia ilość danych produkowanych w różnych serwisach społecznościowych, stronach i serwisach w ciągu jednej minuty – są to dane z 2018 roku, więc można założyć, że obecnie te liczby są jeszcze większe.



Rys. 2.1. Ilość danych produkowanych w ciągu jednej minuty [19]

Poniżej przedstawiono kilka przykładów zastosowań nauki o danych [20]:

- analiza wzorców zachowań klientów przez sklepy internetowe,
- tworzenie spersonalizowanych rekomendacji zakupowych,
- ukierunkowanie marketingu – reklam i promocji,
- optymalizacja łańcucha dostaw i dystrybucji,
- wykrywanie potencjalnych awarii jeszcze przed ich wystąpieniem,
- prognozowanie sprzedaży, kursów walut i akcji,
- optymalizacja tras,
- diagnostyka, np. przesiewowa analiza zdjęć rentgenowskich w celu wykrywania zwiększonego ryzyka rozwoju nowotworu.

Proces obróbki danych można podzielić na kilka etapów:

1. Określenie problemu.
2. Przygotowanie zbioru danych.
3. Przygotowanie danych (m.in. ich czyszczenie).
4. Analiza danych.
5. Prezentacja wyników pracy.

Najistotniejszą częścią tego procesu jest analiza danych, która poza analizą statystyczną i wizualizacją obejmuje również uczenie maszynowe.

## 2.3. Uczenie maszynowe

W większości dziedzin, na przykład w matematyce, można dostrzec zbiór ściśle określonych reguł, dzięki którym przy dostarczeniu odpowiednich danych możliwe jest uzyskanie odpowiedzi na zadawane pytania. Uczenie maszynowe działa nieco odwrotnie – pozwala uzyskiwać reguły na podstawie danych.

Pojęcie uczenia maszynowego wprowadził Arthur Samuel w 1959 roku [21], który rozwijał projekt mający na celu szkolenie zawodników szachowych. Do lat 90. ubiegłego wieku wykorzystywane było ono głównie w prostych grach, następnie znalazła zastosowanie w przeglądarkach internetowych, aż do 2010 roku, kiedy zaczęto stosować uczenie maszynowe powszechnie, rozwijając sieci neuronowe.

### 2.3.1. Istotne pojęcia

Tak jak wspominano wyżej, algorytmy wykorzystywane w uczeniu maszynowym mają zdolność do tworzenia modeli (złożonych z listy instrukcji do wykonania) w oparciu o przeanalizowany zestaw danych – stąd właśnie mowa o uczeniu maszynowym. W celu lepszego zrozumienia dalszej części pracy, warto zwrócić uwagę na następujące pojęcia związane z uczeniem maszynowym i nauką o danych:

**model** – wynik procesu uczenia.

**dane uczące** – zbiór danych, najczęściej część zbioru, która wykorzystywana jest do procesu uczenia.

**próbka** – pojedynczy obiekt zbioru danych, przykładowo jedno ze zdjęć lub jeden z wierszy tabeli.

**cecha** – właściwość, za pomocą której opisywany jest obiekt – na przykład kolumna wartości w przypadku danych tabelarycznych.

**funkcja aktywacji** – zadaniem funkcji aktywacji, jest wprowadzenie nieliniowości do modelu uczenia maszynowego. Nieliniowość zapewnia większą czułość na zależności między klasami.

**overfitting** – sytuacja, w której model uczenia maszynowego jest nadmiernie dopasowany do danych treningowych, co prowadzi do słabej ogólnej zdolności do uogólniania i słabej skuteczności na nowych, nieznanych danych.

### 2.3.2. Podział zbiorów uczących

Dane uczące można przedstawiać na różne sposoby – można podzielić zbiory względem etykietowania oraz struktury.

Zbiory etykietowane (ang. labeled), to takie dane, które zawierają informacje o tym, do której klasy przynależą [22]. Takie dane zostaną wykorzystane w celach uczenia nadzorowanego, przykładowo może być to zbiór danych, zawierający informacje o pacjentach oraz informację czy pacjent jest zdrowy, czy chory. Zbiory nieetykietowane (ang. unlabeled) będą wykorzystywane przy uczeniu nienadzorowanym.

Przy podziale ze względu na strukturę można wyróżnić trzy typy: dane strukturalne, półstrukturalne i niestrukturyzowane [23]. Pierwsze z nich (ang. structured) to takie zestawy danych, które są powiązane relacjami – najczęściej są to relacyjne bazy danych. Drugie (ang. semi-structured) obejmują pojedyncze zestawy danych, takie jak tabele. Trzecie (ang. unstructured) określają zwykle zdane bez żadnej struktury – przykładowo dane multimedialne, jak próbki dźwięku czy zdjęcia [24].

### 2.3.3. Rodzaje uczenia maszynowego

Można wyróżnić trzy główne rodzaje uczenia maszynowego [25]:

1. Uczenie nadzorowane – inaczej budowanie modelu na podstawie etykietowanego zbioru danych. Przykładowo – model dostaje zbiór danych zawierający zdjęcia RTG z etykietami, czy należą one do chorych, czy zdrowych osób. Zadaniem algorytmu jest określenie, czy pacjent jest zdrowy, czy chory, na podstawie nowych zdjęć prześwietleń

rentgenowskich płuc. W najprostszych przypadkach efektem uczenia nadzorowanego będzie przyporządkowanie do jednej z dwóch klas, co określone jest mianem klasyfikacji binarnej.

2. Uczenie nienadzorowane – umożliwia budowę modelu na podstawie nieetykietowanego zbioru danych, który będzie opisywał relacje lub ukryte struktury danych. Znaczącą różnicą jest problem z walidacją takiego uczenia, ponieważ nie jest znany wynik wyjściowy. Dla przykładu, mając zbiór figur geometrycznych, składający się z trójkątów i okręgów, model nie będzie wiedział jaką figurę analizuje, ale po pewnym czasie znajdzie podobieństwa między różnymi trójkątami oraz podobieństwa między różnymi okręgami, tworząc dwie klasy. Można powiedzieć, że ten rodzaj uczenia bardziej przypomina ludzki sposób działania – opiera się on na obserwacji i nauce pewnych wzorców.
3. Uczenie ze wzmacnieniem – model wyposażony jest w zestaw reguł, które pozostają niezmienne podczas procesu uczenia. Model jest poprawiany w każdej kolejnej iteracji przez efekt nagrody (czyli wzmacnienia, które opisuje jakość działania modelu). Najlepszym przykładem będzie system nawigacyjny, gdzie zasady ruchu drogowego pozostają niezmienne, zmieniają się natomiast warunki drogowe i natężenie ruchu, a celem jest dotarcie jak najszybciej do celu.

W związku z tym, że opisany w kolejnym rozdziale zbiór danych, zawiera również etykiety, najbardziej interesującym w kontekście prowadzonych badań jest uczenie nadzorowane i temat wspomnianej już klasyfikacji binarnej.

## 2.4. Algorytmy uczące

Podczas analizowania danych można skorzystać z wielu różnych algorytmów uczących. Warto zwrócić uwagę na to czym się charakteryzują i czym się różnią, by wybrać te, które najlepiej sprawdzą się w przypadku analizy porównawczej:

- Algorytmy regresyjne [ang. regression algorithms] – np. regresja logistyczna, regresja liniowa – zakłada, że dane mogą zostać rozdzielone na klasy za pomocą prostej lub płaszczyzny n-wymiarowej [26].
- Bazujące na instancjach danych [ang. instance-based algorithms] – np. K-najbliższych sąsiadów – wartość interesującego nas punktu zależy od wartości K punktów znajdujących się najbliższej.

- Drzewa decyzyjne [ang. decision tree algorithms] – np. drzewa klasyfikacji i regresji – drzewo składa się z węzłów, w każdym z nich sprawdzana jest wartość jednej z cech i każdy prowadzi do gałęzi. Na końcu gałęzi znajdują się liście – odpowiedzi na hipotezę.
- Algorytmy Bayesowskie [ang. bayesian algorithms] – np. naiwny Bayes, Gaussowski naiwny Bayes – zwane naiwnymi, ze względu na założone uproszczenie, że cechy są od siebie niezależne. Algorytm ten opiera się na prawdopodobieństwie warunkowym klasy, dla danej obserwacji [27].
- Algorytmy klasteryzujące [ang. clustering algorithms] – np. k-średnich – w przypadku tego algorytmu, wstępnie ustalany jest parametr k (liczba grup/klas) i środki zbiorów. Następnie, wykorzystując odległości między punktami a środkami zbioru, przypisujemy obiekty do zbiorów. Następnie powtarzamy cały proces, aż obiekty przestaną się przesuwać między zbiorami.
- Sztuczne sieci neuronowe [ang. artifical neural network] – np. perceptron – podobnie jak w przypadku drzewa, wartość wyjściowa algorytmu zależna jest od wartości w kilku węzłach. Tym razem jednak, każdy węzeł ma określoną wagę, a wartość wyjściowa stanowi sumę wartości z poprzednich warstw przemnożoną przez wagi. Dodatkowo, na wyliczoną sumę nakładana jest tzw. funkcja aktywacji [28].
- Głębokie uczenie [ang. deep learning algorithms] – np. konwolucyjne sieci neuronowe – do opisanych wyżej sieci neuronowej należy dodać warstwę filtra/kernelu, z zastosowaniem głównie przy analizie obrazu czy przetwarzaniu języka [29].
- Algorytmy łączone [ang. ensamble algorithms] np. AdaBoost, las losowy – polega na wykorzystaniu wielu różnych klasyfikatorów w różny sposób, przykładowo poprzez wyczelenie modelu na błędy zwracane ze słabszych klasyfikatorów w kolejnych iteracjach [30].

## 2.5. Metody ewaluacji modelu

Gdy cały model będzie już gotowy, należy ocenić jego skuteczność. Istnieje kilka znanych metod ewaluacji, poniżej przedstawiono kilka z nich. Na potrzeby przedstawionych poniżej wzorów przyjęto następujące skróty:

- TP – ang. true positive – wyniki prawdziwie pozytywne (np. pacjent chory zdiagnozowany jako chory).
- FP – ang. false positive – fałszywie pozytywne (np. pacjent zdrowy zdiagnozowany jako chory).

- TN – ang true negative – prawdziwie negatywne (np. pacjent zdrowy zdiagnozowany jako zdrowy).
- FN – ang false negative – fałszywie negatywne (np. pacjent chory zdiagnozowany jako zdrowy).

Dokładność – ang. accuracy – jest to ilość poprawnie przewidzianych klas, podzielona przez całkowitą ilość testowanych instancji. Najprostsza metoda ewaluacji, niestety nie uwzględnia różnic między istotnością wyników fałszywie dodatnich i fałszywie ujemnych, które są szczególnie ważne w przypadku danych medycznych.

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

Precyzja – ang. precision – informuje, jak dokładny jest model na podstawie odsetku pozytywnie dodatnich wyników ze wszystkich dodatnich. To dobry wskaźnik, gdy wykrywanie fałszywie dodatnich wyników jest istotne, przykładowo wiadomość e-mail zostanie potraktowana jako spam, mimo że nim nie jest.

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

Czułość – ang. recall – informuje o odsetku prawidłowo sklasyfikowanych przypadków wśród wszystkich przypadków pozytywnych (zarówno fałszywie negatywnych, jak i prawdziwie pozytywnych). To dobry wskaźnik, gdy istotne jest wykrywanie wyników fałszywie negatywne, przykładowo gdy transakcja bankowa oszusta zostanie sklasyfikowana jako tradycyjna i nie wzbudzi czujności banku.

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

Wartość F1-score – używana w przypadku, gdy potrzebny jest balans między precyją i czułością – używany jest głównie wtedy, kiedy zarówno przypadki pomyłek fałszywie dodatnich, jak i fałszywie ujemnych są ważne dla działania modelu.

$$\frac{2 \cdot \text{precyzja} \cdot \text{czułość}}{\text{precyzja} + \text{czułość}}$$

Niniejszy rozdział poświęcony jest przede wszystkim teoretycznemu opisowi pracy. Po przejrzeniu istniejących rozwiązań i nauce o danych, przygotowano krótki opis uczenia maszynowego – zarówno najistotniejszych pojęć jak i jego rodzaj. Przedstawiono kilka algorytmów uczących i metody ewaluacji modelu. Kolejnym etapem prac jest wybór zbioru danych, analiza i implementacja oprogramowania.

## 3. Implementacja

Jak wspomniano w części 2.2, cykl życia projektu z dziedziny nauki o danych składa się z kilku faz. W tym rozdziale opisano trzy najbardziej kluczowe – informacje na temat zbioru danych, sposób ich przygotowania i analizę. Analiza porównawcza dotyczy obu parametrów – zarówno nacisku, jak i temperatury oraz obejmuje przeprowadzenie eksperymentów polegających na przygotowaniu i przetrenowaniu siedmiu wybranych modeli uczenia maszynowego.

### 3.1. Wykorzystywane narzędzia

Obecnie programiści mają do dyspozycji setki różnych języków programowania, ale według rankingu TIOBE [31] jednym z najpopularniejszych jest Python. Zapewnia on przede wszystkim dostęp do wielu bibliotek, które znacznie ułatwiają między innymi wizualizację danych. To właśnie w tym celu, w celu wykonania analizy porównawczej zdecydowano się na użycie języka Python – poniżej przedstawiono kilka jego najważniejszych zalet [32, 33]:

- przyjazny początkującym – Python charakteryzuje się prostą składnią i jasną strukturą, co sprawia, że jest się go stosunkowo łatwo nauczyć,
- sprzyja testom – dzięki swojej prostocie, Python idealnie nadaje się dla osób, które wdrażają dużą liczbę testów lub prototypów,
- dostęp do bibliotek – dzięki swojej popularności, Pythonem zainteresowała się szeroka społeczność programistów, co skutkuje świetną dostępnością narzędzi i bibliotek.
- elastyczność – nie tylko można wykorzystać go do wielu zadań – od programowania stron internetowych, po uczenie maszynowe, ale także stosunkowo łatwo można zintegrować go z innymi językami programowania.

W czasie prowadzonych badań wykorzystano kilka istotnych bibliotek [34, 35, 36, 37]:

- NumPy – to popularna biblioteka, która wspomaga obsługiwanie wielu zaawansowanych funkcji matematycznych oraz typów obiektów,
- Pandas – jedna z najważniejszych bibliotek w data science, dzięki której można budować tzw. DataFrame (obiekty, zawierające tablice danych) i wykonywać operacje na bazach danych,
- Matplotlib – najbardziej popularna biblioteka służąca do wizualizacji danych,
- Seaborn – druga z bibliotek wykorzystywana do wizualizacji danych, szczególnie bardziej skomplikowanych wykresów.

Kolejnym popularnym [38] wśród analityków danych językiem programowania jest Scala. W trakcie prac nad implementacją wykorzystano go zarówno do przygotowania danych do analizy statystycznej, jak i do przetworzenia w krótkim czasie dużej ilości danych z zastosowaniem metod uczenia maszynowego. Scala wyróżnia się przede wszystkim ze względu na:

- skalowalność – jest zbudowany na wirtualnej maszynie Javy (JVM), dzięki czemu możliwe jest równoległe wykonywanie procesów – co sprawia, że Scala jest idealnym wyborem przy dużych zbiorach danych,
- wydajność – w przeciwieństwie do Pythona, jest językiem kompilowanym, co zapewnia dużo lepszą wydajność, szczególnie przy skomplikowanych obliczeniach,
- język funkcyjny – w językach funkcyjnych nacisk kładziony jest między innymi na niemutowalność obiektów oraz oparcie programu na funkcjach, dzięki czemu łatwiej uniknąć błędów.

Dodatkowo, podczas uczenia maszynowego wykorzystano platformę Apache Spark [39] i bibliotekę Apache Hadoop [40], które są najczęściej używanych przez analityków danych narzędziami dedykowanymi dla obliczeń rozproszonych.

## 3.2. Zbiór danych

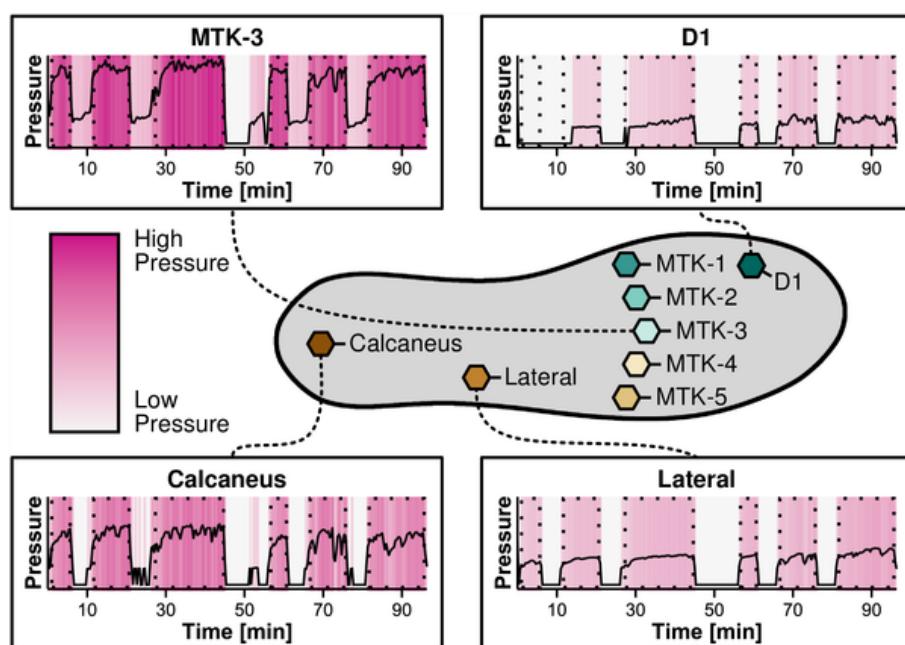
Zbiór danych, który wykorzystano w analizie porównawczej, został przygotowany przez zespół: Niemann U, Spiliopoulou M, Szczepanski T, Samland F, Grützner J. oraz Senk D, w ramach publikacji „Comparative Clustering of Plantar Pressure Distributions in Diabetics with Polyneuropathy May Be Applied to Reveal Inappropriate Biomechanical Stress” [41].

Za pomocą specjalistycznej wkładki wyposażonej w czujniki nacisku zarejestrowano ciśnienie podeszwove w ośmiu miejscach stopy dla 43 osób (18 osób zdrowych i 25 cukrzyków z ciężkim uszkodzeniem neuronu obwodowego).

Czujniki umieszczone w następujący sposób:

- czujnik C – pod piętą,
- czujnik L – pod zewnętrzną, środkową częścią stopy,
- czujnik D1 – pod paluchem,
- czujniki MTK1-MTK5 – odpowiednio pod pierwszą – piątą kośćią śródstopia.

Przedstawione ustawienie widoczne jest również na rysunku 3.1



Rys. 3.1. Schemat ustawienia czujników nacisku wkładki [41]

Badanie polegało na noszeniu wkładki przez 100 minut, w tym przez 70 min podczas stania. W sumie odbyły 43 sesje, co skutkowało zbiorem 86 zestawów danych, w których wartości z czujników rejestrowane były w interwalem czasowym równym 3 sekundy (70% zestawów) lub 20 sekund (30%). W grupie kontrolnej zebrano 106 zestawów danych, które oprócz rejestrowania nacisku, obejmowały również dodać pomiary temperatury z miejsc opisanych na rysunku 3.1.

### 3.3. Przygotowanie danych

Po przyjrzeniu się zbiorowi danych należy zauważyc, że sumarycznie składa się on ze 192 zestawów danych, z czego każdy z nich reprezentowany jest przez jeden plik w formacie .xlsx.

W celu stworzenia modelu uczenia maszynowego zaprojektowane zostały dwie niewielkie bazy danych, w której każdy wiersz reprezentował jedną sesję/zbiór danych. Sumarycznie, każdy plik zawiera 192 wiersze i 31 kolumn, gdzie każda z kolumn reprezentuje jedną z poniższych cech:

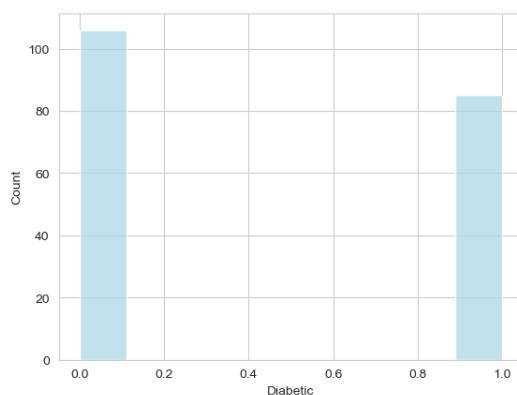
- wartość minimalna, maksymalna i średnia zarejestrowana na każdym z czujników ,
- wartość minimalna, maksymalna i średnia zarejestrowana na każdym z trzech obszarów: pięta, zewnętrzna część śródstopia, przednia część stopy (czujniki MTK1-MTK5 oraz D),
- wartość minimalna, maksymalna i średnia zarejestrowana na całej powierzchni stopy,
- informacja w postaci binarnej, czy dane pochodzą od osoby zdrowej, czy chorej.

Każdą z baz danych zdecydowano się umieścić w pliku typu ORC. Jest to kolumnowy format pliku pamięci, który zapewnia bardzo wysoko wydajny sposób przechowywania danych.

### 3.4. Analiza danych

Po przygotowaniu danych kolejnym etapem jest ich analiza. Przygotowano wykresy, na podstawie których zostanie przeanalizowany cały zbiór danych. Sformułowano także pierwsze wnioski dotyczące korelacji między danymi.

Pierwszym z wykresów jest histogram rozkładu liczby diabetyków (Rys. 3.2) i osób zdrowych, co stanowi jedynie potwierdzenie informacji o zbiorze danych z rozdziału 3.2.



Rys. 3.2. Histogram – ilość sesji osób zdrowych i chorych

### 3.4.1. Wykresy związane z rozkładem nacisku

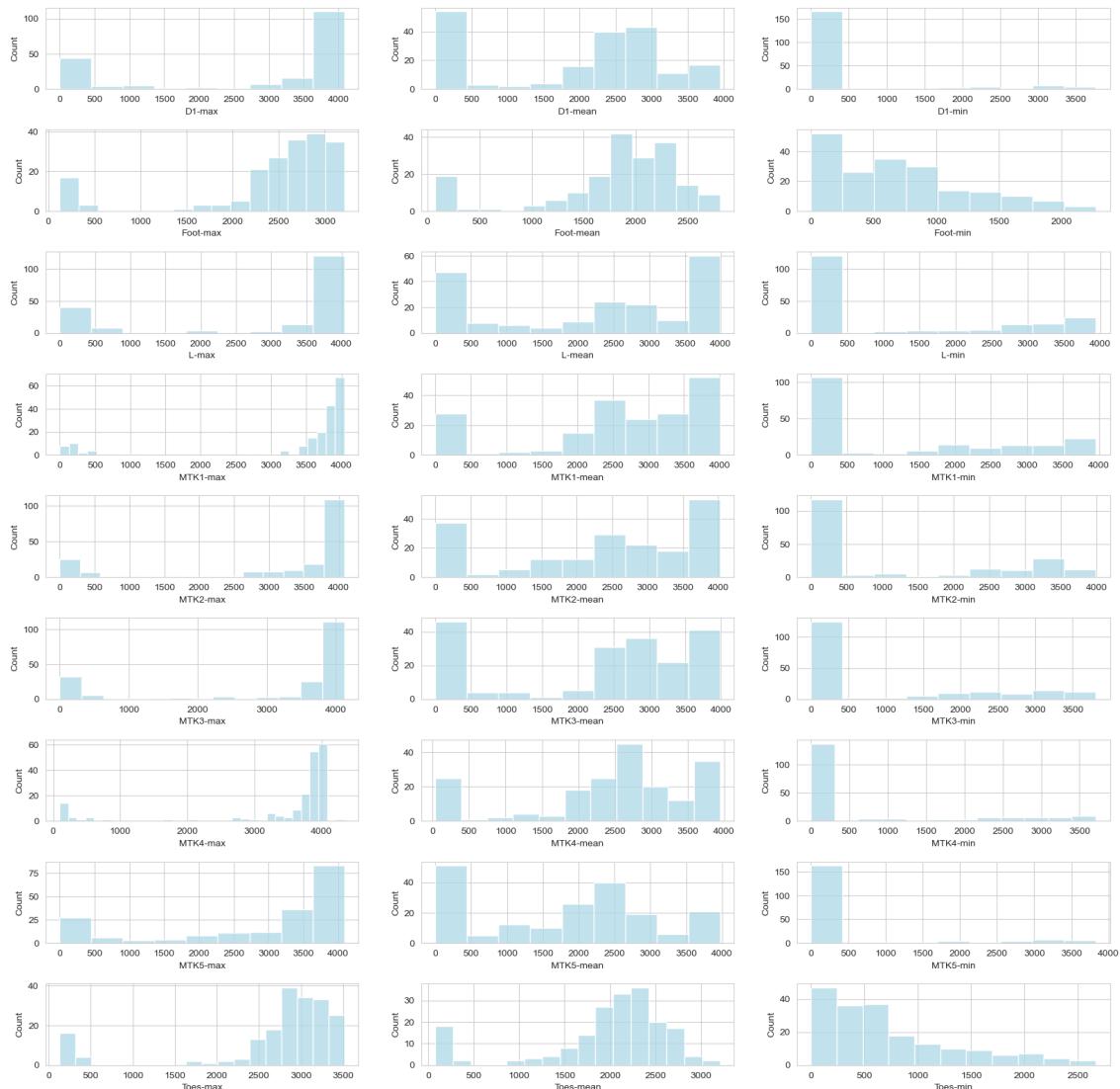
Pierwszym badanym parametrem jest nacisk. Zestaw wykresów pudełkowych pozwala zbadać koncentrację danych, przeanalizować położenie kwartyli, mediany i zmiennych odstających. Wykresy (Rys. 3.3) przygotowano w trzech kolumnach – dla danych minimalnych, maksymalnych i średnich z każdej sesji badań.



Rys. 3.3. Wykresy pudełkowe rozkładu nacisku

Pierwszą ciekawą obserwacją jest fakt, że dane maksymalne czujnika L mają dużo większy rozrzut niż pozostałe – widoczne są większe pudełka na wykresie. Jeśli chodzi o dane średnie, należy zwrócić uwagę na bardzo duże pudełko wykresu czujnika MTK1 oraz L – duży rozrzut danych, ale także na ułożenie mediany, co oznacza, że zdecydowana większość pomiarów

była stosunkowo wysoka. Widoczna jest również asymetria danych – skośność prawostronna. Patrząc ogólnie na wszystkie wykresy, można wysunąć wniosek, że dane koncentrują się na średnich i wyższych wartościach oraz jest niewiele danych odstających. Warto zauważać ciekawą zależność między naciskiem czujników MTK1-MTK5, ponieważ dla kolejnych czujników wartości nacisku są coraz niższe – czyli większą część ciężaru ciała opieramy na wewnętrznych palcach stóp.



Rys. 3.4. Histogramy rozkładu nacisku

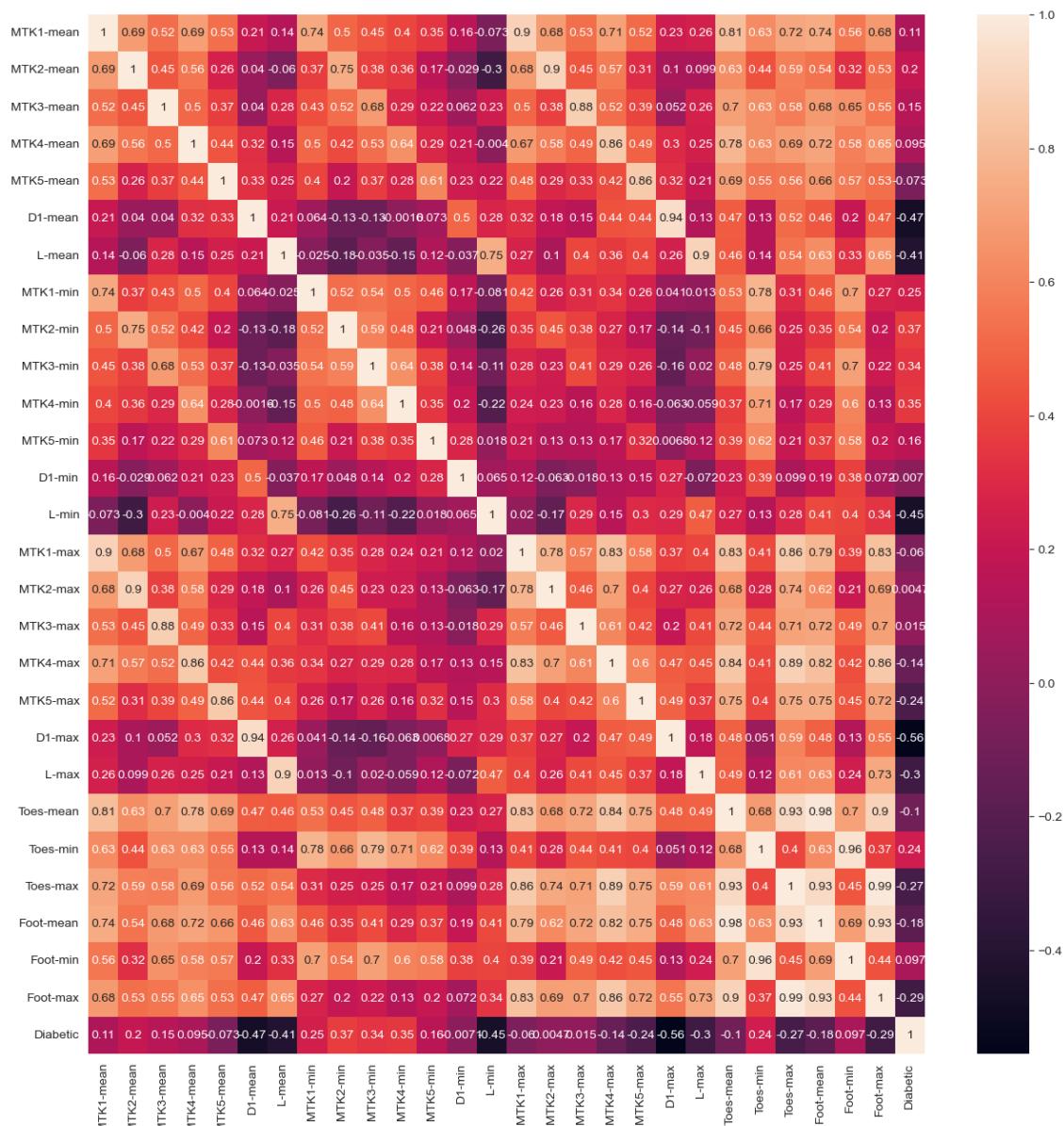
Kolejnymi istotnymi wykresami są histogramy (Rys. 3.4), które pozwalają zbadać rozłożenie danych. Dla danych z kolumny pierwszej – danych maksymalnych – na większości wykresów, większość danych koncentruje się po prawej stronie wykresu, odwrotnie zaś w przypadku danych minimalnych, zgodnie z tym co podpowiada logika. Ciekawą zależnością

jest fakt, że brakuje wartości niskich – większość danych znajduje się z prawej strony wykresu, dodatkowo widać wysokie słupki w okolicy 0 (brak nacisku, zapewne wynikający z podniesienia stóp). Dopiero przy wykresach „Foot”, czyli uśrednionych wartościach ze wszystkich czujników, widać duże zróżnicowanie danych.

Trzecim rodzajem wykresu jest wykres korelacji (Rys. 3.5) pomiędzy poszczególnymi cechami. Za pomocą tego wykresu poszukiwano związku między zmienną określana – „Diabetics”, oznaczającą stan zdrowia pacjenta, a innymi zmiennymi. Największy związek widać zmiennych D1-średniej i D1-maksymalnej oraz L-średniej i L-minimalnej. Należy zaznaczyć, że są to ujemne korelacje, z dodatnich największe zaobserwowano dla wartości MTK2-minimalnej.

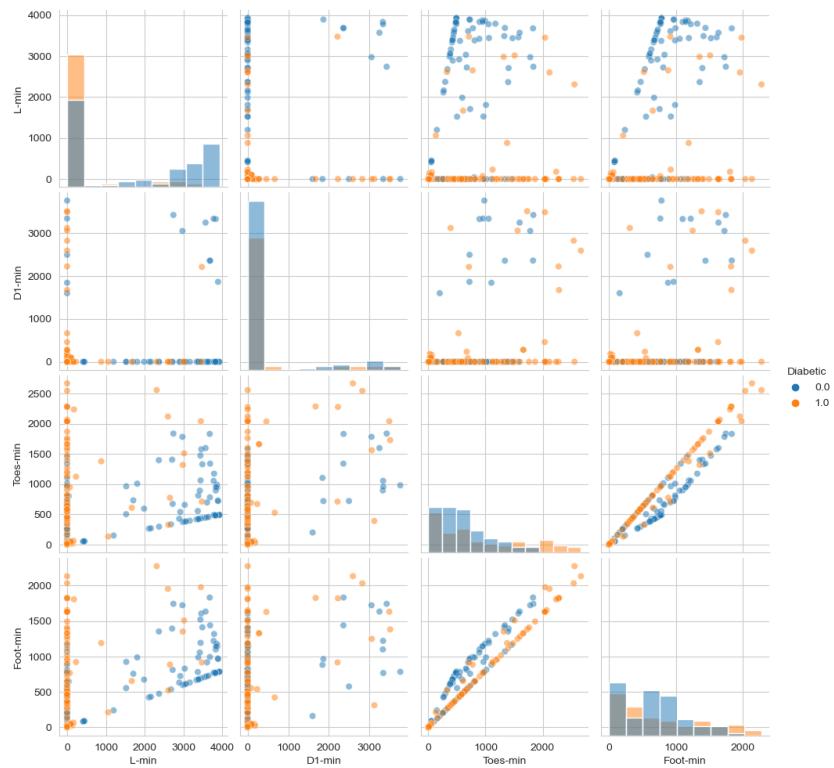
Trzy kolejne wykresy (Rys. 3.6, 3.7, 3.8) to tzw. pairploty – pojedynczy wykres składa się z wielu wykresów, z których każdy przedstawia zależność między dwoma wybranymi zmiennymi. Aby wykresy pozostały dobrze widoczne, zdecydowano się użyć tylko czterech zmiennych – L, D1, sumie z MTK1-MTK5 i sumie ze wszystkich czujników oznaczonej zmienną „Foot”.

Głównym wnioskiem nasuwającym się po analizie przedstawionych wykresów, jest to, że diabetycy mają dużo wyższy wskaźnik minimalnego nacisku na czujnik L, podobnie w przypadku średniego L – jest on raczej wyższy od pomiarów u osób zdrowych. Dla przedstawicieli grupy kontrolnej częściej występowały wyniki bliskie zeru. Jeśli chodzi o wyniki maksymalne – dla czujnika D1-max wyższe wyniki otrzymywali pacjenci z grupy osób chorych, ale wspomniane wyniki są też bardziej rozproszone.



Rys. 3.5. Wykres korelacji między zmiennymi nacisku

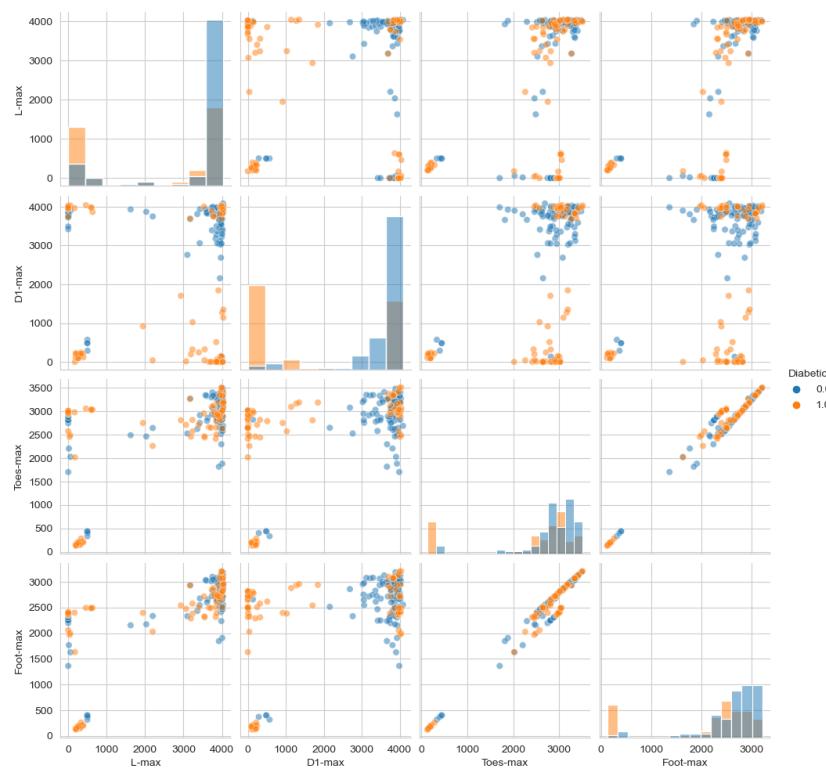
Następne trzy wykresy (Rys.3.9, 3.10, 3.11) – wykresy rozproszenia – pozwalają na sformułowanie podobnych wniosków jak w przypadku analizowania wykresów boxplot - dane powiązane z diabetykami, są dużo bardziej rozproszone. Nie da się jednak wyciągnąć na ich podstawie wniosków dotyczących korelacji między badanymi cechami.



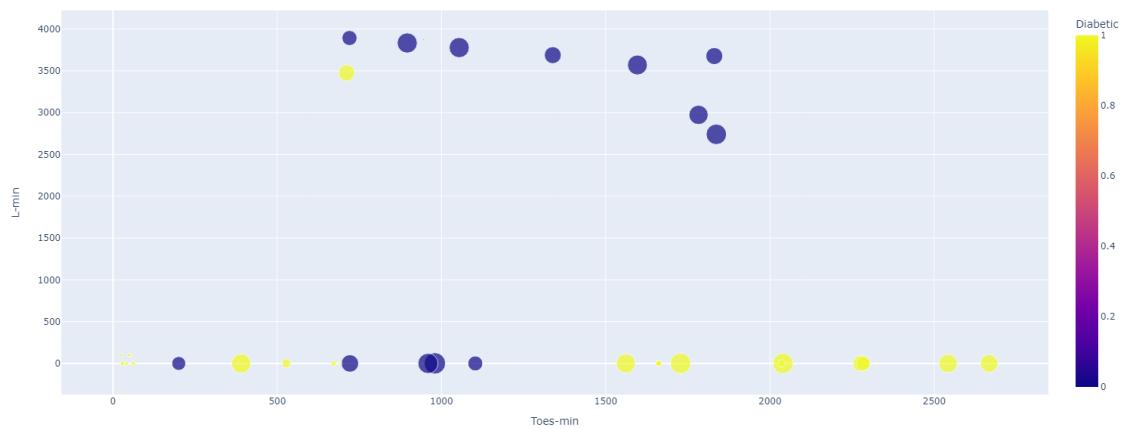
Rys. 3.6. Pairplot minimalnych wartości nacisku



Rys. 3.7. Pairplot średnich wartości nacisku

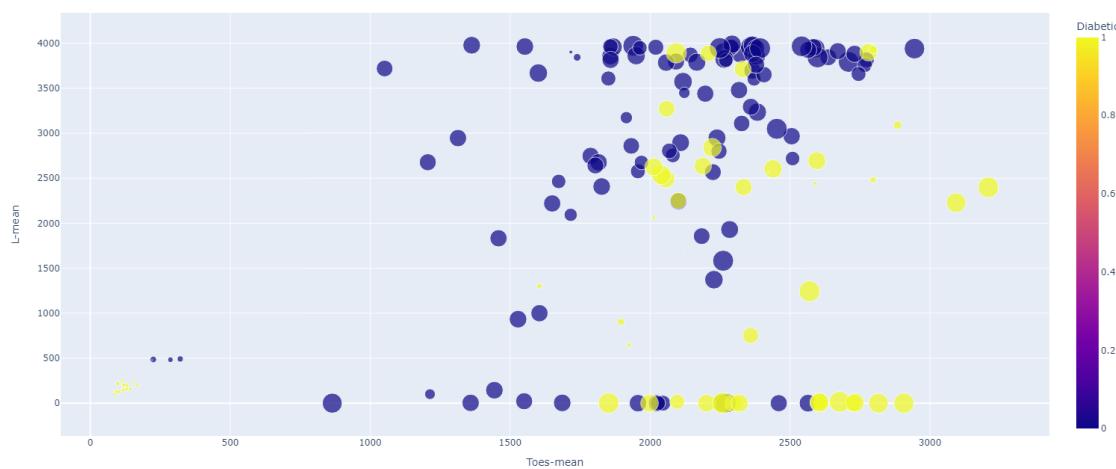


Rys. 3.8. Pairplot maksymalnych wartości nacisku

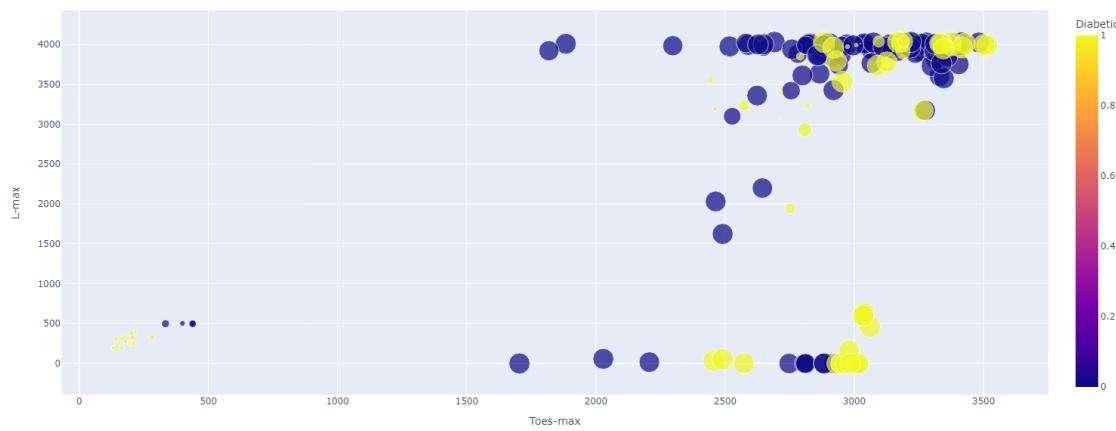


Rys. 3.9. Wykres rozproszenia dla danych minimalnego nacisku

Dwa ostatnie wykresy to wykresy nacisku na różnych czujnikach w zależności od czasu. Przedstawiają one jedną przykładową sesję z grupy cukrzyców i jedną z grupy kontrolnej.

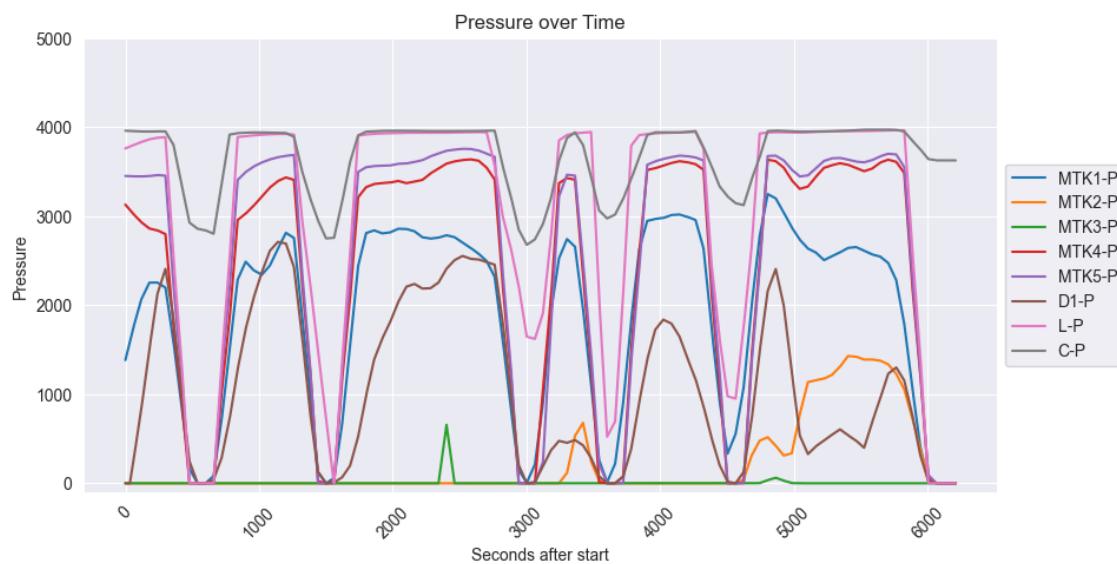


Rys. 3.10. Wykres rozproszenia dla danych średniego nacisku

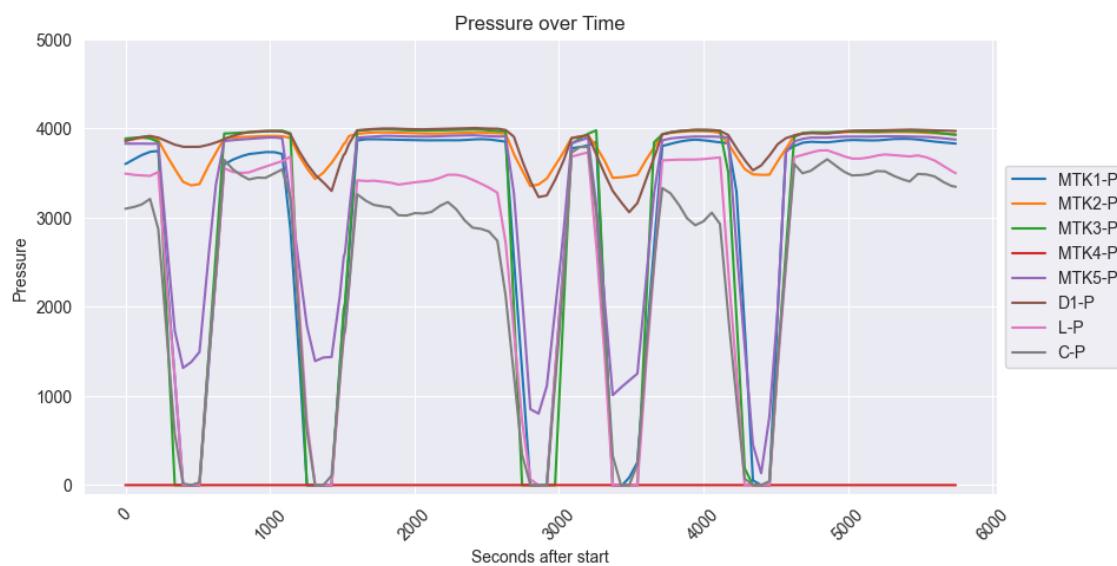


Rys. 3.11. Wykres rozproszenia dla danych maksymalnego nacisku

Pierwszy z nich (Rys. 3.12), należący do zdrowej osoby, ma dużo bardziej zróżnicowane dane na czujnikach, ale również niższe wartości. Drugi (Rys. 3.13) wydaje się być bardziej wyostrzony – dane są dużo bardziej graniczne. Największą różnicę można zaobserwować na czujniku C, który na pierwszym wykresie osiąga wartości od 0 do ok. 1000, a na drugim pełny zakres od 0 do ok. 4000.



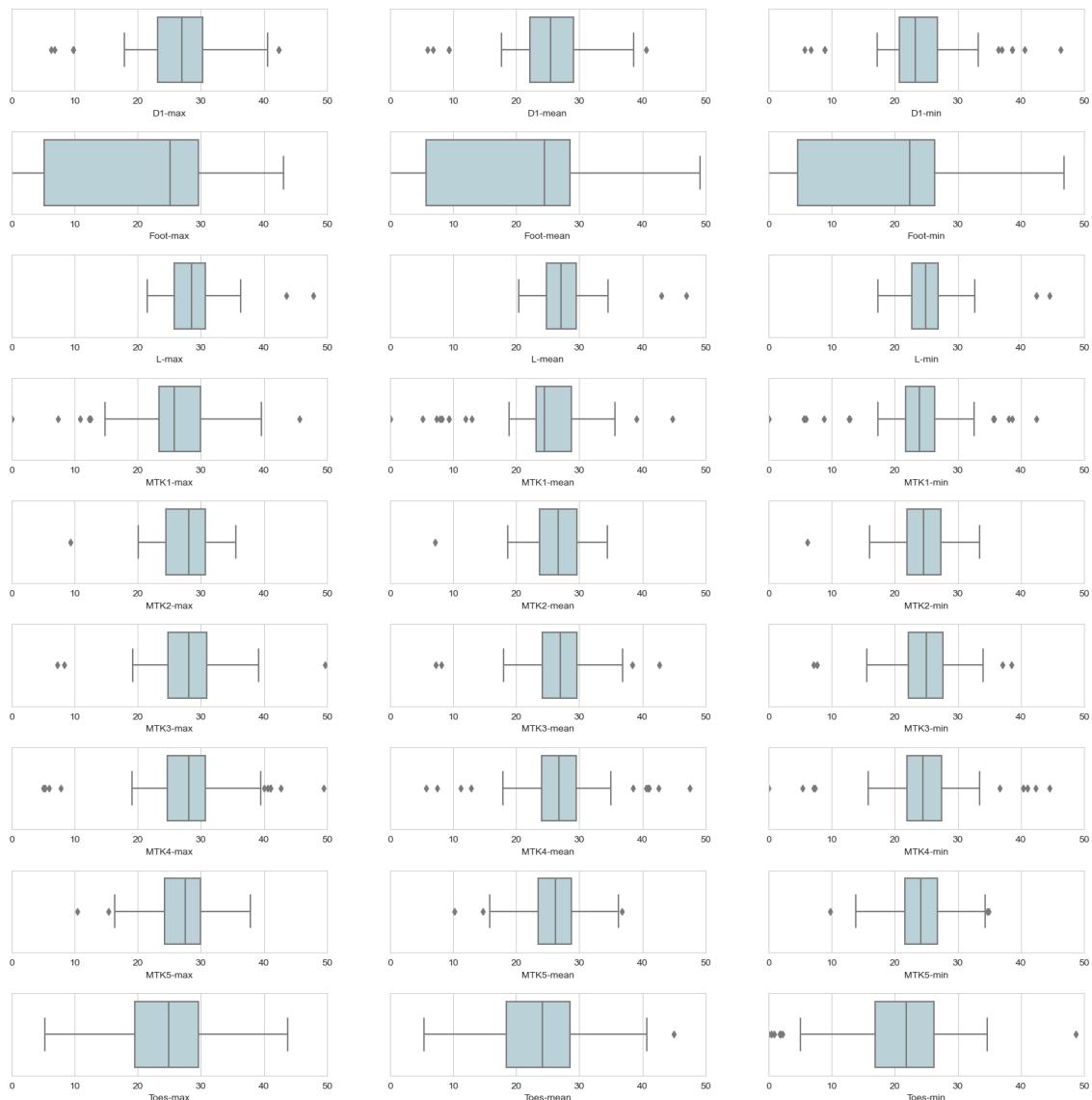
Rys. 3.12. Wykres nacisku w czasie z grupy kontrolnej



Rys. 3.13. Wykres nacisku w czasie z grupy cukrzyców

### 3.4.2. Wykresy związane z rozkładem temperatury

Drugim parametrem, który obejmuje badany zbiór danych, jest temperatura. Czujniki umieszczone są w tych samych miejscach i jest ich tyle samo. Podobnie jak w przypadku nacisku, badano również sumaryczne parametry (sumę ze wszystkich czujników i sumę z MTK1-MTK5).

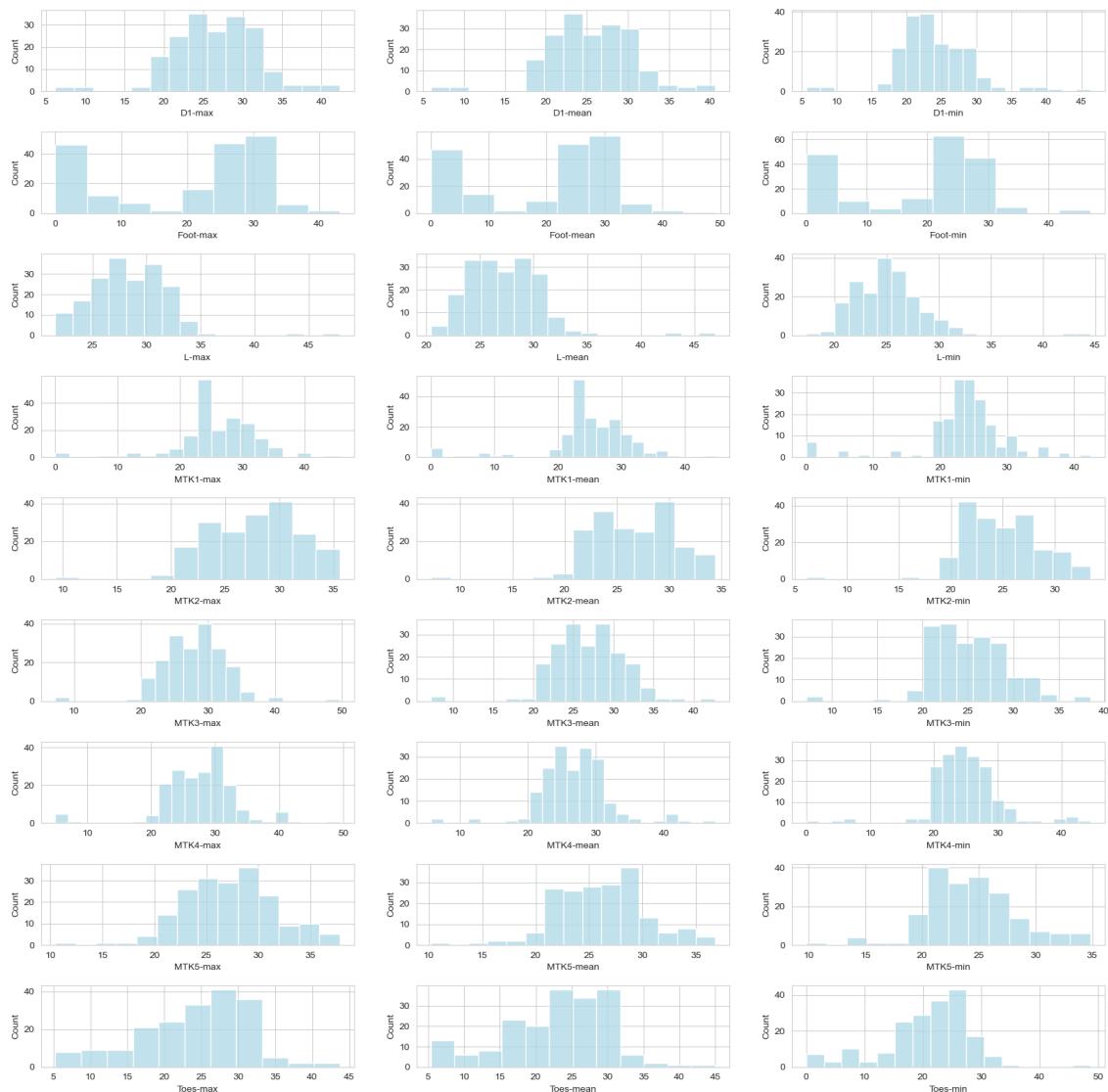


Rys. 3.14. Wykresy pudełkowe rozkładu temperatury

Pierwszymi przygotowanymi wykresami są wykresy pudełkowe (Rys. 3.14). Wykresy, które dotyczą sumy mają dość duże rozmiary, co wynika ze sposobu sumowania danych.

W pozostałych przypadkach nie widać znaczących różnic. Można zwrócić uwagę na dość nisko położoną medianę czujnika MTK1 oraz dużą ilość danych odstających, co świadczy prawdopodobnie o pewnych niedoskonałościach aparatury pomiarowej.

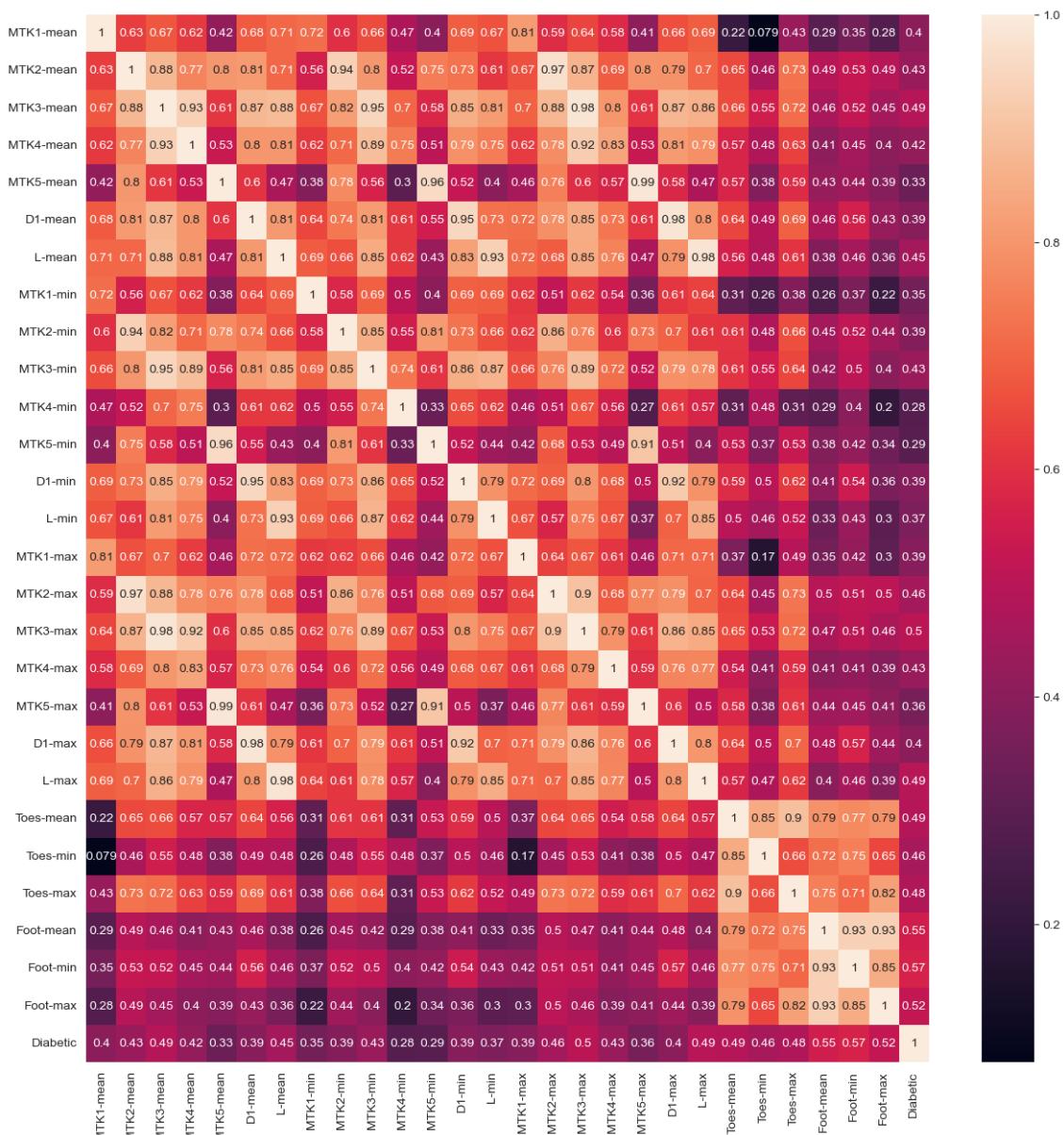
W dalszej kolejności przygotowano histogramy – widać dużo większe zróżnicowanie niż w przypadku badania rozkładu nacisku. Większość danych koncentruje się jednak w centrum wykresu, dane są dużo bardziej symetryczne.



Rys. 3.15. Histogramy rozkładu temperatury

Po wstępnej analizie danych sprawdzono, czy istnieją korelacje między pomiarami temperatury w poszczególnych miejscach, a zmienną wyjaśnianą – w tym celu przygotowano wykres (Rys. 3.16).

Pierwszą obserwacją jest to, że nie widać korelacji ujemnych. Szczególnie wysokie koreacje dotyczą czujników MTK3 oraz L – jest to około 0.5. Co ciekawe, jeszcze wyższe koreacje związane są z minimalną temperaturą zarejestrowaną na całej powierzchni stopy.

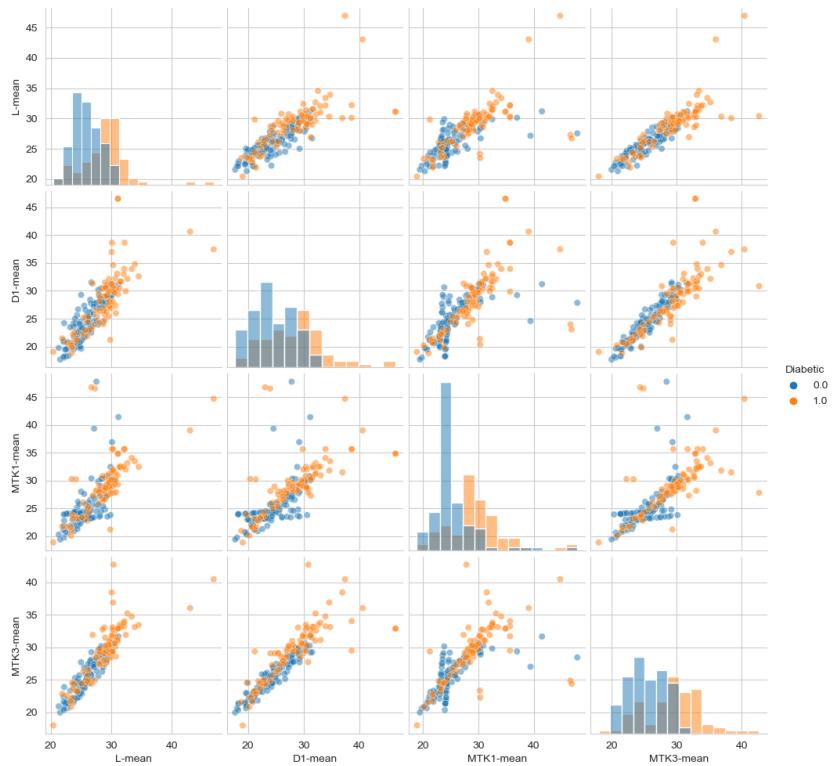


Rys. 3.16. Wykres korelacji między zmiennymi temperatury

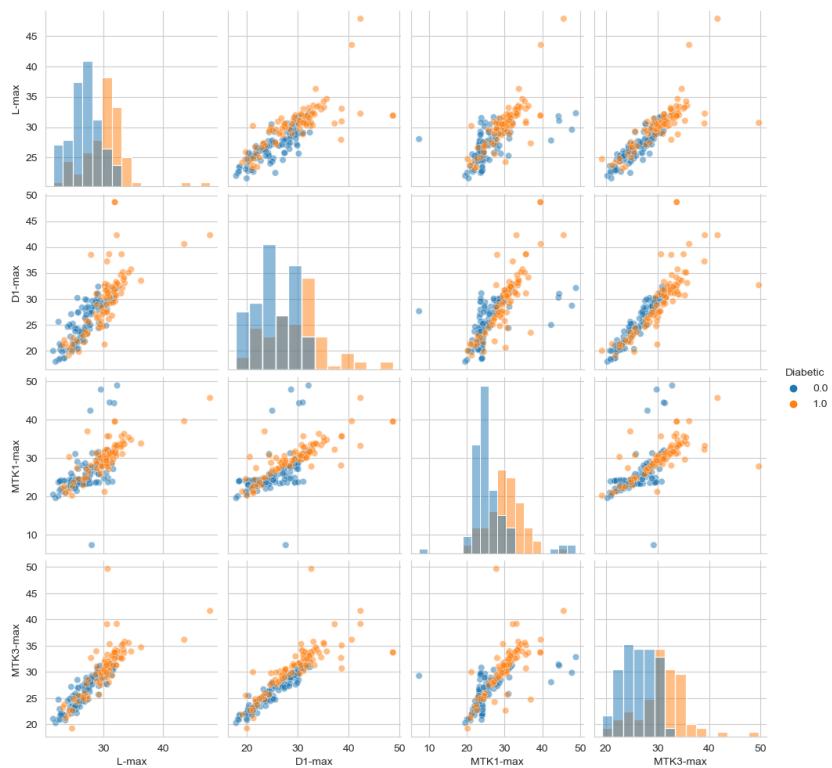
Trzy kolejne wykresy (Rys. 3.17, 3.18, 3.19) to tzw. pairploty. Na wykresie wartości minimalnych, widać pewną zależność, szczególnie na wykresie MTK3 i L – dane dotyczące osób chorych mają na ogół wyższe wartości. Podobne obserwacje widać również na wykresach wartości średnich i maksymalnych.



Rys. 3.17. Pairplot minimalnych wartości temperatury

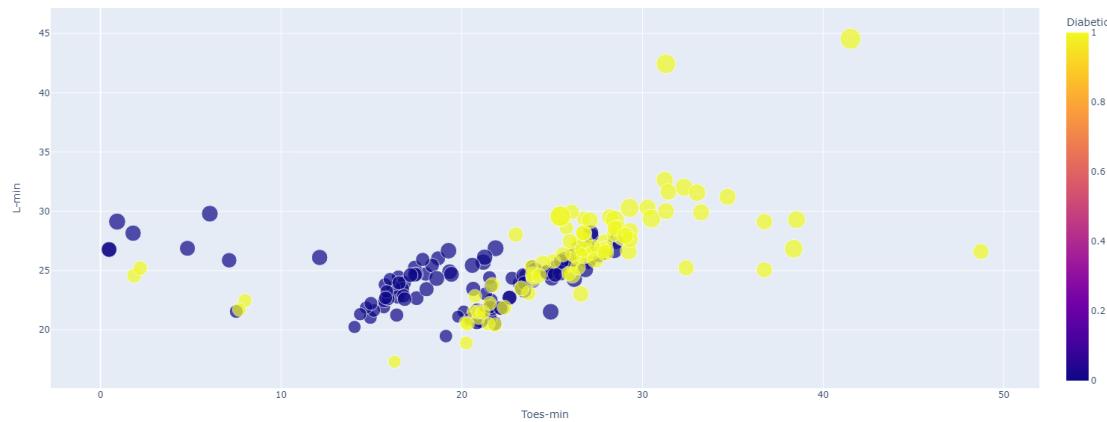


Rys. 3.18. Pairplot średnich wartości temperatury

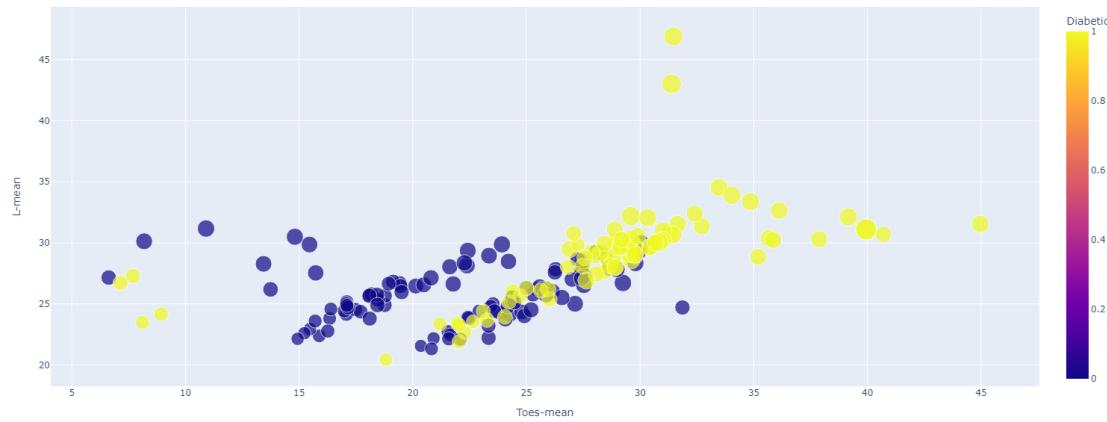


Rys. 3.19. Pairplot maksymalnych wartości temperatury

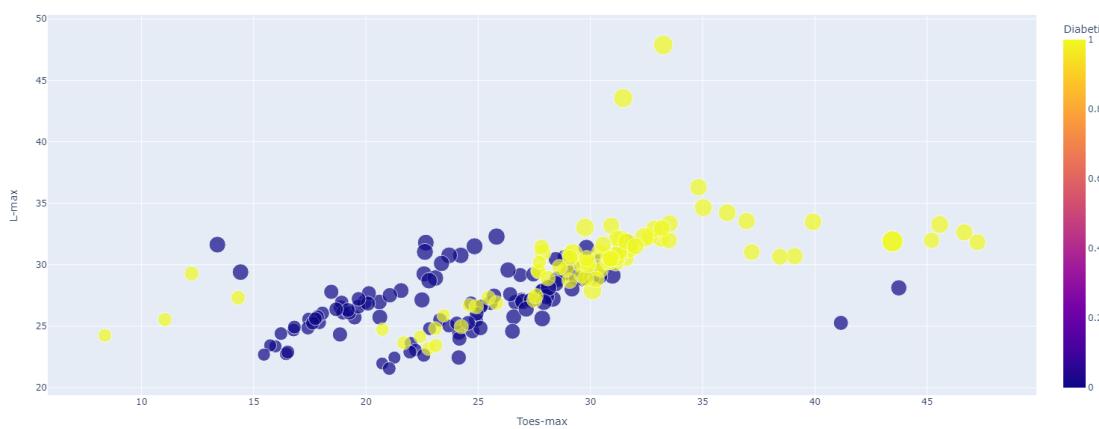
Przedstatni zestaw wykresów to wykresy rozproszenia (Rys. 3.20, 3.21, 3.22). Na każdym z nich widać, czy to dotyczącym minimalnych, średnich czy maksymalnych wartości widać zależność zaobserwowaną już wcześniej – dane pochodzące z grupy kontrolnej są niższe i mniej rozproszone. Można się więc spodziewać dość dobrych wyników predykcji na podstawie temperatury, a być może nawet lepszych niż na podstawie rozkładu nacisku.



Rys. 3.20. Wykres rozproszenia dla danych minimalnej temperatury

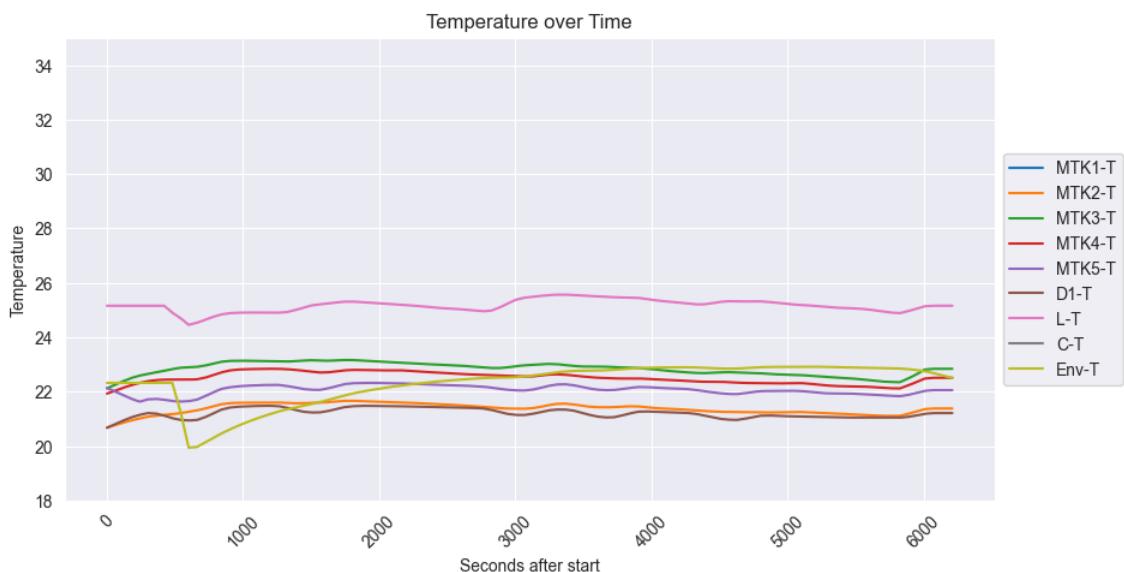


Rys. 3.21. Wykres rozproszenia dla danych średniej temperatury



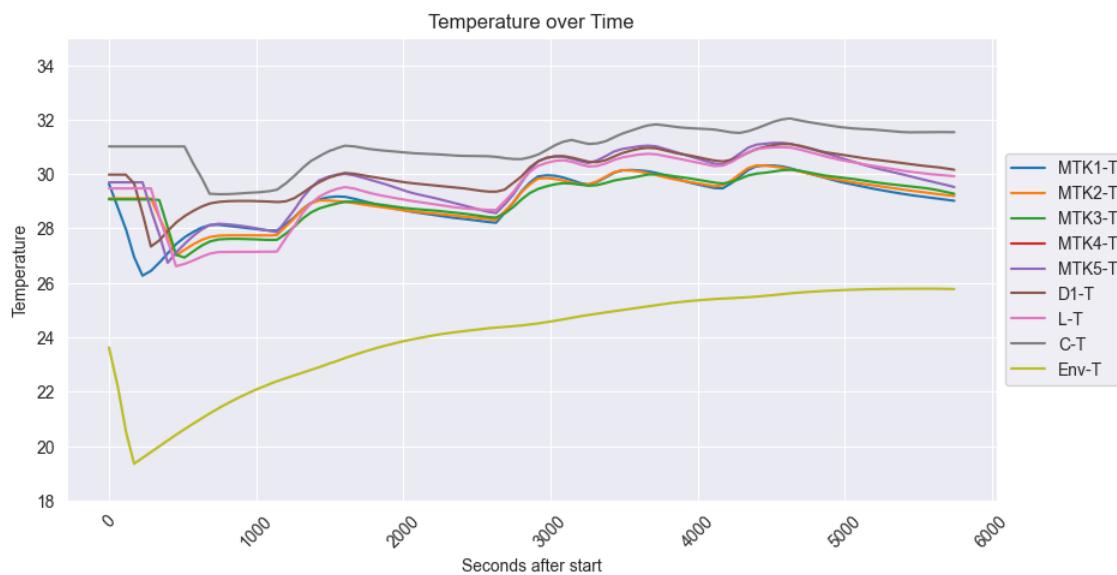
Rys. 3.22. Wykres rozproszenia dla danych maksymalnej temperatury

Na koniec przygotowano wykresy temperatury w zależności od upływu czasu – pierwszy z nich przedstawia temperaturę stopy zdrowego człowieka. Warto zauważyć, że widać dużą liniowość danych, wszystkie są bliskie poziomowi oraz mieszą się w przedziale  $20 - 26^{\circ}\text{C}$ .



Rys. 3.23. Wykres temperatury w czasie z grupy kontrolnej

Zupełnie inaczej prezentują się dane rozkładu temperatury diabetyka – dane nie tylko skokowo rosną i maleją, ale także są dużo wyższe –  $26^{\circ}\text{C}$  to dolna granica obserwacji, a dochodzą one nawet do  $32^{\circ}\text{C}$ .



Rys. 3.24. Wykres temperatury w czasie z grupy cukrzycy

## 3.5. Modele klasyfikacji

Przygotowanie odpowiednich algorytmów uczenia maszynowego wymaga przetestowania wielu modeli oraz parametrów, których wartości często dobierane są eksperymentalnie. Część zastosowanych modeli przedstawiono w poniższych podrozdziałach, umieszczając tam zarówno charakterystykę klasyfikatora, jak i fragment kodu oraz opis ustawianych parametrów.

### 3.5.1. Linear Regression Classifier

Logistic Regression Classifier (klasyfikator regresji logistycznej) [42] jest to liniowy model uczenia maszynowego, który stosowany jest w zadaniach klasyfikacji binarnej (np. przewidywanie czy pacjent zachoruje na chorobę czy nie) oraz wieloklasowej (np. przypisanie zdjęcia do jednej z kilku kategorii).

Klasyfikator ten polega na oszacowaniu prawdopodobieństwa przynależności obserwacji do danej klasy. Do tego celu wykorzystuje się sigmoidalną funkcję aktywacji, która przypisuje wartości pomiędzy 0 a 1. Jeśli wartość ta przekracza pewien próg (domyślnie 0.5), to obserwacja zostaje przypisana do jednej z klas.

W regresji logistycznej wykorzystuje się zmienną zależną binarną (0 lub 1), która określa, do której klasy należy dana obserwacja. Z kolei zmienne niezależne (cechy) są mierzalnymi cechami obserwacji, które służą do przewidywania przynależności do danej klasy.

Klasyfikator regresji logistycznej jest uczony przy użyciu funkcji kosztu, która minimalizuje błąd przewidywania klas. W celu zapobiegania overfittingowi, można stosować regularyzację L1 lub L2, która wprowadza karę za zbyt duże wartości wag modelu.

Regresja logistyczna jest jednym z najczęściej stosowanych algorytmów klasyfikacji w praktyce, ze względu na swoją prostotę, łatwość implementacji oraz szybkość uczenia [43]. Poniższy kod 3.1 to implementacja klasyfikatora typu Logistic Regression Classifier w języku Scala, wykorzystującego bibliotekę Apache Spark. Typ LabeledPoint to klasa z biblioteki Spark MLlib, reprezentująca etykietowany punkt danych, składający się z wektora cech (typu Vector) oraz etykiety (typu Double).

**Listing 3.1.** Logistic Regression Classifier

```
def logisticRegression_classifier(train: RDD[LabeledPoint], test:  
    RDD[LabeledPoint]): Array[Double] = {  
    val logisticRegressionWithLBFGS = new  
        LogisticRegressionWithLBFGS().setNumClasses(2).run(train)  
    val scoreAndLabels = test.map { point =>  
        (logisticRegressionWithLBFGS.predict(point.features), point.label)  
    }  
    effectivenessMeasureMulticlass(scoreAndLabels)  
}
```

W pierwszej linii funkcja tworzy obiekt LogisticRegressionWithLBFGS – klasę implementującą regresję logistyczną wraz z optymalizatorem Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) do optymalizacji funkcji kosztu. Metoda setNumClasses(2) ustawia liczbę klas na 2 – co oznacza, że klasyfikacja jest binarna. Wynik predykcji oraz etykieta punktu są zapisywane jako krotka w zmiennej scoreAndLabels.

### 3.5.2. Naive Bayes Classifier

Naive Bayes Classifier [44] to prosty, ale skuteczny algorytm klasyfikacji, który opiera się na twierdzeniu Bayesa. Naive Bayes jest uważany za „naiwny”, ponieważ zakłada, że wszystkie cechy są niezależne od siebie, co rzadko jest prawdziwe w rzeczywistych problemach klasyfikacji. W przypadku danych zastosowanych w tej pracy oczywiście nie jest to stwierdzenie prawdziwe, ponieważ z całą pewnością istnieją relacje między naciskiem na poszczególne czujniki oraz utworzone grupy czujników.

Naive Bayes wykorzystuje tzw. twierdzenie Bayesa do obliczenia prawdopodobieństwa przynależności do danej klasy na podstawie wartości cech. Algorytm wybiera klasę z najwyższym prawdopodobieństwem. W celu wyznaczenia prawdopodobieństwa dla każdej z klas, wykorzystuje on funkcję gęstości prawdopodobieństwa (PDF) dla każdej cechy w każdej klasie.

Naive Bayes Classifier można podzielić na trzy podstawowe rodzaje: Gaussian Naive Bayes, Multinomial Naive Bayes oraz Bernoulli Naive Bayes.

Gaussian Naive Bayes jest stosowany w przypadku, gdy cechy mają rozkład Gaussa (normalny), natomiast Multinomial Naive Bayes i Bernoulli Naive Bayes stosuje się, gdy zmienne mają rozkład wielomianowy lub Bernoulliego.

W przypadku Multinomial Naive Bayes, cechy są liczbami całkowitymi, które reprezentują liczbę wystąpień danej cechy w dokumencie, podczas gdy w przypadku Bernoulli Naive Bayes, cechy są wartościami binarnymi, które wskazują, czy dana cecha występuje w dokumencie.

Funkcja `naiveBayes_classifier` (Listing 3.2) przyjmuje jako argumenty zbiory danych treningowych i testowych `train` i `test` w formacie `RDD[LabeledPoint]`. Następnie funkcja przekształca wartości cech w zbiorze treningowym, tak aby wszystkie wartości były nieujemne, a następnie tworzy instancję klasyfikatora `NaiveBayes` z ustawieniem wartości `lambda` na 1.0 i typem modelu na „multinomial”.

**Listing 3.2.** Naive Bayes Classifier

---

```
def naiveBayes_classifier(train: RDD[LabeledPoint], test:
    RDD[LabeledPoint]): Array[Double] = {
  val naiveBayes = NaiveBayes.train(train, lambda = 1.0, modelType =
    "multinomial")
  val scoreAndLabels = test.map { point =>
    (naiveBayes.predict(point.features), point.label)
  }
  effectivenessMeasureMulticlass(scoreAndLabels)
}
```

---

Lambda w klasyfikatorze Naive Bayes to parametr regularyzacji, który pomaga zapobiegać overfittingowi. Wartość lambda określa, jak bardzo mniej ważne cechy powinny być pomijane w procesie klasyfikacji. Ustawienie lambda na wartość 1.0 oznacza, że przy obliczaniu prawdopodobieństw, każda cecha jest traktowana jako ważna.

Typ modelu w klasyfikatorze Naive Bayes określa jak algorytm nauczył się funkcji prawdopodobieństwa dla każdej klasy. W Sparku, istnieją dwa typy modelu, „bernoulli” i „multinomial”. W przypadku „multinomial” wartości cech są liczbami całkowitymi, co oznacza, że mogą występować wartości większe niż 1. W tym przypadku, typ modelu „multinomial” został wybrany, ponieważ wartości cech są liczbami całkowitymi, co oznacza, że modelowanie z użyciem „multinomial” będzie bardziej adekwatne do danych.

### 3.5.3. Descision Tree Classifier

Klasyfikator typu Decision Tree Classifier jest jednym z popularnych algorytmów wykorzystywanych do klasyfikacji danych. Algorytm ten buduje drzewo decyzyjne, które na podstawie wartości cech wejściowych klasyfikuje nowe przykłady do jednej z klas.

Proces budowania drzewa zaczyna się od korzenia i polega na rekurencyjnym podziale danych na mniejsze podzbiory, aż do momentu uzyskania liści, czyli węzłów, które nie są już dzielone na mniejsze podzbiory. Każdy węzeł drzewa reprezentuje pewną cechę wejściową, na podstawie której następuje podział danych. Węzły w drzewie są połączone krawędziami, które reprezentują decyzje podjęte na podstawie wartości cechy.

Przy budowie drzewa decyzyjnego, algorytm stara się znaleźć takie podziały danych, które najlepiej rozdzielają przykłady należące do różnych klas. Do tego celu wykorzystywane są różne metryki oceny jakości podziału, takie jak Gini impurity, entropia czy błąd klasyfikacji.

Po zbudowaniu drzewa decyzyjnego, klasyfikacja nowych przykładów polega na przejściu od korzenia do liścia, w którym przypisana jest klasa [45]. Każdy węzeł drzewa reprezentuje pewną cechę wejściową, a na podstawie wartości tej cechy podejmowana jest decyzja o przejściu do lewego lub prawego poddrzewa. W liściach znajdują się etykiety klas, do których przypisane są obserwacje należące do danego liścia.

#### **Listing 3.3. Decision Tree Classifier - wariant 1**

---

```
def decisionTree_classifier(train: RDD[LabeledPoint], test:
    RDD[LabeledPoint]): Array[Double] = {
  val decisionTree = DecisionTree.trainClassifier(train, 2, Map[Int,
    Int](), "gini", 5, 32)
  val scoreAndLabels = test.map { point =>
    (decisionTree.predict(point.features), point.label)
  }
  effectivenessMeasureMulticlass(scoreAndLabels)
}
```

---

#### **Listing 3.4. Decision Tree Classifier - wariant 2**

---

```
def decisionTree_classifier2(train: RDD[LabeledPoint], test:
    RDD[LabeledPoint]): Array[Double] = {
  val decisionTree = DecisionTree.trainClassifier(train, 2, Map[Int,
    Int](), "entropy", 10, 32)
  val scoreAndLabels = test.map { point =>
    (decisionTree.predict(point.features), point.label)
  }
  effectivenessMeasureMulticlass(scoreAndLabels)
}
```

---

Przygotowano dwa (Listing 3.3, listing 3.4) warianty tego modelu, różniące się kryterium podziału drzewa.

- `Map[Int, Int]()` – pusty słownik, który oznacza, że nie ma ograniczeń na liczbę kategorii w każdej kolumnie danych,

- „entropy” oraz „gini” – kryterium podziału drzewa decyzyjnego, które określa, jakie kolumny danych mają być użyte do podziału wewnętrznego. W pierwszym przypadku używane jest kryterium entropii, w drugim impurity w oparciu o entropię,
- 10 oraz 32 – maksymalna głębokość drzewa decyzyjnego,

Maksymalna głębokość drzewa określa maksymalną liczbę warstw w drzewie decyzyjnym, a maksymalna liczba liści określa maksymalną liczbę końcowych liści w drzewie. Te parametry wpływają na złożoność drzewa i mogą pomóc uniknąć overfittingu.

### 3.5.4. Random Forest Classifier

Random Forest Classifier [46] to algorytm uczenia maszynowego, który wykorzystuje zbiór drzew decyzyjnych do klasyfikacji. W ramach algorytmu tworzy się wiele drzew decyzyjnych, gdzie każde drzewo jest budowane na podstawie losowo wybranych podzbiorów danych treningowych i cech. Następnie klasyfikacja nowych przykładów odbywa się przez zastosowanie każdego z drzew i wybranie etykiety, która zostanie przypisana z największym prawdopodobieństwem przez większość drzew.

Podczas budowania drzewa w Random Forest Classifier wybierana jest losowa podgrupa zmiennych objaśniających. Dzięki temu algorytm nie jest podatny na problemy związane z nadmierną dopasowaniem i zapewnia zwykle dobre wyniki predykcyjne.

W ramach klasyfikatora Random Forest Classifier najczęściej stosowanymi parametrami są liczba drzew w lesie, maksymalna liczba cech branych pod uwagę przy budowie drzewa oraz maksymalna głębokość drzewa. Ważnymi cechami Random Forest Classifier są również łatwość w implementacji, wysoka odporność na wartości odstające, a także możliwość obsługi danych kategorycznych.

**Listing 3.5.** Random Forest Classifier language

---

```
def randomForest_classifier(train: RDD[LabeledPoint], test:
    RDD[LabeledPoint]): Array[Double] = {
  val numClasses = 2
  val categoricalFeaturesInfo = Map[Int, Int]()
  val impurity = "gini"
  val maxDepth = 5
  val maxBins = 32

  val model = DecisionTree.trainClassifier(train, numClasses,
    categoricalFeaturesInfo,
    impurity, maxDepth, maxBins)

  val scoreAndLabels = test.map { point =>
```

---

---

```
        (model.predict(point.features), point.label)
    }
    effectivenessMeasureMulticlass(scoreAndLabels)
}
```

---

W zaprezentowanej funkcji (Listing 3.5) tworzony jest pojedynczy drzewo decyzyjne za pomocą metody `trainClassifier` dostarczonej przez Spark, z następującymi ustawieniami:

- `numClasses` – liczba klas wyjściowych (w tym przypadku 2),
- `categoricalFeaturesInfo` – informacja o kolumnach oznaczonych jako kategoryczne,
- `impurity` – miara impurity użyta do wyboru atrybutu dzielącego węzeł (w tym przypadku „gini”),
- `maxDepth` – maksymalna głębokość drzewa,
- `maxBins` – maksymalna liczba kubeków dla atrybutów ciągłych.

Następnie dokonuje się predykcji dla zbioru testowego, a wyniki są porównywane z rzeczywistymi etykietami. Ostateczna skuteczność modelu jest obliczana przez funkcję `effectivenessMeasureMulticlass`.

### 3.5.5. Support Vector Machine Classifier

Support Vector Machine (SVM) [47] to klasyfikator binarny, który w oparciu o analizę danych treningowych tworzy hiperpłaszczyznę rozdzielającą elementy dwóch klas. Dla danych wejściowych, które mają więcej niż dwie klasy, stosuje się podejście „jeden przeciwko reszcie” (ang. one-vs-rest), a pozostałych przypadkach „jeden przeciwko jednemu” (ang. one-vs-one), w którym tworzone są klasyfikatory binarne.

W SVM dane wejściowe są reprezentowane przez wektory cech, które są mapowane na przestrzeń wielowymiarową. Klasyfikator SVM znajduje taką hiperpłaszczyznę, która najlepiej rozdziela elementy różnych klas, tak aby margines między nimi był jak największy. Margines to odległość między hiperpłaszczyzną, a najbliższymi punktami ze zbiorów danych. Im większy margines, tym większa szansa na skuteczne klasyfikowanie nowych danych.

SVM ma wiele wariantów, które różnią się między sobą funkcją kernela (np. liniową, wielomianową, radialną), a także parametrami takimi jak koszt błędu klasyfikacji (C), margines, stopień wielomianu, parametr gamma itp.

**Listing 3.6.** Support Vector Machine Classifier

---

```
def supportVectorMachine_classifier(train: RDD[LabeledPoint], test:
    RDD[LabeledPoint]): Array[Double] = {
```

---

```
val svm = new LinearSVC()
val ovr = new OneVsRest().setClassifier(svm)
val evaluator = new
    MulticlassClassificationEvaluator().setMetricName("accuracy")

val paramGrid = new ParamGridBuilder()
    .addGrid(svm.regParam, Array(0.1, 0.01))
    .addGrid(svm.maxIter, Array(10, 100))
    .addGrid(svm.fitIntercept, Array(true, false))
    .build()

val cv = new CrossValidator()
    .setEstimator(ovr)
    .setEvaluator(evaluator)
    .setEstimatorParamMaps(paramGrid)
    .setNumFolds(3)

val cvModel = cv.fit(train)

val bestModel =
    cvModel.bestModel.asInstanceOf[OneVsRestModel].getClassifier.asInstanceOf[LinearSVC]
val bestRegParam = bestModel.getRegParam
val bestMaxIter = bestModel.getMaxIter
val bestFitIntercept = bestModel.getFitIntercept

val predictions = cvModel.transform(test)
val scoreAndLabels = test.map { point =>
    (predictions.predict(point.features), point.label)
}
effectivenessMeasureMulticlass(scoreAndLabels)
}
```

Aby dobrać odpowiednie parametry dla klasyfikatora SVM, w zamieszczonym fragmencie kodu (listing 3.6) zdecydowano się użyć technik takich jak przeszukiwanie siatki (grid search). Metoda ta polega na wypróbowaniu różnych kombinacji parametrów i wyborze takiej, która daje najlepsze wyniki na zestawie walidacyjnym.

Warto zaznaczyć, że w praktyce dobór odpowiednich parametrów jest kluczowy dla uzyskania dobrych wyników z użyciem SVM. W zależności od problemu i zbioru danych, należy dostosować wartości parametrów, takich jak np. typ funkcji jądrowej, wartość parametru C.

### 3.5.6. Gradient Boosting Classifier

Klasyfikator typu Gradient Boosting Classifier [48] jest metodą uczenia maszynowego z nadzorem, opartą na technice uczenia zespołowego (ensemble learning), która polega na tworzeniu sekwencji prostych modeli uczących, zwanych „boosters”. Są one uzupełniane w kolejnych iteracjach w celu minimalizacji błędu predykcji. W każdej iteracji kolejny prosty model jest dostosowywany do reszt pozostawionych przez poprzednie modele, co oznacza, że wiele słabszych klasyfikatorów jest łączonych w celu uzyskania lepszego modelu. Gradient Boosting Classifier jest jednym z najskuteczniejszych klasyfikatorów w uczeniu maszynowym. W praktyce, Gradient Boosting Classifier może być stosowany do problemów klasyfikacji binarnej lub wieloklasowej.

**Listing 3.7.** Gradient Boosting Classifier - wariant 1

```
def gradientBoosting_classifier(train: RDD[LabeledPoint], test:
    RDD[LabeledPoint]): Array[Double] = {
  val boostingStrategy = BoostingStrategy.defaultParams("Classification")
  boostingStrategy.numIterations = 3 // Note: Use more iterations in
    practice.
  boostingStrategy.treeStrategy.numClasses = 2
  boostingStrategy.treeStrategy.maxDepth = 5
  boostingStrategy.treeStrategy.categoricalFeaturesInfo = Map[Int, Int]()
  val model = GradientBoostedTrees.train(train, boostingStrategy)
  val scoreAndLabels = test.map { point =>
    (model.predict(point.features), point.label)
  }
  effectivenessMeasureMulticlass(scoreAndLabels)
}
```

W funkcji zamieszczonej w listingu 3.7 ustalono trzy kluczowe parametry Gradient Boosting Classifier:

- „numIterations” – określa liczbę iteracji wykonywanych przez klasyfikator. Im większa liczba iteracji, tym większa szansa na poprawę wyniku, ale także większe ryzyko przeuczenia się modelu.
- „treeStrategy.maxDepth” – określa maksymalną głębokość drzewa decyzyjnego. Im większa wartość, tym drzewo będzie bardziej skomplikowane i bardziej dopasowane do danych treningowych. Jednak im bardziej skomplikowane drzewo, tym większe ryzyko przeuczenia się modelu.

- „treeStrategy.categoricalFeaturesInfo” – określa, które cechy (features) w zbiorze danych są kategoryczne. W tym przypadku brak informacji o cechach kategorycznych.

Następnie, klasyfikator Gradient Boosting jest trenowany na zbiorze treningowym i używany do przewidywania etykiet na zbiorze testowym. Wyniki są następnie porównywane z rzeczywistymi etykietami i mierzone za pomocą funkcji effectivenessMeasureMulticlass.

**Listing 3.8.** Gradient Boosting Classifier - wariant 2

---

```
def gradientBoosting_classifier2(train: RDD[LabeledPoint], test:
    RDD[LabeledPoint]): Array[Double] = {
  val boostingStrategy = BoostingStrategy.defaultParams("Classification")
  boostingStrategy.numIterations = 20 // Note: Use more iterations in
  practice.
  boostingStrategy.treeStrategy.numClasses = 2
  boostingStrategy.treeStrategy.maxDepth = 10
  boostingStrategy.treeStrategy.categoricalFeaturesInfo = Map[Int, Int]()
  boostingStrategy.treeStrategy.maxBins = 32
  boostingStrategy.learningRate = 0.1

  val model = GradientBoostedTrees.train(train, boostingStrategy)
  val scoreAndLabels = test.map { point =>
    (model.predict(point.features), point.label)
  }
  effectivenessMeasureMulticlass(scoreAndLabels)
}
```

---

W związku z tym, że w przypadku tego algorytmu odpowiednie wartości parametrów są szczególnie istotne, przygotowano również inny wariant, zaprezentowany w listingu 3.8. Znacznie zwiększo w nim liczbę iteracji oraz głębokość drzewa.

### 3.5.7. K-Nearest Neighbours Classifier

Klasyfikator k-nearest neighbors (kNN) [49] to prosty algorytm uczenia maszynowego, który może być stosowany zarówno do zadania klasyfikacji jak i regresji. W przypadku klasyfikacji, kNN przewiduje klasę obiektu na podstawie k najbliższych sąsiadów w zbiorze treningowym. Kolejne kroki algorytmu przedstawiają się następująco:

1. Oblicz odległość pomiędzy testowym obiektem a każdym elementem ze zbioru treningowego, stosując odpowiednią metrykę, np. euklidesową.
2. Wybierz k elementów treningowych o najmniejszej odległości od testowego obiektu.
3. Przypisz testowy obiekt do klasy, która występuje najczęściej wśród k najbliższych sąsiadów.

4. Parametr  $k$  określa liczbę najbliższych sąsiadów branych pod uwagę w decyzji o przypisaniu klasy. Wartość  $k$  jest zwykle wybierana eksperymentalnie, z uwzględnieniem charakterystyki zbioru danych.

W fragmencie kodu przedstawionym na listingu 3.9 zastosowano KNN.train do utworzenia modelu  $k$ -najbliższych sąsiadów. Zmienna  $k$  określa, ile sąsiadów zostanie użytych do prognozowania etykiet dla nowych obserwacji. Wynik zwracany przez funkcję knnModel.predict to przewidywana etykieta dla nowych danych testowych, a następnie porównywana jest z rzeczywistą etykietą point.label. Ostatecznie, funkcja effectivenessMeasureMulticlass jest używana do obliczenia skuteczności modelu.

---

**Listing 3.9.** k-Nearest Neighbors Classifier

---

```
def kNearestNeighbors_classifier(train: RDD[Vector], test:  
    RDD[LabeledPoint]): Array[Double] = {  
    val k = 1  
    val knnModel = KNN.train(train, k)  
    val scoreAndLabels = test.map { point =>  
        (knnModel.predict(point.features), point.label)  
    }  
    effectivenessMeasureMulticlass(scoreAndLabels)  
}
```

---

Podsumowując, w rozdziale omówiono zastosowane w badaniach metody uczenia maszynowego oraz zbiór danych, na którym opiera się analiza oraz proces przygotowania danych do analizy. Scharakteryzowano zbiór danych i sformułowano wstępne wnioski w zakresie oceny szans na predykcję zachorowania na cukrzycę u badanego pacjenta na podstawie dowolnego parametru – ciśnienia i temperatury. Przygotowano również modele uczenia maszynowego, opisano poszczególne algorytmy, dołączając do nich również odpowiednie fragmenty kodu.



## **4. Wyniki**

Analiza danych, opisana w poprzednim rozdziale, pomogła ocenić zbiór danych oraz możliwości rozróżnienia osób zdrowych i cukrzyków w oparciu o dostarczone dane. Ten rozdział poświęcony jest wykorzystanym metodom uczenia maszynowego, rozpocznie się on od przedstawienia obliczonych korelacji danych, a następnie przedstawiona zostanie ewaluacja algorytmów dla każdego z parametrów – rozkładu nacisku i temperatury. Tabele znajdujące się w kolejnych sekcjach (4.2, 4.3) rozdziału zawierają wyniki eksperymentów, polegających na uruchomieniu przygotowanych i opisanych wcześniej modeli uczenia maszynowego.

### **4.1. Korelacje danych**

Poniższe tabele prezentują korelacje między wszystkimi możliwymi danymi. Uszeregowano je w kolejności malejącej, uwzględniając jedynie skalę korelacji, a nie to, czy jest ona dodatnia czy ujemna.

**Tabela 4.1.** Korelacje wartości czujników nacisku

Nazwa czujnika	Wartość korelacji
L	0,39
D1	0,34
MTK4	0,20
Toes	0,20
Foot	0,19
MTK2	0,19
MTK3	0,17
MTK5	0,16
MTK1	0,14

Rozpoczęto od analizy korelacji ciśnienia. Pierwsza tabela 4.1 przedstawia korelacje dla kolejnych czujników ciśnienia, a druga (4.2) zależności między kolejnymi czujnikami, ale z

dodatkowym podziałem na dane minimalne, maksymalne i średnie wartości. Podobnie jak w przypadku analizy wykresów korelacji, w tym przypadku również wyniki wskazują na wysoką korelację między czujnikami D1-maksymalnym, D1-średnim, L-średnim i L-minimalnym. Ciekawą obserwacją jest fakt, że kolejnym z istotnych cech jest MTK2-minimalna, jednak wartość maksymalna nie wykazuje najmniejszej korelacji. Podobnie w przypadku tabeli 4.1 – najwyższe wartości korelacji powstają w oparciu o czujniki D1, L oraz MTK4.

**Tabela 4.2.** Korelacje cech nacisku

Cecha	Wartość korelacji
D1-max	-0,56
D1-mean	-0,47
L-min	-0,45
L-mean	-0,41
MTK2-min	0,37
MTK4-min	0,35
MTK3-min	0,34
L-max	-0,30
Foot-max	-0,29
Toes-max	-0,27
MTK1-min	0,25
MTK5-max	-0,24
Toes-min	0,24
MTK2-mean	0,20
Foot-mean	-0,18
MTK5-min	0,16
MTK3-mean	0,15
MTK4-max	-0,14
MTK1-mean	0,11
Toes-mean	-0,10
Foot-min	0,10
MTK4-mean	0,10
MTK5-mean	-0,07
MTK1-max	-0,06
MTK3-max	0,02
D1-min	-0,01
MTK2-max	0,00

W dalszej kolejności wykonano podobne tabele dla rozkładu temperatury. Najwyższe wartości korelacji dotyczą sumarycznych wartości z całej powierzchni stopy. Kolejnym istotnym miejscem pomiaru temperatury, jest umiejscowienie czujnika L – wskazują na to obie tabele – 4.3 oraz 4.4.

**Tabela 4.3.** Korelacje wartości cech temperatury

Column	Correlation
Foot-mean	0,53
Foot-max	0,53
Foot-min	0,52
L-max	0,49
L-mean	0,45
Toes-mean	0,41
Toes-max	0,41
MTK3-max	0,40
Toes-min	0,39
MTK3-mean	0,38
MTK2-mean	0,37
L-min	0,37
MTK2-min	0,35
MTK2-max	0,35
MTK3-min	0,34
MTK1-max	-0,29
MTK4-max	0,28
MTK4-mean	0,27
MTK1-mean	-0,26
MTK1-min	-0,26
D1-max	0,26
D1-mean	0,25
MTK4-min	0,21
D1-min	0,21
MTK5-max	0,17
MTK5-mean	0,16
MTK5-min	0,15

**Tabela 4.4.** Korelacje wartości czujników temperatury

Columns' group	Correlation mean
Foot	0,53
L	0,44
Toes	0,40
MTK3	0,38
MTK2	0,36
MTK1	0,27
MTK4	0,26
D1	0,24
MTK5	0,16

## 4.2. Ewaluacja dla rozkładu ciśnienia

Przygotowane modele przetrenowano i przetestowano na czterech różnych zbiorach:

- na wszystkich cechach i czujnikach (zarówno minimalnych, maksymalnych, średnich, jak i sumarycznych),
- na wszystkich cechach z wyłączeniem podsumowujących,
- na podstawie trzech czujników o najwyższych współczynnikach korelacji (tabela 4.1),
- na podstawie pięciu cech o najwyższych współczynnikach korelacji (tabela 4.2).

**Tabela 4.5.** Walidacja modeli – wszystkie cechy rozkładu nacisku

Nazwa modelu	Dokładność	Precyza	Czułość	F1
Linear Regression Classifier	0.805	0.917	0.611	0.733
Naive Bayes Classifier	0.707	0.800	0.444	0.571
Decision Tree Classifier	0.878	0.882	0.833	0.857
Decision Tree Classifier 2	0.756	0.786	0.611	0.688
Random Forest Classifier	0.878	0.882	0.833	0.857
Support Vector Machine Classifier	0.732	0.684	0.722	0.703
Gradient Boosting Classifier	0.878	0.882	0.833	0.857
Gradient Boosting Classifier 2	0.854	0.833	0.833	0.833
K Nearest Neighbors Classifier	0.561	0.000	0.000	0.000

W przypadku modeli rozkładu nacisku testowanego na pierwszym zbiorze (tabela 4.5), trzy modele wykazały te same wartości. Decision Tree Classifier, Random Forest Classifier oraz Gradient Boosting Classifier osiągnęły wartość dokładności na poziomie 88% oraz czułość – 83%. W przypadku drugiego zbioru cech (tabela 4.6) – bez sumarycznych wartości ze wszystkich czujników MTK oraz wszystkich sumarycznych wartości dokładności i czułości modelu są niższe – jest to odpowiednio 84% i 71%.

**Tabela 4.6.** Walidacja modeli – wszystkie cechy rozkładu nacisku bez podsumowujących

Nazwa modelu	Dokładność	Precyzja	Czułość	F1
Linear Regression Classifier	0.742	0.714	0.714	0.714
Naive Bayes Classifier	0.806	0.750	0.857	0.800
Decision Tree Classifier	0.839	0.909	0.714	0.800
Decision Tree Classifier 2	0.774	0.769	0.714	0.741
Random Forest Classifier	0.839	0.909	0.714	0.800
Support Vector Machine Classifier	0.774	0.818	0.643	0.720
Gradient Boosting Classifier	0.839	0.909	0.714	0.800
Gradient Boosting Classifier 2	0.839	0.909	0.714	0.800
K Nearest Neighbors Classifier	0.548	0.000	0.000	0.000

**Tabela 4.7.** Walidacja modeli – pięć najważniejszych cech nacisku

Nazwa modelu	Dokładność	Precyzja	Czułość	F1
Linear Regression Classifier	0.735	0.643	0.692	0.667
Naive Bayes Classifier	0.676	0.563	0.692	0.621
Decision Tree Classifier	0.824	0.733	0.846	0.786
Decision Tree Classifier 2	0.882	0.846	0.846	0.846
Random Forest Classifier	0.824	0.733	0.846	0.786
Support Vector Machine Classifier	0.706	0.600	0.692	0.643
Gradient Boosting Classifier	0.824	0.733	0.846	0.786
Gradient Boosting Classifier 2	0.794	0.688	0.846	0.759
K Nearest Neighbors Classifier	0.618	0.000	0.000	0.000

Wartości ewaluacji modeli opartego o cechy z najwyższym współczynnikiem korelacji przedstawiono w tabeli 4.7. Dokładność modelu spadła do 82%, ale czułość, która jest bardziej istotna dla problematyki badań osiągnęła 85%.

**Tabela 4.8.** Walidacja modeli – trzy najważniejsze czujniki nacisku

Nazwa modelu	Dokładność	Precyzja	Czułość	F1
Linear Regression Classifier	0.844	0.813	0.765	0.788
Naive Bayes Classifier	0.822	0.737	0.824	0.778
Decision Tree Classifier	0.889	0.800	0.941	0.865
Decision Tree Classifier 2	0.822	0.714	0.882	0.789
Random Forest Classifier	0.889	0.800	0.941	0.865
Support Vector Machine Classifier	0.822	0.909	0.588	0.714
Gradient Boosting Classifier	0.889	0.800	0.941	0.865
Gradient Boosting Classifier 2	0.844	0.727	0.941	0.821
K Nearest Neighbors Classifier	0.622	0.000	0.000	0.000

Ostatnim zbiorem cech są wartości trzech czujników (minimalne, maksymalne i średnie) które osiągnęły najwyższe wyniki – tabela 4.8. Dokładność modeli wyniosła 89%, a czułość aż 94% – wynika z tego że efektywność modelu jest lepsza w przypadku zastosowania trzech czujników, niż w przypadku opierania modeli o wszystkie dostarczone.

### 4.3. Ewaluacja dla rozkładu temperatury

Po przygotowaniu modeli i ich efektywności dla rozkładu nacisku, przygotowano podobne zestawienie dla rozkładu temperatury. Podobnie jak w poprzednim przypadku modele trenowano i testowano na czterech zbiorach:

- na wszystkich cechach i czujnikach (zarówno minimalnych, maksymalnych, średnich, jak i sumarycznych),
- na wszystkich cechach z wyłączeniem podsumowujących,
- na podstawie pięciu cechach o najwyższych współczynnikach korelacji (tabela 4.3),
- na podstawie trzech czujników o najwyższych współczynnikach korelacji (tabela 4.4).

Pierwsza tabela – 4.9 wskazuje na wysoką czułość dla modelu „Naive Bayes Classifier” – 95%, jednak należy zwrócić uwagę na niską dokładność tego modelu – 76%. Jedyny model, który wykazał wyższą dokładność to Decision Tree Classifier (78%), jednak wykazuje on niższą czułość – jedynie 67%.

Podobnie w przypadku użycia wszystkich cech z wyjątkiem sumarycznych – najlepszym klasyfikatorem okazał się Naive Bayes Classifier (tabela 4.10) – jednak zarówno jego dokładność jak i czułość okazały się niższe – odpowiednio 71% oraz 90%.

**Tabela 4.9.** Walidacja modeli – wszystkie cechy rozkładu temperatury

Nazwa modelu	Dokładność	Precyzja	Czułość	F1
Linear Regression Classifier	0.676	0.714	0.556	0.625
Naive Bayes Classifier	0.757	0.667	0.947	0.800
Decision Tree Classifier	0.676	0.800	0.444	0.571
Decision Tree Classifier 2	0.784	0.857	0.667	0.750
Random Forest Classifier	0.676	0.800	0.444	0.571
Support Vector Machine Classifier	0.676	1.000	0.333	0.500
Gradient Boosting Classifier	0.703	0.889	0.444	0.593
Gradient Boosting Classifier 2	0.703	0.818	0.500	0.621
K Nearest Neighbors Classifier	0.514	0.000	0.000	0.000

**Tabela 4.10.** Walidacja modeli – wszystkie cechy rozkładu temperatury bez podsumowujących

Nazwa modelu	Dokładność	Precyzja	Czułość	F1
Linear Regression Classifier	0.684	0.684	0.684	0.684
Naive Bayes Classifier	0.711	0.654	0.895	0.756
Decision Tree Classifier	0.711	0.750	0.632	0.686
Decision Tree Classifier 2	0.684	0.706	0.632	0.667
Random Forest Classifier	0.711	0.750	0.632	0.686
Support Vector Machine Classifier	0.553	0.750	0.158	0.261
Gradient Boosting Classifier	0.737	0.800	0.632	0.706
Gradient Boosting Classifier 2	0.737	0.800	0.632	0.706
K Nearest Neighbors Classifier	0.500	0.000	0.000	0.000

Kolejnym zbiór był oparty o pięć cech (tabela 4.11, które wykazały najwyższe korelacje ze zmienną szukaną. W tym przypadku ponownie najlepszym okazał się model Decision Tree Classifier – dokładność tego modelu wynosiła 83%, co oznacza znaczny wzrost, ale odnotowano również spadek czułości modelu do 69%.

Najlepsze efekty przyniosło trenowanie modelu opartego na zbiorze z trzema najważniejszymi czujnikami (tabela 4.12) Model Decision Tree Classifier wykazał najwyższą dokładność – 77%, ale również bardzo wysoką czułość – 85%. Mimo wszystko jest to niższa dokładność niż ta, którą osiągnięto poprzez kreowanie modelu w oparciu o wszystkie dostępne cechy.

**Tabela 4.11.** Walidacja modeli – pięć najważniejszych cech temperatury

Nazwa modelu	Dokładność	Precyzja	Czułość	F1
Linear Regression Classifier	0.629	0.500	1.000	0.667
Naive Bayes Classifier	0.571	0.464	1.000	0.634
Decision Tree Classifier	0.829	0.818	0.692	0.750
Decision Tree Classifier 2	0.743	0.667	0.615	0.640
Random Forest Classifier	0.829	0.818	0.692	0.750
Support Vector Machine Classifier	0.571	0.464	1.000	0.634
Gradient Boosting Classifier	0.829	0.818	0.692	0.750
Gradient Boosting Classifier 2	0.714	0.600	0.692	0.643
K Nearest Neighbors Classifier	0.629	0.000	0.000	0.000

**Tabela 4.12.** Walidacja modeli – trzy najważniejsze czujniki temperatury

Nazwa modelu	Dokładność	Precyzja	Czułość	F1
Linear Regression Classifier	0.590	0.435	0.769	0.556
Naive Bayes Classifier	0.487	0.379	0.846	0.524
Decision Tree Classifier	0.769	0.667	0.615	0.640
Decision Tree Classifier 2	0.795	0.647	0.846	0.733
Random Forest Classifier	0.769	0.667	0.615	0.640
Support Vector Machine Classifier	0.718	1.000	0.154	0.267
Gradient Boosting Classifier	0.718	0.571	0.615	0.593
Gradient Boosting Classifier 2	0.769	0.625	0.769	0.690
K Nearest Neighbors Classifier	0.667	0.000	0.000	0.000

## **5. Podsumowanie**

Po ewaluacji wyników uzyskanych w przeprowadzonych badaniach z wykorzystaniem metod uczenia maszynowego wspomagających diagnostykę cukrzycy w oparciu o nacisk i temperaturę podeszwowej powierzchni stopy, analizie danych i korelacji zweryfikowano zakładane w pracy cele. Ponadto w rozdziale zawarto również opis możliwości rozwoju i dalszego wykorzystania zastosowanych metod uczenia maszynowego, napotkanych problemów i podsumowanie zrealizowanych działań.

### **5.1. Osiągnięte cele**

Podsumowując niniejszą pracę, powrócono do celów pracy opisanych w rozdziale 1.2 i przeanalizowano, czy każdy z nich został osiągnięty.

- Czy na podstawie nacisku lub temperatury podeszwowej części stopy z pomiaru (ok. 2 godziny) można odróżnić osoby zdrowe i chore na cukrzycę?**

Wyniki ewaluacji modelu dla rozkładu nacisku są bardzo dobre - dokładność modelu w najlepszym wypadku wynosiła 89%, a czułość aż 94%. W przypadku rozkładu temperatury dokładność jest niższa (76%), ale czułość nadal wysoka - 95%. Jest to dobry wynik, zwłaszcza biorąc pod uwagę rozmiar zbioru danych, na którym model był trenowany. Można więc przyjąć, że tak sformułowany cel został osiągnięty, choć lepszym sposobem różnicowania osób zdrowych i chorych jest wykorzystanie rozkładu ciśnienia.

- W których częściach stopy najlepiej umieścić czujniki, by skutecznie diagnozować cukrzycę?**

W przypadku rozkładu temperatury najlepszy wynik zostaje osiągnięty, gdy czujniki zostały umiejscowione pod piętą, pod paluchem oraz pod czwartą kośćią śródstopia. W przypadku badania rozkładu temperatury nie jest możliwe uzyskanie odpowiedzi na to pytanie.

– **Która z metod uczenia maszynowego sprawdzi się w tym celu najlepiej?**

W przypadku rozkładu nacisku najlepiej sprawdziła się klasyfikacja Decision Tree Classifier trenowana na zbiorze danym opartym o wartości z trzech najistotniejszych czujników - L, D1 oraz MTK4. W przypadku temperatury - najlepiej sprawdził się „Naive Bayes Classifier” oparty o zbiór składający się ze wszystkich możliwych cech. Dobrze sprawdził się również Decision Tree Classifier, jednak istotniejsza niż dokładność w przypadku specyfiki prowadzonej analizy porównawczej (danych medycznych, predykcji schorzeń) jest czułość modelu.

## 5.2. Możliwości rozwoju algorytmu

Zgodnie z opisem znajdującym się w rozdziale 2.1, algorytmy oparte na parametrach takich jak nacisk wywierany na podłożo w różnych częściach stopy mogą mieć bardzo wiele zastosowań. Dalsze prace nad algorytmem powinny obejmować dużo szerszy zakres danych - mowa tu zarówno o rozmiarze zbioru jak i liczbie cech. Dodanie do bazy danych chorych na inne schorzenia umożliwiłoby stworzenie algorytmu wspomagającego diagnozy i monitorowanie stanu zdrowia na wielu płaszczyznach. Istnieje też duże prawdopodobieństwo, że wraz z upływem czasu powstaną kolejne badania naukowe, w których większa liczba problemów zdrowotnych będzie wykorzystywana do predykcji za pomocą algorytmów opartych o stopy.

W kolejnej wersji algorytmu należy dodać również walidację krzyżową oraz wagę poszczególnych cech. Warto również rozważyć stworzenie sieci neuronowej, dzięki której będzie możliwe bardziej precyzyjne działanie algorytmu.

## 5.3. Napotkane problemy

Głównym problemem w trakcie realizacji badań był zbiór danych. Był on stosunkowo niewielki, a ponadto zawierał wiele danych nieprawidłowych. Pojawiały się odczyty temperatur poniżej  $0^{\circ}\text{C}$ , ale także powyżej  $200^{\circ}\text{C}$ . Co więcej, wyniki pomiarów ciśnienia, były przedstawione w różnych skalach - od 0 do 4096 ( $2^{12}$ ), ale także od 0 do 32768 ( $2^{15}$ ). Pojawiało się również bardzo dużo danych odstających.

Na koniec, warto dodać że maksymalna kadencja biegowa, czyli liczba kroków na minutę to ok 200-220 [50]. Zgodnie z twierdzeniem Nyquista-Shannona (o próbkowaniu) [51], próbki powinny być zbierane co najmniej dwa razy częściej niż okres pomiaru. Pomiar nacisku co 3 sekundy oraz 20 sekund to bardzo niska częstotliwość - odpowiedni sprzęt powinien wykonywać pomiary od 20 do aż 130 razy częściej, aby wychwycić wszystkie istotne zmiany.

Podsumowując, wszystkie zamierzone cele pracy zostały osiągnięte, a przeprowadzone badania wykazują duży potencjał w zastosowaniu metod uczenia maszynowego we wspomaganiu diagnostyki różnych chorób i schorzeń. Opisane w pracy rozwiązanie może być dalej rozwijane i udoskonalenie na wielu płaszczyznach. Pierwszym krokiem powinno być poszerzenie zbioru danych oraz zadbanie o jak największą poprawność danych, na których trenowane są modele.



# Bibliografia

- [1] Wiktoria Kaczmarek-Sondej. „A Comparison of healthy people 2020–LHI topics between student populations within the USA, Germany, and Poland”. Prac. dokt. Hochschule Hannover, 2021.
- [2] Parinun Pangthipampai. „Peace and Wellbeing Base on the Conceptual of Life”. W: *Nimmitmai Review Journal* 5.1 (2022), s. 54–63.
- [3] Diagnostyka. *Choroby cywilizacyjne*. 2022. URL: <https://diag.pl/pacjent/artykuly/choroby-cywiliizacyjne-miazdzyca-cukrzyca-otylosc-jak-badac-jak-przeciwdzialac>.
- [4] Narodowy Fundusz Zdrowia. *Alergia - epidemia XXI wieku*. 2019. URL: <https://www.nfz.gov.pl/aktualnosci/aktualnosci-centrali/alergia-i-astma-filmy-edukacyjne,7363.html>.
- [5] World Health Organization. *Diabets*. 2022. URL: [https://www.who.int/health-topics/diabetes#tab=tab\\_1](https://www.who.int/health-topics/diabetes#tab=tab_1).
- [6] gov.pl. *Ogólnopolski dzień walki z cukrzycą*. 2022. URL: <https://www.gov.pl/web/psse-siedlce/ogolnopolski-dzien-walki-z-cukrzycą-2022>.
- [7] Narodowy Fundusz Zdrowia. *Cukrzyca choroba śmiertelna – prawda czy mit?* 2019. URL: <https://www.nfz-bialystok.pl/aktualnosci-oddzialu/cukrzyca-choroba-smiertelna-prawda-czy-mit-warsztaty-w-suwalkach>.
- [8] Marta Koblańska. *Chorujący na cukrzycę żyją krócej nawet o 12 lat*. 2020. URL: <https://www.termedia.pl/poz/Chorujacy-na-cukrzycę-zyja-krocej-nawet-o-12-lat,38451.html>.
- [9] Caroline Cabral Robinson i in. *Plantar pressure distribution patterns of individuals with prediabetes in comparison with healthy individuals and individuals with diabetes*. 2013.
- [10] Andrew JM Boulton i in. „The global burden of diabetic foot disease”. W: *The Lancet* 366.9498 (2005), s. 1719–1724.
- [11] Nalini Singh, David G Armstrong i Benjamin A Lipsky. „Preventing foot ulcers in patients with diabetes”. W: *Jama* 293.2 (2005), s. 217–228.
- [12] A Shojaie Fard, M Esmaelzadeh i B Larijani. „Assessment and treatment of diabetic foot ulcer”. W: *International journal of clinical practice* 61.11 (2007), s. 1931–1938.

- [13] Probal K Moulik, Robert Mtonga i Geoffrey V Gill. „Amputation and mortality in new-onset diabetic foot ulcers stratified by etiology”. W: *Diabetes care* 26.2 (2003), s. 491–494.
- [14] Christopher D Saudek i in. „A new look at screening and diagnosing diabetes mellitus”. W: *The Journal of Clinical Endocrinology and Metabolism* 93.7 (2008), s. 2447–2453.
- [15] Klaudia Kromołowska i in. „Open-Source Strain Gauge System for Monitoring Pressure Distribution of Runner’s Feet”. W: *Sensors* 23.4 (2023), s. 2323.
- [16] Julian Andres Ramirez-Bautista i in. „Review on plantar data analysis for disease diagnosis”. W: *Biocybernetics and Biomedical Engineering* 38.2 (2018), s. 342–361.
- [17] Thomas H Davenport i DJ Patil. „Data scientist: The sexiest job of the 21st century”. W: *Harvard Business Review* 90.10 (2012), s. 70–76.
- [18] Bernard Marr. *How much data do we create every day? The mind-blowing stats everyone should read*. Forbes. 2018. URL: <https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/?sh=66d83e2351b2> (term. wiz. 2023-04-07).
- [19] Nicole Martin. “How Much Data Is Collected Every Minute Of The Day”. In: (2019).
- [20] Redakcja coderslab. „DATA SCIENCE - co musisz wiedzieć, by zacząć karierę analityka danych”. W: (2022).
- [21] Encyklopedia zarządzania. *Uczenie maszynowe*. 2020. URL: [https://mfiles.pl/pl/index.php/Uczenie\\_maszynowe](https://mfiles.pl/pl/index.php/Uczenie_maszynowe).
- [22] Andrew Y Ng, Michael I Jordan i Yair Weiss. „Discriminative training of Markovian models for speech recognition”. W: *Proceedings of the 2002 Conference on Advances in Neural Information Processing Systems* (2002), s. 267–274.
- [23] Jiawei Han, Micheline Kamber i Jian Pei. „Data mining: concepts and techniques”. W: *Elsevier* 3 (2011).
- [24] Xindong Wu i in. „Data mining with big data”. W: *IEEE Transactions on Knowledge and Data Engineering* 26.1 (2014), s. 97–107.
- [25] Michał Kopciuch, Adam Świtoński i Tomasz Kaczmarek. *Uczenie maszynowe*. Wydawnictwo Akademii Górnictwa-Hutniczej im. Stanisława Staszica w Krakowie, 2021.
- [26] Ethem Alpaydin. *Introduction to Machine Learning*. MIT Press, 2010.
- [27] Jerome Friedman, Trevor Hastie i Robert Tibshirani. *The Elements of Statistical Learning*. Springer, 2001.
- [28] Ian Goodfellow, Yoshua Bengio i Aaron Courville. *Deep Learning*. MIT Press, 2016.

- [29] Yann LeCun, Yoshua Bengio i Geoffrey Hinton. „Deep Learning”. W: *Nature* 521.7553 (2015), s. 436–444.
- [30] Thomas G Dietterich. „Ensemble Methods in Machine Learning”. W: *Multiple Classifier Systems* (2000), s. 1–15.
- [31] TIOBE Software. *TIOBE Index*. [dostęp: 2023-13-14]. 2023. URL: <https://www.tiobe.com/tiobe-index/>.
- [32] Guido Van Rossum. „Python tutorial”. W: *Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands* (1995).
- [33] Guido van Rossum i Fred L Drake Jr. „Python reference manual”. W: *Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands* (1995).
- [34] Stefan Van Der Walt, S Chris Colbert i Gaël Varoquaux. „NumPy: A guide to NumPy”. W: *USA: Continuum Press* (2011).
- [35] Wes McKinney. „pandas: a Foundational Python Library for Data Analysis and Statistics”. W: *Python for High Performance and Scientific Computing* 14.2 (2011), s. 1–9.
- [36] John D Hunter. „Matplotlib: A 2D Graphics Environment”. W: *Computing in Science and Engineering* 9.3 (2007), s. 90–95.
- [37] Michael Waskom. „Seaborn: statistical data visualization”. W: *Journal of Open Source Software* 6.60 (2021), s. 3021.
- [38] Javier Canales Luna. *Top programming languages for data scientists*. [dostęp: 2023-13-14]. 2023. URL: <https://www.datacamp.com/blog/top-programming-languages-for-data-scientists-in-2022>.
- [39] Matei Zaharia i in. „Spark: Cluster Computing with Working Sets”. W: *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*. 2010, s. 10–10.
- [40] Konstantin Shvachko i in. „The hadoop distributed file system”. W: *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE. 2010, s. 1–10.
- [41] Uli Niemann i in. „Comparative Clustering of Plantar Pressure Distributions in Diabetics with Polyneuropathy May Be Applied to Reveal Inappropriate Biomechanical Stress”. W: *PLOS ONE* 11.8 (sierp. 2016), s. 1–12. DOI: [10.1371/journal.pone.0161326](https://doi.org/10.1371/journal.pone.0161326).
- [42] Tom M Mitchell. *Machine learning*. McGraw Hill, 1997.
- [43] David W Hosmer Jr, Stanley Lemeshow i Rodney X Sturdivant. *Applied logistic regression*. John Wiley i Sons, 2013.
- [44] Fabrizio Sebastiani. „Machine Learning in Automated Text Categorization”. W: *ACM Computing Surveys (CSUR)* 34.1 (2002), s. 1–47.

- [45] L. Breiman i in. *Classification and Regression Trees*. Chapman i Hall, 1984.
- [46] Leo Breiman. „Random forests”. W: *Machine learning* 45.1 (2001), s. 5–32.
- [47] Chih-Wei Hsu i Chih-Jen Lin. „A practical guide to support vector classification”. W: *Bioinformatics* 18.12 (2002), s. 1601–1609.
- [48] Jerome H Friedman. „Greedy function approximation: a gradient boosting machine”. W: *Annals of statistics* (2001), s. 1189–1232.
- [49] Xiaohong Wang i in. „K-Nearest Neighbors Based on Sparse Representation and Collaborative Representation for Semi-Supervised Classification”. W: *IEEE Access* 8 (2020), s. 39886–39896.
- [50] Runner’s Blueprint. *How many steps in one mile walking vs. running?* 2017.
- [51] John Lindon. *Encyclopedia of spectroscopy and spectrometry*. Kidlington, Oxford, United Kingdom: Academic Press is an imprint of Elsevier, 2016. ISBN: 978-0-12-803224-4.

# Spis rysunków

2.1	Ilość danych produkowanych w ciągu jednej minuty [19]	13
3.1	Schemat ustawienia czujników nacisku wkładki [41]	21
3.2	Histogram – ilość sesji osób zdrowych i chorych	22
3.3	Wykresy pułkowe rozkładu nacisku	23
3.4	Histogramy rozkładu nacisku	24
3.5	Wykres korelacji między zmiennymi nacisku	26
3.6	Pairplot minimalnych wartości nacisku	27
3.7	Pairplot średnich wartości nacisku	27
3.8	Pairplot maksymalnych wartości nacisku	28
3.9	Wykres rozproszenia dla danych minimalnego nacisku	28
3.10	Wykres rozproszenia dla danych średniego nacisku	29
3.11	Wykres rozproszenia dla danych maksymalnego nacisku	29
3.12	Wykres nacisku w czasie z grupy kontrolnej	30
3.13	Wykres nacisku w czasie z grupy cukrzyków	30
3.14	Wykresy pułkowe rozkładu temperatury	31
3.15	Histogramy rozkładu temperatury	32
3.16	Wykres korelacji między zmiennymi temperatury	33
3.17	Pairplot minimalnych wartości temperatury	34
3.18	Pairplot średnich wartości temperatury	35
3.19	Pairplot maksymalnych wartości temperatury	35
3.20	Wykres rozproszenia dla danych minimalnej temperatury	36
3.21	Wykres rozproszenia dla danych średniej temperatury	36
3.22	Wykres rozproszenia dla danych maksymalnej temperatury	37
3.23	Wykres temperatury w czasie z grupy kontrolnej	37
3.24	Wykres temperatury w czasie z grupy cukrzycy	38

# Spis tabel

4.1	Korelacje wartości czujników nacisku . . . . .	49
4.2	Korelacje cech nacisku . . . . .	50
4.3	Korelacje wartości cech temperatury . . . . .	51
4.4	Korelacje wartości czujników temperatury . . . . .	52
4.5	Walidacja modeli – wszystkie cechy rozkładu nacisku . . . . .	52
4.6	Walidacja modeli – wszystkie cechy rozkładu nacisku bez podsumowujących .	53
4.7	Walidacja modeli – pięć najważniejszych cech nacisku . . . . .	53
4.8	Walidacja modeli – trzy najważniejsze czujniki nacisku . . . . .	54
4.9	Walidacja modeli – wszystkie cechy rozkładu temperatury . . . . .	55
4.10	Walidacja modeli – wszystkie cechy rozkładu temperatury bez podsumowujących	55
4.11	Walidacja modeli – pięć najważniejszych cech temperatury . . . . .	56
4.12	Walidacja modeli – trzy najważniejsze czujniki temperatury . . . . .	56

## **Załączniki**

1. Internetowe repozytorium umieszczone w serwisie Github ([https://github.com/KlaudiaKromolowska/Analiza\\_nacisku\\_i\\_temperatury\\_stop](https://github.com/KlaudiaKromolowska/Analiza_nacisku_i_temperatury_stop)), zawierające następujące katalogi i pliki:
  - 1.1. Katalog „Analiza danych” – [https://github.com/KlaudiaKromolowska/Master\\_thesis-data\\_analysis](https://github.com/KlaudiaKromolowska/Master_thesis-data_analysis)
  - 1.2. Katalog „Uczenie maszynowe” – [https://github.com/KlaudiaKromolowska/Master\\_thesis-machine\\_learning](https://github.com/KlaudiaKromolowska/Master_thesis-machine_learning)
  - 1.3. Plik „Praca dyplomowa” – zawierający niniejszą pracę w formacie pdf – [https://github.com/KlaudiaKromolowska/Master\\_thesis-file](https://github.com/KlaudiaKromolowska/Master_thesis-file)