UNIWERSYTET WARMIŃSKO MAZURSKI W OLSZTYNIE
WYDZIAŁ MATEMATYKI I INFORMATYKI

PAWEŁ LESZCZYŃSKI
Informatyka

Uczenie maszynowe, w klasyfikacji przypadków cukrzycy

Praca inżynierska
wykonana w Katedrze Metod Matematycznych Informatyki

pod kierunkiem dr Pawła Drozdy

Olsztyn 2024

UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

PAWEŁ LESZCZYŃSKI

Computer Science

Machine learning, in diabetes case classification

Engineering Thesis

Carried out in the Chair of
Mathematical Methods of Informatics

Under the guidance of dr Paweł Drozda

Olsztyn 2024

Spis treści

Streszczenie

We współczesnej rzeczywistości zastosowanie sztucznej inteligencji stało się powszechne. Jest ona obecna niemalże w każdym obszarze życia. Niniejsza praca skupia się na praktycznym wykorzystaniu uczenia maszynowego, w rozpoznawaniu wystąpienia cukrzycy u kobiet po dwudziestym pierwszym roku życia. W pracy przedstawiony zostanie proces uczenia nadzorowanego czterech modeli predykcyjnych: sieci neuronowej, k-najbliższych sąsiadów, lasu losowego oraz XGBoost. Każdy z tych modeli powinien być w stanie prowadzić skuteczną klasyfikację danych medycznych. Efektywność modeli zostanie poddana ocenie. Najskuteczniejszy z nich, zostanie zaimplementowany do prostej aplikacji sieciowej. Ta aplikacja umożliwi użytkownikom wprowadzanie swoich danych, na podstawie których przeprowadzona zostanie predykcja wystąpienia cukrzycy.

Abstract

Nowadays, the use of artificial intelligence has become widespread. It is present in almost every area of life. This thesis focuses on the practical use of machine learning in diagnosing the onset of diabetes in group of women over the age of twenty-one. The thesis will present the process of supervised learning, of four predictive models: neural network, k-nearest neighbors, random forest and XGBoost. Each of these models should be able to perform effective classification of medical data. The effectiveness of the models will be evaluated. The most effective one will be implemented in a simple web application. This application will allow users to enter their data, on the basis of which a prediction of diabetes will be made.

1. Wstęp

1.1. Wprowadzenie

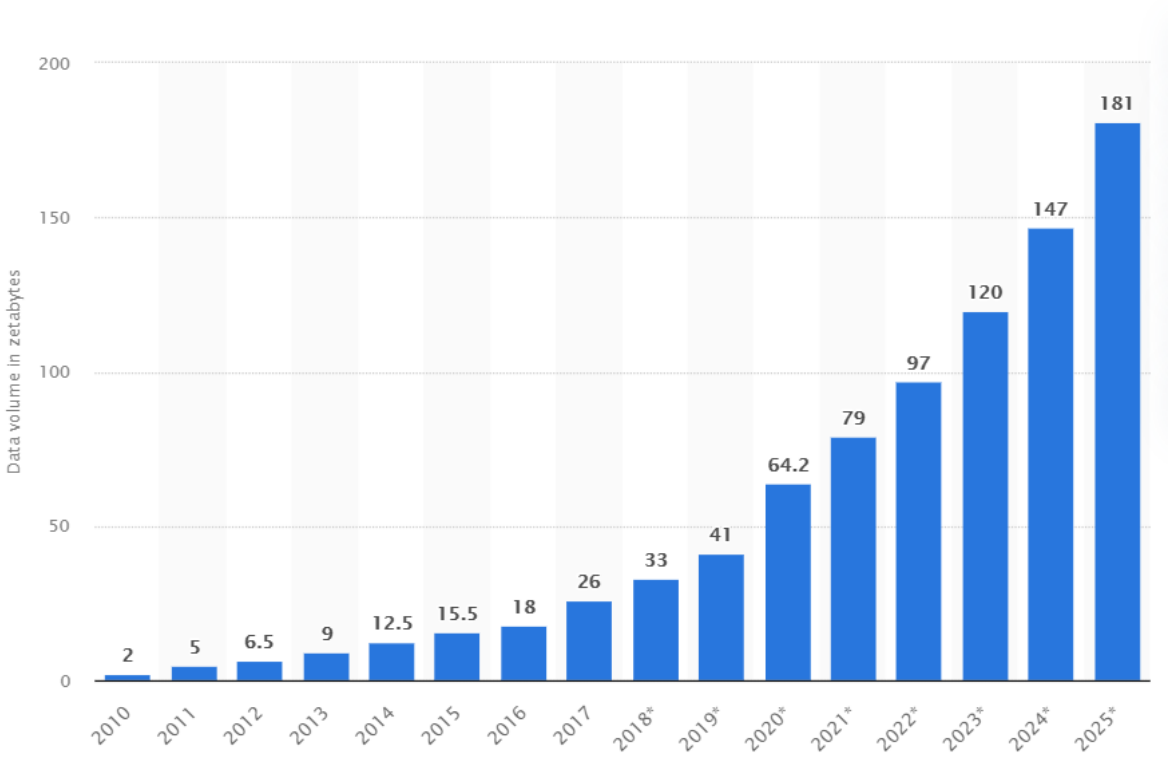
Uczenie maszynowe jest częścią dziedziny sztucznej inteligencji, skupiającą się na przetwarzaniu dużych ilości danych. Wyuczone modele za pomocą odpowiednich algorytmów, są w stanie, analizować i opracowywać wprowadzone do nich dane, w czasie nieporównywalnie mniejszym od człowieka, na dodatek bez jego nadzoru. Na rysunku 1.1 został przedstawiony wykres prognozowanych ilości wytworzonych danych do roku 2025. Prognoza opiera się o przyrost wyprodukowanych danych w latach 2010 – 2020. Jak widać ilość nowych danych, rośnie wykładniczo. Biorąc to pod uwagę, założyć można, że uczenie maszynowe nie jest już tylko pomocą w przetwarzaniu takich ilości danych, a koniecznością.

Technologia oparta na uczeniu maszynowym używana jest już niemalże we wszystkich obszarach rynku. Sprzedawcy z jej pomocą, personalizują oferty sprzedaży dla swoich klientów. Jednocześnie śledząc swoje stany magazynowe i posiłkując się modelami predykcyjnymi, prognozują popyt na posiadane przez nich towary, co pozwala usprawnić logistykę. W obszarze usług finansowych odpowiednie algorytmy pomagają analizować sytuację rynkową, zmniejszając ryzyko nietrafionych inwestycji. Modele predykcyjne są również szeroko używane w dziedzinie cyberbezpieczeństwa, gdzie efektywnie wykrywają podejrzane działania i klasyfikują je jako zagrożenie.

Celem tej pracy jest przedstawienie, praktycznego zastosowania uczenia maszynowego w obszarze medycyny, polegającego na rozpoznaniu wystąpienia cukrzycy, po rozpatrzeniu wprowadzonych danych medycznych. Dowodem na skuteczność uczenia maszynowego w tej dziedzinie, będzie stworzona aplikacja umożliwiająca użytkownikom prowadzenie trafnych predykcji wystąpienia cukrzycy. Aby ten cel zrealizować, zostanie przebadana skuteczność czterech różnych modeli predykcyjnych, a następnie zostanie wyłoniony ten, który najlepiej wykonuje powierzone mu zadanie. Model oceniony jako najlepszy zostanie zaimplementowany w aplikacji.

Dane użyte do wytrenowania modeli, pochodzą z Narodowego Instytutu Cukrzycy oraz Chorób Układu Pokarmowego i Nerek, w Stanach Zjednoczonych. Dotyczą kobiet

po dwudziestym pierwszym roku życia, pochodzących z Indiańskiego plemienia Pima. W sumie uwzględniono osiem zmiennych: liczbę przebytych ciąż, stężenie glukozy dwie godziny po posiłku, ciśnienie krwi, grubość fałdy skórno-tłuszczowej mierzonej na ramieniu, poziom insuliny dwie godziny po posiłku, indeks BMI, wiek i Diabeetes pedigree function (DPF), będącej współczynnikiem prawdopodobieństwa wystąpienia cukrzycy, bazującym na wieku i historii cukrzycy w rodzinie. Dziewiątym parametrem użytym do opisania danych, jest klasa decyzyjna zero lub jeden, odzwierciedlająca wynik negatywny lub pozytywny.



Rysunek 1.1 https://www.statista.com/, prognozowana ilość wytworzonych danych

1.2. Podstawy teoretyczne

W niniejszej pracy poruszona zostanie dziedzina sztucznej inteligencji, jaką jest uczenie maszynowe, a dokładniej zostanie przeprowadzony proces uczenia nadzorowanego. Uczenie maszynowe koncentruje się na tworzeniu algorytmów i trenowaniu modeli, które następnie autonomicznie, bez potrzeby dalszego programowania, wykonują określone zadania.

Odpowiednio wyuczone modele, są w stanie identyfikować zależności i dokonywać predykcji na zupełnie nowych danych.

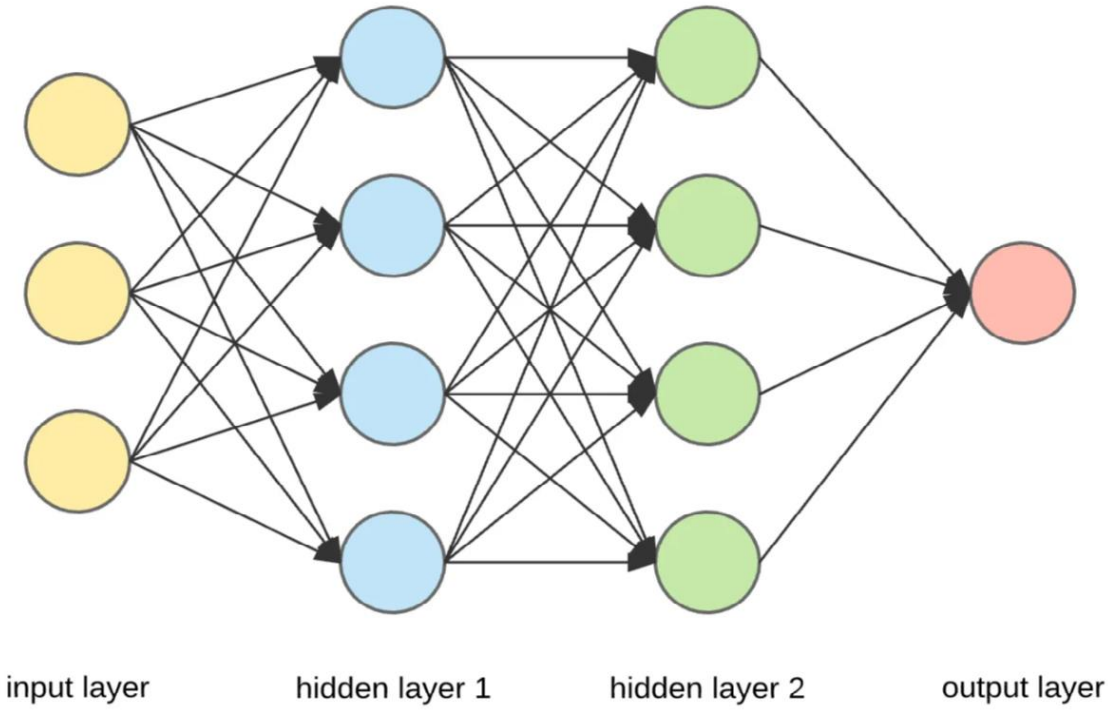
W uczeniu maszynowym istnieje kilka głównych podejść: uczenie nadzorowane, uczenie nienadzorowane oraz uczenie ze wzmocnieniem. Zależnie od zadań stawianych przed modelem, należy dobrać odpowiednie podejście. Jako że w tej pracy, zadaniem modelu będzie predykcja wystąpienia cukrzycy, na podstawie przekazanych mu danych, zastosowano uczenie nadzorowane.

Uczenie nadzorowane polega na trenowaniu modelu, używając do tego zestawu danych, w którym znane są parametry wejścia, oraz odpowiadające im etykiety. W tym przypadku danymi wejściowymi są wyniki badań, przeprowadzonych na grupie kobiet po dwudziestym pierwszym roku życia. Wystąpienie cukrzycy należy traktować jako etykietę. Celem tego typu uczenia jest ustalenie relacji i zależności, pomiędzy danymi, przekazywanymi jako wejście, a odpowiadającymi im etykietami. W procesie uczenia model dostosowywany jest tak, aby minimalizować błąd między przewidywanymi a rzeczywistymi wynikami.

Innym rozróżnieniem modeli, może być rodzaj stawianego przed nimi zadania. Głównymi rodzajami w tej kategorii są: regresja, klasyfikacja, grupowanie. Modele wyuczone na potrzeby pracy, będą rozwiązywać problem klasyfikacji. Klasyfikacja polega na przyporządkowaniu wprowadzanym danym, jednej z określonych wcześniej klas. Ponownie odnosząc się do przypadku tej pracy, będą to klasy „0” lub „1”, reprezentujące kolejno negatywny i pozytywny wynik, w kontekście wystąpienia cukrzycy.

Wśród badanych modeli, jeden z nich się wyróżnia. Sieć neuronowa ma bardzo charakterystyczną postać, gdyż jej budowa ma za zadanie odzwierciedlać działanie biologicznego mózgu i jego neuronów. W celu stworzenia sieci neuronowej, programowane są obiekty mające symulować działanie neuronu. Neurony organizowane są w warstwy. Pierwszą warstwą jest warstwa wejściowa. Jej zadaniem jest przyjęcie danych i przekazanie ich do kolejnych warstw. liczba neuronów znajdujących się w tej warstwie, powinna odpowiadać ilości przekazywanych do niej zmiennych. Dane przekazywane są do warstw wewnętrznych (ukrytych), których może być wiele. Ich ilość zależy od rozwiązywanego problemu i jego poziomu skomplikowania. Na końcu przetworzone dane są przekazywane z ostatniej warstwy wewnętrznej do warstwy wyjściowej, która odpowiada za wygenerowanie wyniku. Liczba neuronów w tej warstwie może się różnić w zależności od rodzaju rozwiązywanego problemu.

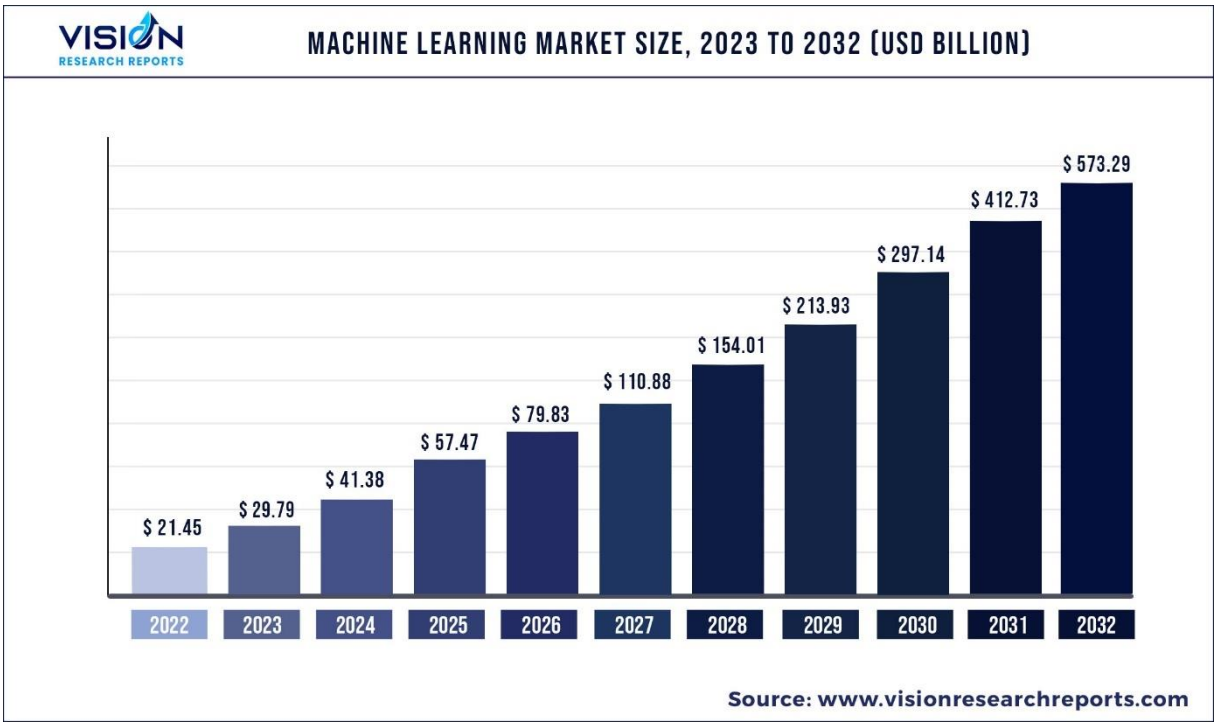
W zadaniach klasyfikacji wystarczy jeden neuron. Przykładowy model sieci neuronowej został zaprezentowany na rysunku 1.2.



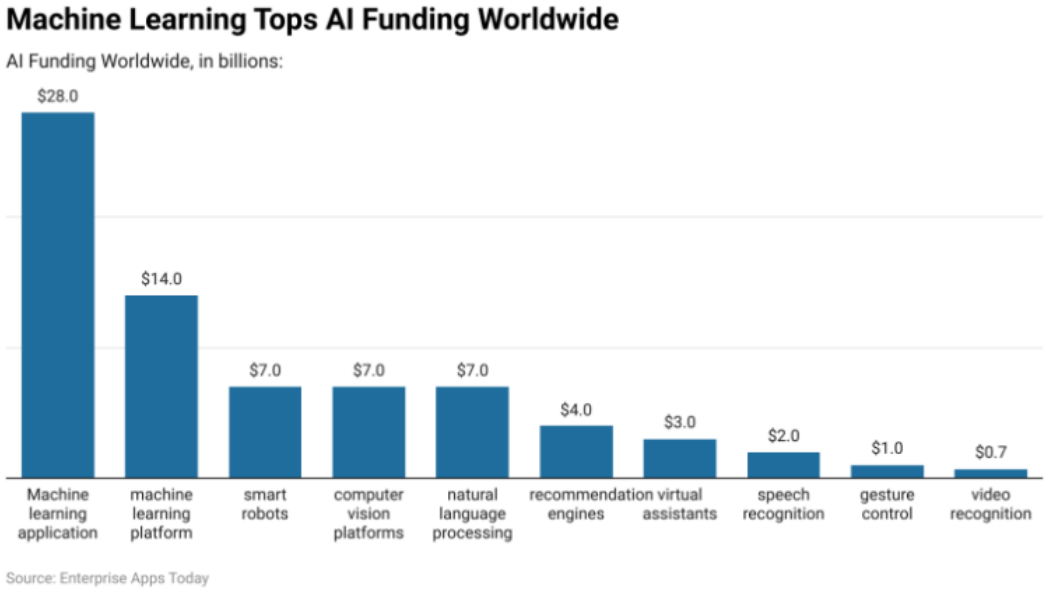
Rysunek 1.2 https://towardsdatascience.com, model sieci neuronowej

Uczenie maszynowe jest bardzo prężnie rozwijającą się dziedziną. Statystyki ukazują, że światowy rynek, związany z uczeniem maszynowym, w przeciągu ostatnich dwóch podwoił swoją wartość. Wielu analityków prognozuje dalszy gwałtowny wzrost w najbliższej dekadzie. Na rysunku 1.3 przedstawiono wartości globalnego rynku uczenia maszynowego, z ostatnich lat, oraz prognozy na nadchodzące lata. W 2023 roku, według Enterprise Apps Today, największą część udziału rynku uczenia maszynowego, stanowiły aplikacje. Rysunek 1.3 przedstawia rozkład udziałów, poszczególnych obszarów wcześniej wspomnianego rynku.

Analizując przytoczone dane, stwierdzić można, że podejście prezentowane w tej pracy, podąża za światowymi trendami. Rynek uczenia maszynowego stale się rozwija. Technologia ta jest używana coraz częściej. Ponadto często implementuje się ją, w przystępnej dla użytkownika formie aplikacji.



Rysunek 1.3 www.visionresearchreports.com, wartość rynku uczenia maszynowego do 2032 roku



Rysunek 1.4 www.enterpriseappstoday.com, wartość poszczególnych obszarów rynku uczenia maszynowego, w roku 2023

2. Technologia

W trakcie realizacji pracy wykorzystano szereg narzędzi, które umożliwiły i ułatwiły, przygotowanie danych, uczenie i optymalizację modeli predykcyjnych oraz ocenę ich skuteczności. W tym rozdziale najważniejsze z nich zostaną wymienione i opisane.

2.1. Python

Cała część programistyczna pracy została wykonana przy użyciu języka Python. Jest to wysokopoziomowy język imperatywny, umożliwiający przeróżne podejścia do programowania, takie jak: programowanie obiektowe, strukturalne czy funkcyjne. Umożliwia on dostęp do ogromnej ilości bibliotek, o przeróżnych funkcjonalnościach. Jednocześnie kod pisany w tym języku, jest prosty i przejrzysty. Wszystko to sprawia, że Python jest aktualnie jednym z najpopularniejszych języków programowania.

2.2. Scikit-learn

Scikit-learn jest biblioteką języka Python, umożliwiającą łatwe i szybkie użycie modeli uczenia maszynowego. Oferuje ona szereg algorytmów, rozwiązujących zarówno problemy klasyfikacji, jak i regresji. Zapewnia również narzędzia, pomagające we wstępnej optymalizacji i obróbce danych, dopasowaniu modeli predykcyjnych do stawianych im zadań czy późniejszej oceny ich skuteczności.

2.3. TensorFlow

TensorFlow jest frameworkiem pozwalającym łatwo i szybko projektować głębokie sieci neuronowe. Zapewnia narzędzia pozwalające na dopasowanie ich konstrukcji i hiperparametrów, do stawianych zadań. Framework został zaprojektowany tak, by w trakcie

uczenia modelu umożliwiał użycie karty graficznej (procesora graficznego), co może znacznie przyspieszyć proces. Dodatkowo może być użyty jako biblioteka, w kilku językach programowania, w tym w języku Python. TensorFlow dzięki swoim funkcjonalnościom, jest najchętniej wybieranym w tej dziedzinie narzędziem.

2.4. Flask

Flask jest lekkim i łatwym w użyciu mikroframeworkiem, napisanym w języku Python. Dostarcza on tylko podstawowe narzędzia do budowy aplikacji sieciowej, jednak umożliwia dołączenie szerokiego wachlarza bibliotek i rozszerzeń. Co istotne, umożliwia on obsługę żądań HTTP oraz szablonów HTML.

2.5. Bootstrap

Bootstrap to framework, który pozwala na proste i szybkie tworzenie responsywnych stron internetowych. System siatki umożliwia podział przestrzeni, na której wyświetlana jest treść, na 12 równych części. Wyświetlanym elementom, poprzez nadanie odpowiedniej klasy w pliku HTML, określa się ilość części, które element powinien zajmować, przy poszczególnych wymiarach otwartej strony. W efekcie strona internetowa będzie posiadała odpowiednią, wygodną strukturę, niezależnie od wymiarów, w których zostanie wyświetlona. Oprócz tego Bootstrap dostarcza gotowe komponenty takie jak panele nawigacyjne, formularze, czy przyciski.

2.6. HTML

HTML jest językiem znaczników, używanym w tworzeniu stron internetowych. Poprzez zastosowanie znaczników, tworzone są elementy, takie jak paragrafy, nagłówki czy też obszary. W efekcie powstaje struktura i układ strony. Umożliwia także zapisanie różnorodnych metadanych.

2.7. CSS

CSS to kaskadowy arkusz stylów, służący do formatowania elementów HTML. Definiuje się w nim reguły, które następnie są używane podczas wyświetlania strony internetowej. Umożliwia manipulowanie szeregiem właściwości wyświetlanej treści. Tyczy się to zarówno tekstu czy innej zawartości, jak i przestrzeni, w której jest ona wyświetlana.

2.8. Inne biblioteki Python

Podczas części programistycznej pracy, zostały wykorzystane również inne, mniejsze biblioteki języka Python:

- Pandas - biblioteka ułatwiająca pracę z danymi, pozwalająca tworzyć struktury danych w postaci dwuwymiarowych tabel (dataframe),
- Matplotlib - biblioteka pozwalająca na wizualizację danych, z wykorzystaniem wykresów i diagramów,
- NumPy – biblioteka umożliwiająca tworzenie tablic n-wymiarowych i wygodne prowadzenie na nich, działań matematycznych.

3. Proces trenowania modeli

W tym rozdziale zostanie przedstawiony proces trenowania wszystkich czterech modeli, a także szczegółowo zostaną opisane wszystkie etapy tego procesu. W pierwszej kolejności omówione zostanie przygotowanie danych, które posłużą do trenowania. Następnie przedstawiony zostanie proces uczenia kolejno: sieci neuronowej, K-najbliższych sąsiadów, lasu losowego i XGboost. W ramach opisu każdego modelu zostanie przedstawiona metodyka doboru istotnych parametrów, mających wpływ na efektywność prowadzenia predykcji.

3.1. Analiza i przygotowanie danych

Pierwszym etapem całego procesu jest analiza i przygotowanie danych, które mają być użyte do trenowania modeli. Od ich jakości, zależy skuteczność predykcji każdego modelu.

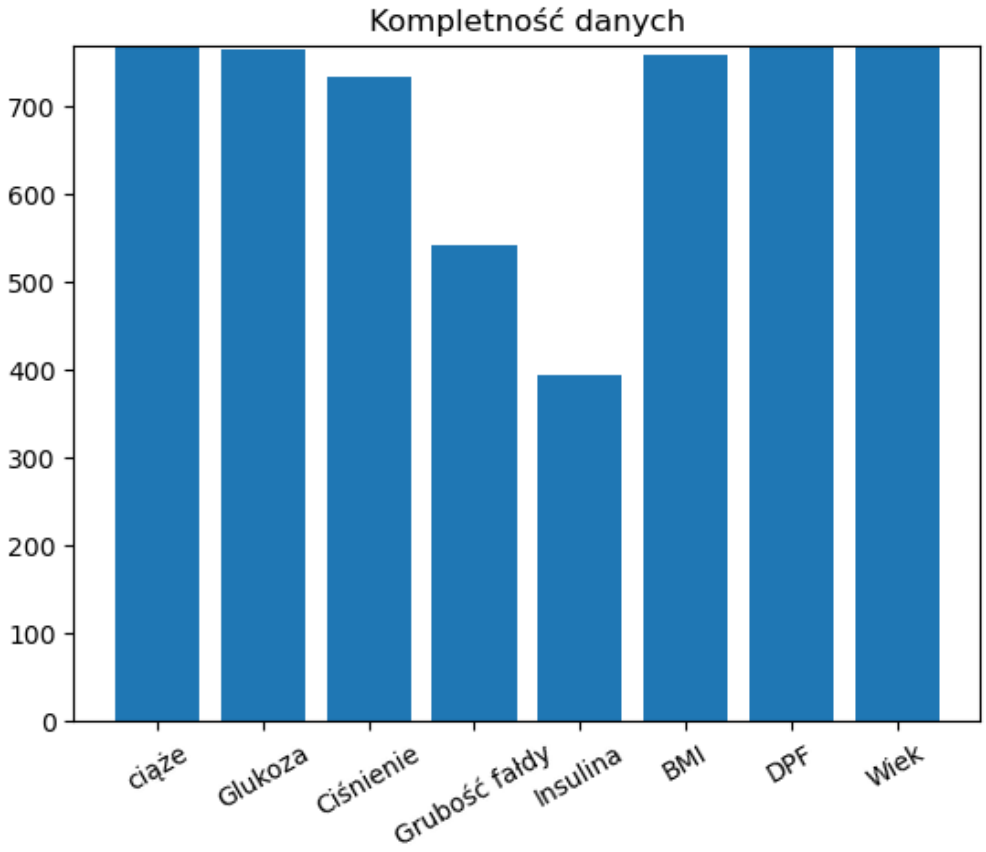
Najpierw została rozpatrzona kompletność danych. W zestawie danych znajdowało się 768 wyników, opisanych ośmioma cechami. Dla pięciu spośród ośmiu parametrów stwierdzono, że dane są niekompletne. Stwierdzone zostały następujące braki:

- 5 wartości poziomu glukozy,
- 35 wartości ciśnienia krwi,
- 227 grubości fałdy skórno-tłuszczowej,
- 374 wartości poziomu insuliny oraz
- 11 wskaźników BMI.

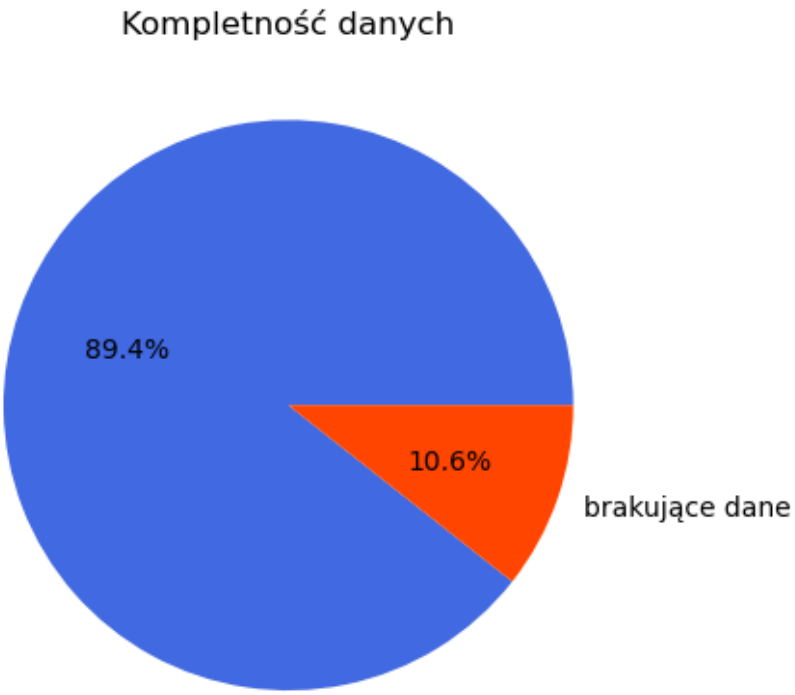
Rysunek 3.1 prezentuje kompletność poszczególnych danych. Rysunek 3.2 przedstawia stosunek danych kompletnych do niekompletnych.

W celu poprawienia skuteczności modelu, brakujące wartości zmiennych w przypadku zbioru wyników pozytywnych zostały uzupełnione medianami zmiennych z tego samego zbioru. Analogicznie postąpiono z wartościami zmiennych w zbiorze wyników negatywnych. Kolejnym podjętym krokiem, było określenie udziału wyników pozytywnych i negatywnych, zawartych w zbiorze danych. Jak widać na rysunku 3.3, zachodzi znaczna dysproporcja pomiędzy ilością wyników poszczególnych klas decyzyjnych. Wynik negatywny diagnozy

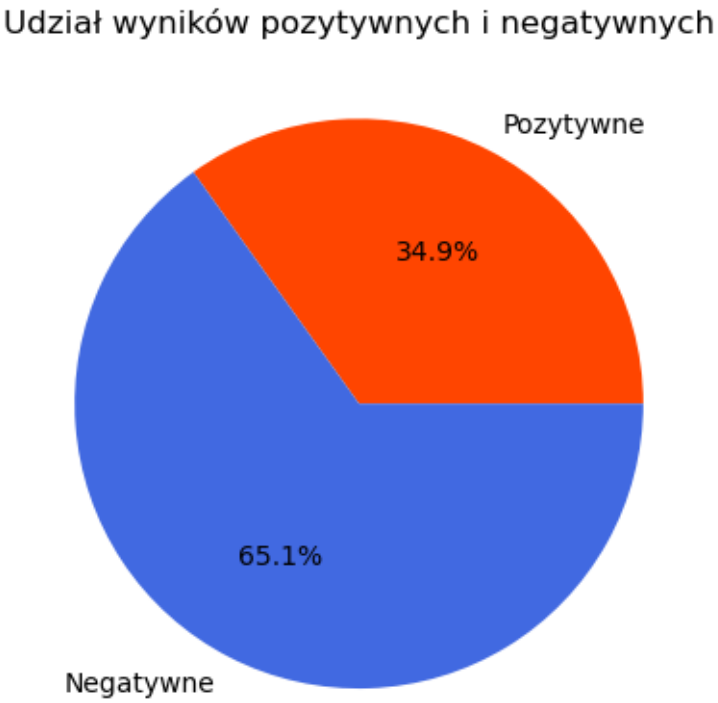
cukrzycy występuje niemalże dwukrotnie częściej niż wynik pozytywny. Taka asymetria w ilości klas decyzyjnych może negatywnie wpłynąć na model, sprawiając, że będzie bardziej skłonny do predykcji klasy z większą reprezentacją. W celu ograniczenia negatywnych efektów zastosowano bibliotekę imblearn, która umożliwiła zrównoważenie klas. Losowe przypadki, należące do mniej licznej grupy z wynikami pozytywnymi, zostały powielone. Działanie te umożliwiło uzyskanie zrównoważonej reprezentacji obu klas decyzyjnych. Ostatecznie oba zbiory liczą po 500 przypadków.



Rysunek 3.1 Kompletność zestawu danych



Rysunek 3.2 Procentowe przedstawienie brakujących danych



Rysunek 3.3 Procentowy udział klas decyzyjnych

W kolejnym kroku dokonano podziału zbioru danych, na grupę uczącą, walidacyjną i testową. Grupa ucząca został użyta do wytrenowania modelu. W tym procesie ustalone zostały zależności pomiędzy wartościami wejściowymi a wynikami. Grupa walidacyjna została wykorzystana do doboru hiperparamterów dla poszczególnych modeli oraz wstępnej oceny ich skuteczności. Ostatni ze zbiorów (grupa testowa), umożliwił ostateczną ocenę modelu, przy użyciu danych, z którymi model nie miał jeszcze styczności na etapie trenowania lub walidacji. Grupy zostały podzielone w następujący sposób: 60% wszystkich danych - grupa ucząca, 20% wszystkich danych - grupa walidacyjna, pozostała część - grupa testowa. Tak przygotowane dane, umożliwiły przystąpienie do procesu trenowania modeli predykcyjnych.

3.2. Model sieci neuronowej

Pierwszym rozpatrywanym modelem jest sieć neuronowa. Sieć użyta do badań w tej pracy, została stworzona przy pomocy biblioteki Tensorflow. Przez wzgląd na charakterystykę danych, na których model będzie pracował, do budowy sieci użyto warstwy wejściowej z ośmioma neuronami i dwóch warstw wewnętrznych. Warstwy te były w pełni połączone (dense). Użyto w nich funkcji aktywacji ReLU. Po warstwach wewnętrznych dodano warstwę wyjściową z funkcją aktywacji sigmoid. Zasadą działania funkcji aktywacji ReLU (rectified linear unit) jest następująca reguła: jeżeli suma wejść jest większa niż zero, zwrócona zostanie wartość tej sumy, w przeciwnym razie zwrócone zostanie zero. Funkcji tej, przypisuje się efektywność obliczeniową i zdolność do nauki skomplikowanych wzorców. Funkcja sigmoid przekształca zadaną wartość na wartość z zakresu (0,1). W efekcie zwraca ułamek, który można interpretować jako prawdopodobieństwo. W celu klasyfikacji przypadków ułamek ten zostanie dalej zaokrąglony do wartości „0” lub „1”. Pomiędzy warstwą pierwszą i drugą oraz drugą i wyjściową dodano warstwy porzuceniowe (droput layers), mające za zadanie ograniczyć zjawisko nadmiernego dopasowania modelu.

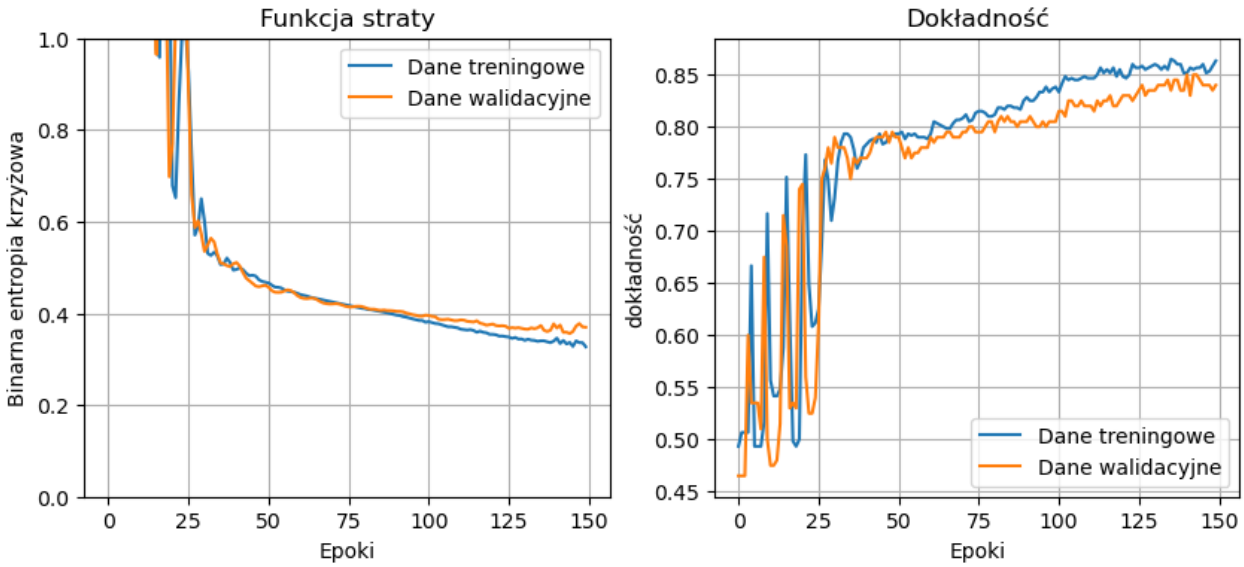
Tworząc model, należało rozpatrzeć szereg hiperparametrów. Oceniając skuteczność sieci neuronowej, rozpatrzono następujące parametry:

- ilość neuronów w pojedynczej warstwie wewnętrznej,
- prawdopodobieństwo porzucenia (dropout),
- tempo uczenia ,

- ilość epok.

Skuteczność modelu rozpatrywana była na podstawie osiągniętej dokładności i funkcji straty, zaprezentowanych na rysunku 3.4. Modelem, który zgodnie z tym kryterium okazał się najlepszy, był model o następujących parametrach:

- posiadający po128 neuronów w warstwach wewnętrznych,
- o tempie uczenia 0,01,
- trenowany przez 150 epok,
- z zerowym prawdopodobieństwem porzucenia (dropout).



Rysunek 3.4 Krzywe uczenia sieci neuronowej

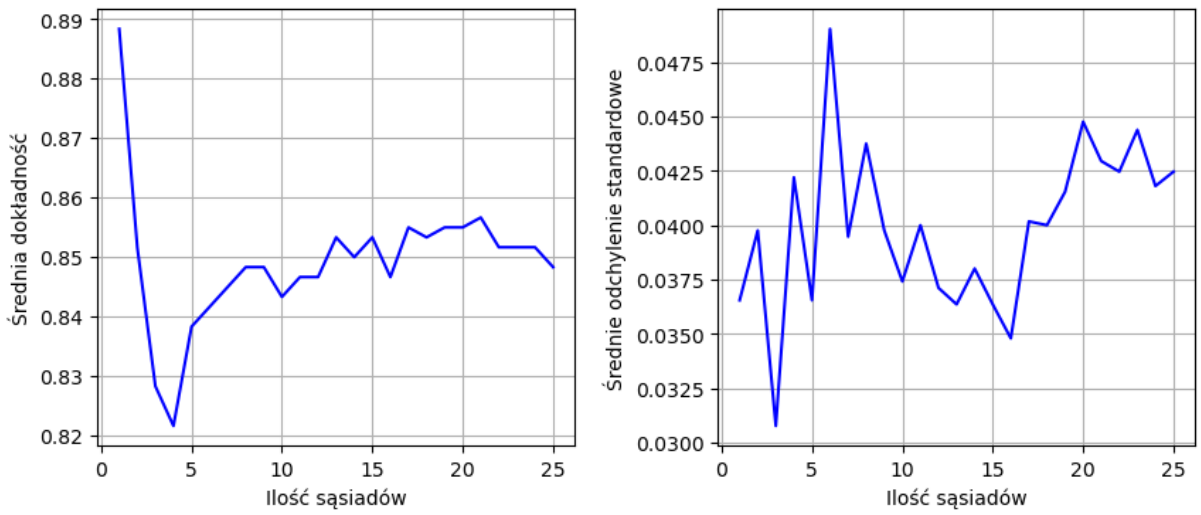
Obserwując funkcję straty można, zauważyć podobieństwo przebiegu krzywej dla danych treningowych i walidacyjnych. Wykresy nie odbiegają znacznie od siebie wartościami. Na tej podstawie można uznać, że walidacja została przeprowadzona na reprezentatywnym zbiorze danych, a model nie został przeuczony. Ostatecznie, badając predykcje modelu na ostatniej grupie (testowej), uzyskano funkcję straty na poziomie 0,37 i dokładność na poziomie 0,84.

3.3. Model K-najbliższych sąsiadów

Drugim poddanym pod rozpatrzenie modelem, jest model K-najbliższych sąsiadów. To stosunkowo prosty algorytm rozpatrujący wartość przewidywanego wyniku, w oparciu o klasy decyzyjne k-najbliższych sąsiadów.

W trenowanym modelu do określenia odległości do sąsiadów zastosowano odległość euklidesową. Aby dobrać optymalną ilość sąsiadów, przeprowadzono badanie porównujące średnie dokładności i odchylenia standardowe predykcji. Badanie przeprowadzono na dziesięciu podzbiorach, które uzyskano poprzez podział grupy danych uczących. Wyniki badania przedstawiono na rysunku 3.5.

Przy wyborze liczby rozpatrywanych sąsiadów odrzucono wartości parzyste, aby uniknąć potencjalnych konfliktów w sytuacji, gdy ilości sąsiadów z etykietą "1" i "0" byłyby sobie równe. Na podstawie otrzymanych danych, został wybrany model rozpatrujący piętnastu najbliższych sąsiadów. Ten wybór zdaje się oferować równowagę między wysoką dokładnością a niskim odchyleniem standardowym. Mimo że model rozważający jednego najbliższego sąsiada charakteryzuje się wysoką dokładnością, to niesie on ze sobą większe ryzyko niepoprawnej klasyfikacji.



Rysunek 3.5 Wyniki badań zmiennej K, w modelu K-n. sąsiadów

3.4. Model lasu losowego

Trzecim rozpatrywanym modelem jest las losowy. Las losowy opiera się na metodzie zespołowej (ensemble learning). W trakcie uczenia tworzonych jest wiele słabszych modeli. W tym wypadku modele te są drzewami decyzyjnymi. Dla każdego drzewa losuje się podgrupę danych z grupy uczącej. Jednej tak wylosowanej podgrupy, używa się do uczenia jednego drzewa decyzyjnego. Podczas uczenia wybierany jest najlepszy zbór cech, na podstawie

których, drzewo podejmuje decyzję. Ostateczna predykcja prowadzona jest poprzez głosowanie pojedynczych drzew.

Podczas tworzenia modelu, należało dobrać następujące parametry:

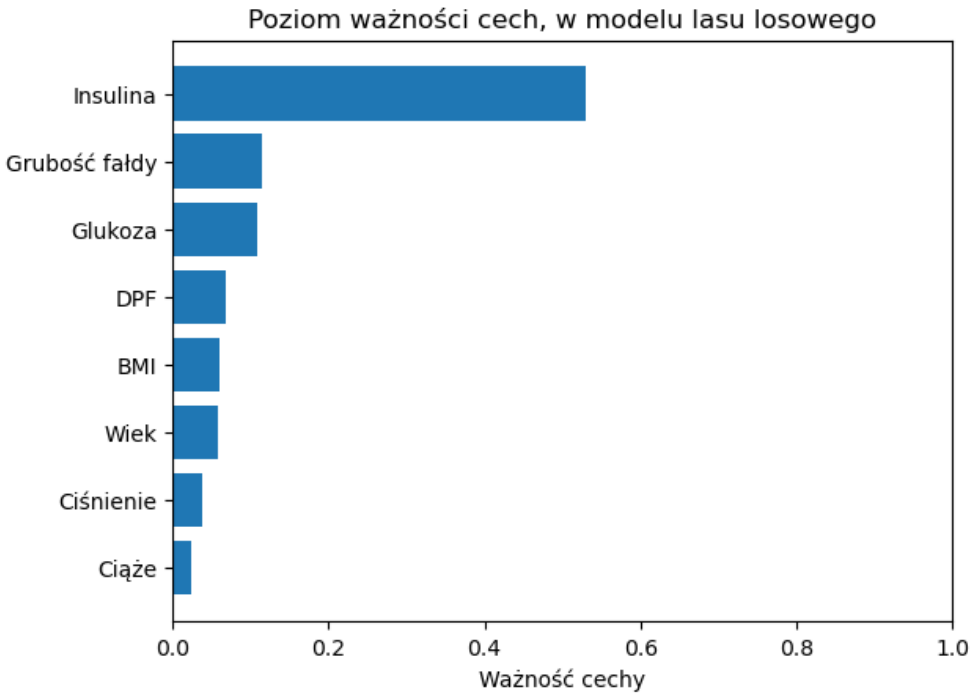
- maksymalną głębokość pojedynczego drzewa decyzyjnego,
- maksymalną liczbę cech branych pod uwagę przez algorytm,
- minimalną liczbę próbek wymaganych do podziału węzła,
- liczbę drzew decyzyjnych.

Po przeprowadzonych testach ustalono, że najlepsze parametry to:

- brak maksymalnej głębokości pojedynczych drzew,
- maksymalnie pięć rozpatrywanych cech,
- podział przy minimalnie dwóch próbkach,
- sto drzew decyzyjnych w lesie.

Test przeprowadzono, przy użyciu narzędzia gridsearch, z biblioteki Sci-kit learn. Jakość modeli oceniano na podstawie średniego wyniku walidacji krzyżowej, badanej na 10 podgrupach wydzielonych z grupy danych uczących.

Biblioteka scikit-learn, z pomocą której wytrenowany został model, umożliwia pobranie dla każdej z badanych cech jej ważności. Poziom ważności każdej z ośmiu cech, użytych do uczenia modelu, został przedstawiony na rysunku 3.6. Jak widać, zdecydowanie najważniejszą cechą jest poziom insuliny dwie godziny po spożyciu posiłku. Wszystkie pozostałe cechy są dużo mniej istotne. W początkowej ewaluacji danych stwierdzony został brak aż 374 wartości opisujących wyniki badania poziomu insuliny. W związku z tym, przed faktycznym użyciem takiego modelu w predykcji wystąpienia cukrzycy, zestaw danych użytych do uczenia, powinien zostać uzupełniony prawdziwymi wartościami, przedstawiającymi poziom insuliny badanych osób.



Rysunek 3.6 Wykres poziomu ważności cech, w lesie losowym

3.5. XGBoost

Ostatnim badanym modelem, jest model XGBoost- Extreme Gradient Boosting, czyli model ekstremalnego wzmacniania gradientowego. Podobnie jak las losowy opiera się on o metodę zespołową. W trakcie uczenia dodawane są kolejno nowe, słabsze modele. W tym konkretnym przypadku są to drzewa decyzyjne. Każde kolejne drzewo powinno poprawiać błędy poprzedniego. XGBoost jest modelem zdolnym do analizy złożonych relacji, zachodzących w zbiorze danych. Jednocześnie jest on bardzo odpornym na zjawisko przeuczenia.

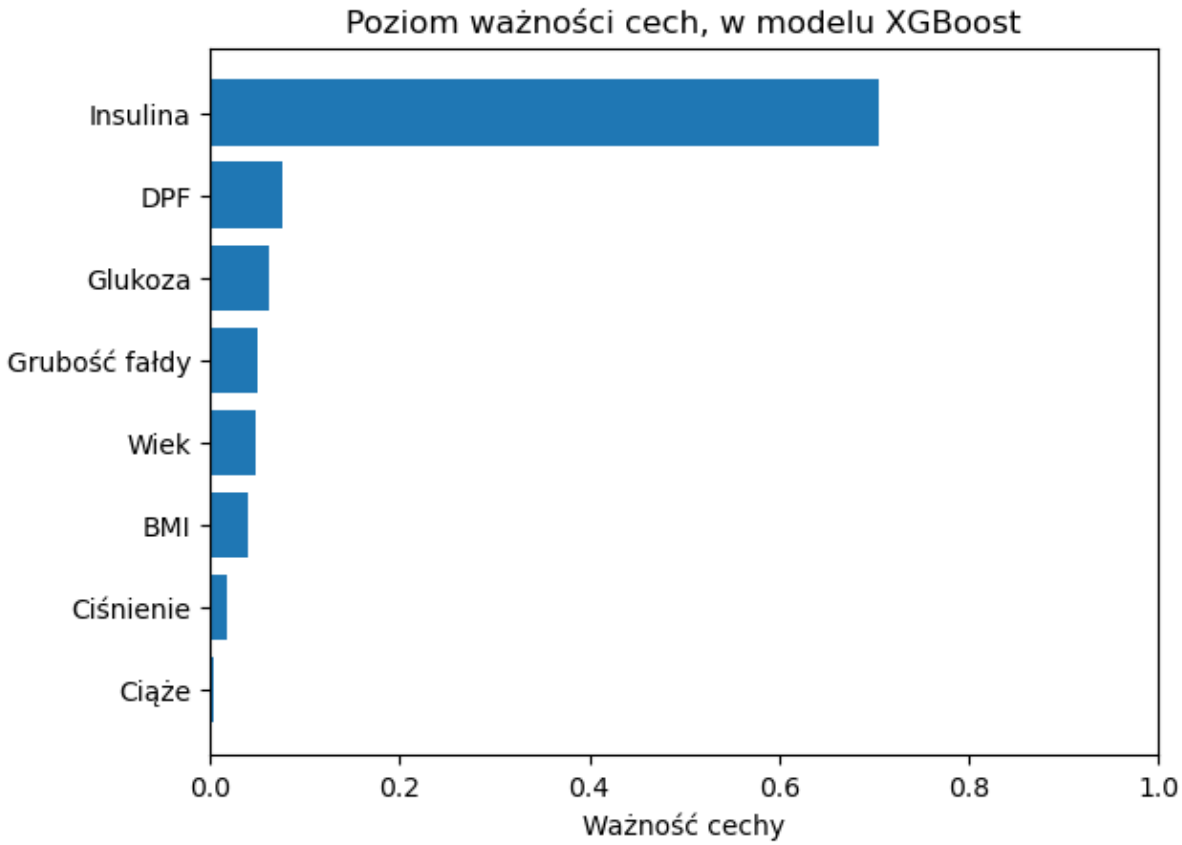
W celu poprawnego wytrenowania modelu podobnie jak w poprzednich przypadkach należało zbadać jego jakość przy różnych parametrach. Ustalane w procesie uczenia parametry to:

- tempo uczenia,
- minimalna liczba próbek wymagana do podziału drzewa,
- maksymalna głębokość drzewa,

- współczynnik próbkowania instancji uczących,
- ilość użytych słabszych modeli.

Do wyłonienia najlepszego zestawu parametrów ponownie zostało użyte narzędzie gridsearch, a ocenę przeprowadzono na podstawie średniego wyniku walidacji krzyżowej na 10 podgrupach, wydzielonych z grupy danych uczących.

Analogicznie do lasu losowego, dla tego modelu również został stworzony wykres ważności poszczególnych cech, który jest przedstawiony na rysunku 3.7. Podobnie jak w poprzednim modelu najistotniejszym parametrem okazał się poziom insuliny dwie godziny po spożyciu posiłku. Jak widać, jest on decydującym czynnikiem w procesie predykcji. W tym modelu jego ważność nawet wzrosła. Podobnie jak w poprzednim przypadku pozostałe cechy są zdecydowanie mniej ważne.



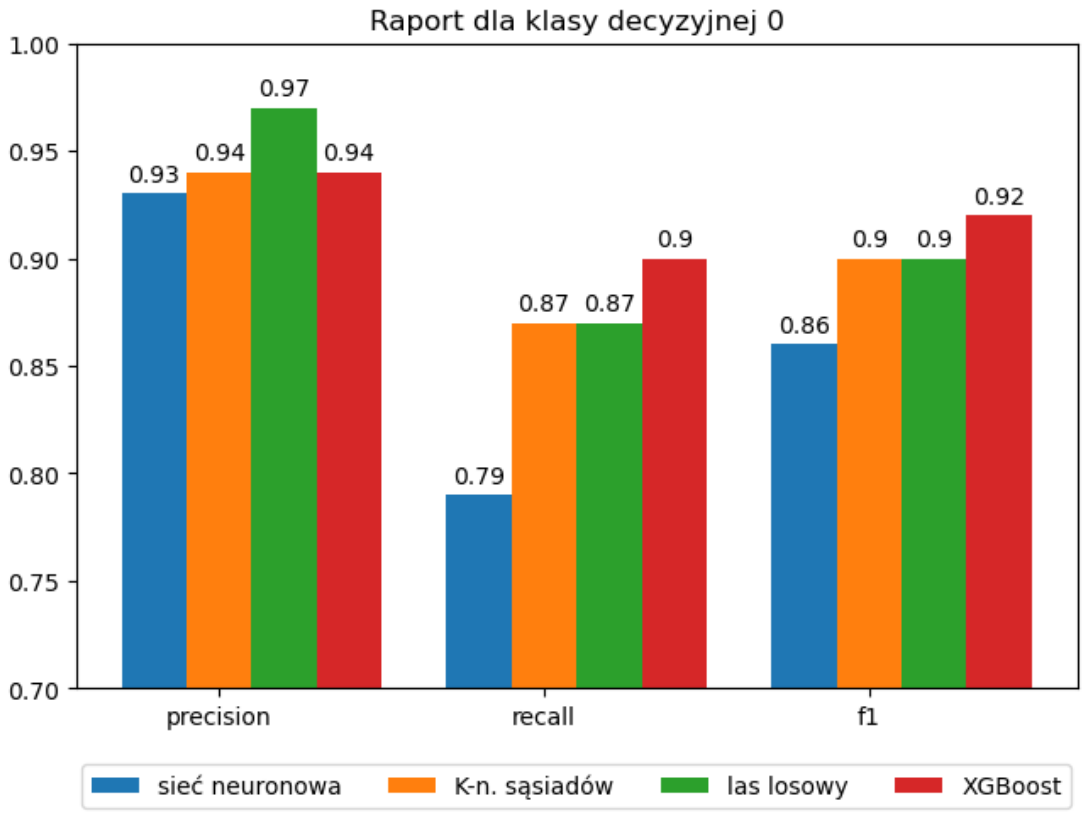
Rysunek 3.7 Wykres poziomu ważności cech, w XGBoost

3.6. Wybór modelu

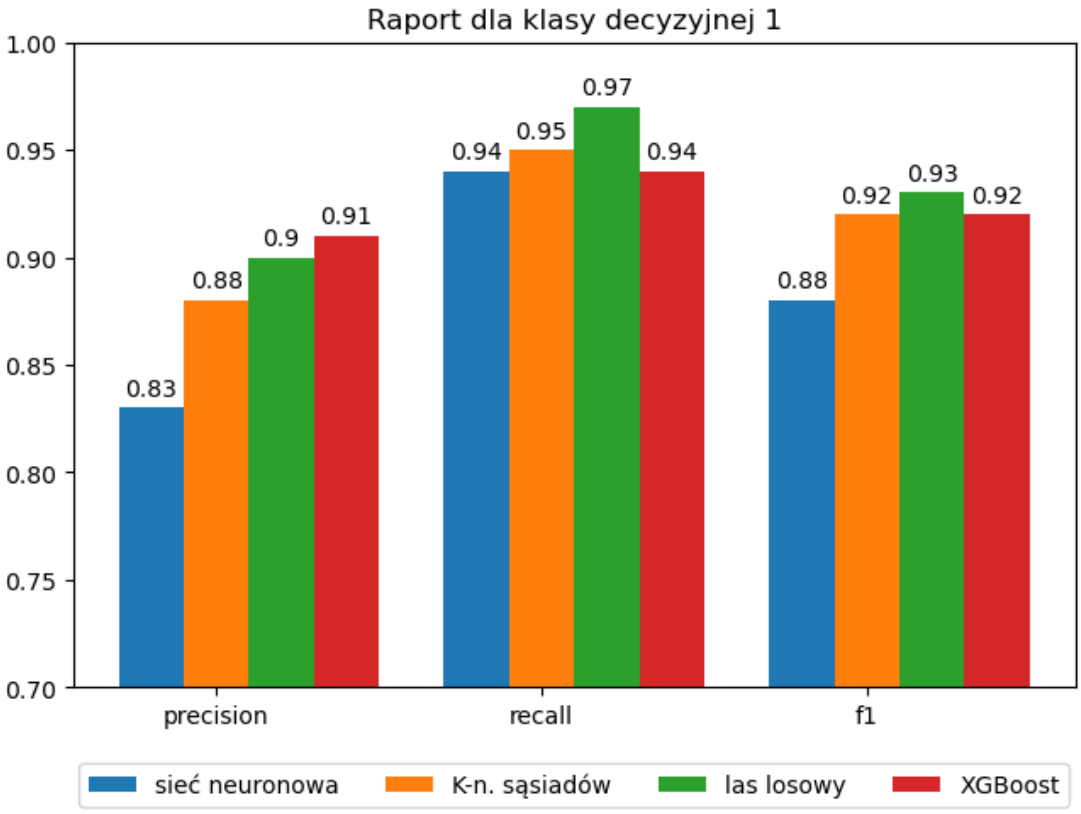
Po wytrenowaniu wszystkich czterech modeli należało je ze sobą porównać i wyłonić najlepszy. W tym celu używając każdego z nich, przeprowadzono predykcję na zbiorze danych testowych. Następnie każdy ze zbiorów wyników predykcji, został porównany ze zbiorem prawdziwych etykiet, a do każdej pary, stworzono raport, z użyciem biblioteki Scikit-learn. Wartości kryteriów z raportów zostały zaprezentowane na rysunkach: 3.8, 3.9 i 3.10 , wizualizując odpowiednio wyniki dla:

- klasy decyzyjnej 0,
- klasy decyzyjnej 1 ,
- średniej wyników z obu klas decyzyjnych.

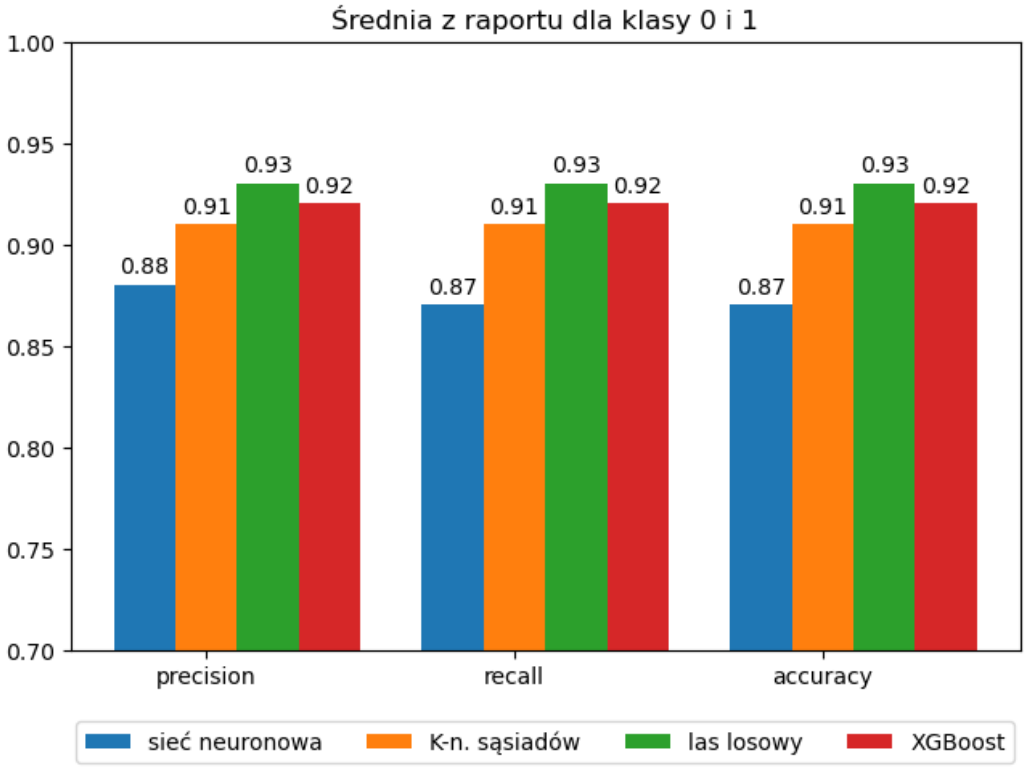
W grupie testowej znalazło się 97 przypadków, w których wynik był negatywny pod względem wystąpienia cukrzycy i 103, w których wynik był pozytywny.



Rysunek 3.8 Raporty modeli, dla klasy decyzyjnej 0



Rysunek 3.9 Raporty modeli, dla klasy decyzyjnej 1



Rysunek 3.10 Raporty modeli, dla średnich wartości obu klas decyzyjnych

Jak widać, pod uwagę zostały wzięte cztery kryteria:

- precision,
- recall,
- f1 score,
- accuracy.

Pierwsze kryterium – precision, określa procent przypadków, które zostały poprawnie określone zgodnie z przyjętą klasą decyzyjną. Na przykładzie sieci neuronowej, 93% predykcji, w których model rozpoznał klasę decyzyjną „0” - wynik negatywny, było zgodne z prawdziwą wartością etykiety i 83% predykcji wyniku pozytywnego zostało określone poprawnie. Kolejnym kryterium jest recall. Wskazuje ono, jaki procent klasy decyzyjnej, został rozpoznany. Ponownie na przykładzie modelu sieci neuronowej: 79% wszystkich wyników, które powinny zostać sklasyfikowane jako „0”, zostało poprawnie określone w ten sposób. Z wyników, które powinny zostać sklasyfikowane jako „1”, 94% zostało oznaczone w ten sposób. W celu uogólnienia oceny jakości średnich wyników z obu klas zastosowano accuracy. Accuracy jest stosunkiem wszystkich dobrze rozpoznanych etykiet, do ilości wszystkich przeprowadzonych predykcji. Do uogólnienia wyników dla poszczególnych klas decyzyjnych, użyto f1 score, obliczane w następujący sposób:

$$f1\ score = 2 \cdot \frac{precision\ \cdot\ recall}{precision + recall}$$

Dokładne wyniki poszczególnych modeli zostały przedstawione poniżej w tabelach:

- Dla klasy decyzyjnej „0” – tabela 1,
- Dla klasy decyzyjnej „1” – tabela 2,
- Dla uśrednionych wyników obu klas decyzyjnych – tabela 3.

Tabela 1 wyniki dla klasy decyzyjnej 0

model	precision	recall	f1
sieć neuronowa	0.93	0.79	0.86
K-n. sąsiadów	0.94	0.87	0.9
las losowy	0.97	0.87	0.9
XGBoost	0.94	0.9	0.92

Tabela 2 wyniki dla klasy decyzyjnej 2

model	precision	recall	f1
sieć neuronowa	0.83	0.94	0.88
K-n. sąsiadów	0.88	0.95	0.92
las losowy	0.9	0.97	0.93
XGBoost	0.91	0.94	0.92

Tabela 3 uśrednione wyniki dla obu klas decyzyjnych

model	precision	recall	accuracy
sieć neuronowa	0.88	0.87	0.87
K-n. sąsiadów	0.91	0.91	0.91
las losowy	0.93	0.93	0.93
XGBoost	0.92	0.92	0.92

Analizując uzyskane wyniki, można zauważyć, że wszystkie modele predykcyjne odniosły lepsze rezultaty w określaniu wyniku pozytywnego, niż negatywnego. Powodowane to może z być dużą różnicą w ilości danych, dla poszczególnych klas decyzyjnych. Wyników z klasą decyzyjną „1” (pozytywny), było prawie o połowę mniej od tych z klasą „0” (negatywny). Ilość klas została zrównoważona, poprzez losowe powtórzenie rekordów z odpowiednią klasą. W efekcie część danych w grupie testowej mogła zostać powtórzona w grupie uczącej, a same dane były mniej różnorodne. Oznacza to, że wynik może być zawyżony. Spośród czterech wytrenowanych modeli, sieć neuronowa i K-najbliższych sąsiadów wypadły zdecydowanie gorzej niż las losowy i XGBoost, w praktycznie wszystkich rozpatrywanych kryteriach. W związku z tym nie brano ich dalej pod uwagę. Natomiast w przypadku lasu losowego i XGBoost, otrzymane raporty trzeba było poddać głębszej analizie, ponieważ żaden z nich, nie jest jednoznacznie lepszy. W wyniku analizy stwierdzono, że w rozpatrywanym przypadku, zaimplementowany do aplikacji powinien zostać las losowy. Jego wyniki, w rozpatrywaniu klasy decyzyjnej „1”, są lepsze niż te, otrzymane z użyciem XGboost. Najistotniejszym czynnikiem, wydaje się bardzo wysoki poziom recall, który wyniósł 97%. Oznacza to, że 97% danych, które pochodziły od osób z cukrzycą, zostało poprawnie rozpoznanych i oznaczonych jako wyniki pozytywne. Las losowy uzyskał też niewielką przewagę, przy rozpatrzeniu uśrednionych wyników obu klas decyzyjnych.

W prowadzonych badaniach czynnikami mogącymi mieć negatywny wpływ na jakość predykcji modeli mogły być:

- niekompletność danych
- niezrównoważona ilość danych

Wpływ pierwszego czynnika, mogącego mieć negatywny wpływ na model predykcyjny został zminimalizowany poprzez uzupełninie danych odpowiadającymi im medianami. Przypuścić jednak można, że różnią się względem prawdziwych wartości, co obniża dokładność predykcji. Szczególną uwagą powinny zostać obdarzone, wyniki badania insuliny po dwóch godzinach od spożycia posiłku, gdyż dwa najlepsze modele- las losowy i XGBoost, nadały tej cesze najwyższą wagę. Na dodatek waga ta, była zdecydowanie większa, od wagi pozostałych cech. Drugim czynnikiem była niezrównoważona ilość danych dla poszczególnych klas decyzyjnych. Reprezentacja wyników pozytywnych była bowiem znacząco mniejsza od negatywnych. Zwiększenie jej do ilości wyników negatywnych, prawdziwymi, unikalnymi wynikami, zamiast powieleniem już istniejących, mogłoby mieć pozytywny wpływ, na każdy z wytrenowanych modeli. Możliwym też jest, że dokładność predykcji, może zostać podniesiona, poprzez wprowadzenie pewnego uogólnienia do danych. Niektóre dane, przykładowo te określające BMI, poziom glukozy, czy insuliny, można by zastąpić opisowo, lub dodać opis, definiujący je w kontekście odstawania od przyjętych normy. Przykładowo: znacznie powyżej normy, powyżej normy, w normie, poniżej normy, znacznie poniżej normy. Jednak nie wiadomo czy taki zabieg przyniósłby oczekiwany efekt.

4. Aplikacja AmIDiabeete

Model predykcyjny z najlepszymi wynikami, czyli las losowy, został zapisany przy pomocy biblioteki Joblib, a następnie zaimplementowany w prostej aplikacji sieciowej. Aplikacja została nazwana AmIDiabeete, w tłumaczeniu: CzyJestemCukrzykiem. Sama aplikacja prezentuje możliwości wykorzystania uczenia maszynowego jako narzędzia wspomagającego proces rozpoznania cukrzycy. Na dodatek narzędzie te, może być dostępne dla wszystkich. Co prawda przez wzgląd na specyfikę danych, w wiarygodny sposób przeprowadzić można tylko predykcję wystąpienia cukrzycy u kobiet po 21 pierwszym roku życia. Jednak z dostępem do innych zestawów danych, nie byłoby przeszkód, by aplikację rozwinąć i umożliwić korzystanie z niej innym grupom. Oczywistym jest, że aplikacja nie stawia diagnozy medycznej. Wyniki jej, zwłaszcza pozytywne, można jednak traktować jako ostrzeżenie oraz motywację do konsultacji z lekarzem. Możliwym też, byłoby użycie modeli predykcyjnych, w placówkach medycznych, w celu klasyfikacji wystąpienie cukrzycy, a nawet innych schorzeń, w oparciu o posiadane już przez placówkę dane pacjentów.

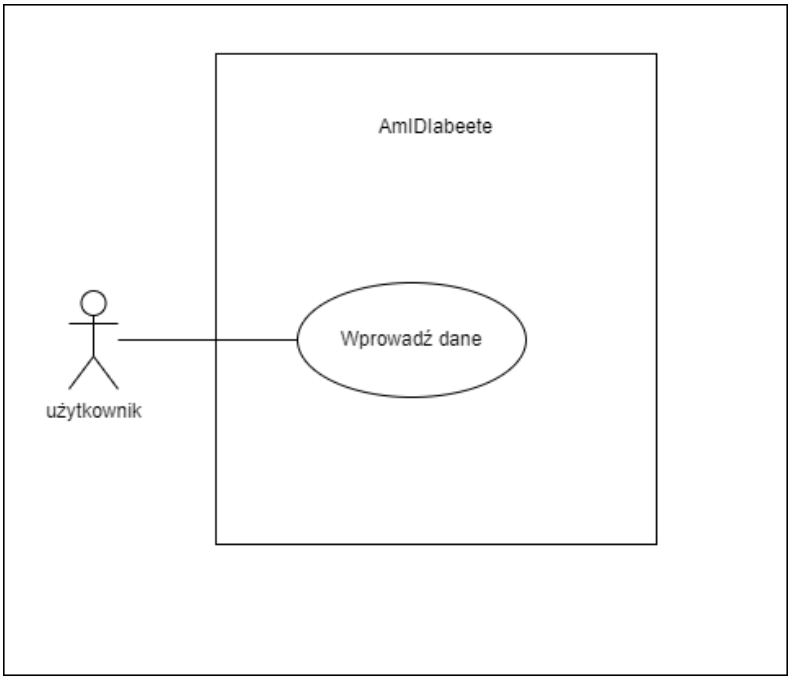
4.1. Konstrukcja aplikacji

Warstwa użytkownika (frontend), została stworzona w HTML 5. Do jej stylowania, użyto kaskadowych arkuszy stylów CSS, oraz frameworku Bootstrap. Dzięki zastosowaniu tych narzędzi, stworzona strona jest estetyczna i reaktywna, co znacznie zwiększa komfort podczas przeglądania.

Warstwa backend, została stworzona, przy użyciu frameworku Flask. Dane wprowadzane do formularza przesyłane są na serwer za pomocą metody POST. W kolejnym kroku następuje sprawdzenie kompletności. W przypadku, gdy wartość DPF nie została przekazana, zostaje jej przypisana mediana z zestawu danych, wykorzystanych do uczenia modelu. Następnie przekazane dane zostają umieszczone w tablicy i w tej formie przekazane do wcześniej wytrenowanego modelu. Na ich podstawie model wykonuję predykcję. Następnie wyświetlany

jest nowy szablon HTML, który w zależności od tego czy model zwrócił „0” (reprezentujące wynik negatywny), czy „1” (reprezentujące wynik pozytywny) wyświetla stosowny komunikat.

4.2. Diagram przypadków użycia



Rysunek 4.1 Diagram przypadków użycia

Przypadek użycia: Wprowadź dane

Aktor: Użytkownik

Opis: Użytkownik z użyciem aplikacji, dokonuje predykcji wystąpienia cukrzycy.

Warunki wstępne: Otwarty widok z formularzem.

Przebieg:

1. Użytkownik wprowadza dane do formularza i zatwierdza wprowadzone dane.
2. System sprawdza poprawność wprowadzonych danych.
   2.a. Wprowadzone dane są poprawne.
      2.a.1. System za pomocą modelu przeprowadza predykcję, a użytkownikowi zostaje wyświetlony jej wynik.
   2.b. Wprowadzone dane są niepoprawne.
      2.b.1 System informuje o błędzie w podanych danych.

4.3. Opis działania aplikacji

Pierwszym ekranem, który widzi użytkownik aplikacji, jest ekran z formularzem, przedstawiony na rysunku 4.2. Użytkownik powinien uzupełnić formularz stosownymi danymi:

- ilością przebytych ciąż,
- poziomem glukozy dwie godziny po spożyciu posiłku,
- ciśnieniem krwi,
- poziomem insuliny dwie godziny po spożyciu posiłku,
- BMI,
- DPF,
- wiekiem.

Oprócz DPF wszystkie dane muszą koniecznie zostać uzupełnione przez użytkownika. W przypadku nieuzupełnienia którejś z nich wyświetlone zostanie odpowiednie powiadomienie. DPF natomiast jest parametrem, który może zostać wypełniony lub może pozostać niewypełniony. W sytuacji, gdy użytkownik nie chce podawać DPF, komórkę powinien pozostawić pustą. Każda z komórek formularza musi zostać uzupełniona w zależności od wprowadzanego parametru liczbą całkowitą lub zmiennoprzecinkową. W przypadku uzupełnienia niewłaściwą wartością zostanie wyświetlony komunikat o błędzie. Wysłanie ankiety możliwe będzie dopiero gdy wszystkie wymagane pola w formularzu zostaną

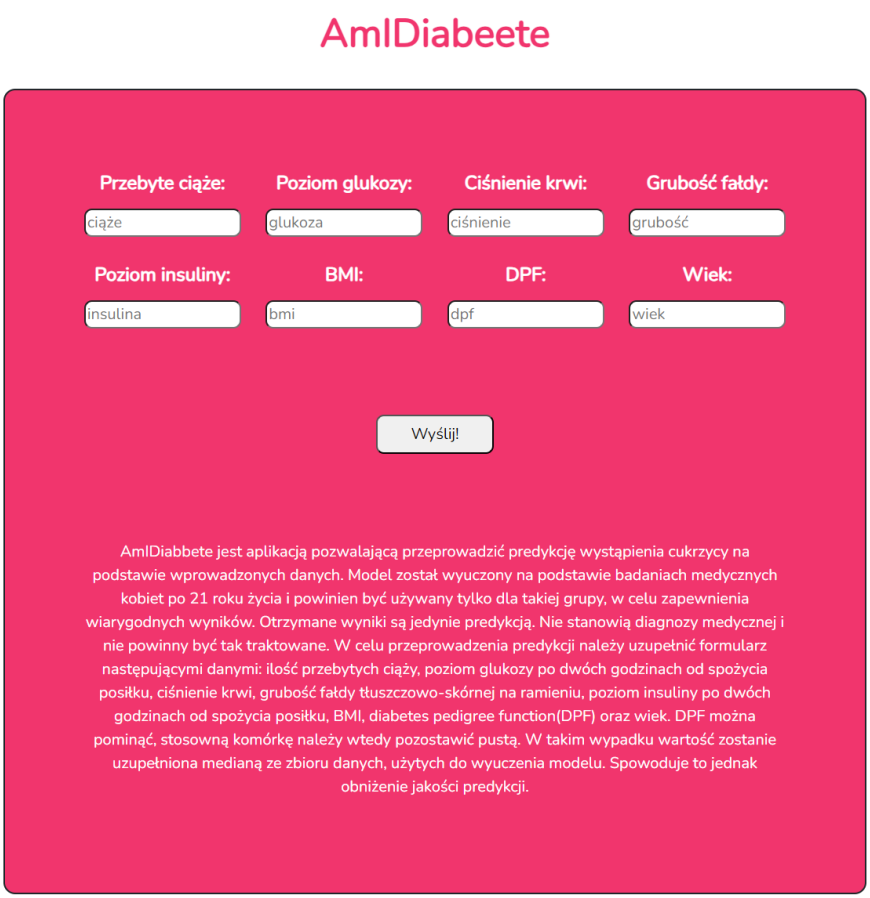
poprawnie uzupełnione. Pod ankietą, którą uzupełnia użytkownik, znajduje się informacja, opisująca krótko działanie i przeznaczenie aplikacji, o następującej treści:

„ AmIDiabbete jest aplikacją pozwalającą przeprowadzić predykcję wystąpienia cukrzycy na podstawie wprowadzonych danych. Model został wyuczony na podstawie badań medycznych kobiet po 21 roku życia i powinien być używany tylko dla takiej grupy, w celu zapewnienia wiarygodnych wyników. Otrzymane wyniki są jedynie predykcją. Nie stanowią diagnozy medycznej i nie powinny być tak traktowane. W celu przeprowadzenia predykcji należy uzupełnić formularz następującymi danymi: ilość przebytych ciąży, poziom glukozy po dwóch godzinach od spożycia posiłku, ciśnienie krwi, grubość fałdy tłuszczowo-skórnej na ramieniu, poziom insuliny po dwóch godzinach od spożycia posiłku, BMI, diabetes pedigree function(DPF) oraz wiek. DPF można pominąć, stosowną komórkę należy wtedy pozostawić pustą. W takim wypadku wartość zostanie uzupełniona medianą ze zbioru danych, użytych do wyuczenia modelu. Spowoduje to jednak obniżenie jakości predykcji.”

Po wysłaniu formularza i analizie danych, użytkownikowi wyświetlany jest jeden z komunikatów:

- informujący o podejrzeniu wystąpienia u niego cukrzycy,
- informujący o braku podstaw do takich przypuszczeń.

Oba komunikaty, zostały przedstawione na rysunkach 4.3 i 4.4 . Użytkownik ma możliwość powrotu do strony głównej zawierającej formularz, poprzez kliknięcie w odpowiednio opisany przycisk „powrót na stronę główną”.



Rysunek 4.2 Ekran startowy aplikacji AmIDiabeete

Rysunek 4.3 Ekran informujący o negatywnym wyniku predykcji

Rysunek 4.4 Ekran informujący o pozytywnym wyniku predykcji

5. Podsumowanie

W pracy przedstawiono pełen proces rozwiązania problemu klasyfikacji przypadków wystąpienia cukrzycy u kobiet po dwudziestym pierwszym roku życia. Opisana też została analiza danych użytych do uczenia modeli predykcyjnych. Wskazano nieprawidłowości w wykorzystanym zestawie danych tj. niekompletność oraz niezrównoważoną ilość danych, dla poszczególnych klas decyzyjnych. Opisano następujące działania, które podjęto w celu zminimalizowanie negatywnego efektu na jakość predykcji:

- powielenie danych w mniej licznej klasie decyzyjnej (oversampling)
- zastąpienie braków medianami, odpowiednimi dla poszczególnych klas decyzyjnych.

Z użyciem przygotowanych danych wyuczone zostały cztery różne modele predykcyjne. Opisane zostało dopasowanie ich parametrów oraz kryteria oceny, podczas wyboru najlepszych z nich. Ostatecznie porównano wszystkie cztery modele oraz poddano je ocenie, używając do tego jednolitych kryteriów. Po przeprowadzeniu analizy raportów dla każdego modelu wybrany został las losowy. Został on uznany za najlepszy do prognozowania wystąpienia cukrzycy w oparciu o wybrane dane medyczne. Osiągnięte wyniki uznane zostały za satysfakcjonujące. Model został zaimplementowany z sukcesem w prostej aplikacji sieciowej, umożliwiającej swoim użytkownikom przeprowadzenie predykcji, na podstawie wprowadzonych przez nich danych. Pomimo tego, że wyników predykcji nie należy traktować jako diagnozy medycznej, mogą one być traktowane jako wiarygodny sygnał ostrzegawczy dla użytkownika. Główne cele i założenia: wytrenowanie modelu umożliwiającego wiarygodną klasyfikację wystąpienia cukrzycy oraz implementacja modelu do ogólnodostępnej aplikacji sieciowej, zostały spełnione.

5.1. Plany rozwoju

Stworzona na potrzeby projektu aplikacja AmIDiabeete umożliwia przeprowadzenie wiarygodnej predykcji dla kobiet po dwudziestym pierwszym roku życia. Najistotniejszym kierunkiem rozwoju jest poszerzenie grupy użytkowników. Pierwszą planowaną zmianą jest dodanie kolejnego modelu, zdolnego do prognozowania wystąpienia cukrzycy wśród mężczyzn. W tym celu trzeba będzie pozyskać stosowne dane medyczne, dotyczące tej grupy. W takim przypadku konieczna będzie także modyfikacja aplikacji. W rozbudowanej aplikacji użytkownik będzie miał możliwość wyboru płci. W zależności od wybranej płci wyświetlana będzie stosowna ankieta, a wprowadzone dane będą analizowane przez odpowiedni model predykcyjny.

Planowane jest również poprawienie jakości predykcji dla kobiet po 21 roku życia. Użyte do wytrenowania modelu dane, nie były kompletne. Najwięcej brakujących wartości stwierdzono w wynikach poziomu insuliny, który okazał się też być najważniejszą cechą rozpatrywaną przez model. Uzyskanie dostępu do kompletnych danych oraz zbioru z większą ilością zbadanych osób, szczególnie osób ze zdiagnozowaną cukrzycą, umożliwiłoby wytrenowanie modelu o wyższej dokładności.

Spis rysunków

Bibliografia

1. Python, dokumentacja https://docs.python.org/pl/3/.
2. Bootstrap 5, dokumentacja https://getbootstrap.com/docs/5.3/getting-started/introduction/.
3. Sci-kit learn, dokumentacja https://scikit-learn.org/stable/.
4. TensorFlow, dokumentacja https://www.tensorflow.org/api_docs.
5. R. C. Martin, Clean Code, 2009.
6. P. Drozda, K. Sopyła, Accelerating SVM with GPU: The State of the Art ICAISC 2016, In book: Artificial Intelligence and Soft Computing, pp.624-634.
7. T. M. Mitchell, The Discipline of Machine Learning, School of Computer Science Carnegie Mellon University Pittsburgh, 2006.
8. Matplotlib, dokumentacja https://matplotlib.org/stable/index.html.
9. Flask, dokumentacja https://flask.palletsprojects.com/en/3.0.x/.
10. A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2022.
11. A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, 2012
12. A. Ng, Machine Learning Yearning, 2018
13. M. R. Mufid, A. Basofi, M. U. H. Al Rasyid, I. F. Rochimansyah, A. rokhim, Design an MVC Model using Python for Flask Framework Development, 2019
14. J. D. Kelleher, B. M. Namee, A. D'Arcy, Fundamentals of Machine Learning for Predictive Data Analytics, 2015
15. A. Kapoor, A. Gulli, S. Pal, F. Chollet, Deep Learning with Tensorflow and Keras: Build and deploy supervised, unsupervised, deep, and reinforcement learning models, 2022
16. G.Hackeling, Mastering Machine Learning with scikit-learn, 2017