

Katolicki Uniwersytet Lubelski Jana Pawła II
Wydział Nauk Ścisłych i Nauk o Zdrowiu
Instytut Matematyki i Informatyki

Informatyka, studia stacjonarne II stopnia

Szymon Mrozowski

Nr. Albumu: 146453

Wykorzystanie technik uczenia maszynowego w medycynie

Praca magisterska
napisana na seminarium **Algorytmizacja i modelowanie**
pod kierunkiem
dr Doroty Pylak

Lublin, 2022r

Spis treści

1. Wstęp	5
Cel Pracy	5
Struktura Pracy	6
Podstawowe pojęcia	6
2. Machine learning	8
Historia	9
Machine learning w medycynie	13
Ogólne zastosowanie	15
3. Data Mining	17
Historia	17
Zastosowanie	19
Proces analizy danych	21
Ustalenie celów	21
Zbieranie danych	21
Czyszczenie danych	22
Analizowanie danych	23
Interpretacja wyników	24
4. Język programowania	25
Historia	25
IDE	26
Jupyter Notebook	27
5. Analiza danych	28
Prezentacja przykładowych danych	29
Czynniki wpływające na choroby serca	31
Modele predykcji	35

6. Aplikacja.....	38
Interfejs użytkownika	38
Implementacja	42
Obserwacje	48
7. Podsumowanie.....	51
Literatura.....	53

1. Wstęp

Zdrowie jest jedną z najważniejszych, jeżeli nie najważniejszą wartością w życiu człowieka. Od zarania dziejów ludzie starali się jak mogli leczyć choroby, wykorzystując zasoby naturalne lub to, co znaleźli w swoim środowisku. W dzisiejszych czasach medycyna bardzo się rozwinęła. Coraz więcej rozumiemy na temat powstawania chorób, ich leczenia, a także profilaktyki, na przykład poprzez szczepienia czy wykrywanie genów zwiększających ryzyko pojawienia się chorób nowotworowych. Przykładem takiego genu jest BRCA1. Niektóre jego mutacje sprzyjają powstawaniu raka piersi. Pomimo tak bardzo rozwiniętej technologii, według danych z 2017 roku każdego dnia umiera około 150 000 ludzi. Około 90% tych zgonów jest spowodowane różnymi chorobami. Główną przyczyną zgonów, zarówno w Polsce, jak i na świecie, są choroby układu krążenia. Jest to grupa chorób związanych z zaburzeniem pracy serca oraz naczyń krwionośnych. Od ponad 20 lat znajduje się one na pierwszym miejscu najpowszechniejszych chorób ludzkości zabijając około 17 milionów ludzi każdego roku, niemal dwa razy więcej niż nowotwory. Dodatkowo liczba zgonów z ich powodu stale rośnie od wielu lat. Jest to problem, który wymaga znalezienia szybkiego rozwiązania, a w tym pomóc może data mining (z ang. Analiza danych). Dzięki uczeniu maszynowemu oraz analizie danych możemy stworzyć modele, które są w stanie przedwcześnie wykrywać choroby układu krążenia u pacjentów na podstawie ich wyników badań lub dodatkowo ich trybu życia. Jest to niezwykle istotne ponieważ wcześniejsze wykrycie choroby oraz szybsze rozpoczęcie leczenia diametralnie zwiększa szanse pacjenta na długie i zdrowsze życie.

Cel Pracy

Praca ma na celu przedstawienie czym jest machine learning oraz analiza danych, jaki one mają wpływ na dzisiejszy świat, a także w jaki sposób mogą one wspomóc medycynę i lekarzy we wczesnym wykrywaniu niektórych chorób serca. Część praktyczna przedstawia proces analizy danych w praktyce. Są tam analizowane dane pacjentów chorych na serce przy użyciu narzędzia Jupyter Notebook. Natomiast informacje, które uzyskamy z analizy danych zostaną wykorzystane przy stworzeniu programu desktopowego do wykrywania chorób serca. Aplikacja będzie w stanie analizować dane pacjenta i na ich podstawie diagnozować jego stan. Dodatkowo program będzie udostępniać raport przedstawiający bardziej szczegółowy wgląd w

wyniki modelu predykcyjnego. Jak dobrze wiemy sztuczna inteligencja już od wielu lat jest szeroko wykorzystywana w różnych dziedzinach. Jednak czy jest ona w stanie pomóc nam w leczeniu jakichkolwiek chorób? Czy da się ją wykorzystać w medycynie? Czy może już jest ona tam wykorzystywana. Na te pytania postaramy się odpowiedzieć w tej pracy.

Struktura Pracy

Struktura pracy jest podzielona na 7 rozdziałów. Wstęp stanowi wprowadzenie do pracy. Przedstawia cele pracy oraz wyjaśnia podstawowe pojęcia. Dwa początkowe rozdziały opisują część teoretyczną dotyczącą uczenia maszynowego oraz analizy danych. Jest w nich opisana ich historia, sposób działania, ogólne zastosowanie oraz jakie mają zastosowanie w medycynie. Czwarty punkt przedstawia język programowania Python, który został wykorzystany w pracy do stworzenia programu oraz przy analizie danych pacjentów. Dalsza część pracy związana jest z częścią praktyczną.

Część praktyczna pracy została podzielona na dwie części. Pierwsza część dotyczy analizy danych pacjentów przy użyciu narzędzia Jupyter Notebook. Przedstawia ona wykorzystanie data mining w praktyce. Znajduje się tam wiele wykresów oraz istotnych informacji dotyczących samego zestawu danych, które udało się uzyskać dzięki ich dogłębnej analizie. Uzyskane wyniki są również dołączone do pracy magisterskiej. Druga część jest związana ze stworzeniem programu komputerowego, który ma za zadanie stwierdzić czy dany pacjent jest chory na podstawie jego wyników badań. Rozdział ten opisuje dokładnie cały program. Przedstawia on interfejs użytkownika wraz z opisem poszczególnych elementów. Są tam również przedstawione najważniejsze biblioteki i metody wykorzystane przy implementacji jak na przykład działanie lasu decyzyjnego z biblioteki scikit-learn. Dodatkowo umieszczone są tam spostrzeżenia, które powstały podczas pisania i testowania aplikacji. Ostatni rozdział podsumowuje całą pracę magisterską i odpowiada na pytania zadane we wstępie.

Podstawowe pojęcia

Praca wykorzystuje pojęcia, skróty oraz angielskie nazwy związane ze sztuczną inteligencją. Dla lepszego i łatwiejszego zrozumienia pracy poniżej zostały przedstawione i wyjaśnione najczęściej wykorzystywane sformułowania:

Machine learning (ML)– uczenie maszynowe,

Deep learning – uczenie głębokie, jest ono ściśle związane z sieciami neuronowymi,

Data Mining – proces analizowania danych,

Model – model predykcji, jak na przykład drzewo decyzyjne,

Indeks Giniego – inaczej wskaźnik Giniego wykorzystywany w drzewie decyzyjnym. Pomaga on w optymalnym podziale drzewa,

IDE - zintegrowane środowisko programistyczne,

Open-source – termin, który oznacza, że kod źródłowy programu oraz sam program są dostępne publicznie za darmo.

Biblioteka – zbiór gotowych klas i metod, które można dodać do projektu i z nich korzystać.

2. Machine learning

Najpierw w skrócie wytłumaczmy, co oznacza pojęcie machine learning (z ang. uczenie maszynowe). Uczenie maszynowe skupia się na tworzeniu programów komputerowych, które są w stanie analizować dane i uczyć się na ich podstawie. Dzięki takiej umiejętności bliskiej ludzkiemu procesowi poznawczemu jesteśmy w stanie pisać systemy, które potrafią uczyć się same — nabywać doświadczenie tak samo, jak ludzie. To jest też głównym celem uczenia maszynowego, by stworzyć system zdolny to pełnej samodzielności. Czyli taki, który samodzielnie się uczy, zdobywa doświadczenie oraz sam podejmuje decyzje. Możemy wyróżnić trzy główne rodzaje uczenia, które są wykorzystywane w machine learningu.

Pierwszy rodzaj to uczenie nadzorowane. Algorytm uczy się na podstawie danych, które są oznaczone poprawną odpowiedzią. Dzięki temu algorytm od razu wie, co oznacza dany zbiór danych. Przeanalizowanie wielu takich oznaczonych informacji daje algorytmowi duże prawdopodobieństwo na znalezienie poprawnej odpowiedzi. Przykładem może być uczenie maszyny rozpoznawania psów i kotów. Do nauki wybieramy zdjęcia, które mają opis przedstawiający, co się znajduje na obrazku. Algorytm wiedząc, co jest na obrazku, będzie uczył się cech charakterystycznych danego zwierzęcia. Po skończonej nauce, jeżeli podamy zdjęcie kota już bez opisu, algorytm będzie w stanie rozpoznać kota.

Drugi rodzaj to uczenie nienadzorowane, czyli takie, gdzie algorytm nie dostaje poprawnych odpowiedzi. Sam musi wywnioskować na podstawie danych treningowych, jaka jest poprawna odpowiedź lub jak skategoryzować dane wejściowe. Dobrym przykładem obrazującym problem może być klasteryzacja (grupowanie). Jej celem jest znalezienie naturalnych grup lub klastrów w taki sposób, aby dane wejściowe przypisać do odpowiedniej grupy, która jest podobna do innych danych na podstawie wcześniej zdefiniowanego podobieństwa.

Uczenie wzmocnione jest trzecim rodzajem uczenia maszynowego. Polega ono na tym, że algorytm dostaje z góry ustalony zestaw reguł lub działań. Następnie dokonuje analizy i wykorzystuje reguły tak, aby osiągnąć pożądany efekt. Dobrze obrazuje to aplikacja AlphaGo

Zero stworzona w 2017 roku¹. Jest to aplikacja, która potrafi grać w Go. Do jej uczenia wykorzystano jedynie reguły gry. Aplikacja nie uczyła się jak inne tego typu aplikacje na podstawie historycznych rozegranych partii między ludźmi. Do działania program dostał jedynie podstawowe zasady gry. Następnie rozgrywał partie grając sama ze sobą. W ciągu kilku kolejnych dni pokonał inny program, mimo że był to program, który zwyciężył z Lee Sedol zawodowym graczem Go. A już po ponad miesiącu AlphaGo Zero zostało najlepszym graczem go na świecie tylko i wyłącznie dzięki samodzielnej nauce – bez interakcji z ludźmi. Przy uczeniu wzmocnionym możemy często natknąć się na słowo „Q-learning”. Oznacza to uczenie się optymalnego sposobu poruszania się po środowisku, poprzez oszacowanie, które dostępne działanie w danym momencie doprowadzi do najlepszego wyniku, zdefiniowanego przez nagrody otrzymywane za osiągnięcie pewnego rezultatu.

Po zapoznaniu się z pojęciem ML, możemy teraz przejść do krótkiego omówienia historii jego powstania.

Historia

Słowo „Machine learning” po raz pierwszy zostało użyte przez Artura Samuela w 1952 roku. Samuel był amerykańskim pionierem w dziedzinie sztucznej inteligencji oraz gier komputerowych. Stworzył on bowiem program do grania w warcaby, który potrafił się uczyć w trakcie swojego działania². Wykorzystał do tego komputer IBM 701. Miał do dyspozycji niewielką ilość pamięci więc, zamiast sprawdzać każde możliwe ustawienie pionków, wykorzystał on algorytm alpha beta do zminimalizowania ilości obliczeń. Działa on tak, że jeśli sprawdzając kolejny ruch napotka chociaż jeden stan, który daje gorszy wynik niż poprzedni ruch, to kończy analizowanie takiego ruchu i sprawdza następny. Takie działanie przyspiesza działanie programu oraz zmniejsza ilość niepotrzebnych obliczeń. Samuel wystawił program, aby grał sam ze sobą i w ten sposób uczył się gry. W roku 1961 jego program wygrał przeciwko Robertowi Nealeyowi³, który był ówczesnie znanym mistrzem warcabów. Później okazało się, że Nealey nie był tak dobry jak mówiono, co nie przeszkodziło jednak dużemu rozgłosowi, i w rezultacie dało lepszy start uczeniu maszynowemu.

¹ Aplikacja AlphaGo: https://en.wikipedia.org/wiki/AlphaGo_Zero [Dostęp: 15.02.2021]

² Wolfgang Ertel, Introduction to Artificial Intelligence, 2017 rok

³ Mecz w warcaby <http://www.fierz.ch/samuel.htm> - [Dostęp: 23.03.2021]

W roku 1957 powstał pierwszy algorytm wykorzystujący model neuronów mózgowych, nazwany Mark I Perceptron. Była to prosta sieć neuronowa stworzona do rozpoznawania obrazów. Jednak jej działanie nie było tak dobre, jak się spodziewano. Dodatkowo praca Minskyego i Paperta z 1969 wskazująca duże ograniczenia perceptronów wzmocniła zanik zainteresowania sztuczną inteligencją. Jednakże nie do końca mieli oni rację. Faktycznie zwykły perceptron składający się z jednego neuronu mógł wykonywać tylko podstawowe działania jak AND lub OR, które można było przeprowadzić prościej przy użyciu nieskomplikowanych i bardziej wydajnych metod. To czego brakowało, to dodanie kolejnej warstwy neuronów do perceptronu. Użycie jednej dodatkowej warstwy pozwala na przeprowadzanie dodatkowych operacji (np. XOR). Jako pierwsi wielowarstwowy perceptron stworzyli A. Ivakhnenko i V. Lapa. Otworzyło to nową drogę rozwoju sieci neuronowych. Między innymi dzięki tej pracy Ivakhnenko jest dzisiaj również określany mianem ojca deep learningu.

W latach 70' została opracowana propagacja wsteczna, jest to niezwykle istotny element wykorzystywany w sieciach neuronowych. Pozwala ona na dostosowanie wag pomiędzy ukrytymi warstwami neuronów, co przyspiesza i polepsza naukę programu. Dzisiaj jest ona powszechnie używana w deep learningu. W kolejnych latach maszynowego uczenie maszynowe rozwijało się dynamicznie. Między innymi pojawił się zdalnie sterowany pojazd ze Stanford⁴, który mógł poruszać się autonomicznie przy użyciu mapowania trójwymiarowego i nawigacji. Kunihiko Fukushima opublikował pracę⁵ o hierarchicznej wielowarstwowej sieci neuronowej zdolnej do wykrywania wzorców. Systemy oparte na jego pracy są powszechnie wykorzystywane w dzisiejszych czasach do rozpoznawania obrazów.

Kolejnym ważnym krokiem były próby stworzenia systemu, który będzie w stanie rozmawiać jak człowiek oraz rozpoznawać ludzką mowę. Wczesne początki tych badań sięgają lat 60. Pod koniec dekady system rozpoznawania mowy został udoskonalony tak, aby nie trzeba było się zatrzymywać po każdym słowie. Jednak w tamtych czasach komputery nadal nie posiadały wystarczającej mocy obliczeniowej. Dopiero w latach 90 nastąpił przełom. Dużym krokiem w przód był system LSTM zaproponowany przez Jürgena Schmidhubera i Seppa

⁴ Pojazd ze Stanford: <http://cyberneticzoo.com/cyberneticanimals/1960-stanford-cart-american> [Dostęp 15.04.2021]

⁵ Kunihiko Fukushima, Neocognitron: A hierarchical neural network capable of visual pattern recognition. 1988 rok

Hochreitera⁶. Sieci LSTM (Long Short-Term Memory) są typem rekurencyjnej sieci neuronowej w skrócie RNN. Rekurencyjne sieci neuronowe zawierają w sobie pętle, pozwalające na przechowywanie informacji. Takie rozwiązanie pozwala na wykorzystanie rozumowania z poprzednich doświadczeń, aby przewidzieć nadchodzące wydarzenia. Dzięki temu można przetwarzać bardziej skomplikowane zdania. Wykorzystuje się je w złożonych dziedzinach problemowych, takich jak tłumaczenie maszynowe, rozpoznawanie mowy i wiele innych. Dziś są one szeroko stosowane, między innymi przez takie firmy jak Google, na przykład do tworzenia transkrypcji z mowy na tekst. Według wpisu na oficjalnym blogu Google, nowy model zmniejszył liczbę błędów w transkrypcji o 49%⁷. Również Google Translate od 2016 korzysta z LSTM, co poprawiło jakość tłumaczonych tekstów o 60%. Sieć LSTM jest także wykorzystywana w wirtualnych asystentach głosowych takich jak Siri czy Alexa. Uczenie maszynowe to nie tylko sieci neuronowe. Wykorzystywane są również modele predykcji, takie jak regresja logistyczna czy drzewa decyzyjne, które zostały użyte w części praktycznej tej pracy. Modele takie, mimo że są prostsze od sieci neuronowych, często mogą dawać lepsze wyniki w szczególności, gdy ilość danych treningowych nie jest duża. W 1995 roku Tin Kam Ho, kierownik działu statystyki i badań nad uczeniem, opublikowała prace na temat Random decision forest (z ang: losowy las decyzyjny)⁸. Las ten uzyskała łącząc razem kilka drzew decyzyjnych. Dzięki temu zabiegowi otrzymała wyniki, które były znacznie dokładniejsze od tych, które można uzyskać dla pojedynczego drzewa decyzyjnego. Między innymi dzięki tej pracy Tin Kam Ho w 1999 otrzymała nagrodę 'ICDAR Young Scientist Award' przyznaną za wybitny wkład w analizę i rozpoznawanie dokumentów.

Dzisiaj coraz częściej pojawia się określenie „deepfake”. Zyskało ono dużą popularność między innymi dzięki użytkownikom Reddita, którzy w amatorski sposób korzystali z tego algorytmu wytwarzając własne nierzadko wysokiej jakości podrabiane filmy ze znanymi osobami. Samo określenie „deepfake” powstało w 2017 roku i pochodzi od jednego z użytkowników Reddita. Deepfake oznacza film, w którym została podmieniona twarz osoby na inną, zazwyczaj kogoś znanego na przykład polityka. Oprócz części wizualnej deepfake potrafi również podrobić głos osoby, dzięki czemu film wygląda bardziej realnie. Ostatnio pojawiło się

⁶ Sepp Hochreiter i Jürgen Schmidhuber, Long Short-Term Memory. 15.11.1997 rok

⁷ Oficjalny blok Google: <https://blog.google/products/google-voice/neon-prescription-or-rather-new> [Dostęp: 15.04.2021]

⁸ Tin Kam Ho, Random decision forests. 1995 rok

kilka filmów na instagramie oraz TikToku od użytkownika ‘deepmocruse’, gdzie pokazany jest Tom Cruise w różnych sytuacjach zawierających krótkie monologi. Jakość tych filmów jest fenomenalna. Oglądając je nie da się stwierdzić, że nie jest to prawdziwy Tom Cruise. Jego twarz, ruchy, głos oraz zachowanie wyglądają realistycznie. Nawet, gdy wykonuje szybkie ruchy, na przykład przy nakładaniu okularów i kapelusza, nie widać żadnych artefaktów, które wskazywałyby na podrobienie filmu. Początki deepfake sięgają 1997 roku, kiedy to powstał program do przerabiania filmów stworzony przez Christophera Breglera, Michela Covella i Malcolma Slaneya. Potrafił on dostosowywać ruchy twarzy oraz ust do podanego audio. Autorzy twierdzili, że można będzie wykorzystać ten algorytm na przykład przy dubbingu, aby film wyglądał bardziej realnie. Jednak jak nietrudno się domyślić pomysł znalazł inne zastosowanie m.in. do siania dezinformacji oraz do zabawy. Obecnie istnieją narzędzia takie jak face2face, które potrafią manipulować obrazem w czasie rzeczywistym. Należy więc pamiętać, że nawet podczas transmisji na żywo obraz, który widzimy może być automatycznie manipulowany przez algorytmy sztucznej inteligencji.

Deepfake należy do dziedziny nauki Computer vision (z ang. Rozpoznawanie obrazów), która obejmuje przetwarzanie, analizowanie i rozpoznawanie obrazów w formie cyfrowej. Głównie jest to rozpoznawanie elementów znajdujących się na obrazie, czyli klasyfikacja. Obecnie każdy smartphone ma aplikacje, które potrafią rozpoznawać twarz oraz nakładać filtry na obraz z kamery w czasie rzeczywistym. Facebook korzysta z algorytmów, które potrafią automatycznie tagować to co znajduje się na zdjęciu. Rozpoznawanie obrazów dla ludzi jest trywialnym zadaniem. Jeszcze zanim człowiek nauczy się chodzić potrafi rozpoznawać kształty, kolory, ludzi, swoich rodziców czy zwierzęta. Jednak ten sam proces dla maszyny jest bardziej skomplikowany niż może się wydawać. Wymaga dużej ilości danych, mocy obliczeniowej oraz bardziej zaawansowanych algorytmów, by móc nauczyć maszynę klasyfikowania zdjęć, rozpoznawania krawędzi, zwierząt czy ludzi. Rozpoznawanie obrazów jak i ogólnie uczenie maszynowe odgrywa również dużą rolę w medycynie, gdzie już od jakiegoś czasu jest wykorzystywane do wspomagania pracy lekarzy. W kolejnym podrozdziale przedstawimy więcej informacji na ten temat.

Machine learning w medycynie

Uczenie maszynowe szybko przenika do wielu obszarów w branży opieki zdrowotnej, od diagnozy, opracowywania leków aż do epidemiologii. Oprogramowanie oparte na AI może stwierdzić, czy pacjent cierpi na określoną chorobę, nawet zanim pojawią się wyraźne objawy. W badaniu opublikowanym w czasopiśmie naukowym Nature z 2018 roku pt. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography⁹ (z ang. Kompleksowe badanie przesiewowe w kierunku raka płuc z wykorzystaniem trójwymiarowego głębokiego uczenia na nisko dawkowej tomografii komputerowej klatki piersiowej) użyto algorytmu uczenia maszynowego, który wykorzystywał aktualne i wcześniejsze zdjęcia tomografii komputerowej pacjenta, aby przewidzieć ryzyko zachorowania na raka płuc. Przedstawione wyniki są bardzo zadowalające. Algorytm uzyskał wydajność ok 94%, co stwarza ogromne możliwości optymalizacji procesu badań przesiewowych przy pomocy komputera, automatyzacji i uczenia maszynowego.

Każdy człowiek jest inny. Jego organizm może reagować inaczej na te same leki. Dlatego też niezwykle istotne jest spersonalizowane podejście w medycynie. Jednakże, przy ogromnej liczbie pacjentów trudno jest przeanalizować z dużą dokładnością historie ich chorób, listę spożywanych leków czy wyniki wszystkich badań, również tych z przeszłości. Machine learning jest doskonałym narzędziem do analizowania tego typu danych i wyciągania odpowiednich wniosków w sposób automatyczny. Lekarze w niektórych krajach już korzystają z tego typu udoskonaleń. Pomaga to w lepszym doborze leków, czy nawet odpowiednich dawek. Przykładem może być firma IBM i Watson Oncology¹⁰, która jest wiodącą instytucją, pomagającą podejmować decyzje dotyczące leczenia. Wykorzystuje ona informacje medyczne i historię pacjenta, aby optymalizować wybór leczenia. Jak już zostało wspomniane w poprzednim podrozdziale uczenie maszynowe zajmuje się również rozpoznawaniem obrazów, co ma duże znaczenie również w medycynie. Przykładem może być projekt Microsoftu InnerEye¹¹, który zajmuje się narzędziami diagnostycznymi do analizy obrazów medycznych 3D. Szpital w

⁹ Nature, End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography, 2019r

¹⁰ Watson Oncology: <https://www.ibm.com/watson-health/solutions/cancer-research-treatment> [Dostęp: 12.03.2021]

¹¹ InnerEye: <https://www.microsoft.com/en-us/research/project/medical-image-analysis> [Dostęp: 12.03.2021]

Cambridge wykorzystuje tę technologię do dokładnej identyfikacji guzów na skanach pacjentów, skracając czas przetwarzania tomografii komputerowych i planowania leczenia nawet o 90%.

Metody uczenia maszynowego dostarczają narzędzi, które mogą usprawnić proces odkrywania nowych leków niemal na wszystkich jego etapach od wstępnego przesiewania związków leków, aż do przewidywania sukcesu na podstawie czynników biologicznych. Przykładem może być walidacja celów, identyfikacja biomarkerów czy analiza cyfrowych danych patologicznych w badaniach klinicznych. Opracowanie leku często wymaga dekady badań i miliardowych inwestycji, zanim trafi on na rynek. Dzieje się tak, ponieważ proces rozpoczyna się od przeanalizowania przestrzeni chemicznej obejmującej miliardy związków chemicznych i wybrania tych z pożądanymi cechami. Problem ten rozwiązuje południowokoreański startup Standigm¹². Firma ta stworzyła system, który bada przestrzeń chemiczną wygenerowaną przez sztuczną inteligencję w celu otrzymania nowych związków o pożądanym właściwościach. Następnie system interpretuje dane w celu odkrycia możliwych ścieżek rozwoju leków. Rozwiązania oferowane przez startup eliminują niepewność w procesie odkrywania leków, oszczędzając także czas i koszty produkcji.

Technologie oparte na sztucznej inteligencji i uczeniu maszynowym są dziś również wykorzystywane do monitorowania i przewidywania epidemii na całym świecie. Naukowcy mają obecnie dostęp do ogromnej ilości danych zbieranych z satelitów, mediów społecznościowych i informacji ze stron internetowych w czasie rzeczywistym. ML pomaga analizować te informacje i przewidywać wszystko, od pojawienia się nowych ognisk epidemii na przykład malarii, po poważne przewlekłe choroby zakaźne. Przewidywanie tych epidemii jest szczególnie pomocne w krajach trzeciego świata, ponieważ brakuje w nich kluczowej infrastruktury medycznej i systemów edukacyjnych. Doskonałym tego przykładem jest ProMED-mail¹³. Jest to internetowa platforma sprawozdawcza, która monitoruje rozwijające się i pojawiające się choroby oraz dostarcza raporty w czasie rzeczywistym na temat ich występowania.

¹² Startup Standigm: <https://www.standigm.com> [Dostęp: 23.03.2021]

¹³ Platforma sprawozdawcza: <https://promedmail.org> [Dostęp: 19.03.2021]

Ogólne zastosowanie

Częściowo zastosowanie uczenia maszynowego zostało opisane w poprzednim podrozdziale, a także w części o historii. W bieżącym podrozdziale zostaną zaprezentowane pozostałe przypadki wykorzystania ML w obecnych czasach.

Na początek usługi finansowe, ponieważ oprócz dużych firm technologicznych, to właśnie sektor bankowości i finansów jest jedną z wiodących branż w zakresie wdrażania sztucznej inteligencji. Uczenie maszynowe ma tutaj ogromny potencjał. Pomaga ono bankom i instytucjom finansowym w podejmowaniu lepszych decyzji. Potrafi śledzić wzorce wydatków klientów, zauważyć zamknięcie konta, zanim to nastąpi, przeprowadzić analizę rynku, a co najważniejsze algorytmy mogą łatwo zidentyfikować problemy i reagować na nie w czasie rzeczywistym. Przykładem może być wykrywanie różnych przestępstw finansowych przez wyszkoloną sieć neuronową. Wykorzystują to firmy takie jak Sanction Scanner¹⁴, oferujące usługi z zakresu przeciwdziałaniu praniu brudnych pieniędzy, z których korzysta wiele firm i banków. Fakt, że technologie oparte na uczeniu maszynowym dają zaawansowany wgląd w rynek, pozwala zarządzającym funduszami zidentyfikować konkretne zmiany rynkowe znacznie wcześniej w porównaniu z tradycyjnymi modelami inwestycyjnymi. Przydaje się to na przykład przy tradingu lub inwestowaniu w akcje firm. Dzięki monitorowaniu wyników handlu i wiadomości w czasie rzeczywistym oraz analizowaniu setek źródeł danych jednocześnie ML może przewidzieć wzrost lub spadek cen akcji, co daje wyraźną przewagę na rynku. Innym szybko rozwijającym się elementem w tej branży jest Robo-advisor (z ang. automatyczny doradca). Bazuje on na technologiach uczenia maszynowego oraz tradycyjnych technikach przetwarzania danych, aby tworzyć dla swoich użytkowników portfele finansowe i rozwiązania, takie jak handel, inwestycje, plany emerytalne itp. Dzięki temu, że Robo-advisor jest tani oraz łatwy w użytkowaniu stanowi on świetną opcję dla początkujących inwestorów.

Machine learning obecnie wykorzystywany jest powszechnie w analizowaniu danych np. klientów. Firmy takie jak Amazon, Facebook czy Netflix zbierają ogromne ilości danych na temat swoich klientów. Netflix według strony statista.com¹⁵ w 4 kwartale 2020 posiadał 203

¹⁴ Sanction Scanner: <https://sanctionscanner.com> [Dostęp: 22.03.2021]

¹⁵ Ilość użytkowników Netflix: <https://www.statista.com/statistics/250934/quarterly-number-of-netflix-streaming-subscribers-worldwide> [Dostęp: 22.03.2021]

miliony subskrybentów. Ponad 75% aktywności widzów opiera się na spersonalizowanych rekomendacjach. Netflix zbiera wiele informacji, aby stworzyć szczegółowe profile swoich klientów. Między innymi gromadzi dane dotyczące daty i godziny oglądania filmów, z jakich urządzeń był oglądany film oraz czy klient obejrzał film do końca. Dzięki analizowaniu tych danych Netflix może lepiej dostosowywać swoje usługi, rekomendować lepsze filmy i programy telewizyjne. Jest to niezwykle istotne, by utrzymać klienta dłużej na swojej platformie i by był on bardziej zadowolony z oferowanych usług. Przekłada się to na to, że klient chętniej wydaje więcej pieniędzy. Byłoby to oczywiście niemożliwe bez potężnego algorytmu ML, który jest w stanie automatycznie analizować wszystkie te dane i podejmować odpowiednie działanie.

Tak jak było już wspomniane wcześniej uczenie maszynowe jest szeroko stosowane przy rozpoznawaniu obrazów. Obecnie jest wiele firm, które korzystają z tej technologii. Przykładem może być Google i ich aplikacja Google Lense. Pozwala ona na rozpoznawanie tego co znajduje się na zdjęciu. Nie tylko rozpoznaje ona poszczególne obiekty, ale potrafi także zwrócić bardziej szczegółowe dane jak nazwę rośliny czy rasę psa. Aplikacja ta posiada jeszcze jedną przydatną funkcję. Potrafi rozpoznawać tekst na zdjęciu. Dzięki temu mamy możliwość tłumaczenia w czasie rzeczywistym lub po prostu kopiowania tekstu bez potrzeby ręcznego przepisywania. Google Lens nie jest jedyną taką aplikacją. Obecnie pojawia się coraz więcej programów skierowanych do użytkowników smartfonów, jak na przykład LeafSnap, który potrafi rozpoznawać około 90% znanych nam roślin, czy TapTapSee aplikacja stworzona dla ludzi niedowidzących, która mówi na głos to co widzi na zdjęciu czy na nagrany filmie. Rozpoznawanie zdjęć jest także popularne w social mediach. Facebook wykorzystuje tę technologię do automatycznego rozpoznawania i tagowania tego co znajduje się na zdjęciu. Facebook rozpoznaje twarze ludzi i oznacza je na zdjęciach. Google, Amazon i wiele innych firm udostępnia chmury obliczeniowe z algorytmami AI, które są w stanie rozpoznawać wszelkie treści znajdujące się na zdjęciach. Daje to jeszcze prostszy dostęp do potężnych zasobów i szybszego rozwijania tego typu technologii.

3. Data Mining

Dane są nieodzowną częścią budowania modeli uczenia maszynowego. Dzisiejszy cyfrowy świat składa się głównie z danych, a każdego dnia szacuje się, że produkujemy około 2.5 miliona terabajtów nowych informacji. Ponadto większość obecnie dostępnych danych w Internecie powstała w przeciągu kilku / kilkunastu ostatnich lat. Pokazuje to jak ogromną ilość informacji produkujemy poprzez samo korzystanie z Internetu, a ilość ta wzrasta i będzie ciągle wzrastać wraz ze wzrostem populacji i dostępności szybkiego Internetu. Jednakże dane te nie są ujednolicone, co sprawia, że ciężko jest wydobyć z nich cenne informacje. Proces przekształcania surowych danych w użyteczne informacje nazywa się właśnie *data mining*. Wiąże się to z analizą wzorców, anomaliami oraz korelacjami w dużych zbiorach danych przy użyciu odpowiednich algorytmów. Dzięki temu przedsiębiorstwa mogą dowiedzieć się więcej o swoich klientach, opracować bardziej efektywne strategie biznesowe, a w konsekwencji wykorzystać zasoby w bardziej optymalny sposób.

Historia

Termin "data mining" po raz pierwszy został użyty w latach 90-tych, ale jego historia jest znacznie dłuższa. W procesie data mining wykorzystuje się między innymi statystykę, komputery oraz uczenie maszynowe. Ich historia powstania jest po części historią powstania data mining. Większość źródeł przedstawia wiek XVIII jako wczesny początek analizy danych. Wtedy to angielski matematyk Thomas Bayes opracował teorię prawdopodobieństwa, którą dzisiaj znamy jako twierdzenie Bayesa. Stanowi ono fundament data miningu, ponieważ pozwala na zrozumienie złożonych rzeczywistości warunkujących się nawzajem. Przez kolejne dekady powstawały nowe algorytmy, metody jak regresja, a także koncept maszyny zdolnej do przetwarzania informacji przedstawiony przez Alana Turinga. W 1943 Warren McCulloch i Walter Pitts w pracy zatytułowanej „A logical calculus of the ideas immanent in nervous activity” (z ang. Logiczny rachunek idei immanentnych w aktywności nerwowej)¹⁶ stworzyli konceptualny model sieci neuronowej. Sieci neuronowe stanowią kolejny ważny element, który

¹⁶ Warren S. McCulloch & Walter Pitts: A logical calculus of the ideas immanent in nervous activity. 1943r

dzisiaj jest szeroko stosowany w data mining do automatycznego analizowania ogromnych ilości danych.

Kolejnym ważnym przełomem w historii jest SVM (Support vector machines), który został zaproponowany przez Bernharda E. Boser, Isabelle'a M. Guyona i Vladimira N. Vapnika. SVM jest algorytmem zaliczanym do nadzorowanego uczenia maszynowego zdolnym do przeprowadzania klasyfikacji, regresji, oraz wykrywania wartości skrajnych. Może on rozwiązywać problemy liniowe i nieliniowe i dobrze sprawdza się w wielu praktycznych problemach. W skrócie działa on tak, że tworzy linię, która rozdziela dane na klasy.

W 1989 powstał nowy termin "Knowledge Discovery in Databases" (z ang. Odkrywanie wiedzy w bazach danych) w skrócie KDD. Został on użyty przez Gregory'ego Piatetsky-Shapiro. Współtworzył on pierwsze warsztaty pod nazwą KDD. Knowledge Discovery in Databases oraz Data Mining stały się obszarem wspólnego zainteresowania naukowców zajmujących się uczeniem maszynowym, statystyką, inteligentnymi bazami danych, obliczeniami o wysokiej wydajności czy wizualizacją danych. Szybki wzrost ilości danych i informacji stworzył potrzebę wydobywania wiedzy z baz danych. Kluczowe problemy w KDD obejmują kwestie reprezentacji, złożoności wyszukiwania, wykorzystania wcześniejszej wiedzy, wnioskowania statystycznego i algorytmów do analizy ogromnych ilości danych. Warsztaty KDD szybko musiały zmienić format na konferencje z otwartą frekwencją ze względu na duże zainteresowanie. Dziś powszechnie uważa się że było to najbardziej wpływowe forum dla badań nad odkrywaniem wiedzy i analizą danych.

Bazy danych oraz tzw. hurtownie danych również odgrywają dużą rolę w data mining, dlatego też warto o nich wspomnieć. Dzięki zaawansowanym systemom bazodanowym, możliwe jest przechowywanie i przeszukiwanie petabajtów danych. W latach 80-tych ubiegłego wieku powstały hurtownie danych, które umożliwiły proces transformacji danych do systemów wspomagających podejmowanie decyzji biznesowych. Dzięki zebraniu danych z różnych źródeł, poukładaniu ich i umieszczeniu w jednym miejscu można znacznie łatwiej przeanalizować dane i wyciągnąć odpowiednie wnioski. Data mining potrzebuje duże ilości dobrej jakości danych, a hurtownia danych właśnie to zapewnia. W 2005 roku pojawiła się publikacja pt. "Competing on Analytics" odnośnik (z ang. Konkurencja na rynku analitycznym). Opisywała ona pojawienie się nowej formy „konkurencji” opartej na szerokim zastosowaniu analityki danych i

podejmowaniu decyzji w oparciu o fakty. Firmy powoli zaczęły odchodzić od tradycyjnych metod podejmowania decyzji, na rzecz zaawansowanej analizy statystycznej oraz modelowania predykcyjnego. Dziś określamy to nazwą „business intelligence” lub w skrócie BI.

Zastosowanie

Niezależnie jakiego typu firmy korzystają z data mining, głównie wykorzystują one ten proces do analizowania zebranych danych, by móc podjąć lepsze decyzje biznesowe, zmniejszać wydatki i zwiększyć dochody firmy. Poniżej zostało opisane kilka przykładów wykorzystania analizy danych.

Wiele supermarketów w Polsce i na świecie oferuje klientom bezpłatne karty lojalnościowe, które dają dostęp do obniżonych cen. Karty ułatwiają sklepom śledzenie, kto, co, kiedy i w jakiej cenie kupił. Po przeanalizowaniu danych sklepy mogą je wykorzystać do tworzenia ofert dostosowanych do nawyków zakupowych klientów oraz do podejmowania ważnych decyzji jak na przykład, kiedy wystawić produkty na sprzedaż lub kiedy sprzedać je po innej cenie.

Obecnie jednym z najbardziej popularnych platform streamingowych do muzyki jest Spotify. Pod koniec 2020 roku platforma posiadała 345 milionów aktywnych użytkowników¹⁷. Aby utrzymać tak ogromne zainteresowanie istotne jest optymalne zarządzanie platformą. W tym przypadku także z pomocą przychodzi data mining. Śledzona jest aktywność użytkowników, np. z jakiego urządzenia korzystają, jakiego rodzaju muzyki słuchają i ile czasu spędzają w aplikacji. Dodatkowo wykorzystuje się techniki przetwarzania języka naturalnego. Pomaga to Spotifyowi analizować dane związane z daną piosenką, które zawierają informacje takie jak artysta, tekst czy gatunek. Poprzez przyjęcie takich technik Spotify upewnia się, że jego subskrybenci otrzymują to, czego chcą. Platforma oferuje m.in. funkcję „Odkryj w tym tygodniu”, gdzie co tydzień pojawia się inny zestaw piosenek wybranych według preferencji użytkownika. Dzięki temu platforma stale zapewnia klientom nowe ulubione piosenki, a więc użytkownicy się nie nudzą i nie odchodzą do konkurencji.

¹⁷ Spotify, statystyka: <https://investors.spotify.com> [Dostęp: 19.04.2021r.]

Szpital gromadzą ogromną ilość różnych danych, z których większość stanowią elektroniczne rejestry medyczne. Eksploracja tych danych staje się coraz bardziej popularna i niezbędna w służbie zdrowia. Dane wytwarzane przez służbę zdrowia nie mogą być przetwarzane i analizowane tradycyjnymi metodami ze względu na ich złożoność. Data mining w tym przypadku daje możliwość, aby m.in. leczyć pacjentów przy pomocy medycyny predykcyjnej, oceniać skuteczność leczenia, zarządzać służbą zdrowia czy wspomagać diagnozy i doradzać lekarzom w podejmowaniu decyzji klinicznych. Systemy wykorzystujące data mining mogą również podpowiadać, które leki i procedury medyczne okazują się nieskuteczne i warto je zastąpić czymś lepszym. Producenci leków mogą wykorzystywać aplikacje eksploracji danych do dostosowywania i ulepszania swoich leków lub rozwoju sprzętu, przeprowadzania testów i sprawdzania, jakie produkty i usługi będą najbardziej pożądane w przyszłości. Analiza danych może również wspomóc wykrywanie wszelkiego rodzaju oszustw, nadużyć oraz marnotrawstwa w sektorze opieki zdrowotnej. Zdarzają się sytuacje, gdzie ludzie chcą uzyskać receptę na leki bez wyraźnej potrzeby. Takie sytuacje mogą zdarzyć się, gdy dany lek jest w pewien sposób uzależniający. Posiadając wystarczającą ilość informacji o pacjencie, oprogramowanie może powiedzieć lekarzowi, czy dana osoba naprawdę potrzebuje leku, czy też próbuje zdobyć go nielegalnie. Usługi laboratoryjne w USA w 2017 roku wygenerowały 87.3 mld dolarów przychodu co stanowi pewien potencjał do nadużyć i oszustw¹⁸. Szacuje się, że straty w tym sektorze mogą przekraczać dziesiątki milionów dolarów amerykańskich. Dlatego też powstają takie inicjatywy jak HFPP – Healthcare Fraud Prevention Partnership (z ang. Partnerstwo na rzecz zapobiegania oszustwom w opiece zdrowotnej). Jest dobrowolne publiczno-prywatne partnerstwo, które pomaga wykrywać i zapobiegać oszustwom w opiece zdrowotnej poprzez wymianę danych i informacji oraz analizę danych. Aktualnie inicjatywa ta posiada ponad 200 partnerów takich jak rząd federalny, agencje stanowe, organy ścigania, prywatne plany ubezpieczeń zdrowotnych itp., które razem współpracują, by zapobiegać oszustwom.

¹⁸ Analiza usług laboratoriów medycznych pod kątem oszustw w USA
<https://www.cms.gov/files/document/download-clinical-laboratory-services-white-paper.pdf> [Dostęp: 19.09.2021]

Proces analizy danych

Proces eksploracji danych jest wieloetapowy. Aby wyciągnąć użyteczne informacje z surowych danych należy przejść przez cały proces data mining, co nie zawsze może być proste. Chociaż wiele firm posiada dużo danych, nie wszystkie z nich są użyteczne lub niezbędne do realizacji konkretnego celu projektu. Największym problemem w analizie danych nie jest brak dostępnych danych, ale sam długi i skomplikowany proces analizy. Po pierwsze, należy ustalić cele projektu, czyli czego chcemy się dowiedzieć analizując dane. Następnie dane są zbierane i przechowywane w hurtowni danych. Kolejnym krokiem, jest czyszczenie i procesowanie danych. Analitycy identyfikują dane, usuwają duplikaty, lub niespójne dane, które mogłyby zniekształcić analizę. Na koniec interpretuje się uzyskane dane, tworzy prezentacje i wykresy, które odpowiadają na pytania postawione na samym początku. Aby lepiej zrozumieć cały proces poniżej zostało dokładnie opisane pięć głównych kroków data mining.

Ustalenie celów

Analizowanie danych jest długim i żmudnym procesem, który musi mieć jakiś ważny powód. Ustalając jakie informacje chcemy uzyskać analizując dane tworzony jest jasny plan działania. Analitycy, informatycy i pozostałe osoby związane z procesem analizy danych z góry wiedzą jak procesować dane, które dane są potrzebne, a które można usunąć. Brak sprecyzowania jasnych celów może powodować poważne problemy w całym procesie eksploracji danych. W tym kroku firmy najczęściej stawiają ważne dla nich pytania na przykład – „Jak możemy zwiększyć sprzedaż naszych głównych produktów przy wykorzystaniu obecnych zasobów?” lub „Jak zniżki wpływają na zachowania zakupowe naszych klientów”. Postawienie odpowiednich pytań (celów) jest bardzo istotne. Od tego zależy czy na końcu całego procesu firma uzyska użyteczne informacje, które mogą pomóc jej w dokonaniu lepszych decyzji, zwiększyć dochody lub umocnić swoją pozycję na rynku.

Zbieranie danych

Po zdefiniowaniu jasnych celów należy zacząć zbierać odpowiednie dane. Informacje te mogą pochodzić z różnych źródeł: wewnętrznych baz danych firmy, marketingu, systemów CRM oraz ERP. Są to źródła danych, które zawierają informacje na temat klientów i ich

interakcji z firmą, dane dotyczące sprzedaży, finansów oraz wiele innych. Do analizy stosuję się również dane pochodzące ze źródeł zewnętrznych takie jak media społecznościowe, by dowiedzieć się więcej na temat zainteresowania ludzi marką i produktami firmy. Ze źródeł zewnętrznych może również uzyskać informacje na temat trendów ekonomicznych, przyszłości obecnych oraz przyszłych zainteresowań społeczeństwa. Danych do zebrania jest zawsze pod dostatkiem, najważniejsze w tym punkcie jest to, by wiedzieć skąd je uzyskać oraz jakie informacje będą przydatne.

Czyszczenie danych

Zebrane dane nigdy nie będą wolne od wad dlatego też w tym kroku są one poddawane procesowi czyszczenia oraz normalizacji. Często zdarzają się duplikaty, niekompletne dane lub uszkodzone wartości, które mogłyby spowodować niepoprawne wyniki analizy danych. Dlatego jest to bardzo ważny krok i zajmuje najwięcej czasu. Wady danych mogą mieć wiele przyczyn, na przykład często użytkownicy nie chcą podawać prawdziwych informacji takich jak data urodzenia przy rejestracji lub pozostawiają niektóre pola puste. Ponieważ jest kilka rodzajów wad danych zostały stworzone osobne metody to radzenia sobie z nimi¹⁹.

- Puste wartości – tę wadę można rozwiązać na kilka sposobów. Najprostszą metodą jest usunięcie całego rekordu, gdy jest on nie kompletny. Jednak jest to nie praktyczne, jeżeli większość rekordów zawiera brakujące wpisy. Można wtedy takie brakujące informacje oszacować i uzupełnić jeżeli nie są one bardzo istotne w przeciwnym wypadku mogą znacząco wpłynąć na finalny wynik analizy. Najbardziej pożądanym sposobem jest zaprojektowanie analizy danych w taki sposób, aby mogła ona pracować z brakującymi wartościami.
- Niepoprawne wartości – w tym przypadku problem polega na wykryciu i skorygowaniu niepoprawnych wartości. Problem ten może się często pojawiać, gdy informacje pochodzą z różnych źródeł i są zapisane w różnych formatach. Na przykład pole z imieniem może zawierać samo imię, imię i nazwiskiem lub tylko inicjały. Mogą też pojawiać się sprzeczności w danych na przykład, gdy w polu *kraj* widnieje *Polska* to pole z nazwą miasta nie może być *Chicago*. Aktualnie istnieje wiele narzędzi, które pomagają

¹⁹ Charu C. Aggarwal, Data mining – The textbook, Springer, 2015 rok

wykrywać nieprawidłowe wpisy, chociaż nadal często wykryte błędy trzeba korygować ręcznie.

- Skalowanie i normalizacja – jest to przekształcanie oryginalnych danych numerycznych takich jak wiek czy zarobki do określonego przedziału. Najczęściej są to przedziały rzędu $[3, -3]$ lub $[0, 1]$. Przekształcanie to jest konieczne, aby móc wykorzystać te dane w algorytmach uczenia maszynowego. Najczęściej stosuje się funkcje min-max, daną następującym wzorem:

$$y = \frac{x - \min}{\max - \min},$$

Równanie 1 Funkcja min-max

gdzie x określa pojedynczą wartość, \min jest wartością najmniejszą, a \max największą w danym zbiorze danych. Po użyciu tej funkcji dane zostaną znormalizowane do przedziału $[0, 1]$. W przypadku gdy wartość minimalna lub maksymalna nie jest znana, wykorzystuje się funkcję Zero-Mean określoną formułą

$$y = \frac{x - \mu}{\sigma},$$

Równanie 2 Funkcja Zero-Mean

gdzie μ jest wartością średnią elementów całego zbioru, a σ oznacza odchylenie standardowe

Analizowanie danych

Oczyszczone i znormalizowane dane są już gotowe do przeprowadzenia analizy. W tym kroku wyciąga się wszelkie niezbędne informacje na temat zebranych danych. Odkrywa się ukryte wzorce i zależności. Istnieje wiele programów i technologii, które pomagają w wizualizacji danych, tworzeniu raportów czy predykcji. Najbardziej powszechne są narzędzia Business Intelligence jak np.: Datapine czy Tableau. Są to potężne narzędzia, które potrafią operować na ogromnej ilości danych. Jednak do analizy można wykorzystać prostsze narzędzia takie jak język R lub Python, które również świetnie nadają się do tych celów. Zebrane

informacje pod postacią raportów, tabel i wykresów mogą zostać przekazane dalej odpowiednim jednostkom, by na ich podstawie podejmowano odpowiednie decyzje biznesowe.

Interpretacja wyników

Ostatnim krokiem jest interpretacja wyników analizy danych. W tym kroku uzyskuje się odpowiedzi na pytania postawione na samym początku. Czytając uzyskane raporty oraz wykresy firmy dowiadują się cennych informacji, na podstawie których podejmują ważne decyzje dotyczące produkcji, sprzedaży czy prowadzenia marketingu, by zwiększyć przychody w jak najbardziej efektywny sposób. Dane te są niezwykle cenne i z punktu widzenia firm warte przeprowadzenia całego skomplikowanego i żmudnego procesu. Tak jak było opisane w poprzednim podrozdziale na temat zastosowań, wiele firm takich jak Spotify, Netflix supermarkety czy nawet szpitale bardzo dużo zyskują analizując zebrane dane. Poprawna interpretacja wyników nie jest prosta. Osoba, która tym się zajmuje musi znać różnice między przyczynowością, przypadkami i korelacją, a także mieć na uwadze skąd pochodzą dane i to, że mogą one nadal zawierać błędy. Dlatego, też powstały dwie główne metody interpretacji: jakościowa i ilościowa:

- Interpretacja jakościowa – w tej metodzie dane są opisywane za pomocą tekstu. Inaczej są to dane kategoryczne, które można podzielić na podstawie cech fizycznych, pochodzenia lub czegośkolwiek, co nie jest opisane liczbowo. Czyli mogą być to dane takie jak kolor włosów, typ samochodu, rasa czy pochodzenie etniczne,
- Interpretacja ilościowa – jest ona ściśle związana z danymi liczbowymi. Odnosi się do zestawu procesów, w których analizowane są dane liczbowe. Najczęściej wiąże się to z wykorzystaniem modelowania statystycznego, takiego jak odchylenie standardowe, mediana, regresja czy analiza predykcyjna.

Decyzje podejmowane na podstawie zebranych danych dają znacznie lepszy wynik. Według badań firm, które analizują swoje dane do podejmowania decyzji są średnio o 5% bardziej wydajne i zyskowe od konkurentów²⁰. Analiza i interpretacja danych są więc kluczowe dla opracowania solidnych wniosków i podejmowania bardziej świadomych decyzji, które przynoszą pożądany efekt.

²⁰ A. McAfee, E. Brynjolfsson. Big Data: The Management Revolution, 2012 r

4. Język programowania

Praca magisterska powstała między innymi przy pomocy języka Python oraz kilku narzędzi z nim związanych takich jak PyCharm, i Jupiter Notebook. Python jest językiem programowania ogólnego przeznaczenia, co oznacza, że może być wykorzystywany do wielu rodzajów programowania. Obejmuje to między innymi strony internetowe, oprogramowania desktopowe, backend, data science czy tworzenie skryptów systemowych. Wśród popularnych języków jak Java czy C++ wyróżnia go to, że jest prostszy. Dzięki temu nie tylko można się go szybko nauczyć, ale także pisanie samego kodu zajmuje mniej czasu. Dzieje się tak między innymi dlatego, że Python wykorzystuje dynamiczne typowanie oraz dynamiczne wiązanie. W tym języku nie określa się typów zmiennej. Są one dynamicznie typowane w momencie uruchomienia programu. Jedna zmienna w trakcie działania programu może przyjmować różne typy i nie będzie powodować to, jakichkolwiek błędów. Jest to duża zaleta Pythona, gdyż upraszcza to kod jednak kosztem czasu kompilacji. Dynamiczne typowanie wymaga od kompilatora dodatkowej pracy, ponieważ to nie programista, a kompilator musi przypisać typ do zmiennej, co wydłuża jego czas pracy. Jednak w większości przypadków, znacznie ważniejsze jest szybkie napisanie kodu od bardzo szybkiego działania aplikacji. Python posiada wiele użytecznych bibliotek dostępnych za darmo, które można z łatwością dodać do projektu. Jest to możliwe dzięki systemowi zarządzania pakietami o nazwie pip. Można go wywołać z linii komend i prosto pobrać brakujące biblioteki, a następnie przy użyciu słowa kluczowego *import* dodać je do projektu. Według indeksu TIOBE²¹, który jest wskaźnikiem popularności języków programowania, język Python aktualnie znajduje się na drugim miejscu tuż za językiem C. Python wyprzedził również Javę, która od wielu lat znajduje się w czołówce najbardziej popularnych języków programowania.

Historia

Python został opracowany przez Guido van Rossuma, holenderskiego programistę pod koniec lat 80. Van Rossum pracował wtedy nad rozwojem języka programowania ABC. Jak sam wspomina język ABC miał bardzo duży wpływ na rozwój Pythona. Van Rossum w wolnym

²¹ Indeks TIOBE - <https://www.tiobe.com/tiobe-index> [Dostęp: 24.08.2021]

czasie po koniec 1989 roku zaczął pisać prosty interpreter dla własnego języka skryptowego. Mając już spore doświadczenie przy tworzeniu języka programowania ABC van Rossum chciał stworzyć język, który będzie miał wszystkie zalety ABC, pomijając przy tym jego wady. Niedługo później, bo już w lutym 1991 roku Guido van Rossum opublikował pierwszą wersję Pythona. Posiadała ona wiele funkcjonalności takich jak obsługa wyjątków czy obiektowość. Dodatkowo oprócz typów prostych język posiadał również listę czy słownik, gdzie można przechowywać dane w postaci klucz i wartość. Pierwsza pełna wersja z numerem 1.0 została wydana w 1994 roku. Posiadała ona dodatkowo takie narzędzia jak lambda, map czy filter, które można było znaleźć na przykład w języku Lisp. Van Rossum założył inicjatywę Computer Programming for Everybody (z ang. Programowanie komputerowe dla każdego). Miała ona na celu sprawić, by programowanie było łatwiej dostępne i prostsze dla ludzi. Do tego celu Python był doskonałym narzędziem, gdyż jest on prosty w nauce i posiada prostą składnię. Przez kolejne lata były wydawane kolejne wersje. W 2000 roku została wydana wersja 2.0 z wieloma nowymi funkcjonalnościami między innymi: garbage collector, unicode string oraz dodatkowymi metodami dla typu string. Osiem lat później została wydana wersja 3.0. Język ten jest wspierany i ciągle rozwijany. Średnio co ponad rok wydawana jest kolejna wersja. Na ten moment najnowsza wersja Pythona to 3.10. Przyszłość Pythona wygląda bardzo dobrze. Dzięki swojej prostocie i uniwersalności ciągle zyskuje na popularności, będąc aktualnie równie popularny co język Java.

IDE

PyCharm jest zintegrowanym środowiskiem służącym do pisania programów w języku Python. Został on stworzony przez bardzo znaną czeską firmę JetBrains, która posiada wiele narzędzi do programowania jak na przykład IntelliJ – jeden z najbardziej popularnych IDE dla języka Java. PyCharm jest bardzo rozbudowany, dostarcza wiele niezbędnych narzędzi dla programistów, między innymi łączenie z bazą danych, integracja z innymi narzędziami jak github czy AWS, automatyczna refaktoryzacja kodu, podświetlanie błędów oraz wiele innych. Pozwala on na łatwe debugowanie poprzez wstawianie tzw. breakpointów. Dzięki temu programista może dokładniej przeanalizować działanie kodu oraz wyłapać co dokładnie powoduje błędy. PyCharm upraszcza również pisanie testów. Prostym skrótem klawiszowym *ctrl* + *shift* + *t* można automatycznie stworzyć osobny plik z przygotowaną klasą, w której można

pisać testy. IDE jest bardzo wszechstronne i jest dostępne w kilku wersjach. Z PyCharm Community można korzystać w pełni za darmo dzięki licencji Apache License 2.0. Wersja Community posiada mniej funkcjonalności, jednak wciąż jest ona wystarczająca do nauki i pisania aplikacji. Program, który jest częścią tej pracy został napisany przy użyciu tej wersji. Licencja Apache pozwala na dowolne wykorzystanie tego oprogramowania również w projektach komercyjnych. PyCharm jest dostępny również w wersji Ultimate, która jest płatna w formie subskrypcji. Posiada mnóstwo dodatkowych funkcjonalności, których nie ma w wersji Community i również można ją wykorzystywać w komercyjnych projektach. JetBrains udostępnia swoje narzędzia również w wersji edukacyjnej. Posiada ona wszystko to co ma pełna i płatna wersja Ultimate. Jest ona dostępna dla uczniów i nauczycieli i służy jedynie do nauki. Licencja wersji edukacyjnej nie pozwala na wykorzystanie oprogramowania w celach komercyjnych.

Jupyter Notebook

Jupyter Notebook to kolejne istotne narzędzie, wykorzystane przy tworzeniu aplikacji. Jest to open source'owa aplikacja webowa, która umożliwia pisanie i udostępnianie dokumentów łączących kod, wyniki oraz wizualizacje. Jest wykorzystywany przede wszystkim w badaniach, analizie i wizualizacji danych, uczeniu maszynowym czy modelowaniu numerycznym. Jupyter pozwala na korzystanie nie tylko z języka Python, ale również innych języków jak np.: Julia, Python i R. Stąd też powstała nazwa Jupyter – z połączenia nazw języków, które obsługuje. Należy zaznaczyć, że obecnie Jupyter obsługuje około 100 języków wliczając w to MatLab, Processing czy język Scala. Jupyter to aplikacja typu serwer-klient można więc ją uruchomić nie tylko na swoim komputerze, ale również na serwerze i udostępnić ją poprzez Internet.

5. Analiza danych

Analiza danych rozpoczyna część praktyczną tej pracy. Analiza danych lub inaczej data mining jest procesem przekształcania surowych danych w użyteczne informacje. Teoria tego procesu została przedstawiona w rozdziale Data Mining. W tym rozdziale proces ten zostanie użyty w praktyce do przeanalizowania zestawu danych pacjentów chorych na serce. Jak zostało powiedziane wcześniej data mining można podzielić na 5 kroków: ustalenie celów, zebranie danych, czyszczenie danych, analiza oraz interpretacja.

Zacznijmy więc od pierwszego kroku. Głównym celem tej analizy jest odnalezienie optymalnego modelu predykcji oraz ustalenie, które parametry mają największy wpływ na chorobę serca. Te informacje będą dla nas kluczowe przy tworzeniu aplikacji desktopowej do predykcji czy dany pacjent jest chory.

Kolejny krok to zbieranie danych. Zastaw danych, który został wybrany to „Cleveland database” ze strony internetowej kaggle.com. Zawiera ona dane 303 pacjentów. Dla każdego pacjenta opisano 14 różnych parametrów np.: wiek, rodzaj bólu klatki piersiowej, poziom cholesterolu etc. W tej bazie jest również kolumna, która wskazuje czy dany pacjent ma chore serce, czy nie co jest istotne do nauczania modelu predykcji.

Trzecim punktem jest czyszczenie danych. Rozważany zestaw danych, został już wstępnie oczyszczony, więc ten punkt może zostać pominięty.

Następnym krokiem jest analiza danych. Do tego celu wykorzystano narzędzie Jupyter Notebook oraz język Python. Jest to niewielkie narzędzie, które pozwala na pisanie skryptów w języku Python w taki sposób, aby zachowana była struktura dokumentu. Jupyter Notebook wspiera również język markdown, dzięki któremu można czytelnie opisywać poszczególne bloki kodu. Otrzymane pliki HTML oraz ipynb (Jupyter notebook) są również dołączone do tej pracy. Wszystkie poniższe diagramy i wykresy pochodzą z analizy danych, która została wykonana w narzędziu Jupyter Notebook. Analizę danych najlepiej rozpocząć od sprawdzenia ogólnej struktury danych oraz określenia, co oznaczają poszczególne parametry, by lepiej zrozumieć analizowane dane.

Prezentacja przykładowych danych

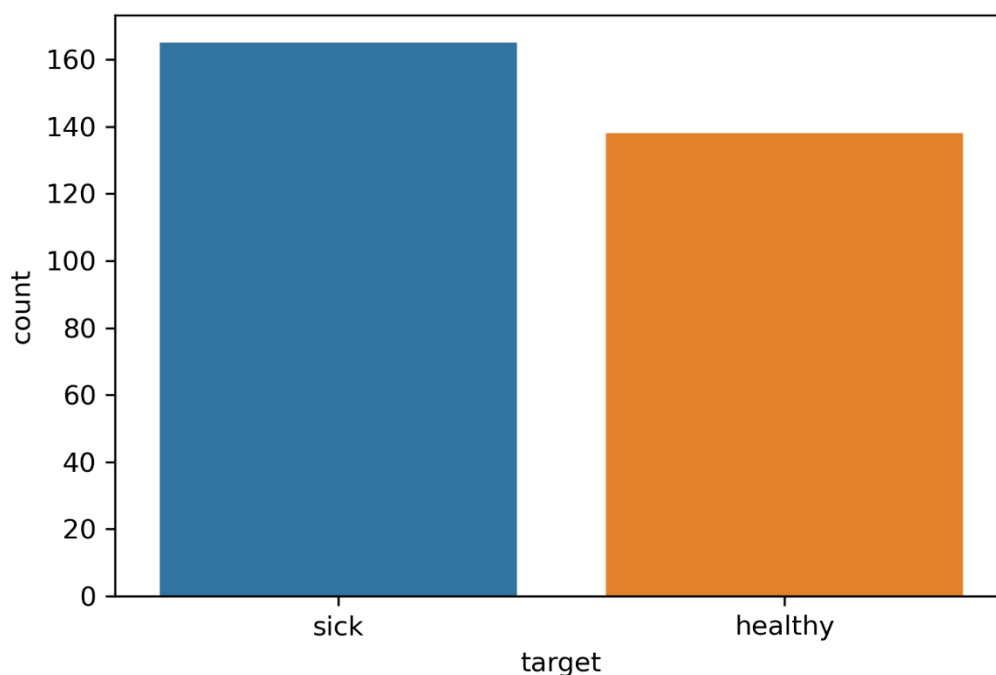
Tabela 1 Struktura bazy danych pacjentów

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
56	1	1	120	236	0	1	178	0	0.8	2	0	2	1

Powyżej został przedstawiony schemat wykorzystanej bazy danych pacjentów. Zestaw danych posiada łącznie 303 rekordy. Każdy rekord posiada 14 kolumn określające różne parametry związane z pacjentem. Opiszemy kilka z nich:

- Age – wiek pacjenta w latach,
- Sex – płeć (1 - mężczyzna, 0 - kobieta),
- cp – rodzaj bólu w klatce piersiowej (0 - typical angina, 1 - atypical angina, 2 - non-anginal pain, 3 - asymptomatic),
- trestbps – spoczynkowe ciśnienie krwi (mm Hg przy przyjęciu do szpitala),
- chol – poziomu cholesterolu mg/dl,
- fbs – poziom cukru we krwi osoby na czczo (1 – poziom większy niż 120 mg/dl, 0 – poniżej 120 mg/dl),
- exang – czy wysiłek fizyczny prowadzi do dławicy piersiowej (1 – tak, 0 - nie),
- thal – zaburzenie krwi zwane talasemią (1 – w normie, 2 - stała wada, 3 - uleczalna wada),
- target – czy wykryto chorobę serca (1 – tak, 0 - nie).

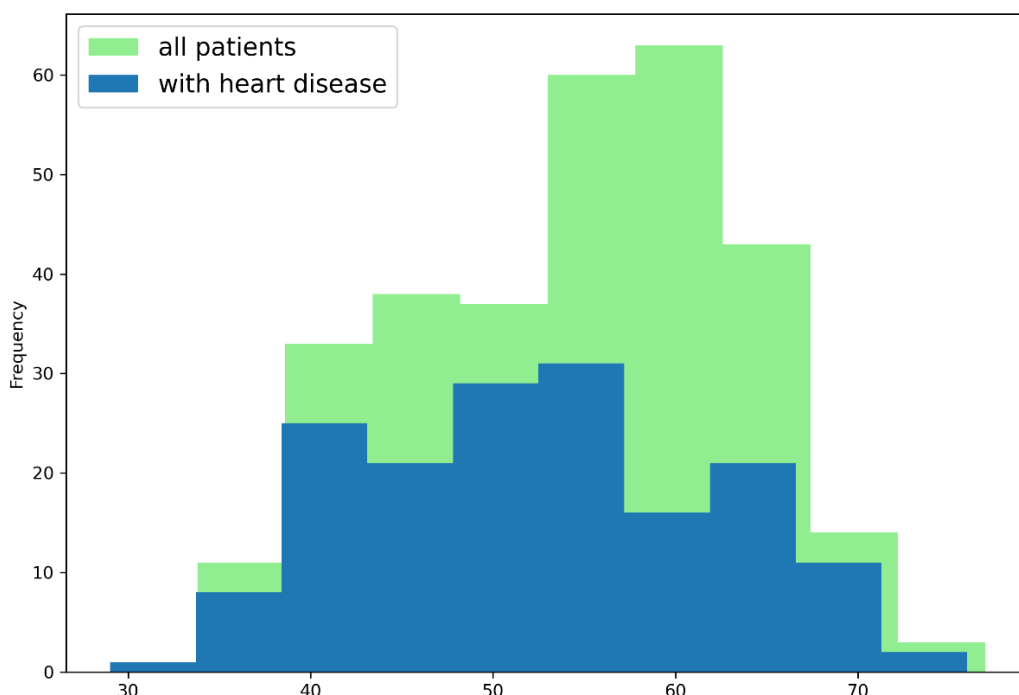
Analizę danych rozpoczęto od sprawdzenia czy dane są zbalansowane tzn. czy liczba chorych i zdrowych osób jest mniej więcej podobna. Odpowiedni balans danych jest istotny do otrzymania poprawnych wyników z analizy.



Rysunek 5.1 Balans bazy danych

Jak widać na powyższym diagramie słupkowym liczba osób chorych i zdrowych jest podobna, przy czym osób chorych(niebieski słupek) jest nieznacznie więcej niż zdrowych (pomarańczowy słupek). Dokładnie mamy 165 osób chorych i 138 zdrowych. Różnica jest nieznaczna, możemy więc stwierdzić, że zestaw danych jest odpowiednio zbalansowany.

Następny istotny czynnik, który warto wziąć pod uwagę to wiek pacjentów. Na serce najczęściej skarżą się osoby starsze, jednak coraz więcej słyszy się o młodszych osobach dotkniętych problemami z sercem. Możemy więc przeanalizować jak rozkłada się wiek w zestawie danych.



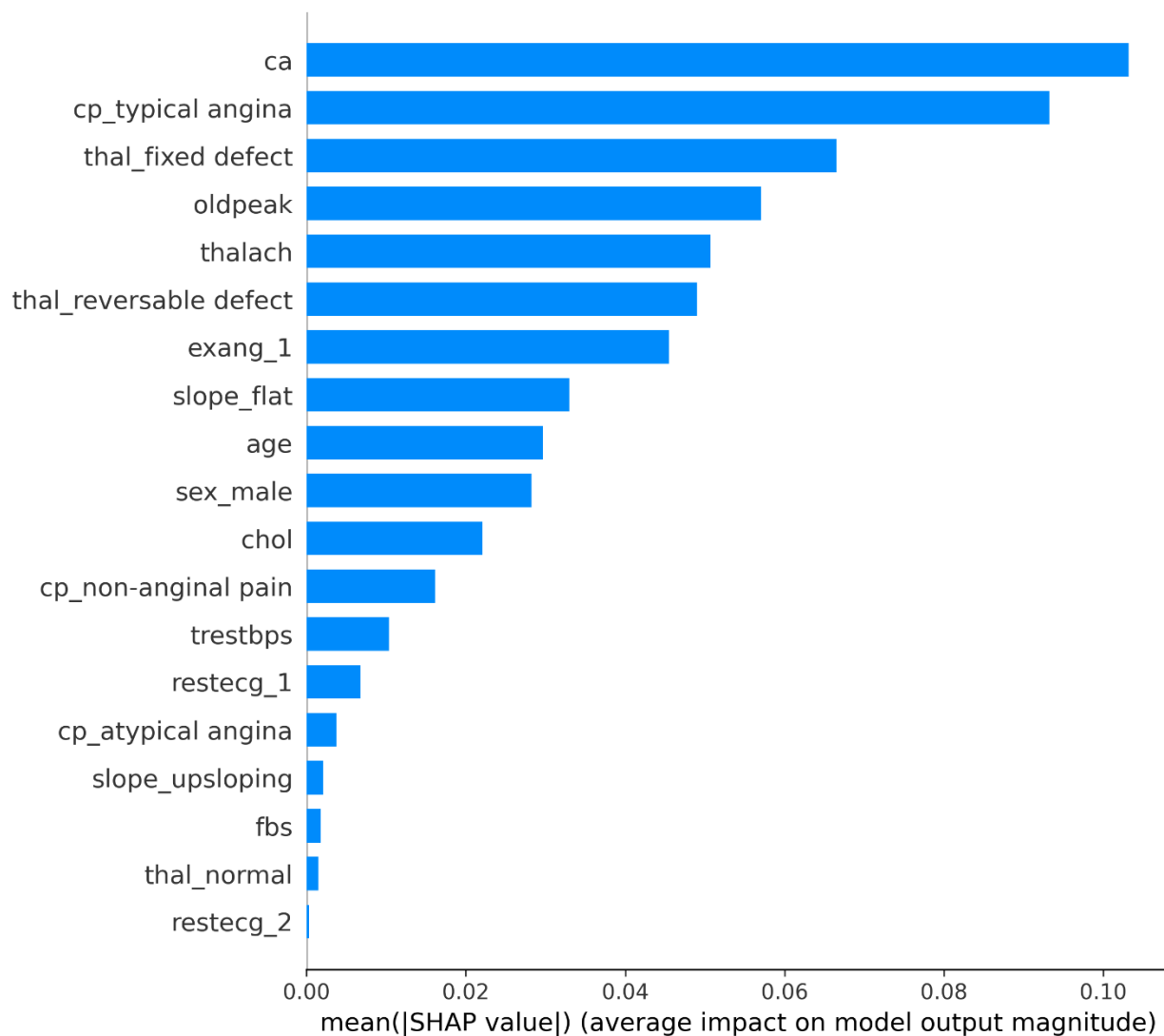
Rysunek 5.2 Histogram wieku pacjentów

Histogram wskazuje, że najwięcej pacjentów jest w przedziale wiekowym pomiędzy 55 a 65 lat. Natomiast w przedziale 40 - 65 lat jest najwięcej pacjentów, u których wykryto chorobę serca. Bardzo interesująca jest mniejsza liczba osób chorych w wieku powyżej 65 lat. Jednak jest to tylko przykładowy niewielki zbiór danych osób z pewnego obszaru biorących udział w badaniu, więc trudno jest znaleźć przyczynę i wyciągać ogólne wnioski dla populacji. W zestawie danych przeważającą grupą są mężczyźni. Stanowią oni około 68% wszystkich pacjentów. Mężczyzn jest więcej zarówno w grupie pacjentów chorych, jak i zdrowych.

Czynniki wpływające na choroby serca

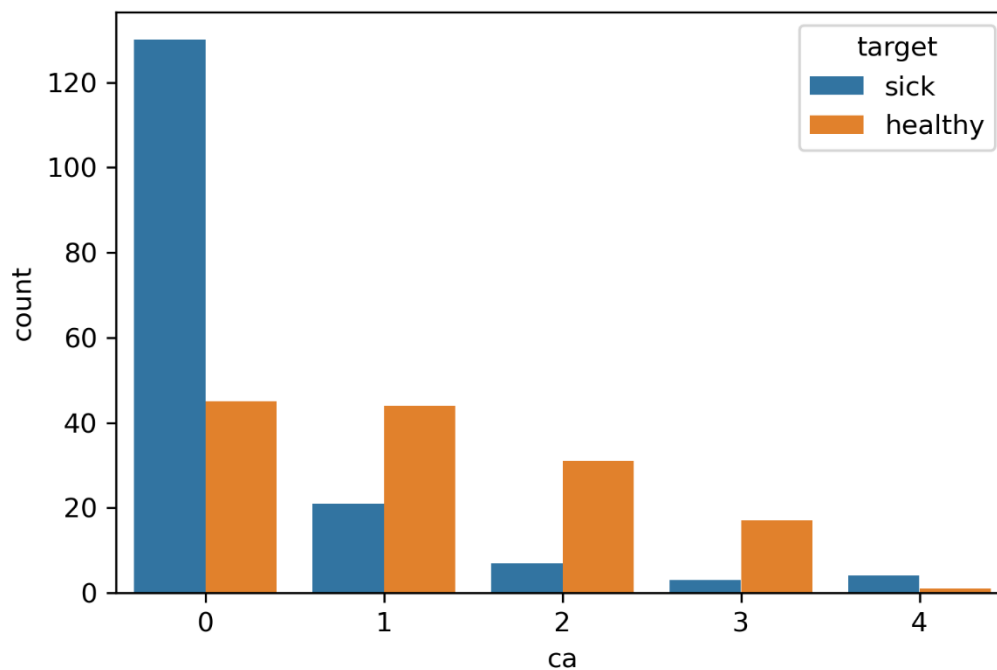
Po przeanalizowaniu kilku kolejnych parametrów podjęto próbę określenia, które czynniki mają największy wpływ na choroby serca. Do tego celu wykorzystany został algorytm losowego lasu decyzyjnego oraz biblioteka SHAP²², która pomaga opisać dane wyjściowe dowolnego modelu uczenia maszynowego. Natomiast do wizualizacji wykresu wykorzystana została metoda `summary_plot` pochodząca również z biblioteki SHAP. Po dodatkowym przetworzeniu danych uzyskano poniższy wykres.

²² Biblioteka SHAP: <https://shap.readthedocs.io/en/latest/index.html> [Dostęp: 18.09.2021]



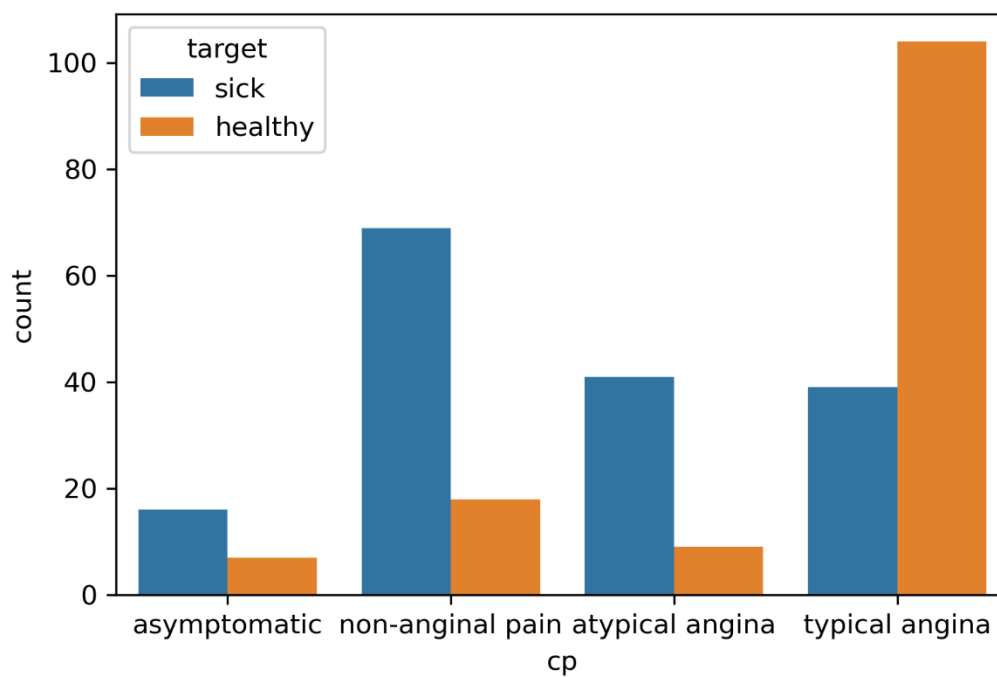
Rysunek 5.3 Ranking parametrów

Widać, że na pierwszym miejscu plasuje się parametr *ca*. Po sprawdzeniu tej wartości można zauważyć, że ponad 70% pacjentów z *ca* równym 0 ma chore serce, dlatego też parametr ten ma bardzo duży wpływ na występowanie chorób serca. Poniższy wykres przedstawia jak parametr *ca* wpływa na wynik pacjenta. Bardzo dobrze widać, że przy wartości *ca* równej 0 niebieski słupek wskazujący liczbę osób chorych jest największy. Natomiast im większa jego wartość tym mniejsza liczba osób chorych. Przy wartości parametru *ca* równej 4 proporcja znowu się odwraca. Jednak liczba pacjentów z tą wartością jest bardzo mała, dlatego ciężko jest jednoznacznie stwierdzić co na to wpływa.



Rysunek 5.4 Wpływ parametru CA na wynik

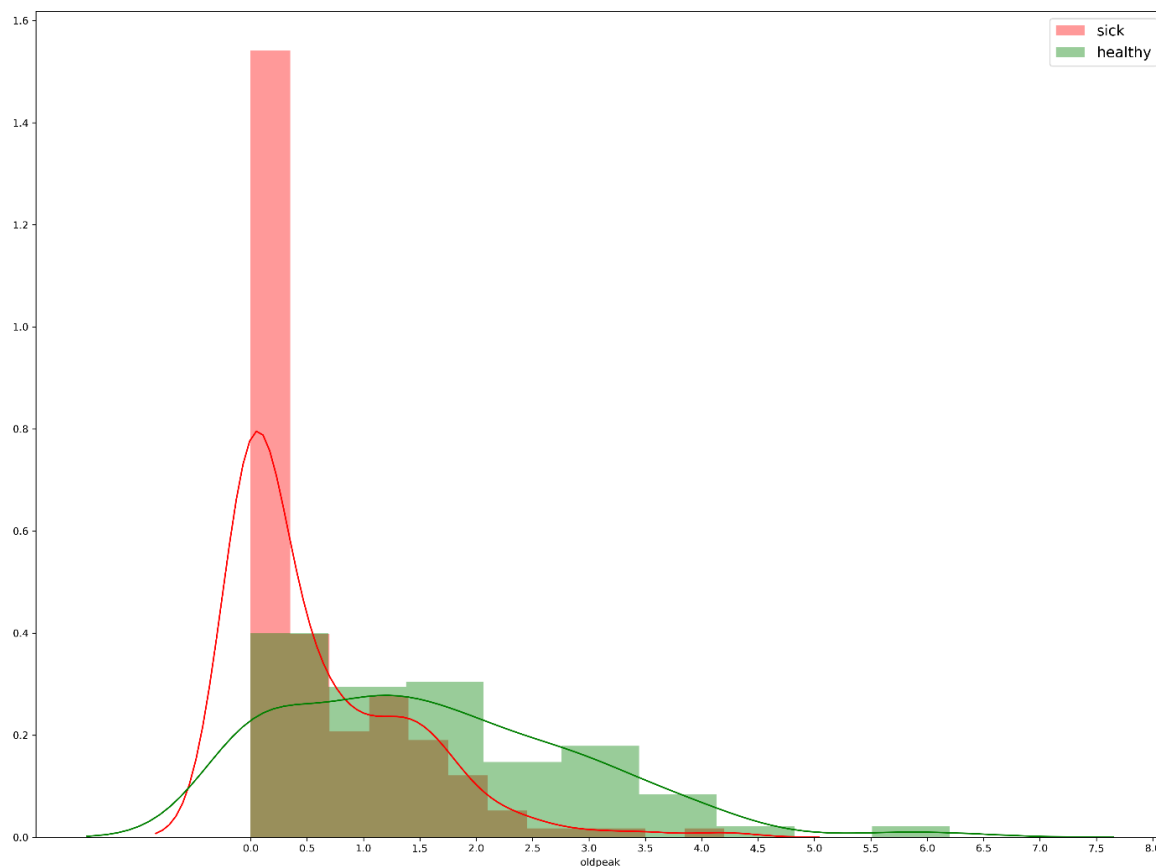
Następnym badanym czynnikiem jest *cp* lub inaczej *chest pain* (z ang. ból ból w klatce piersiowej)



Rysunek 5.5 Wpływ bólu w klatce piersiowej na wynik

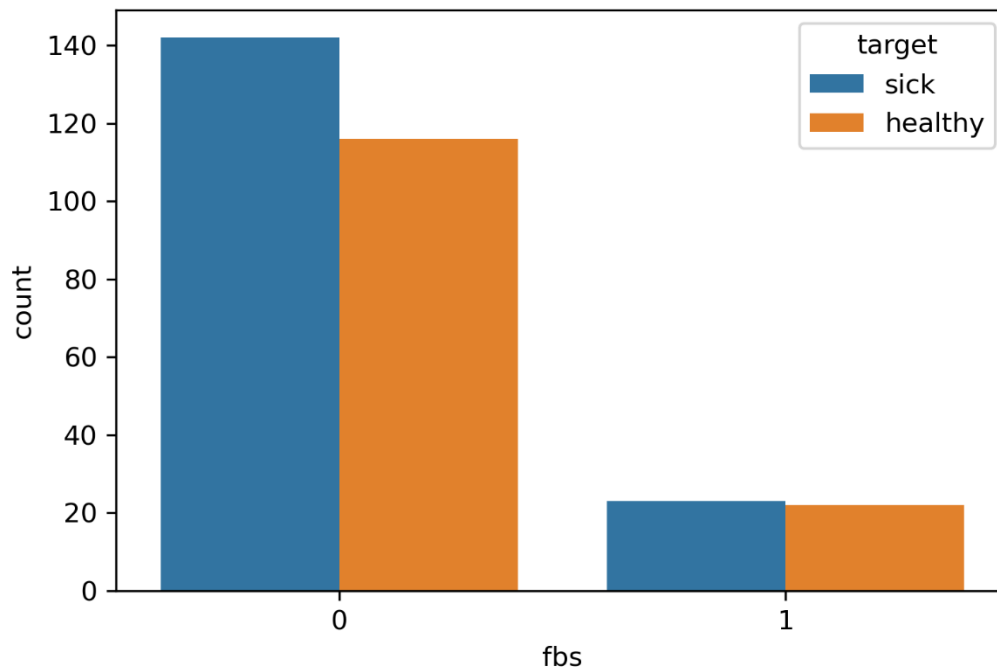
Po dokładnym sprawdzeniu tego parametru można zauważyć, że znaczna większość osób z *typical angina* (dławica piersiowa) nie ma choroby serca. Dlatego też pojawiła się ona tutaj na drugim miejscu. Brak *typical angina* (posiadanie innego rodzaju bólu w klatce piersiowej) ma znaczący wpływ na końcowy wynik pacjenta. W grupie osób z bólem typu *asymptomatic* (bezobjawowy), *non-anginal* (niedławicowy) oraz *atypical* (atypowa dusznica) przeważająca liczba pacjentów ma wykrytą chorobę serca, tak jak można to zaobserwować na powyższym wykresie.

Na trzecim miejscu jest parametr *thal* o wartości *fixed defect*. A na kolejnym miejscu jest wartość *oldpeak*, aby dowiedzieć się dokładniej jakie wartości tego pola zwiększają prawdopodobieństwo posiadania choroby serca należy spojrzeć na poniższy wykres. Pokazuje on, że osoby chore mają niższe wartości pola *oldpeak*., natomiast pacjenci z wartościami bliskimi zero najczęściej chorują na serce. Dla wartości 2.5 i więcej liczba chorych spada niemal do zera.



Rysunek 5.6 Wpływ parametru *oldpeak* na wynik

Najmniej istotne okazały się wartości takie jak *restecg* (spoczynkowy pomiar elektrokardiograficzny) o wartości 2 czy parametr *fbs*, który przedstawia poziom cukru na czczo. Na wykresie przedstawionym poniżej widać, że liczba chorych i zdrowych jest bardzo podobna bez względu na to czy parametr *fbs* jest równy 0 czy 1. Warto tutaj wspomnieć, że te wyniki tyczą się jedynie badanego zbioru danych i mogą nie przekładać się w pełni na rzeczywistość.



Rysunek 5.7 Wpływ parametru *fbs* na wynik

Modele predykcji

Po przeanalizowaniu danych i odnalezieniu parametrów mających największy wpływ na występowanie chorób serca można przejść do kolejnego istotnego aspektu, czyli sprawdzenia, które modele predykcji działają najlepiej na tym zbiorze danych. Aktualnie istnieje wiele rodzajów modeli predykcji. Część z nich jak np. regresja liniowa może być stosowana tylko do danych liniowych, inne jak regresja logistyczna do danych logistycznych. Są także modele, jak na przykład drzewa decyzyjne, które są w stanie pracować bez względu na typ danych,. Bardziej zaawansowanym modelem predykcji jest sieć neuronowa. Jest to model połączony z wielu wierzchołków, które mogą być porównane do neuronów w ludzkim mózgu. Sieci neuronowe

ucząc się potrafią rozpoznawać ukryte wzorce, czy relacje między danymi. Sieć może grupować dane oraz je klasyfikować. Zdawało by się, że sieci neuronowe świetnie się nadają do kategoryzowania pacjentów. Jednak sieci neuronowe, by działać poprawnie potrzebują bardzo dużej ilości danych treningowych. Dla mniejszych zestawów danych mniej złożone modele radzą sobie znacznie lepiej. Dodatkowo sieci neuronowe nie dostarczają szczegółowych informacji na temat w jaki sposób osiągnięty został wynik. Wykorzystane zostały jednak do testów porównawczych wraz z innymi popularnymi modelami predykcji. Wybrano taki oto zestaw metod użytych na potrzeby niniejszej pracy:

- Regresja liniowa,
- Regresja logistyczna,
- Naiwny klasyfikator bayesowski,
- K najbliższych sąsiadów,
- Losowy las decyzyjny,
- Sieć neuronowa.

Dane zostały podzielone w sposób losowy na grupę treningową (80%) i testową (20%). Podział ten często jest przyjmowany przy tworzeniu tego typu modeli. Przy tworzeniu losowego lasu decyzyjnego wykorzystane zostały dodatkowe parametry: $max_depth = 5$, $n_estimator = 4$ oraz $random\ state = 20$. Otrzymano je testując różne wartości w poszukiwaniu najlepszej kombinacji parametrów. Do stworzenia sieci neuronowej wykorzystano natomiast narzędzie GridSearchCV, które pomaga dobrać parametry do uzyskania lepszych rezultatów modeli. Sieć neuronowa została zbudowana z trzech warstw ukrytych, gdzie każda posiada 50 neuronów. Po przetestowaniu wszystkich modeli osiągnięto następujące wyniki:

Tabela 2 Wyniki poszczególnych modeli

Typ wyniku	Regresja liniowa	Regresja logistyczna	GaussianNB	KNN	Losowy las decyzyjny	Sieci neuronowe
Treningowy	54.40 %	85.95 %	85.54 %	77.69 %	91.74 %	85.54 %
Testowy	32.26 %	73.77 %	67.21 %	72.13 %	81.97 %	73.77 %

Tabela przedstawia dwa rodzaje wyników: treningowy oraz testowy. Są to wyniki otrzymane dla dwóch różnych grup danych. Jak było wspomniane wcześniej, dane zostały podzielone na część treningową oraz testową. Grupa danych treningowych jest większa i służy do uczenia modelu. Natomiast dane testowe to takie, których model nie widział wcześniej i służą do przetestowania jak dobrze model się nauczył rozpoznawać informacje. Dlatego też to wynik z danych testowych ma większe znaczenie. W powyższej tabeli można zauważyć, że regresja liniowa poradziła sobie najgorzej otrzymując 32.3% skuteczności. Oznacza to, że model ten jedynie w 1/3 przypadków miał rację. Pozostałe modele poradziły sobie znacznie lepiej otrzymując wyniki na poziomie 70%. Również sieć neuronowa uzyskała wynik na poziomie 73% procent. Najlepiej jednak poradził sobie model losowego lasu decyzyjnego, który uzyskał 82% skuteczności. Oznacza, że z 82% prawdopodobieństwem jest w stanie przewidzieć czy dany pacjent jest chory na serce na podstawie jego wyników badań. Oczywiście 82% to o wiele za mało, by można było użyć taki system na przykład w szpitalu. System ten w 18% przypadków, by się mylił sprawiając, że osoby chore niepoddane by były odpowiedniemu leczeniu. Należy zaznaczyć, jak już było wspomniane wcześniej, że ten zbiór danych jest bardzo mały, co bardzo ogranicza możliwości modeli predykcji. Mając dostęp do dużo większej bazy danych - 1000 lub więcej pacjentów, model mógłby się nauczyć przywidywać chorobę ze znacznie większą skutecznością. Posiadając już wiedzę, który model jest najlepszy oraz jakie parametry są najbardziej istotne można przejść do tworzenia aplikacji desktopowej.

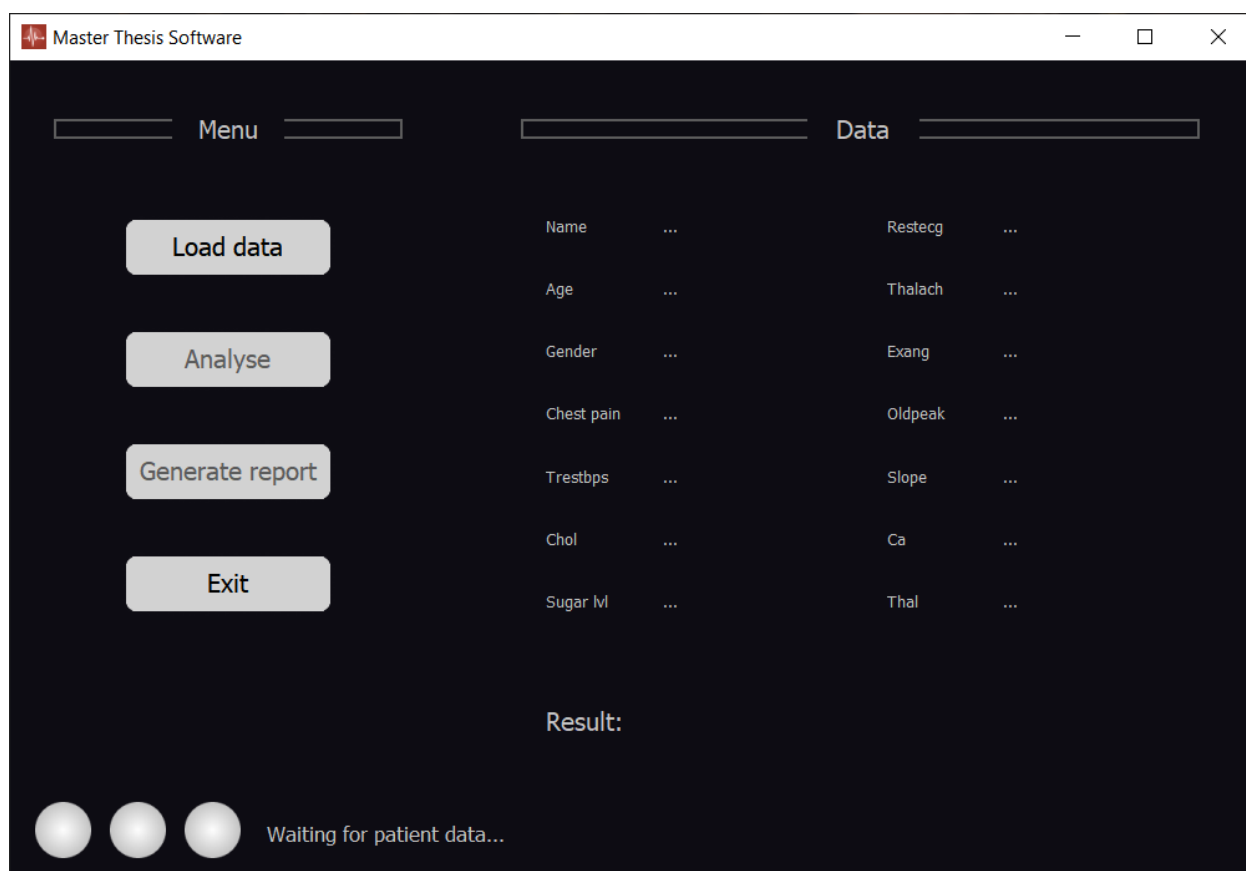
6. Aplikacja

Aplikacja stanowi praktyczną część pracy oraz służy do zaprezentowania omawianego tematu. Przedstawia ona działanie nauczonego modelu na podstawie wprowadzonych w poprzednim rozdziale danych do predykcji chorób serca. Aplikacja skupia w sobie wszystko to co zostało omówione w pracy. Do analizowania danych pacjentów wykorzystuje machine learning, a dokładnie algorytm losowego lasu decyzyjnego, który został wybrany w poprzednim rozdziale jako model, który najlepiej sobie poradził z tym zadaniem. Został on nauczony przy użyciu danych zebranych i przetworzonych za pomocy data mining. Program został napisany w języku Python przy pomocy IDE PyCharm. Do stworzenia interfejsu graficznego została użyta biblioteka PyQt, a dodatkowo elementy graficzne zostały wykonane za pomocą programu Figma. Cały program jest dostępny tylko w języku angielskim. Interfejs użytkownika jest prosty i przejrzysty, dzięki czemu korzystanie z aplikacji jest bardzo proste. Program został przetestowany na danych „Cleveland database”, które zostały opisane w poprzednim rozdziale. Do testów wybrano te dane, na których model się nie uczył. Wyniki są zadowalające. W znacznej większości wykonanych program podawał poprawny wynik. Ponadto aplikacja potrafi wygenerować raport przedstawiający więcej szczegółów na temat rozumowania modelu predykcji oraz jak poszczególne parametry wpływają na wynik pacjenta. Należy zaznaczyć, że program służy jedynie do demonstracji przedstawionego tematu pracy oraz możliwości sztucznej inteligencji i modeli predykcji. Aby stworzyć w pełni działający i skuteczny program przede wszystkim potrzebna byłaby duża ilość danych treningowych oraz przeprowadzenie odpowiednich badań, by skorygować parametry.

Interfejs użytkownika

Interfejs został stworzony w taki sposób, aby był jak najprostszy i intuicyjny w użytkowaniu. Program posiada jedno główne okno, które jest podzielone na dwie części. Po lewej stronie znajduje się menu z czterema przyciskami. Natomiast po prawej stronie są wypisane wszystkie dane pacjenta takie jak imię i nazwisko oraz pozostałe parametry pochodzące z wyników różnych badań. Poniżej została umieszczona grafika przedstawiająca główne okno aplikacji. Interfejs wykorzystuje ciemny motyw, by był mniej męczący dla wzroku,

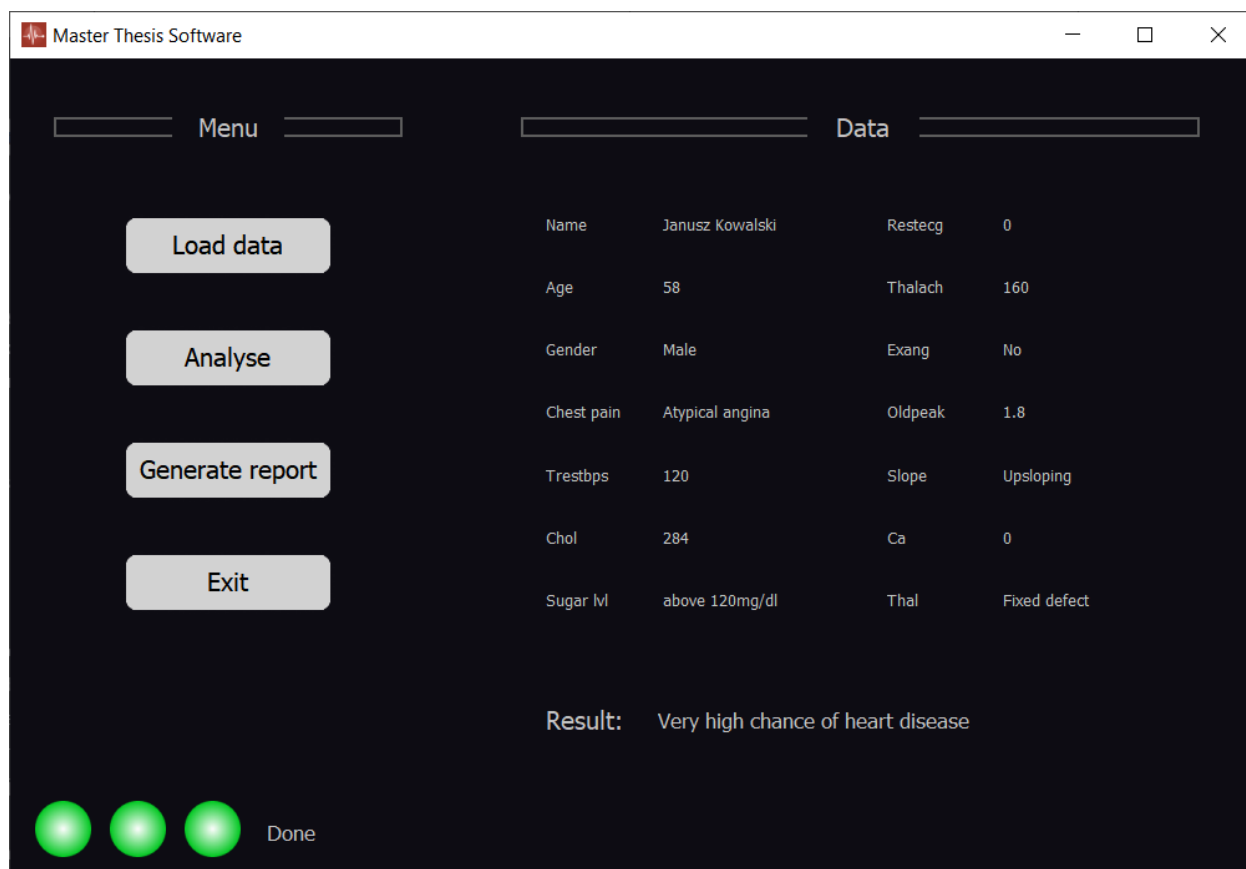
a dodatkowo przyciski oraz napisy są wystarczająco duże, by można było je z łatwością rozczytać.



Rysunek 6.1 Interfejs użytkownika

Menu składa się z czterech przycisków. Część z nich jest początkowo zablokowana. Pierwszy przycisk „Load data” służy do załadowania pliku z danymi pacjenta. Po naciśnięciu uruchamiane jest okno wyboru pliku. Obsługiwane są pliki w formacie csv lub txt. Wewnątrz plików powinny znajdować się dwa wiersze – pierwszy z nazwami poszczególnych parametrów, a drugi z ich wartościami. Jeżeli nazwy parametrów się nie zgadzają lub wartości mają niepoprawny format, to zostanie wyświetlony odpowiedni komunikat o błędzie. Po poprawnym załadowaniu danych informacje z pliku wyświetlane są z prawej strony okna programu. Dodatkowo odblokowuje się drugi przycisk w menu do przeprowadzenia analizy. Przycisk o nazwę „Analyse” służy do rozpoczęcia procesu analizy danych pacjenta. Ponieważ algorytm nie jest bardzo skomplikowany oraz użyta ilość danych jest dość mała proces jest niemal natychmiastowy. W polu „Result” pojawia się wynik analizy. Może przyjąć on trzy wartości,


które oznaczają, że – pacjent jest chory na serce, pacjent jest zdrowy lub wymagane są dodatkowe badania, by wykluczyć chorobę serca. W tym miejscu również może pojawić się komunikat o błędzie, jeżeli nie został załadowany model. Może się tak zdarzyć, jeżeli program nie znajdzie pliku binarnego z modelem w katalogu *data* w głównym folderze aplikacji.



Rysunek 6.2 Interface po poprawnej analizie danych pacjenta

Program oprócz podstawowego wyniku pacjenta jest w stanie przedstawić bardziej szczegółowy opis analizy wyników pacjenta. Po udanej analizie odblokowuje się trzeci przycisk z napisem „*Generate report*”. Przycisk ten uruchamia nowe okno, w którym pojawiają się dodatkowe informacje. Można tam znaleźć wynik mówiący czy pacjent jest chory lub nie, prawdopodobieństwo choroby serca wyrażone w procentach – im większa wartość tym większe ryzyko choroby serca. Raport przedstawia także trzy parametry, które najbardziej wpłynęły na wynik. Dodatkowo generowana jest tabelka opisująca jak każdy z parametrów wpływa na wynik końcowy pacjenta. Dla lepszej czytelności parametry, które mają pozytywny wpływ na wynik są

oznaczone kolorem czerwonym, a parametry z negatywnym wynikiem oznaczone są kolorem zielonym.

 Report ×

Patient has very high risk of heart disease.

Model probability: 93.5%

The most significant parameters that determine a positive result are:

- thal 29.8%
- thalach 20.6%
- ca 15.4%

Table showing how each parameter affects the final result

	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
score	-0.000138	-0.00175	0.127	0.058	0.0147	-0.0365	0.095	0.184
rate	0.02%	0.28%	20.57%	9.40%	2.38%	5.93%	15.41%	29.80%
patient	0	0	160	0	1.8	1	0	2

<>

Save

Cancel

Rysunek 6.3 Raport ze szczegółowymi wynikami pacjenta

Wygenerowany raport można zapisać do pliku pdf klikając na przycisk „save” w dolnym prawym rogu okna z raportem. Okno można wyłączyć klikając na przycisk *cancel*. Należy pamiętać, że główne okno aplikacji jest zablokowane i nie będzie reagować, dopóki jest uruchomione drugie okno z raportem lub wyborem pliku. Aby zakończyć działanie aplikacji należy wybrać ostatni przycisk w menu z napisem „Exit”.

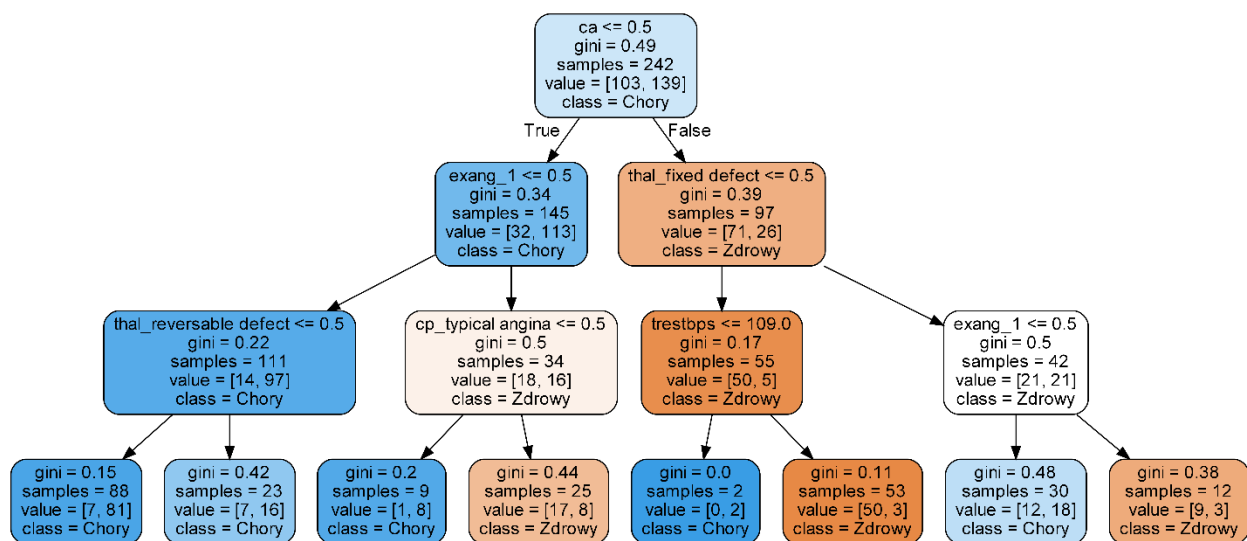
Implementacja

Aplikacja została napisana w języku Python z wykorzystaniem powszechnie dostępnych dla niego bibliotek. Interfejs użytkownika został stworzony przy użyciu biblioteki PyQt, która jest bardzo popularna i można przy jej pomocy stworzyć przyjazne użytkownikowi GUI. Do implementacji losowego lasu decyzyjnego została wykorzystana bardzo dobrze znana biblioteka `scikit-learn`²³, która posiada w sobie gotowe modele predykcji oraz wiele innych użytecznych metod wykorzystywanych przy analizie danych i sztucznej inteligencji. Aplikacja składa się z 12 plików z kodem Python’a, trzech plików z testami jednostkowymi, jednego pliku binarnego będącego modelem oraz kilku plików graficznych. Jako architekturę po części wykorzystano MVP (Model, View, Presenter). Architektura ta dobrze nadaje się do rozdzielania logiki od interfejsu graficznego. Dzięki temu kod związany z wyglądem aplikacji został wydzielony do plików *view.py* oraz *window.py* natomiast logika, która go obsługuje znajduje się w pliku *presenter.py*. Aplikacja wykorzystuje pliki zewnętrzne między innymi do przechowywania modelu, odczytu i zapisu danych, dlatego też został wydzielony osobny plik *file.py*, który zawiera całą implementację związaną z operacjami na plikach. Do ładowania modelu została wykorzystana biblioteka *pickle*. Jest to prosta biblioteka, która pozwala na łatwy zapis i odczyt obiektów z plików binarnych. Do zapisu raportu w formacie pdf aplikacja korzysta z biblioteki FPDF. Jest to jedna z kilku dostępnym bibliotek w pythonie, która pozwala na tworzenie plików PDF.

Podczas pisania każdej aplikacji bardzo istotne są testy. Pozwalają one na sprawdzenie poszczególnych metod oraz całej aplikacji czy zachowuje się tak jak powinna. Dzięki testom mogą być szybko wykryte i poprawione wszelkie niepoprawne zachowania jak zwracanie niepoprawnych wartości, niepoprawna walidacja danych itp. W języku Python domyślnym narzędziem do testów jest biblioteka *unittest*. Pozwala ona na pisanie testów jednostkowych, które sprawdzają działanie poszczególnych metod. Unittest jest bardzo podobny do JUnit, który pochodzi z języka Java ponieważ jest nim inspirowany. Dzięki temu programista znający Javę nie powinien mieć problemu z testami jednostkowymi w Pythonie.

²³ Biblioteka `scikit-learn`: <https://scikit-learn.org/> [Dostęp: 18.09.2021]

Jak było wspomniane wcześniej aplikacja korzysta z open source'owej biblioteki scikit-learn. Dokładniej korzysta z implementacji losowego lasu decyzyjnego, który zapewnia ta biblioteka. Losowy las wykorzystuje uczenie nadzorowane. Można go wykorzystać do regresji, jak i również klasyfikacji czy dany pacjent jest chory czy zdrowy. Losowy las decyzyjny zbudowany jest w oparciu o wiele drzew decyzyjnych. Łącząc je można uzyskać znacznie dokładniejszy wynik. Drzewo decyzyjne to algorytm, który stosuje się do przewidywania lub klasyfikacji zdarzeń na podstawie pewnych reguł decyzyjnych. Jak nazwa wskazuje algorytm ten ma strukturę drzewiastą, tak jak to można zaobserwować na poniższym obrazku. Jest to przykład drzewa, które zostało zasilone danymi pacjentów chorych na serce. Takie same drzewa decyzyjne zostały wykorzystane w programie. Do wizualizacji drzewa wykorzystane zostało narzędzie o nazwie graphviz, które jest open source'ową biblioteką do Pythona umożliwiającą wizualizację grafów.



Rysunek 6.4 Drzewo decyzyjne

Drzewo decyzyjne zaczyna się w jednym punkcie, który nazywamy korzeniem. Korzeń za pomocą węzłów łączy się z kolejnymi wierzchołkami. Natomiast wierzchołki, z których nie wychodzą żadne węzły nazywane są liśćmi. Korzeń, jak i pozostałe wierzchołki odpowiadają na jakieś pytanie, a następnie dzielą zestaw danych. W tym przypadku dane są dzielone na dwie części ponieważ są dwa możliwe rezultaty – pacjent jest chory lub pacjent jest zdrowy. Aby lepiej zrozumieć działanie algorytmu zaczniemy od korzenia. W drzewie decyzyjnym korzeń odgrywa kluczową rolę ponieważ to w nim powinno znajdować się pytanie, które ma największy

wpływ na końcowy wynik. Na rysunku w korzeniu widzimy $ca \leq 0.5$. Jest pytanie decyzyjne, które sprawdza czy parametr ca jest mniejszy lub równy 0.5. W tym zestawie danych ca przyjmuje wartości 0, 1, 2, 3 lub 4. Czyli tak naprawdę korzeń sprawdza czy ca jest równe 0. Jeżeli tak to przechodzi strzałką (węzłem) na lewo, jeżeli nie to na prawo. Samples = 242 – oznacza liczbę danych w wierzchołku w tym przypadku w korzeniu. Value [103, 139] określa jak został podzielony zestaw danych na podstawie pytania. Kolejny bardzo ważny wskaźnik to Gini. Pojawia się on w korzeniu, każdym wierzchołku oraz liściach. Indeks Gini jest jedną z metod używanych w algorytmach drzewa decyzyjnego. Algorytm drzewa decyzyjnego zadaje sekwencje pytań, tak że z każdym pytaniem dane są dzielone na mniejsze podzbiory, aż w końcu w ostatnim elemencie czyli liściu podejmowana jest ostateczna predykcja. Indeks Gini pomaga w optymalnym podziale danych na kolejne podzbiory.

$$I_G(n) = 1 - \sum_{i=1}^J (p_i)^2$$

Równanie 3 Indeks Giniego

Indeks Giniego dla wierzchołka n jest równy 1 minus suma prawdopodobieństw wystąpienia każdego wyniku (w tym przypadku są dwa możliwe wyniki – chory lub zdrowy) podniesionego do kwadratu. Aby lepiej zrozumieć to równanie obliczymy indeks Giniego dla korzenia. Mamy

$$I_{root} = 1 - \left[\left(\frac{103}{242} \right)^2 + \left(\frac{139}{242} \right)^2 \right],$$

$$I_{root} = 1 - (0.181 + 0.330),$$

$$I_{root} = 1 - 0.511 = 0.489.$$

Wartość $\frac{103}{242}$ to prawdopodobieństwo oznaczające brak choroby, a $\frac{139}{242}$ to prawdopodobieństwo wystąpienia choroby. Powyższe wyniki zostały zaokrąglone do trzech miejsc po przecinku, by równanie było bardziej czytelne. Otrzymany wynik to 0.489, co po zaokrągleniu daje ten sam wynik, który jest widoczny w korzeniu na rysunku 6.4. Z korzenia wychodzą dwa węzły. Wierzchołek lewy ma w sobie 145 próbek danych. Są to dane, które odpowiedziały pozytywnie

na pytanie z korzenia tzn. w tym wierzchołku wszystkie dane mają parametr *ca* równy 0. Natomiast wierzchołek prawy ma 97 próbek a dane, które tam są mają wartość *ca* większą niż 0. Kolejne wierzchołki odpowiadają na kolejne pytania jak na przykład czy parametr *exang* jest równy 1 lub czy pacjent ma ból klatki piersiowej typu „*typical angina*”. W każdym wierzchołku drzewo decyzyjne poszukuje takich wartości parametrów, które powodują największą redukcję współczynnika Giniego i na podstawie tego tworzy pytanie. W przykładzie korzeń posiada indeks Giniego równy 0.49. Aby poprawnie obliczyć indeks Giniego dla pary wierzchołków należy skorzystać z następującego wzoru.

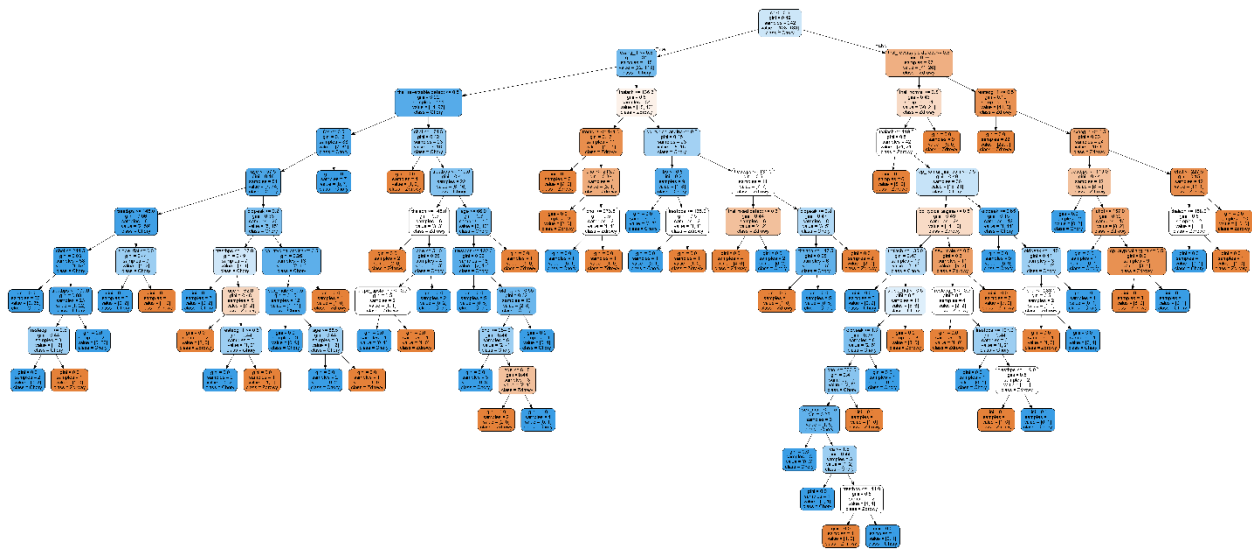
$$I = \frac{n_{\text{lewy}}}{n_{\text{rodzic}}} * I_{n_{\text{lewy}}} + \frac{n_{\text{prawy}}}{n_{\text{rodzic}}} * I_{n_{\text{prawy}}}.$$

Gdzie za n_{lewy} należy podstawić liczbę próbek danych w lewym wierzchołku, n_{rodzic} to rodzic wierzchołka lewego w tym przypadku jest to korzeń, natomiast $I_{n_{\text{lewy}}}$ oznacza indeks Giniego lewego wierzchołka. Poniżej zostały podstawione dane i obliczony indeks Giniego dla pierwszej pary wierzchołków wychodzącej z korzenia:

$$I = \frac{145}{242} * 0.34 + \frac{97}{242} * 0.39,$$

$$I = 0.60 * 0.36 + 0.40 * 0.39 = 0.372.$$

W ten sposób można obliczyć indeks Giniego dla każdej pary wierzchołków. Im głębiej tym wskaźnik ten będzie niższy. Wskaźnik Giniego może przyjąć wartość 0. Oznacza to, że wierzchołek jest całkowicie czysty, co zapewnia, że żadna losowo wybrana próbka danych nie zostałaby źle sklasyfikowana. Drzewo decyzyjne jest podatne na przepełnienie tzn. jeżeli nie ograniczy się głębokości drzewa to może ono rosnąć do momentu aż każda obserwacji uzyska osobny liść. Na kolejnym obrazku zostało zwizualizowane to samo drzewo co na rysunku 6.4, jednak tym razem bez ograniczenia głębokości drzewa.



Rysunek 6.5 Drzewo decyzyjne bez ograniczenia głębokości

Drzewo to jest znacznie większe od poprzedniego. Posiada aż 13 poziomów i 66 liści. Każdy liść posiada wskaźnik Giniego równy zero. Jednak to nie oznacza, że algorytm będzie perfekcyjnie klasyfikował każdą próbkę danych. Przepelnienie sprawia, że drzewo niejako dopasowuje się do danych treningowych sprawiając, że może bardzo dobrze klasyfikować dane, na których się uczyło, ale każdy nowy zestaw danych będzie dla niego bardzo problematyczny. Z drugiej strony zbyt duże ograniczenie głębokości drzewa sprawia, że drzewo nie będzie się w stanie nauczyć nawet rozpoznawać danych treningowych. Jest więc to bardzo istotne, by znaleźć odpowiedni balans pomiędzy tymi cechami, a także wykorzystać wiele drzew, by uśrednić wynik i zminimalizować potencjalny błąd jaki może się pojawić w pojedynczych drzewach. Takie połączenie wielu drzew jest właśnie nazywane lasem decyzyjnym. Oprócz uśredniania wyniku las wykorzystuje do treningu drzewa losowo wybranej próbki danych, mogą się one również powtarzać w tym samym drzewie. Dzięki temu osiągnane wyniki powinny być znacznie lepsze. Możemy teraz przejść do implementacji najistotniejszych metod.

```

18
19     def run(self, data):
20         if self.ready:
21             matrix = createMatrix(data)
22             prediction = self.randomForest.predict(matrix)
23             probability = self.randomForest.predict_proba(matrix)
24             self.result = generateResult(prediction, probability[0][1])
25             return prediction, probability[0][1]
26

```

Rysunek 6.6 Implementacja modelu predykcji

Wewnątrz powyższej metody wykorzystywany jest model losowego lasu decyzyjnego. Jest ona wywoływana przy analizie danych pacjenta. Metoda przed uruchomieniem analizy sprawdza czy zmienna *ready* jest ustawiona na *true*. Jest to istotne, gdyż może się zdarzyć, że model nie jest załadowany, co doprowadziłoby do błędu *NullPointerException*. Metoda *run* przyjmuje w parametrze dane pacjenta. Są one następnie przekształcane w tablicę dwuwymiarową za pomocą metody *createMatrix*. Jest to niezbędne ponieważ w następnej linijce wywoływana jest metoda *predict* z klasy *RandomForestClasifier*, która przyjmuje taki format danych. Pod zmienną *prediction* znajduje się wynik analizy danych pacjenta. Może on przyjąć dwie wartości: 0 lub 1, gdzie 1 oznacza, pozytywny wynik czyli wykryto chorobę serca lub 0, które oznacza, że pacjent jest zdrowy. Kolejna zmienna *probability* przechowuje wynik metody *predict_proba*. Metoda ta zwraca prawdopodobieństwo wystąpienia choroby. W kolejnej linii znajduje się wywołanie metody pomocniczej *generateResult*. Na podstawie predykcji oraz prawdopodobieństwa generuje ona podstawowy opis wyniku pacjenta. Możemy go zobaczyć w oknie głównym aplikacji przy napisie *Result*. Natomiast sama metoda zwraca dwie wartości – predykcje oraz prawdopodobieństwo. Są one później wykorzystywane przy generowaniu raportu.

```

9
10     def generate(self, patient_data: DataFrame, prediction, probability):
11         explainer = TreeExplainer(self.model)
12         shap_values = explainer.shap_values(patient_data)
13         reasoning = DataFrame(shap_values[1], columns=patient_data.columns.values)
14         utils = ReasoningUtils(reasoning, patient_data)
15         message = self._createMessage(prediction, probability, utils)
16
17         return Report(message[0], message[1], message[2], utils.reasoning, probability)
18

```

Następna metoda, którą warto zaprezentować to *generate* z klasy Reporter. To tutaj jest tworzony szczegółowy raport dotyczący analizy danych pacjenta oraz wpływu jaki mają poszczególne parametry na wynik. Taka szczegółowa analiza jest możliwa dzięki wykorzystaniu *TreeExplainer* z biblioteki SHAP. Wspiera on większość modeli opartych na drzewie z biblioteki scikit-learn w tym losowy las decyzyjny. SHAP oblicza wpływ każdego parametru korzystając między innymi z kombinatoryki. Posiada on metoda *shap_values*, która przyjmuje dane pacjenta, a jako wynik zwraca wartości dla każdego parametru pacjenta. Wartości te przedstawiają jaki mają wpływ na wynik drzewa decyzyjnego. Przyjmują one formę wartości zmiennoprzecinkowych i mogą być dodatnie lub ujemne w zależności czy wpływają negatywnie czy pozytywnie na wynik. Następnie wartości te są parsowane na obiekt *DataFrame*. Generowane są wiadomości tekstowe dotyczące szczegółowego opisu analizy oraz ostatecznie zwracany jest obiekt *Report*, który przyjmuje kilka parametrów. Obiekt ten jest później przesyłany do klasy, która wyświetla okno z raportem.

Obserwacje

Podczas pisania i testowania programu zauważono, że wybór odpowiedniego modelu ma duży wpływ dla otrzymanych wyników. Umiejętne dobranie oraz skalibrowanie parametrów jest bardzo istotne. Model, użyty w programie to losowy las decyzyjny, który ma ustawioną głębokość równą 4, jest złożony z pięciu drzew oraz ma ustawiony parametr *randomState* na 20. Parametr ten kontroluje losowość próbek danych, z których budowane są drzewa. Z bazy danych pacjentów wybrano losowo jednego pacjenta, zwanego pacjentem X. Testując działanie programu i analizując jego wyniki zauważone zostały pewne rozbieżności, jeżeli chodzi o wykorzystanie modelu lasu decyzyjnego z różnymi parametrami. Domyślny model w raporcie stwierdził, że pacjent X ma 91.1% prawdopodobieństwa na chorobę serca oraz największy wpływ na wynik mają parametry *thal* (27.1%), *thalach* (21.2%) oraz *ca* (14.4%). Drugi las decyzyjny, który badano miał głębokość ustawioną na 3, był złożony z 6 drzew, natomiast parametr *randomState* był taki sam jak poprzednio. Ten model dla danych pacjenta X stwierdził, że ma on 66.6% prawdopodobieństwa na chorobę oraz największy wpływ na wynik mają parametry *thalach* (19.2%) *thal* (17.4%) i *exang* (12.6%). Różnica w prawdopodobieństwie wynosiła 32.5 procenta. Model także wskazał trochę inne parametry, które miały największy wpływ na wynik.

Tabela 3 Wyniki modelu lasu decyzyjnego z różnymi ustawieniami

Liczba drzew	Głębokość	Bootstraping	RandomState	Wynik	Parametry
5	4	tak	20	91.1%	thal - 27.1% thalach - 21.2% ca - 14.4%
6	3	tak	20	66.6%	thalach - 19.2% thal - 17.4% exang - 12.6%
10	3	tak	null	66.1%	thal - 19.3% cp 18.1% ca 16.7%
5	4	nie	—	77.7%	ca - 28.9% cp - 18.6% exang – 10.6%

Powyższa tabelka wskazuje wykorzystanie modelu lasu decyzyjnego z różnymi parametrami takimi jak ilość drzew czy głębokość. Bootstraping oznacza, że drzewa są trenowane przy użyciu losowo wybranych próbek danych. Dodatkowo niektóre próbki danych mogą być wielokrotnie użyte w tym samym drzewie. Wynik oznacza prawdopodobieństwo wystąpienia choroby w procentach. Natomiast kolumna o nazwie „Parametry” przedstawia jakie parametry model wskazał jako najbardziej istotne. Można zauważyć, że prawdopodobieństwo oscyluje pomiędzy 66 a 91 procent. Czyli każdy model poprawnie wykrył chorobę dla tego pacjenta. Modele te testowane były również na danych innych pacjentów i za każdym razem wskazywały one poprawny wynik. Jednak prawdopodobieństwo, a także wskazanie najbardziej wpływowych parametrów było inne w zależności od użytego modelu.

Model, który ma wyłączony bootstraping, wydaje się wskazywać jako kluczowe parametry te, które są położone najwyżej na rysunku 5.3. Czyli te, które powinny mieć największy wpływ na chorobę według przeprowadzonej analizy. Przykładowo około 70% pacjentów chorych ma parametr *ca* równy 0. Natomiast większość zdrowych osób ma ten parametr większy od zera. W poprzedniej tabeli można zauważyć, że tylko model z wyłączonym parametrem bootstraping wskazał *ca* oraz *cp* jako najbardziej istotne parametry powodujące chorobę serca u tego pacjenta. Testując ten model udało się osiągnąć 88.5% efektywności. To

znaczy, że model poprawnie zdiagnozował pacjenta w 88.5 procenta przypadków. Widać, że ustawienie odpowiednich parametrów przy tworzeniu lasu decyzyjnego ma istotny wpływ na osiągane wyniki. Przy pomocy znacznie większej liczby badań i danych, oraz po odpowiednim skorygowaniu parametrów aplikacja w teorii mogłaby z lepszą skutecznością wstępnie diagnozować pacjentów oraz przedstawiać, które parametry pacjenta wskazują na chorobę.

7. Podsumowanie

Uczenie maszynowe oraz analiza danych jest niesamowitym narzędziem, które ma szerokie zastosowanie w obecnych czasach. Możemy sobie nie zdawać sprawy, jak duży ma ono wpływ na nasze codzienne życie. Uczenie maszynowe i analiza danych jest wykorzystywana, gdy robimy zakupy przez Internet, dostajemy listę polecanych produktów, utworów muzycznych czy nawet nowych znajomych na podstawie naszej aktywności w sieci. Uczenie maszynowe nie jest wykorzystywane tylko w social mediach, e-commerce czy finansach, ale również w medycynie. Data mining pomaga w leczeniu ludzi. Wspiera sektor medyczny także w wielu innych aspektach. Od wspieranie produkcji leków, dzięki czemu mogą być one tańsze i skuteczniejsze, aż po walkę z oszustwami w branży medycznej. Uczenie maszynowe oraz analiza danych mają bardzo szerokie zastosowanie i ogromny wpływ na dzisiejszy świat.

Analiza danych jest niezwykle przydatna. Posiadanie dużej ilości danych to tylko połowa sukcesu, ponieważ to proces analizy danych potrafi wyciągnąć niezwykle przydatne informacje z surowych danych. Dzięki zastosowaniu data mining oraz narzędzia Jupyter notebooka udało się dokładnie przeanalizować bazę danych „*Cleveland database*” z danymi pacjentów chorych na serce. Uzyskane informacje były niezbędne do napisania aplikacji desktopowej, która wykorzystuje model losowego lasu decyzyjnego do diagnozowania pacjentów. Korzystając z analizy danych można wyciągnąć ogromne ilości informacji nawet z prostych zestawów danych. Narzędzie Jupyter notebook oraz język Python są proste do nauki i są bardzo przydatne nie tylko do przeprowadzania analizy i wizualizacji danych, ale również do programowania i przedstawiona kodu w formie dokumentu. Jak się okazuje, język Python ma naprawdę szerokie zastosowanie. W pracy został użyty nie tylko do samej analizy i wizualizacji danych, ale również do stworzenia aplikacji desktopowej z prostym i nowoczesnie wyglądającym interfejsem użytkownika.

Ostatnią częścią pracy było napisanie aplikacji desktopowej. Została ona napisana w języku Python przy użyciu biblioteki PyQt. Dzięki przeprowadzonej wcześniej analizie danych oraz wykorzystaniu języka Python napisanie programu nie przysporzyło wielu problemów. Aplikacja miała diagnozować pacjentów czy są chorzy na serce na podstawie ich wyników badań. Do tego celu został wykorzystany model losowego lasy decyzyjnego, który został nauczony na podstawie zestawu danych „*Cleveland database*”. Model ten został wybrany,

ponieważ podczas analizy danych losowy las decyzyjny osiągnął najlepsze wyniki przebijając nawet sieci neuronowe. Z przeprowadzonych testów wynika, że oprogramowanie w znacznie większości poprawnie diagnozuje pacjentów chorych na serce. Oprócz samej diagnozy program generuje dodatkowo raporty pokazujące bardziej szczegółowy wgląd w to, dlaczego pacjent został zdiagnozowany tak, a nie inaczej. Należy tutaj jednak zaznaczyć, że program służy jedynie do demonstracji przedstawionego tematu pracy oraz możliwości sztucznej inteligencji i modeli predykcji. Aby stworzyć w pełni działający i skuteczny program przede wszystkim potrzebna jest ogromna ilość danych treningowych oraz przeprowadzenie wielu odpowiednich badań, by ustawić odpowiednie parametry. Wtedy takie oprogramowanie mogłoby wskazywać dokładniejsze wyniki i być wykorzystane przez lekarzy przy wstępnym diagnozowaniu pacjentów.

Literatura

1. A. McAfee, E. Brynjolfsson. Big Data: The Management Revolution, 2012 r
2. Analiza usług laboratoriów medycznych pod kątem oszustw w USA
<https://www.cms.gov/files/document/download-clinical-laboratory-services-white-paper.pdf>
3. Biblioteka scikit-learn: <https://scikit-learn.org>
4. Biblioteka SHAP: <https://shap.readthedocs.io>
5. Charu C. Aggarwal, Data mining – The textbook, Springer, 2015 rok
6. Historyczny mecz w warcaby: <http://www.fierz.ch/samuel.htm>
7. Index TIOBE: <https://www.tiobe.com/tiobe-index>
8. InnerEye: <https://www.microsoft.com/en-us/research/project/medical-image-analysis>
9. Kunihiko Fukushima, Neocognitron: A hierarchical neural network capable of visual pattern recognition. 1988 rok
10. Nature, End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography, 2019r
11. Oficjalny blok Google: <https://blog.google/products/google-voice/neon-prescription-or-rather-new>
12. Platforma sprawozdawcza: <https://promedmail.org>
13. Pojazd ze Stanford: <http://cyberneticzoo.com/cyberneticanimals/1960-stanford-cart-american>
14. Sanction Scanner: <https://sanctionscanner.com>
15. Sepp Hochreiter i Jürgen Schmidhuber, Long Short-Term Memory. 15.11.1997
16. Startup Standigm: <https://www.standigm.com>

17. Statystyka Netflix: <https://www.statista.com/statistics/250934/quarterly-number-of-netflix-streaming-subscribers-worldwide>
18. Statystyka Spotify: <https://investors.spotify.com>
19. Tin Kam Ho, Random decision forests. 1995 rok
20. Warren S. McCulloch & Walter Pitts: A logical calculus of the ideas immanent in nervous activity. 1943r
21. Watson Oncology: <https://www.ibm.com/watson-health/solutions/cancer-research-treatment>
22. Wikipedia, AlphaGo: https://en.wikipedia.org/wiki/AlphaGo_Zero
23. Wolfgang Ertel, Introduction to Artificial Intelligence, 2017 rok