



Zachodniopomorski  
Uniwersytet Technologiczny  
w Szczecinie



Wydział  
Elektryczny

**Patryk Frankowski**

nr albumu: 30570

kierunek studiów: Automatyka i Robotyka

forma studiów: studia stacjonarne II stopnia

**BADANIE WYBRANYCH ALGORYTMÓW SZTUCZNEJ INTELIGENCJI  
W CELU PREDYKCJI ZDARZEŃ  
NA PODSTAWIE DANYCH METEOROLOGICZNYCH**

**THE STUDY OF SELECTED ARTIFICIAL INTELLIGENCE ALGORITHMS TO  
PREDICT EVENTS BASED ON METEOROLOGICAL DATA**

Praca dyplomowa magisterska

napisana pod kierunkiem:

**dr inż. Artura Wollek**

Katedra Sterowania i Pomiarów

Data wydania tematu pracy: .....

Data złożenia pracy: .....

Szczecin, 2018

**OŚWIADCZENIE  
AUTORA PRACY DYPLOMOWEJ**

Oświadczam, że praca dyplomowa magisterska pt. „Badanie wybranych algorytmów sztucznej inteligencji w celu predykcji zdarzeń na podstawie danych meteorologicznych” napisana pod kierunkiem: dr inż. Artura Wollek jest w całości moim samodzielnym autorskim opracowaniem sporządzonym przy wykorzystaniu wykazanej w pracy literatury przedmiotu i materiałów źródłowych.

Złożona w dziekanacie Wydziału Elektrycznego treść mojej pracy dyplomowej w formie elektronicznej jest zgodna z treścią w formie pisemnej.

Oświadczam ponadto, że złożona w dziekanacie praca dyplomowa ani jej fragmenty nie były wcześniej przedmiotem procedur procesu dyplomowania związanych z uzyskaniem tytułu zawodowego w uczelniach wyższych.

Szczecin, dn. .....  
.....  
podpis dyplomanta

**Streszczenie pracy**

W pracy dyplomowej opisano wykonanie oraz testowanie systemu zawierającego wybrane algorytmy sztucznej inteligencji, którego celem jest predykcja zdarzeń na podstawie danych meteorologicznych. Informacje dostarczane są za pomocą przenośnego systemu pomiarowego skonstruowanego w ramach pracy inżynierskiej autora. Analizowane dane to: temperatura otoczenia, ciśnienie atmosferyczne, wilgotność powietrza, temperatura wody, położenie geograficzne, czas i data.

**Słowa kluczowe**

Parametry pogodowe, predykcja, algorytm, data science, uczenie maszynowe

**Abstract**

This thesis describes construction and testing of the system which contain chosen algorithms of artificial intelligence which objective is to predict events based on meteorological data. The informations are supplied by a mobile measuring system which was constructed as part of BEng Thesis of the author. The analyzed data are: temperature of environment, atmospheric pressure, humidity of air, temperature of water, geographical positioning, time and date.

**Keywords**

Weather parameters, prediction, algorithm, data science, machine learning

## Spis treści

Wprowadzenie .....	6
1. Metody analizy danych w celu predykcji zdarzeń.....	8
1.1. Definicja sztucznej inteligencji.....	8
1.2. Historia i rozwój sztucznej inteligencji .....	9
1.3. Działalności i zastosowania sztucznej inteligencji .....	11
1.4. Data Science i uczenie maszynowe w celach predykcji .....	13
1.4.1. Data Science.....	13
1.4.2. Metody uczenia maszynowego.....	15
1.4.3. Etapy tworzenia systemów uczenia maszynowego w celu predykcji.....	19
1.4.3.1. Wstępne przetwarzanie danych .....	20
1.4.3.2. Uczenie .....	26
1.4.3.3. Ewaluacja .....	38
2. Konstrukcja urządzenia pomiarowego .....	46
2.1 Projekt elektroniki, konstrukcja płyty PCB oraz obudowy.....	46
2.2 Oprogramowanie systemu pomiarowego .....	57
3. Projekt aplikacji systemu do analizy algorytmów uczenia maszynowego w celu predykcji zdarzeń.....	61
3.1. Microsoft Azure Machine Learning Studio.....	61
3.2. Integracja z urządzeniem pomiarowym oraz baza danych .....	64
3.3. Wstępne przetwarzanie danych.....	66
3.4. Etap uczenia i ewaluacji modelu analitycznego .....	79
3.5. Zestawienie wyników i ich analiza .....	83
3.6. Operacyjonalizacja modelu predykcyjnego .....	101
Wnioski.....	104
Załącznik A. Schemat elektryczny urządzenia.....	105
Załącznik B. Listing programu .....	107
Bibliografia .....	115
Spis tabel .....	116
Spis rysunków .....	117

## Wprowadzenie

Pogoda potrafi mieć znaczący wpływ na wybrane zdarzenia. Jakość i ilość plonów, zachowanie zwierząt, jakość powietrza, a nawet samopoczucie często są uzależnione od panujących warunków atmosferycznych. Tutaj Autor pracy stawia pytania:

- *Na podstawie przewidywanych parametrów pogodowych, z jaką dokładnością można byłoby przewidzieć wybrane zdarzenia?*
- *Czy obserwacje ciągu zdarzeń są w stanie prognozować nam pogodę w najbliższym czasie?*

Aby na nie odpowiedzieć, należy wziąć pod uwagę kilka dziedzin naukowych.

W celu przeprowadzenia badań Autor pracy postanowił stworzyć własną bazę danych, na podstawie zebranych wielkości zmierzonych urządzeniem skonstruowanym przez siebie w ramach pracy inżynierskiej pt. „*Projekt przenośnego wodoszczelnego systemu do pomiaru parametrów pogodowych*”. Badane parametry pogodowe to: temperatura i wilgotność powietrza, temperatura wody oraz ciśnienie atmosferyczne. Za pomocą modułu GPS, zapisywane są: czas, data i położenie geograficzne. Dodatkowo urządzenie rejestruje aktywność badanego zdarzenia.

Poprawne skonstruowanie bazy danych oraz analiza zawartych w niej informacji jest kolejną częścią niniejszej pracy. W tym etapie niezbędne są narzędzia, które oferują dział sztucznej inteligencji. Jest to dziedzina, m.in. informatyki, która rozpoczęła preźny rozwój w wieku XX i nadal odkrywane są nowe obszary tej nauki. *Data Science* wspólnie z *Machine Learning* obejmują techniki automatycznego pozyskiwania wiedzy. Inteligentne systemy informatyczne charakteryzują się zdolnością do uczenia się, polegającym na odpowiednim doborze algorytmów, począwszy od dokładnej analizy i do precyzowania informacji w bazie danych, poprzez wykorzystanie wzorów klasyfikujących w celu uzyskania żądanego wyników. Jest to proces skomplikowany, ze względu na: różnorodność jakości i ilość uzyskanych danych oraz bardzo dużą liczbę możliwych do wykorzystania algorytmów sztucznej inteligencji.

### Cel pracy

Celem pracy jest przeprowadzenie analizy rezultatów predykcji zdarzeń korzystając z wybranych algorytmów z dziedziny sztucznej inteligencji. Dane niezbędne do przeprowadzenia badań uzyskano z samodzielnie zaprojektowanego i skonstruowanego urządzenia rejestrującego pomiary parametrów atmosferycznych, informacje o położeniu geograficznym, datę, czas oraz wartości przekazywane przez użytkownika.

### Zakres pracy

System ma mieć możliwość wnioskowania predykcji zdarzeń, na podstawie analizy uzyskanych wartości parametrów pogodowych, położenia geograficznego, czasu i daty oraz danych przekazywanych przez użytkownika. Ponadto badania poprawności final-

## Metody analizy danych w celu predykcji zdarzeń

Predykcja determinuje pewien proces wnioskowania, w określonym czasie, przyszłych wartości zmiennych losowych, w przypadku braku znajomości wielkości wyjściowej. Umiejętność predykcji zdarzeń stwarza możliwości przeciwdziałania nieuchianym wydarzeniom (np. tornada, trzęsienia ziemi), planowania kolejnych decyzji (np. ruchy na giełdzie), czy też przeprowadzania badań w wybranej tematyce (np. migracja zwierząt, zmiany naprężenia konstrukcji stalowych). Człowiek wykorzystując swoją inteligencję i doświadczenie jest w stanie przewidywać pewne zdarzenia. Każdy z nas doświadczył niejednokrotnie skutecznej predykcji, często nie zdając sobie z tego faktu sprawy. Najprostszym przykładem może być tzw. „łamanie w kościach”, które na ogół zapowiada pogorszenie się warunków atmosferycznych (doświadczenie), czy też upadek cegłówki z danej wysokości, co spowoduje jej pęknięcie (inteligencja). Te wszystkie procesy myślowe i intuicyjne dające do przewidywania pewnych zdarzeń staramy się opisać odpowiednimi algorytmami. Umożliwia to wykonywanie dokładniejszych i bardziej skomplikowanych predykcji. Między innymi z takich powodów powstała dziedzina informatyki i matematyki zwana Sztuczną Inteligencją [1].

### 1.1. Definicja sztucznej inteligencji

W 1956r., w trakcie projektu *Dartmouth Summer Research Project*, John McCarthy przekonał obecnych naukowców do sformułowania *sztuczna inteligencja* jako nowej dziedziny nauki. Obecnie, większość ludzi jest już przyzwyczajona do tego wyrażenia, aczkolwiek przez wiele lat istniały debaty nad zmianą nazwy tegoż działu. Wielu przeciwników stwierdzało, iż samo pojęcie inteligencji nie jest dobrze zdefiniowane, a część „sztuczna” przed jakimkolwiek słowem nie brzmi pozytywnie. Stwarzało to możliwości żartowania o „sztucznej głupoty” i „naturalnej inteligencji”. Jednak, gdy tylko opadła kurz nad dysputą o poprawnym nazewnictwie, naukowcy skupili się na zdefiniowaniu celów sztucznej inteligencji. Badaniami w tej dziedzinie mogą towarzyszyć różne motywacje. Wyróżnia się dwa podstawowe czynniki:

- **motywacja na poziomie inżynierskim** - polega na koncentracji w tworzeniu systemów, mających na celu wyręczenie człowieka w pracach wymagających od niego inteligencji,
- **motywacja psychologiczno-filozoficzna** – opiera się na konstruowaniu mechanizmów działających jak ludzki umysł, w celu lepszego poznania praw rządzących ludzką inteligencją.

W związku z tym ewoluowały dwie definicje pod względem efektu wysiłku intelektualnego wymaganego w tej dziedzinie:

1. **Słaba sztuczna inteligencja** – tworzy metody rozwiązuające zadania, które człowiek jest w stanie rozwiązać z wykorzystaniem własnego intelektu,
2. **Silna sztuczna inteligencja** – są to systemy, które samoistnie myślą na poziomie inteligencji ludzkiej lub nawet ją przewyższają [1, 2, 3].

W świecie nauki o sztucznej inteligencji znacznie przeważa motywacja inżynierska. W związku z tym można jej przypisać sformułowanie zaproponowane przez jednego z ojców tej dziedziny, Marvina Minsky'ego: „*Sztuczna inteligencja jest nauką o maszynach realizujących zadania, które wymagają inteligencji, gdy są wykonywane przez człowieka*” [1].

## 1.2. Historia i rozwój sztucznej inteligencji

Pewne pytania, które można dziś zaliczyć do dziedziny sztucznej inteligencji, takie jak: *czy procesy myślowe można opisać matematycznymi wzorami? Czy można je zautomatyzować?* Nurtowały one filozofów już w starożytności. Jednakże, przez brak rozbudowanych mechanizmów do tworzenia i realizacji zaawansowanych algorytmów, nauki o sztucznej inteligencji poczyniły mały progres, aż do początków XX wieku [1].

W 1950r. Alan M. Turing zaproponował test, który miał odpowiedzieć na pytanie, kiedy można uznać system stworzony przez człowieka za inteligentny. Wówczas powstała tak zwana *gra imitacyjna*. Polegała ona na tym, że tester (człowiek) prowadził jednocześnie dialog z innym człowiekiem i komputerem. Usunięto wszelkie sposoby prostej identyfikacji interloktora m.in. udzielane odpowiedzi, czy to od maszyny, czy drugiego człowieka były wyświetlane na ekranie w formie tekstu. Po ustalonym czasie, tester miał za zadanie zidentyfikować obu rozmówców. Zdaniem Turinga, jeśli w tym momencie przesłuchujący nie był w stanie odróżnić człowieka od komputera na podstawie ich odpowiedzi, to sztuczna inteligencja jest nieodróżnialna od ludzkiej. Test Turinga, był jedną z pierwszych prób określenia sztucznej inteligencji, nadal wykorzystywany w dzisiejszych czasach [2].

Istnieją dwie teorie odnośnie powszechnie uznawanego roku narodzin sztucznej inteligencji. W 1955r. powstał system *Logic Theorist*, który udowodnił 38 z 52 pierwotnych twierdzeń zawartych w „*Principia Mathematica*”, zaprojektowany przez dwóch naukowców: Allena Newella oraz Herberta Simon'a. W kolejnym roku w Dartmouth odbyła się konferencja pod hasłem „*Każdy aspekt uczenia się lub innej cechy inteligencji może być opisany na tyle precyzyjnie, aby mogła go wykonać maszyna*”, gdzie McCarthy przekonał zebranych do wyrażenia „*sztuczna inteligencja*”, jako nowej dziedziny nauki. Mimo, iż istnieje dylemat odnośnie dokładnej daty narodzin AI (ang. *Artificial Intelligence*), pewnym jest, że przypada ona na połowę lat 50. XX wieku [1, 2].

Na przestrzeni lat 50. i 60. sztuczna inteligencja bardzo szybko ewoluowała. Część naukowców skupiła się na tworzeniu komputerów i języków programowania umożliwiających rozwiązywanie wysoko skomplikowanych operacji. W 1957r. F. Ro-

senblatt przedstawił perceptron – najprostszy model sieci neuronowej, który określał przynależność wartości wejściowych do jednej z dwóch klas (tzw. klasyfikator binarny). Prekursorami sztucznej inteligencji byli m.in. wyżej wspomniani Allan Newell i Herbert Simon, którzy już w 1957r. stworzyli pierwszą wersję GPS, czy też McCarthy, który w 1958r. zaprezentował język programowania *LISP* (ang. *LIS Processing*) używany przez wiele lat w środowisku badaczy sztucznej inteligencji. Badania opierały się wówczas na tworzeniu programów rozwiązywających problemy z wykorzystaniem logiki. W 1966r. Joseph Weizenbaum skonstruował *ELIZA*, który był pierwszym systemem komunikującym się z człowiekiem poprzez komunikator tekstowy. Tak preżny rozwój sztucznej inteligencji został zahamowany w 1969r., kiedy to Marvin Minsky oraz Seymour Papert z MIT, przedstawili książkę „*Perceptrons*”, w której wskazali poważne ograniczenia perceptronu Rosenblatta. Był on wtenczas podstawą wielu systemów sztucznej inteligencji. Czas poświęcony projektowaniu i tworzeniu nowych mechanizmów, został znacznie zastąpiony dysputą między naukowcami o sztucznej inteligencji w sferze, można by rzec – filozoficznej [1, 2].

Lata 70. zmieniły kurs postępu nauki o sztucznej inteligencji. Wyraźnym rozwojem w tym czasie charakteryzowały się systemy *eksperckie*. Ten kierunek AI to tzw. po-dejście oparte na wiedzy. Systemy ekspertowe, jak sama nazwa wskazuje, bazują na mądrości i doświadczeniu ekspertów (ludzi). Cały mechanizm w pierwszej kolejności związany był z projektem *DENDRAL*, prowadzonym przez Edwarda Pegenbauma oraz Joshua Ledberga na Stanford University, rozpoczętym już w poprzednim dziesięcioleciu. Systemy ekspertowe wyróżniały się na owe czasy tym, że skupiały się na rozwiązywaniu problemów w określonym obszarze zastosowania systemu sztucznej inteligencji. Cała struktura bazuje na wiedzy ekspertów w danej dziedzinie (sformalizowanej w postaci danych) oraz wyposażona jest w mechanizmy wnioskowania. System ekspertowy założenia powinien pracować w środowisku ze specjalistami i rozwiązywać problemy niezależnie od nich. Poprawność działania mechanizmu sprawdza się porównując uzyskane wyniki symulacji z wiedzą ekspertów. Dział sztucznej inteligencji pod koniec lat 70. (poza opisanym wyżej systemem eksperckim) cechował się głównie przystępowniem rozwoju. Naukowcy byli w stanie z wykorzystaniem systemów uzyskać wyniki złożonych zadań matematycznych. Jednakże z pozoru łatwe czynności t.j. rozpoznanie twarzy lub unikanie przeszkód w trakcie przemieszczania się robotów, znacznie przerażały moc obliczeniową ówczesnych komputerów [1, 2].

Na początku lat 80., sztuczna inteligencja wyszła poza obszar tylko badań naukowych. Pierwsze korporacje zaczęły inwestować w systemy ekspertowe, a nawet tworzyć wewnętrzne działy AI. Zauważono, że ta relatywnie młoda dziedzina nauki może przynieść duże usprawnienia oraz korzyści majątkowe. Dla przykładu: w Digital Equipment Corporation zaczęto korzystać z systemu XCON w 1980r. Po upływie sześciu lat stwierdzono w tejże korporacji zmniejszenie wydatków o ok. 40mln USD rocznie tylko dzięki wdrożeniu mechanizmów sztucznej inteligencji. Powstały pierwsze firmy specjalizujące się w konstrukcji maszyn oraz oprogramowania dla systemów ekspertowych. Rządy m.in. Japonii, Wielkiej Brytanii i Stanów Zjednoczonych przeznaczały kolosalne fundusze w rozwój sztucznej inteligencji. Niestety, tak duże finansowanie nie przyniosło

oczekiwanych rezultatów, co w istocie spowodowało kolejny kryzys w postępie badań w naukach AI. Główne zarzuty dotyczyły nadal zbyt wąskich dziedzin życia, w których te systemy mogły mieć zastosowanie. Dodatkowym bodźcem skutującym przyhamowaniem rozwoju sztucznej inteligencji było pojawienie się na rynku komputerów osobistych, które skutecznie wyparły systemy eksperckie [1, 2].

Od lat 90. zauważalny jest wyjątkowo szybki rozwój sztucznej inteligencji. Głównym tego powodem jest zwiększająca się moc obliczeniowa coraz to nowszych komputerów. Także znaczny postęp technologii usprawnia rozwój nauk z działu AI zmniejszając niezbędne fundusze w porównaniu do wcześniejszych dziesięcioleci. Lata 90. charakteryzowały się przede wszystkim rozwojem nowych algorytmów w tej dziedzinie, t.j. algorytm lasów losowych, algorytm wektorów nośnych. Sztuczna inteligencja wówczas zaczęła łączyć się z nowymi dziedzinami, jak robotyka przemysłowa, logistyka, ekonomia, medycyna. Tworzyło to nowe możliwości, często skutecznie zastępując ludzką logikę i znacznie skracając czas wnioskowania żądanych wyników.

Jeszcze niedawno AI była to nauka spotykana tylko w zaawansowanych laboratoriach i dużych korporacjach. Dziś każdy może mieć styczność ze sztuczną inteligencją, m.in. w telefonach komórkowych, gdzie istnieją systemy rozpoznawania mowy albo wyszukiwarkach internetowych tj. Google. W 2011r. na rynku pojawił się Siri – pierwszy osobliwy zautomatyzowany asystent oparty na algorytmach sztucznej inteligencji. Ten i podobne jemu systemy potrafią adaptować swoje odpowiedzi oraz rekomendacje w zależności od preferencji użytkownika, bez potrzeby programowego wprowadzania informacji. Należy wspomnieć, że dobór odpowiednich algorytmów wymaga dużego zbioru danych. Ten problem został zminimalizowany wraz z upowszechnieniem się Internetu. Można stwierdzić, że sztuczna inteligencja obecnie się „zapętla”. Jest niezbędnym narzędziem, do tworzenia baz danych poprzez zautomatyzowane zbieranie informacji, do ich analizy, a także odpowiedniego wykorzystania uzyskanych wniosków w celu przekazywania spersonalizowanych rekomendacji. Reakcja użytkownika na proponowane wskazówki jest kolejną informacją do tejże bazy danych, która adaptuje nowe algorytmy, aby coraz to lepiej przedstawiać ofertę zainteresowanemu.

### 1.3. Dział i zastosowania sztucznej inteligencji

Sztuczna inteligencja dzięki swojemu potencjałowi i coraz mniejszym ograniczeniom technicznym jest łączona z innymi dziedzinami. Powoduje to powstawanie coraz to nowszych działań tej nauki. Jednakże można wyróżnić cztery podstawowe gałęzie sztucznej inteligencji zajmujące się odrębnymi zadaniami.

Ludzka inteligencja charakteryzuje się umiejętnością wnioskowania pewnych faktów na podstawie uzyskanych danych. Przykładem może być właściciel sądu, który znając ilość plonów, ceny rynkowe oraz wszelkie koszty związane m.in. z transportem jest w stanie wywnioskować zysk ze zbiorów. Ludzie potrafią także wyznaczyć trajektorię ruchu rakiety, posiadając dane o masie, ilości spalania paliwa, aerodynamice itp. Każde zadanie wiąże się z różną jakością i ilością danych. Wykorzystując komputery,

które mają znacznie większą moc obliczeniową od człowieka, można rozwiązywać problemy zawierające ogólną liczbę zapisanych informacji. Obecnie wyróżnia się dwa systemy polegające na wnioskowaniu posługujące się logiką [1]:

- System ekspercki – otrzymując bieżące informacje i korzystając z wartości przekazanych w formie bazy danych jest w stanie logicznie wnioskować o innych faktach. Takie rozwiązanie jest wykorzystywane m.in. w diagnostyce uszkodzeń maszyn, wskazując na miejsce i rodzaj awarii bazując na wcześniej zebranych informacjach [1],
- System automatycznie dowodzący twierdzeń – sprawdza możliwość udowodnienia założonej tezy na bazie zapisanej wiedzy w postaci stwierdzeń i formuł. Jest on wykorzystywany głównie w badaniach matematycznych [1].

Kolejnym aspektem ludzkiego intelektu jest umiejętność dokonywania najlepszych wyborów w przekroju wszystkich potencjalnych rozwiązań. W trakcie gry w szachy (notabene gry, która często bywa ważnym wzorcem do tworzenia systemów sztucznej inteligencji) każdy z uczestników ma wiele możliwości ruchu. Aby odnieść zwycięstwo, należy wybrać najlepszą z opcji uwzględniając przy tym wiele aspektów, jak np. zmniejszenie możliwości lub wręcz wymuszanie ruchu przeciwnika, a przede wszystkim przewidywanie sekwencji manewrów. Najbardziej znane algorytmy, przeszukujące przestrzeń możliwych wyborów, to [1]:

- Algorytmy genetyczne (ewolucyjne) – złożone są z populacji osobników (potencjalnych wyborów). Po reprodukcji w każdym z kolejnych pokoleń pozostają osoby, które wg wybranego kryterium są najlepsze. Często algorytmy genetyczne wykorzystywane są w celach przewidywania ruchów na giełdzie [1],
- Algorytmy wyboru ruchu w grze – polegają na dokonywaniu wyboru na podstawie intelligentnej analizy przestrzeni sekwencji ruchów wykonywanych przez oponenta. Najlepszym przykładem jest wcześniej opisana gra w szachy, w której to w 1997r. po raz pierwszy program komputerowy Deep Blue pokonał najlepszego gracza w historii – Garrię Kasparowa. W dzisiejszych czasach poziom arcymistrzowski potrafi uzyskać gry nawet w telefonach komórkowych [1],
- Algorytmy wyboru sekwencji – podobnie jak w algorytmach wyboru ruchu w grze, są w stanie dokonać optymalnych sekwencji wyborów. Wykorzystywane są m.in. w wyznaczaniu trajektorii ruchu robota, w otoczeniu pełnym przeszkód [1].

Kolejnym z podstawowych działań sztucznej inteligencji są systemy rozmyte. Często człowiek nie jest w stanie podać konkretnej odpowiedzi na dane zadanie, opisując coś, jako „duże”, „niewiele”, „ciepło” itp. Podobne odpowiedzi można uzyskać za pomocą systemów rozmytych. Dwoma podstawowymi terminami z tej dziedziny są: pojęcie rozmyte oraz reguły rozmyte. Te pierwsze definiuje funkcję przynależności wartości do danego wyrażenia. Reguły rozmyte wnioskują na podstawie obiektów określonych pojęciami rozmytymi. Algorytmy z tej dziedziny wykorzystywane są w celach regulacji pewnych procesów oraz detekcji awarii systemów [1].

Często na pozór proste czynności dla człowieka nie są w pełni osiągalne dla konstruowanych przez ludzi maszyn i systemów. Jednym z powodów jest brak możliwości opisania algorytmami pewnych procesów myślowych. Przykładem jest umiejętność rozpoznawania obrazów, której nie można w stanie wprost wyrazić pewnymi zasadami, gdyż nie ma pełnej wiedzy o tym, jak ona przebiega w mózgu człowieka. Dodatkowym problemem jest to, że nowe obrazy można odkrywać każdego dnia. Mimo, że są to nieznane wcześniej informacje, można m.in. zidentyfikować znajdujące się na nich obiekty. Mózg ludzki jest w stanie adaptować i generować na bieżąco nowe „zasady” umożliwiające rozpoznawanie całych obrazów, czy też ich szczegółów. Uczenie maszynowe jest dziedziną sztucznej inteligencji, której zadaniem jest uaktualnianie reguł do zmieniającego się otoczenia. Zamiast tworzyć systemy rozwijające poszczególne zadania, gromadzone są dane opisujące prognozowane wyniki i wykorzystywane są w celach treningowych konkretnych algorytmów uczenia maszynowego. Systemy uczące się korzystają z technik automatycznego pozyskiwania wiedzy. Mechanizmy uczenia maszynowego często wykorzystywane są w optymalizacji pracy sterowników, tworzenia systemów rozpoznawania głosów, czy też obrazów oraz eksploracji zasobów Internetu. Jako że ta dziedzina sztucznej inteligencji jest wykorzystana przez Autora w większej części jego pracy magisterskiej, zostanie jej poświęcony osobny podrozdział [1, 4, 5].

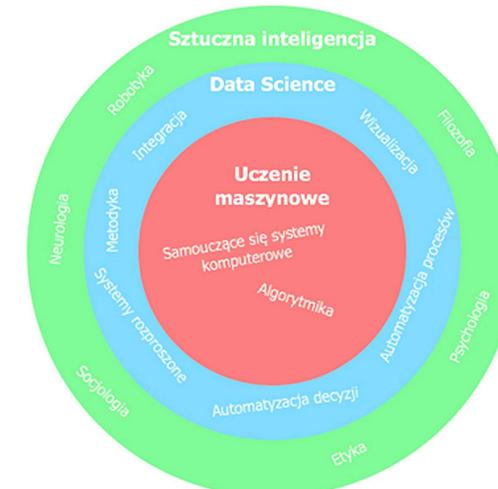
#### 1.4. Data Science i uczenie maszynowe w celach predykcji

Jak już Autor wspomniał we wstępie rozdziału, możliwość przewidywania pewnych zdań może mieć ogromny wpływ na poprawne planowanie kolejnych ruchów. Przykładowo, sprzedawca rowerów z doświadczenia posiada już wiedzę o tym, że sprzedaż jest najwyższa w okresie wiosennym. Wiadomym jest, że przed tym czasem musi zaoferować się w większą liczbę rowerów. Aby zminimalizować zbędne koszty i zmaksymalizować zysk należałoby zakupić dokładnie tyle sprzętu, ile uda się sprzedać. Ważną wiadomością w celu odpowiedniego zamówienia jest zestawienie sprzedaży z poprzednich lat. Jednakże wiele czynników ma wpływ na optymalny dobór ilości rowerów, które mogą mieć odniesienia do zebranych danych. Może to być m.in. prognoza pogody (np. przedłużająca się zima), nowe punkty sprzedaży w okolicy, dostępne promocje w innych sklepach, a także wprowadzenie innowacyjnych modeli rowerów. W tym i innych przykładach pomocne może być stworzenie modeli predykcyjnych [5].

##### 1.4.1. Data Science

Trzonem systemów przewidujących jest baza danych. Data Science jest relatywnie nową dziedziną informatyki, która zakłada automatyczne przetwarzanie i analizę informacji z wykorzystaniem komputerów. W czasach, gdy liczba cyfrowych danych rośnie o ok. 30% rocznie, zawód taki jak *data scientist* jest jednym z najbardziej poszukiwanych na rynku pracy, a także bardzo dobrze płatnym. Datyfikacja (gromadzenie danych) tyczy się każdego z nas. Poprzez jakiekolwiek działanie na komputerze, smartfonie, czy

innym urządzeniu połączonym z Internetem, generują się informacje, które są zapisywane i wykorzystywane przez zainteresowane branże. Coraz więcej instytucji dostrzega „ukrytą moc” danych, dzięki którym mogą planować z dużą dokładnością kolejne kroki w zarządzaniu nie narażając się na utratę finansów, klientów itp. Praktycznie każda większa korporacja posiada w swoich szeregach specjalne działy zajmujące się zbieraniem i przechowywaniem danych, na podstawie, których następnie formułowane są hipotezy. Częstym błędem jest mylenie pojęć „Data Science” i „uczenie maszynowe”, w których rozróżnieniu może pomóc rysunek 1.1. [5].



Rysunek 1.1. Sztuczna inteligencja, Data Science i uczenie maszynowe – działy  
Źródło: opracowanie własne na podstawie [7]

Jak widać na przedstawionej ilustracji, Data Science można określić jako dziedzinę sztucznej inteligencji, która łączy się z wieloma dziedzinami nauki i nie ogranicza się tylko do branży technologicznej. Najczęściej *data scientist* (zwany także „mistrzem danych”) może zostać osoba z dużą wiedzą o statystyce, matematyce, informatyce, wizualizacji danych, a także wiedzą ukierunkowaną w badanym sektorze, czym odróżnia się od statystyka, czy też analityka danych. Nie istnieje ścisła definicja Data Science. Można to sformułować określić jako interdyscyplinarną dziedzinę nauki, zajmującą się pozyskiwaniem informacji, ich analizą i wizualizacją oraz wnioskowaniem bazując na danych, korzystając z metod: eksploracji danych, statystycznych, uczenia maszynowego i analizy predykcyjnej [5, 8].

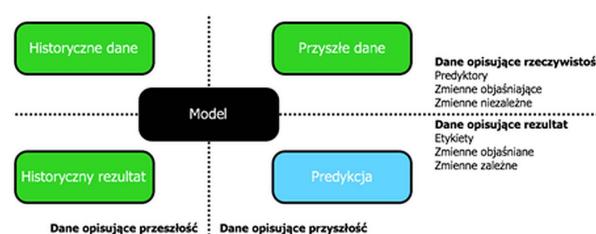
#### 1.4.2. Metody uczenia maszynowego

Stworzenie modelu przewidującego jest przykładem systemu, który jest praktycznie niemożliwy do skonstruowania tak, aby działał on optymalnie za pierwszym razem. Idea uczenia maszynowego oparta jest na ciągłym, zautomatyzowanym treningu sterownika danego modelu. Tak jak w przypadku człowieka, który na początku życia nie potrafi wykonać wielu czynności (m.in. chodzić), ale opanowuje tą umiejętność w miarę ciągłej praktyki. Uczenie maszynowe wyręcza nas w analizie olbrzymich baz danych, do których często dodawane i aktualizowane są kolejne informacje. Jest to istotna dziedzina informatyki, która odgrywa coraz większą rolę w codziennym życiu [1, 6].

Wyróżnia się trzy podstawowe metody uczenia maszynowego:

- uczenie nadzorowane,
- uczenie nienadzorowane,
- uczenie poprzez wzmacnianie.

W pierwszej z nich, człon „nadzorowany” odnosi się do faktu, że część pożądanego sygnałów wyjściowych (etykiet) jest znana. W uczeniu nadzorowanym, systemowi dostarcza się dwóch informacji: dane wejściowe oraz dane wyjściowe. W tej metodzie celem mechanizmu jest wnioskowanie oczekiwanych odpowiedzi ucząc się i tworząc pewną funkcję odwzorowującą zależność pomiędzy zadanymi zbiorami. Rysunek 1.2 prezentuje cztery podstawowe bloki związane z informacjami wykorzystywanymi w uczeniu nadzorowanym z wyróżnionymi podziałami na czas oraz typ danych.



Rysunek 1.2. Uczenie nadzorowane – podział na informacje

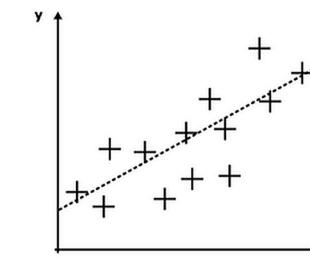
Źródło: opracowanie własne na podstawie [7]

Dodatkowo zielonym kolorem zaznaczone są elementy, wymagające czennego udziału projektanta systemu predykcyjnego (m.in. przygotowanie odpowiedniego zbioru informacji). Zadaniem modelu, poprzez analizę historycznych danych i rezultatów, jest wywnioskowanie predykcji po zaimplementowaniu mu nowych danych. Ilustracja 1.3 przedstawia przykład opisujący zależność danych na podstawie rysunku 1.2.

Rysunek 1.3. Uczenie nadzorowane – przykład podziału na informacje

Źródło: na podstawie [7]

Często uczenie nadzorowane wykorzystywane jest w celu klasyfikacji, dla którego typowym przykładem jest filtr antyspamowy w skrzynkach elektronicznych. Poprzez dostarczanie informacji systemowi oznaczając niechcianą wiadomość trenuje się dany model, który z czasem jest w stanie rozróżnić wiadomości wartościowe od tych niepożądanych. Filtr antyspamowy jest przykładem klasyfikacji binarnej, gdy określa przynależność tylko do dwóch klas. Metody predykcyjne wykorzystujące algorytmy uczenia nadzorowanego wykorzystywane są także do tzw. klasyfikacji wielokryterialnej, kiedy to przypisywane są etykiety do nowych klas. Sztandarowym przykładem takiej klasyfikacji jest m.in. rozpoznanie odręcznego pisma, gdzie występuje tyle klas, ile jest możliwych do zidentyfikowania znaków. Poniższa ilustracja (rysunek 1.4) przedstawia ogólny przykład nauczania nadzorowanego, gdzie przerywana linia reprezentuje regułę binarnej klasyfikacji wartości (krzyżyki i kółka) pomiędzy dwoma zbiorami ( $x_1, x_2$ ).

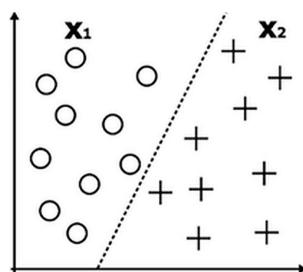


Rysunek 1.4. Przykład klasyfikacji binarnej

Źródło: opracowanie własne na podstawie [6]

Należy jednak podkreślić, że rysunek 1.4 ilustruje prosty przykład w przypadku dwóch wymiarów i łatwo zauważalnej granicy pomiędzy wartościami. W rzeczywistości często analizie poddana jest większa ilość informacji w wielu wymiarach [6].

W metodzie nadzorowanej uczenia maszynowego poza klasyfikacją wyróżnia się tzw. *analizę regresji*. Metoda ta polega na prognozowaniu wyników ciągłych. Celem analizy regresyjnej jest znalezienie relacji pomiędzy danymi zmiennymi objaśniającymi (element prognostyczny) oraz ciągłą zmienną objaśnianą (element prognostowany). Prostym przykładem jest model przewidujący wyniki z pewnego egzaminu w przeszłości, bazujący na zapisanych rezultatach poprzednich testów. Zaimplementowana baza danych po raz kolejny służy do trenowania systemu. Rysunek 1.5 ilustruje przykładowe wyniki [6].



Rysunek 1.5. Przykład regresji liniowej

Źródło: opracowanie własne na podstawie [6]

Wykorzystując zmienne objaśniające (osi x) i zmienne objaśniane (osi y) wyznaczono prostą przebiegającą w jak najmniejszej odległości od danych punktów (wyniki z poprzednich egzaminów). Najczęściej w celu określenia tejże linii stosuje się metodę najmniejszych kwadratów. Aby prognozować wyniki wystarczy skorzystać z punktu przecięcia prostej z osiami współrzędnych oraz jej kąta nachylenia [6].

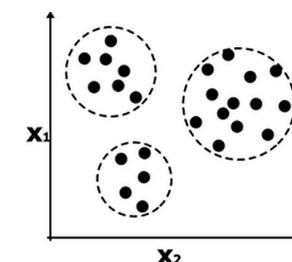
Kolejnym rodzajem uczenia maszynowego jest tzw. uczenie nienadzorowane. Posiadając już wiedzę o uczeniu nadzorowanym, można wydedukować, że podstawową różnicą pomiędzy tymi metodami jest brak znajomości właściwej odpowiedzi. Wykorzystywane w tym przypadku bazy danych zawierają nieoznakowane informacje wejściowe często o nieznanej strukturze. Opisywana metoda jest natomiast niejednokrotnie elementem wstępny uczenia nadzorowanego. Rysunek 1.6 przedstawia podstawowe bloki związane z informacjami w uczeniu nienadzorowanym.



Rysunek 1.6. Uczenie nienadzorowane – podział na informacje

Źródło: opracowanie własne na podstawie [7]

Uczenie nienadzorowane używane jest często w celu grupowania (zwane też klasteryzacją, bądź analizą skupień). Pozwala to uzyskać podstawowe informacje o danych, które są głównym trzonem uczenia maszynowego. Poprzez tworzenie nowych podzbiorów eksponowane są podobieństwa i różnice pomiędzy danymi. Daje to wiele możliwości, m.in. w przypadku grupowania klientów ze względu na ich zainteresowania, umożliwia to planowanie konkretnej akcji marketingowej celującej w żądaną grupę odbiorców. Rysunek 1.7 ilustruje prosty przykład klasteryzacji na trzy grupy na podstawie podobieństw i różnic ( $x_1, x_2$ ) [6].



Rysunek 1.7. Przykład grupowania za pomocą klasteryzacji

Źródło: opracowanie własne na podstawie [6]

Często pracuje się z potężnymi bazami danych, których informacje są wielowymiarowe (każda dana przypisana jest do dużej ilości wartości pomiarowych). Uzyskanie żądanego wyniku w tym przypadku jest utrudnione ze względu na ograniczoną pojemność pamięci, czy też efektywność obliczeniową funkcji uczenia maszynowego. Tak ogromne bazy danych zawierają także informacje, które są zbędne w żądanej predykcji. Są to tzw. szумy, które mogą negatywnie wpływać na wyniki, a także znacznie wydłużać czas obliczeń (ze względu na dodatkową ilość niechcianych danych, które algorytm także musi uwzględnić). Kolejna dziedzina uczenia nienadzorowanego – redukcja wymiarowości, jest narzędziem, które umożliwia kompresję ilości danych zachowując najważniejsze informacje. Poza ewidentną optymalizacją działania algorytmów uczenia maszynowego, redukcja wymiarowości wykorzystywana jest także w celu lepszej wizualizacji danych. Wielowymiarowość cech jest ciężko wyobrażalna dla człowieka, a te funkcje umożliwiają kompresję np. wykresu trójwymiarowego do dwuwymiarowej przestrzeni (przykładem tego może być rysunek 1.7).

Trzecim rodzajem uczenia maszynowego jest *uczenie przez wzmacnianie*. Jego celem jest poprawa skuteczności systemu, składającego się z regulatora i agenta, poprzez interakcje ze środowiskiem. Uczenie przez wzmacnianie można częściowo powiązać z uczeniem nadzorowanym, gdyż omawiana metoda także korzysta ze sprzężenia zwrotnego informacji, zawierającego tzw. sygnał nagrody. Dostarcza on do systemu dane dotyczące środowiska modelu. Nie są to wzorcowe etykiety, jak w przypadku uczenia nadzorowanego, lecz wartości określające skuteczność pomiaru działania

przez funkcję nagrody. Obserwując funkcjonowanie środowiska regulator trenuje szereg czynności w celu zwiększenia nagrody, co ilustruje rysunek 1.8 [6].



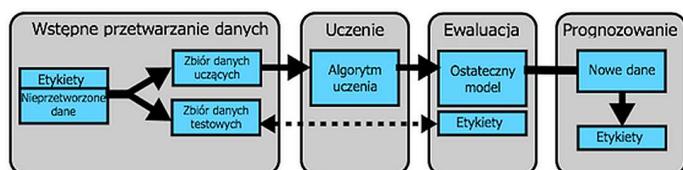
Rysunek 1.8. Schemat oddziaływania w metodzie uczenia przez wzmacnianie

Źródło: opracowanie własne na podstawie [6]

Przykładem metody uczenia przez wzmacnianie jest silnik zautomatyzowanej gry w szachy. W tym przypadku komputer dobiera swoje kolejne ruchy na podstawie obserwacji aktualnego stanu na planszy, a nagrodą może być zwycięstwo, jak i porażka na koniec partii.

#### 1.4.3. Etapy tworzenia systemów uczenia maszynowego w celu predykcji

Model predykcyjny składa się z kilku etapów, który ogólny zarys przedstawia rysunek 1.9.

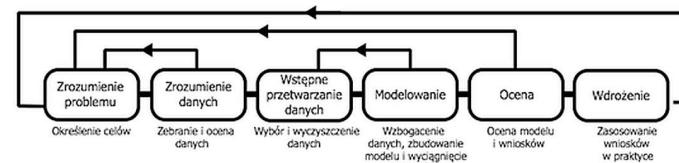


Rysunek 1.9. Schemat etapów modelu predykcyjnego

Źródło: opracowanie własne na podstawie: [6]

Niniejszy podrozdział skupia się na opisaniu poszczególnych etapów i napotkanych problemach w trakcie pracy nad modelem predykcyjnym.

Należy wspomnieć, że nauki Data Science proponują szereg metod pracy nad konstrukcją systemu przewidującego. Najpopularniejszą metodą jest tzw. CRISP-DM (ang. Cross Industry Standard Process for Data Mining), która opisuje etapy pracy data mining umożliwiając w określonych miejscach korekcję błędów. Schemat metodyki dla powyższej normy przedstawiono na rysunku 1.10.



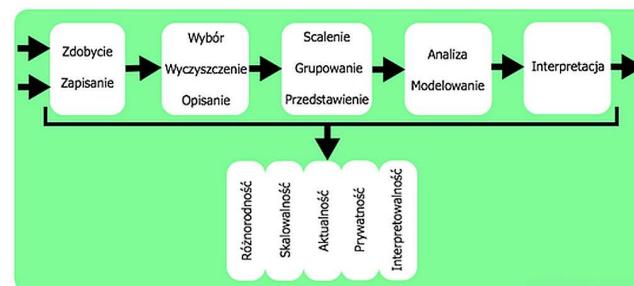
Rysunek 1.10. Schemat metodyki CRISP-DM

Źródło: opracowanie własne na podstawie [5]

Na podstawie powyższego rysunku można wywnioskować, że etap modelowania, który wydawałby się najważniejszym, jest jednym z końcowych kroków eksploracji danych. Często istotnym popełnianym błędem jest pominięcie postawienia konkretnych pytań przed przystąpieniem do zadania, a tym samym brak zrozumienia problemu. Kolejnym krokiem jest wstępna analiza danych, definiująca przydatność poszczególnych informacji w celu osiągnięcia założonego zadania.

#### 1.4.3.1. Wstępne przetwarzanie danych

Rzadko, kiedy się zdarza, że analizowana baza danych zawiera tylko ustrukturyzowane dane bez żadnych szumów. Dlatego bardzo ważnym etapem tworzenia modelu predykcyjnego jest wstępne przetwarzanie danych. W praktyce czas poświęcony ocenie i przygotowaniu informacji treningowych zajmuje do 80% całkowitego czasu modelowania systemu przewidującego. *Eksploracja danych* (ang. data mining) dzieli się na kilka etapów, podczas których można napotkać wiele problemów, co przedstawiono na rysunku 1.11.



Rysunek 1.11. Etapy eksploracji danych oraz związane z nimi problemy

Źródło: opracowanie własne na podstawie [5]

Ważnym aspektem konstruowania modelu predykcyjnego jest wiadomość o pochodzeniu analizowanych informacji. Często projektanci takiego systemu korzystają

z udostępnionych baz danych np. MySQL, PostgreSQL, czy też SQL Server. Przedstawione dane powinny być w postaci tabelarycznej, aby istniała możliwość prostej wstępnej analizy. Wiersze takiej tabeli opisują obserwacje, a kolumny ich atrybuty. Jeżeli schemu tworzy się taką bazę, należy podjąć decyzję o formacie zapisu pliku. Ze względu na konstrukcję najpopularniejszych platform uczenia maszynowego zaleca się korzystanie z formatów: CSV (ang. *Comma-Separated Values*), TSV (ang. *Tab-Separated Values*) oraz JSON (ang. *JavaScript Object Notation*). Po identyfikacji źródła danych i sprawdzeniu aktualności badanych wartości, a także zapisaniu pliku w odpowiednim formacie, należy określić typ poszczególnych danych. Można wyróżnić wartości liczbowe, tekstowe, czy też mieszane, ale z punktu widzenia *Data Science* zwraca się szczególną uwagę na zmienne kategoryczne (jakościowe) i numeryczne (ilościowe). Rysunek 1.12 przedstawia przykład tabeli zapisanej w formacie CSV ze wskazanymi sformułowaniami wyjaśnionymi do tej pory w tym podrozdziale.

Osoba,Płeć,Wiek,Masa,Wzrost,BMI,Kategoria	Nagiówka opisującej kolejne atrybuty
Janek,M,29,78,1.85,22.79,Prawidłowa	Zarejestrowane obserwacje
Magda,K,34,51,1.67,18.29,Niedowaga	

Rysunek 1.12. Przykładowa baza danych zapisana w formacie CSV.

Źródło: opracowanie własne na podstawie [5]

Na podstawie powyższej ilustracji można stwierdzić, że wartości: „Osoba”, „Płeć” i „Kategoria” są przykładami danych tekstowych, a „Wiek”, „Masa”, „Wzrost” i „BMI” przedstawiają dane liczbowe. Wartościami danych numerycznych, są wyłącznie liczby, z wykorzystaniem, których możliwe jest przeprowadzanie operacji arytmetycznych. W tej kategorii wyróżnia się tzw. zmienne numeryczne ciągłe, których zbiór wartości jest nieprzeliczalny oraz zmienne numeryczne dyskretnie, gdzie liczba wartości jest skończona. Zmiennymi kategorycznymi mogą być zarówno liczby jak i dane innych typów. Charakteryzuje je to, że zbiór wartości jest zawsze ograniczony, oraz że określenie odległości między informacjami jest możliwe tylko na podstawie przyjętego modelu (np. kolory na podstawie ustalonej palety barw). Prawidłowy opis i określenie typu każdej z danych jest niezbędnym etapem w procesie projektowania modelu predykcyjnego umożliwiającym prawidłową interpretację danych poprzez wyeksponowanie ich różnorodności [5, 7].

Skalowalność jest często zapominanym, aczkolwiek bardzo ważnym etapem wstępnego przetwarzania danych. Należy jednak podkreślić, że nie jest to niezbędny krok w każdym rodzaju modelowania, np. algorytmy lasów losowych potrafią skutecznie zniwelować błąd niepoprawnej skali wartości. Dla przykładu przyjęto, na podstawie rysunku 1.12, że kategoria „Wiek” może przyjmować wartości od 10 do 80 lat, zaś kategoria „Wzrost” zawiera dane w zakresie 1,3 m + 2,05 m. W dalszej kolejności modelu, w przypadku korzystania np. z funkcji sumy kwadratów błędów (m.in. w celach klasyfikacji), cechy te, mogłyby być różnie interpretowane ze względu na różny czas opty-

malizacji wag, z powodu różnicy błędów. Zadaniem projektanta systemu predykcyjnego jest decyzja o skorzystaniu bądź pominięciu etapu skalowania danych. Wyróżnia się dwie metody sprowadzania danych do jednakowej skali: normalizacja i standaryzacja. Pierwsza z nich polega na przedstawieniu wartości w zakresie [0, 1]. Operacja ta polega na przeskalowaniu każdej kolumny cech w odniesieniu do wartości krańcowych, co można przedstawić wzorem:

$$x_{norm}^{(i)} = \frac{x^i - x_{min}}{x_{max} - x_{min}} \quad (1.1)$$

gdzie:

$x_{norm}^{(i)}$  – nowa wartość,

$x^i$  – próbka,

$x_{min}$  – minimalna wartość próbki w danej kolumnie,

$x_{max}$  – największa wartość.

Chociaż normalizacja jest bardzo często wykorzystywana metodą w procesie skalowalności danych, to standaryzacja może okazać się praktyczniejszym rozwiązaniem. Wyznacza ona, bowiem tzw. rozkład normalny, gdzie średnią wartością jest „0” przy odchyleniu standardowym „1”. Proces standaryzacji prezentuje się wzorem:

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x} \quad (1.2)$$

gdzie zmienna  $\mu_x$  określa średnią wartość z danej kolumny, a  $\sigma_x$ , jest związany z odchyleniem standardowym tejże kategorii. Przykładowe wyniki normalizacji i standaryzacji przedstawione są w poniższej tabeli [6].

Tabela 1.1. Przykładowe wyniki procesu normalizacji i standaryzacji

Źródło: opracowanie własne na podstawie [6]

DANE WEJŚCIOWE	DANE NORMALIZOWANE	DANE STANDARYZOWANE
0,0	0,0	-1,336876
1,0	0,2	-0,802637
2,0	0,4	-0,253721
3,0	0,6	0,253721
4,0	0,8	0,802637
5,0	1,0	1,336876

Prywatność danych (zwana też wrażliwością danych), związana jest bezpośrednio z prawem i przepisami sformalizowanymi, a także niesformalizowanymi. Istnieje szereg ustaw opisujących zasady ochrony danych osobowych (np. RODO), ale wrażliwość danych tyczy się także m.in. tajemnic zawodowych (lekarskiej, bankowej itd.). Do tego typu informacji mogą należeć: m.in.: PESEL, numer karty kredytowej, stan zdrowia, adres zamieszkania. Należy zwrócić jednak uwagę na fakt, że nielegalne rozprze-

strzenianie danych może odbywać się także pośrednio, bazując na zebranych informacjach. Jako przykład, posłuży tabela 1.2 [7].

**Tabela 1.2.** Przykład danych w celu opisu problemu prywatności danych

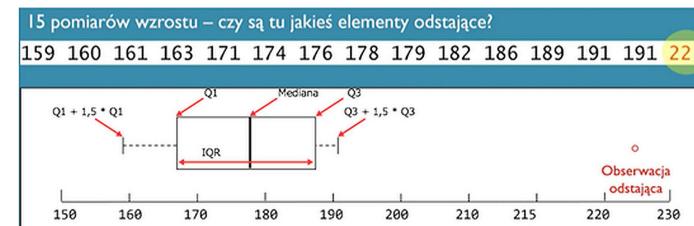
Źródło: opracowanie własne na podstawie [7]

WAGA (KG)	WZROST (M)
82	1,92
78	1,83
39	1,46
103	2,01
68	1,76

Na podstawie wartości zawartych w tabeli 1.2. nie ma możliwości odpowiedzenia na pytanie: który z badanych uczestników testu to Jan Kowalski? Wydawałoby się, że prywatność danych osobowych w tym przykładzie jest niezagrożona. Ażkolwiek posiadając dodatkową информацию i formułując trochę inaczej to samo pytanie, np. który z badanych uczestników testu to Jan Kowalski – lat 11? Łatwo jest wydedukować, że z dużym prawdopodobieństwem jest to osoba, której dane znajdują się na trzeciej pozycji w tabeli 1.2. Jako przykład złamania pośrednio zasad wrażliwości danych można przytoczyć słynną aferę z udziałem sieci „Netflix” i „IMDB”. Pierwsza z firm upubliczniała zanonimizowane dane swoich użytkowników, zawierające informacje odnośnie preferencji oglądanych filmów. Porównując te wartości do powszechnie dostępnych danych firmy „IMDB” w łatwy sposób można było zidentyfikować imię, nazwisko, a nawet adres IP komputerów każdego z ankieterów firmy „Netflix”. Jest to kolejny przykład uświadamiający, jaką potężną moc skrywają w sobie informacje, a także przykład na to, jak ostrożnym należy być przy tworzeniu i udostępnianiu baz danych [7].

Interpretowalność modeli jest aspektem, który wybiega poza standardową analizę wyników proguzy. Pozwala ona na określenie jakości modelu odpowiadając na pytanie: dlaczego model dostarczył taką, a nie inną odpowiedź. W etapie interpretowalności nie wykorzystuje się żadnych wzorów ani technik, a bazuje się na ciekawości i wiedzy ludzkiej. Człowiek z natury lubi wiedzieć, dlaczego coś działa w dany sposób, zbierając coraz większą ilość informacji z otaczającego świata. Stwierdza się także, że poprawna interpretacja modeli buduje społeczne zaufanie do technologii, gdyż oferuje nam często niepodważalne dowody na tezy postawione przed tworzeniem projektu predykcyjnego. Istnieją przykłady, gdzie interpretowalność jest narzucona przez pewne branżowe wymogi. Coraz częściej można spotkać się z sytuacją, gdzie w medycynie wykorzystuje się nowoczesne systemy w celu zidentyfikowania choroby. Jednakże konkretne stwierdzenie przypadłości i podjęcie odpowiedniego leczenia byłoby niemożliwe bez interpretacji otrzymanych wyników przez wykwalifikowanego lekarza. Innym przykładem może być branża bankowa, gdzie interpretowalność wyników, często jest podstawą i dowodem na udzielenie, bądź odmowy udzielenia kredytu [7].

Wstępne przetwarzanie danych, poza opisanymi wyżej zadaniami, jest etapem, w którym projektant systemu musi sprostać błędem technicznym baz danych. Często brakuje poszczególnych danych, co może być spowodowane m.in. usterką urządzenia rejestrującego, brakiem udzielonej odpowiedzi bądź też błędem operatora. Konstruktor modelu prognostycznego ma kilka możliwości zareagowania na opisywanego problem: usunąć wszystkie zaobserwowane wartości związane z brakującą informacją, automatycznie zastąpić pustą komórkę wg wybranego wzorca (np. mediana, średnia, określona wartość) lub zignorować błąd, jeśli nie ma to znaczącego wpływu na kolejne etapy modelu. Innymi możliwymi usterkami w bazie danych mogą być tzw. elementy odstające (ang. *outlier*). Przykładowo może być to zbiór zawierający pomiary zewnętrznej temperatury powietrza w okresie zimowym. Pośród wszystkich wartości, można wyróżnić pojedyncze zapisy, które ewidentnie nie pasują do określonej pory roku, bądź są nienaturalne w naszym środowisku (np. 28 °C zimą, czy też -224 °C). Elementy odstające mogą być oznaką słabej jakości danych, a poprzez ich analizę można zlokalizować i zidentyfikować błąd powodujący takie zapisy. *Outliery* nie muszą być automatycznie uznawane za usterkę i eliminowane bądź zastępowane w bazie danych. W zależności od przeprowadzanych badań mogą one być oznakowane, jako anomalie i być bardzo cenne w całym procesie modelu prognostycznego. Dlatego też projektant systemu musi analizować indywidualnie każdy element odstający dobierając odpowiednią strategię (np. zastępowanie wartością średnią, usuwanie). Należy nadmienić w tym miejscu, że bardzo pomocnym narzędziem (nie tylko w wykrywaniu *outlierów*) w Data Science są tzw. wykresy pudełkowe, których przykład jest przedstawiony na rysunku 1.13 [7].



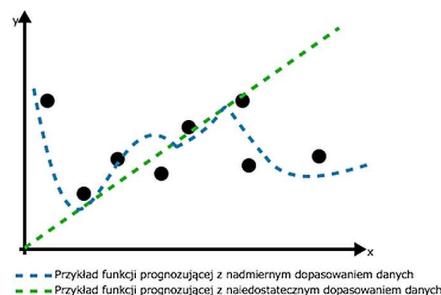
**Rysunek 1.13.** Tworzenie wykresu pudełkowego

Źródło: opracowanie własne na podstawie [7]

W celu utworzenia wykresu pudełkowego, w pierwszej kolejności należy wyliczyć medianę z wartością zawartą w bazie danych. Następnie wyznacza się kwartyle (na rysunku 1.13: „Q1” i „Q3”), które są granicami tzw. przedziału międzykwartylowego (na rysunku 1.13: „IQR”). Ostatecznie, wyliczając i zaznaczając tzw. wąsy, zarysuje się ostateczny kształt wykresu pudełkowego. Wszystkie wartości wybiegające poza jego obszar mogą oznaczać błędą informację bądź anomalię [7].

Ważnym aspektem w tworzeniu modeli predykcyjnych jest także wielkość użytej bazy danych. Teoretycznie ocenę poprawnej ilości informacji należałoby wykonać przed

przystąpieniem do kolejnych etapów, ale często ten błąd można zauważać dopiero przy uzyskaniu pierwszych wyników, a więc na końcu projektu. Zbyt mała ilość danych może spowodować uproszczenie modelu, nie oddając tym samym rzeczywistości. W przeciwnym przypadku, gdy tych wartości jest za wiele, można natknąć się na problemy: z ograniczeniami sprzętu przechowującego dane, z bardzo wydłużonym czasem budowy modelu i wykonywanych przez niego operacji oraz ze zjawiskiem *nadmiernego dopasowania* (ang. *overfitting*). Przykład niedostatecznego i nadmiernego dopasowania przedstawia rysunek 1.14 [7].



Rysunek 1.14. Przykład zjawiska *underfittingu* i *overfittingu*

Źródło: opracowanie własne na podstawie [7]

W zależności od przypadku badanych danych, gorszym zjawiskiem może być niedostateczne lub nadmierne dopasowanie. Można to w łatwy sposób stwierdzić, dodając nowe informacje do zbioru danych (nowe punkty na przykładowym wykresie przedstawionym w rysunku 1.14) i obliczyć ich odległość od danej funkcji charakteryzującej się *overfittingiem* lub *underfittingiem*. W przypadku niedostatecznego dopasowania, rozwiązaniem problemu może być wykorzystanie innego algorytmu, intelligentne powiększenie ilości informacji w zbiorze danych lub dodanie nowych informacji do opisu obserwacji. Gdy model predykcyjny jest zbyt szczegółowy, można skorzystać z szeregu technik optymalizacyjnych, często udostępnionych w danych programach do modelowania systemów uczenia maszynowego. W trakcie manipulacji ilością danych należy zwracać szczególną uwagę na zachowanie reprezentatywności całego zbioru [7].

Niniejszy podrozdział przedstawia m.in. złożoność zadań związanych z etapem wstępnej analizy i przetwarzania danych. Należy nadmienić, że opisano tylko najczęstsze problemy napotykane w trakcie przygotowywania informacji do modelu predykcyjnego. Wiele współczesnych systemów uczenia maszynowego posiada gotowe moduły, których zadaniem jest wprowadzanie żądanego modyfikacji w bazie danych. W przypadku braku zautomatyzowanych systemów korekcji wartości, istnieje możliwość indywidualnego stworzenia takich opcji, najczęściej korzystając z języków programowania t.j. Python i R [6, 7].

Ostatnim krokiem w etapie wstępnego przetwarzania danych jest podział informacji na zbiór danych uczących i testowych. Jest to relatywnie prosta czynność, aczkolwiek niezbędna i jedna z najważniejszych w procesie modelowania systemu predykcyjnego. Etap ten polega na losowym rozdzieleniu wykorzystanego zbioru danych, wyznaczając stosunek przynależności informacji do danego podzbioru. Oczywiście istnieje jeszcze szereg parametrów wpływających na podział, wśród których do najważniejszych można zaliczyć tzw. stratyfikację, która polega na segregacji danych z uwzględnieniem proporcji pomiędzy klasami. Zbiór danych uczących służy do analizy i przeprowadzenia szeregu operacji algorytmizacji, w celu zidentyfikowania zależności między danymi, ostatecznie wyznaczając odpowiednią funkcję m.in. klasyfikującą, bądź progностyczną. Jakość tegoż wzoru oceniana jest na podstawie zbioru danych testowych w fazie ewaluacji modelu (przedstawia to rysunek 1.9) [6].

#### 1.4.3.2. Uczenie

Etap uczenia polega na doborze odpowiednich algorytmów sztucznej inteligencji. Celem jest wynosowanie na podstawie zbioru danych uczących pewnej reguły, która będzie spełniała założone zadanie. W niniejszym podrozdziale, autor pracy postara się opisać podstawowe i najczęściej wykorzystywane algorytmy sztucznej inteligencji w procesie uczenia maszynowego.

W zależności od założonego celu, wyróżnia się cztery podstawowe grupy algorytmów, których zadaniem może być: detekcja anomalii, klasteryzacja (grupowanie), predykcja kategorii oraz predykcja wartości. Rysunek 1.15 przedstawia ten podział (w języku angielskim), wraz z nazwami konkretnych algorytmów wg firmy Microsoft. Ze względu na ilość informacji zawartych na rysunku, jest on przedstawiony na całości kolejnej strony.

W celu wybrania odpowiedniego algorytmu do założonego zadania należy rozpatrzyć kilka zagadnień. Projektant modelu musi zdecydować o dokładności wyniku. Nie zawsze nie jest niezbędnym uzyskanie konkretnej wartości, a wystarczy odpowiedź z pewnym przybliżeniem. Pozwala to na znaczone skrócenie czasu obliczeń, a także jest to naturalny sposób uniknięcia problemu nadmiernego dopasowania wartości. Bezpśrednio z dokładnością modelu związany jest właśnie kolejny aspekt doboru odpowiedniego algorytmu, czyli czas szkolenia. Zależny jest on od użytej funkcji a także ilości analizowanych danych. Czas uczenia systemu może waahać się od paru sekund do wielu godzin. Następnie należy zdecydować o liniowości wyniku [5, 7].



Rysunek 1.15. Microsoft Azure Machine Learning: Algorithm Cheat Sheet  
 Źródło: na podstawie [9]

Rysunek 1.14 przedstawia przykład liniowej i nieliniowej odpowiedzi. Decyzja ta ma znaczący wpływ na dokładność wyników i czas nauczania modelu, dlatego musi być podjęta po głębszej analizie informacji zawartych w bazie danych i sformułowaniu żądanej odpowiedzi. W trakcie tworzenia systemu uczenia maszynowego, projektant korzystając z danego modułu często ma do czynienia z doborem parametrów związanych z wybraną opcją. Wpływ na działanie danego algorytmu ma m.in. współczynnik tolerancji błędu, liczba iteracji, czy też poszczególne preferencje mające oddziaływanie na zachowanie takiej funkcji. Parametry wybranego algorytmu często są zależne od wzorów, które go opisują, dlatego potrafi to być niezwykle ciężka decyzja poprzedzona poświęceniem długiego czasu na dokładne zapoznanie się z jego działaniem (biorąc także pod uwagę ilość możliwych kombinacji). Platformy uczenia maszynowego zazwyczaj oferują podstawowe, najczęstsze ustawienia opcji algorytmów, które mogą być wystarczające dla danego modelu. Istnieją także moduły, których zadaniem jest analiza działania wybranej funkcji w różnej konfiguracji tychże parametrów, aczkolwiek znacznie zwiększa to czas całego procesu. Finalnie, należy zapoznać się ze strukturą oferowanych algorytmów, aby optymalnie je obrać. Każda z funkcji oferuje inny sposób działania, lepiej radząc sobie m.in. z czasem uczenia, wnioskowaniem dokładniejszych prognoz, czy też z większą lub mniejszą ilością danych. Tabela 1.3 obrazuje bardzo ogólnie charakterystyki poszczególnych algorytmów, udostępnionych na platformie Microsoft: Azure Machine Learning Studio [7, 9].

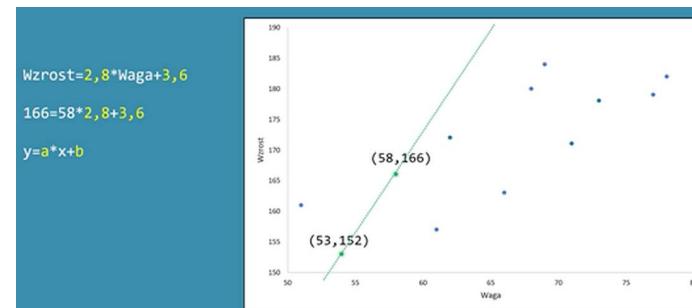
Tabela 1.3. Charakterystyki poszczególnych algorytmów  
 Źródło: opracowanie własne na podstawie: [9]

Algorytm	Dokładność	Czas szkolenia	Liniowość	Liczba parametrów	Uwagi
Klasyfikacja binarna (dwuklasowa)					
Regresja logistyczna		B. dobry	B. dobra	5	
Wzmacniane drzewo decyzyjne	B. dobra	Dobry		6	Duże zużycie pamięci
Las decyzyjny	B. dobra	Dobry		6	
Dżungla decyzyjna	B. dobra	Dobry		6	Małe zużycie pamięci
Sieć neuronowa	B. dobra			9	Możliwość dodatkowego dostosowania
Uśredniony perceptron	Dobra	Dobry	B. dobra	4	
Maszyna wektorów nośnych (SVM)		Dobry	B. dobra	5	B. dobry przy dużej ilości danych
Lokalnie głęboka SVM	Dobra			8	B. dobry przy dużej ilości danych
Maszyna punktu Bayes'a		Dobry	B. dobra	3	
Klasyfikacja wieloklasowa					
Regresja logistyczna		B. dobry	B. dobra	5	
Las decyzyjny	B. dobra	Dobry		6	
Dżungla decyzyjna	B. dobra	Dobry		6	Małe zużycie pamięci
Sieć neuronowa	B. dobra			9	Możliwość dodatkowego dostosowania

One vs all	-	-	-	-	Zależność parametrów od wybranych algorytmów klasyfikacji binarnej
<b>Regresja</b>					
Regresja liniowa		B. dobry	B. dobra	4	
Bayes'owska regresja liniowa		Dobry	B. dobra	2	
Las decyzyjny	B. dobra	Dobry		6	
Wzmocniony las decyzyjny	B. dobra	Dobry		5	B. dobry przy dużej ilości danych
Szybki las kwantylowy	B. dobra	Dobry		9	
Sieć neuronowa	B. dobra			9	Możliwość dodatkowego dostosowania B. dobry w celu prognozowania wartości
Regresja Poissona			B. dobra	5	
Regresja porządkowa					B. dobry w rankingowej predykcji
<b>Detekcja anomalii</b>					
Maszyna wektorów nośnych (SVM)	Dobra	Dobry		2	Posiada wiele funkcji
Analiza głównych składowych (PCA)		Dobry	B. dobra	3	

Pierwszą z „rodzin” algorytmów uczenia maszynowego jest tzw. *klasyfikacja dwuklasowa*. Celem modelu korzystającego ze wzorów tego zbioru jest znalezienie klasyfikatora, a więc pewnej funkcji oddzielającej poszczególne grupy wartości na dwie części. Możliwości przydzielania danych do wielu podzbiorów stwarzają algorytmy z rodziny klasyfikacji wieloklasowej. Funkcje określane, jako regresyjne wnioskują predykcję konkretnych wartości, a nie tylko przynależność do danej grupy. Ostatnim algorytmem, użytym w pracy magisterskiej Autora sa te, których zadaniem jest detekcja anomalii. Wiele algorytmów uczenia maszynowego spełnia swoją rolę w różnych zadaniach, dlatego poniżej opisano funkcje w wybranej przez autora pracy kolejności [7, 9].

Pojecie regresji określa metodę statystyczną, która umożliwia wyznaczanie funkcji określającej zmienne objaśniane na podstawie zmiennych objaśniających. Wyróżnia się szereg sposobów na opisanie żądanej reguły, stąd też istnieje wiele rodzajów regresji. Najpopularniejszym algorytmem wykorzystywanym w celu predykcji (przynależności do jednej z dwóch kategorii) jest tzw. *regresja liniowa*. Często projektanci takich systemów rozpoczynają swoje badania od przetestowania tejże metody, ze względu na małe wymagania obliczeniowe (a więc krótszy czas uczenia). Dzięki łatwej interpretowalności wyników, regresja liniowa umożliwia w prosty sposób wnioskowanie zależności między zmiennymi objaśniającymi i objaśnianymi. W przypadku niedopasowania tego algorytmu do danego modelu, regresja liniowa może stanowić dobrą bazę do dalszej optymalizacji systemu. Rysunek 1.16 przedstawia możliwy wynik i wyznaczenie funkcji w algorytmie regresji liniowej [7].



Rysunek 1.16. Przykład regresji liniowej

Źródło: na podstawie [7]

Jak przedstawia rysunek 1.16 funkcja regresji liniowej wyznaczana jest na podstawie analizy zależności między badanymi zbiorami wartości dobierając odpowiednie wartości wag. Współczynnik kierunkowy („a”), decyduje o kącie nachylenia funkcji względem osi x, a tzw. wyraz wolny („b”), określa punkt przecięcia z osią y. Najczęściej w jej wyznaczeniu korzysta się z metody najmniejszych kwadratów. Wówczas powstała linia jest szukaną predykcją, na podstawie, której można przewidzieć np. wzrost człowieka bazując na informacji o jego wagie. W odróżnieniu od powyższego przykładu, w rzeczywistości badany problem zawiera o wiele więcej danych, które nie zawsze umożliwiają wyznaczenie liniowej zależności. Ogólny zapis wzoru regresji liniowej przedstawia się następująco [7, 9]:

$$y = a_1x_1 + a_2x_2 + \dots + a_nx_n + b \quad (1.3)$$

Bardzo podobnym w działaniu algorymem jest *bayesowska regresja liniowa*. Podstawową różnicą pomiędzy opisywanymi funkcjami jest ustalanie wartości parametrów nie poprzez metodę najmniejszych kwadratów, a z wykorzystaniem *estymacji bayesowskiej* [7, 9].

Metoda probabilistyczna jest powszechnie znaną strukturą predykcyjną. Przykładem może być przewidywanie opadów deszczu, które mogą być na poziomie 60%. Wiadomość jest, że opady będą lub nie, ale poziom prawdopodobieństwa ich wystąpienia jest wyciągany z wyników w przeszłości, np. 6 razy na 10 przypadków padał deszcz. Wartości takiego prawdopodobieństwa są zależne od częstości występowania klas i jest to tzw. prawdopodobieństwo *a priori* (bezwarunkowe). Istnieje także pojęcie prawdopodobieństwa warunkowego i jest ono zależne od kontekstu. Jako przykład może posłużyć chęć wykrycia niechcianych wiadomości w skrzynce elektrycznej. System może wywnioskować, że gdy w wiadomości pojawia się słowo „okazja” w większości przypadków oznacza to spam. Prawdopodobieństwo warunkowe (zajście zdarzenia A pod warunkiem zajścia zdarzenia B) wylicza się z następującego wzoru [5, 7, 9]:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad (1.4)$$

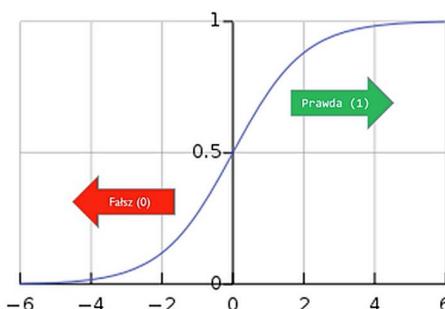
Podstawą konstrukcji modelu probabilistycznego w algorytmach uczenia maszynowego jest twierdzenie brytyjskiego matematyka Thomasa Bayesa, które przedstawia się następująco:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (1.5)$$

gdzie  $A$  i  $B$  to zdarzenia oraz  $P(B) > 0$ , przy czym  $P(A|B)$  jest prawdopodobieństwem warunkowym (wystąpienie zdarzenia  $A$  o ile wystąpi zdarzenie  $B$ ), a  $P(B|A)$  opisuje prawdopodobieństwo wystąpienia zdarzenia  $B$  o ile wystąpi zdarzenie  $A$  [5, 7].

Maszyna punktów Bayesa jest algorymem uczenia maszynowego opartym na regule maksymalizacji prawdopodobieństwa *a posteriori*. W tym systemie przypadek jest klasyfikowany do tego zbioru, dla którego wartość prawdopodobieństwa jest największa. Zadaniem maszyny punktów Bayesa jest, więc znalezienie takiego klasyfikatora, który trafnie przydziela wartości do odpowiednich klas [5].

Kolejnym popularnym algorytmem w uczeniu maszynowym jest regresja logistyczna. Wykorzystuje się ją, jako klasyfikator binarny oraz także, jako klasyfikator wieloklasowy. Bazą regresji logistycznej jest funkcja logistyczna, której celem jest przekształcenie wartości liczbowej na wartość z zakresu (0,1). Często wynik tego algorytmu jest przedstawiany w postaci funkcji sigmoid. Zadaniem konstruktora systemu predykcyjnego, z wykorzystaniem regresji logistycznej, jest odpowiedni dobór punktu odcięcia, który określa granicę pomiędzy dwoma wynikami. Przykład funkcji regresji logistycznej przedstawiono na rysunku 1.17 [5, 7, 9].



Rysunek 1.17. Przykład regresji logistycznej

Źródło: na podstawie [7]

Jak ilustruje rysunek 1.17 wszystkie wyniki przekształcone przez funkcję logistyczną zawarte są w przedziale (0,1), a końcowy wniosek (fałsz czy prawda) określone

są poprzez podział na zbiory, których granicą jest punkt odcięcia (w tym przypadku 0,5). Regresja logistyczna charakteryzuje się szybkim działaniem nawet w przypadku dużych zbiorów danych, aczkolwiek jej wadą jest stałe radzenie sobie, gdy zbiory nie są liniowo separowalne. Jak wspomniano wyżej, algorytm ten może być wykorzystywany także do *klasyfikacji wieloklasowej*. W tym przypadku, zamiast funkcji logistycznej używa się funkcji Softmax, która wylicza prawdopodobieństwo dla każdej wartości tak, aby ich suma dała 1. Należy nadmienić, że obecne platformy uczenia maszynowego często umożliwiają korzystanie z klasyfikatorów binarnych w celach klasyfikacji wieloklasowej, w przypadku, gdy one same nie mają takiego odpowiednika, którego przykładem jest regresja logistyczna. Do tego celu służy moduł korzystający ze strategii *jeden przeciw wszystkim* (ang. *one-vs-all*, lub *one-vs-rest*). Zasada działania bazuje na tworzeniu kolejnych podzbiorów z uzyskanych odpowiedzi, a ich wyniki są porównywane do reszty klas. Predykcja jest wnioskowana na podstawie najlepszych końcowych wyników [5, 7, 9].

Regresja Poissona jest przykładem algorytmu uczenia maszynowego, który w celach predykcyjnych korzysta z funkcji rozkładu Poissona. Jest to powszechna metoda statystyczna, wykorzystywana głównie w zamiarze przewidywania rzadko występujących zdarzeń niezależnych od czasu, jaki upłynął od ostatniego takiego zajścia. Przykładem użycia regresji Poissona może być predykcja awarii urządzenia lub wystąpienia choroby genetycznej. Algorytm ten jest specyficzną metodą, ze względu na odmienną interpretację wyników w porównaniu do innych funkcji regresyjnych, w której charakterystycznym jest waga wpływu analizy wniosków przez specjalistę. Rozkład Poissona przedstawia się następująco [5, 9]:

$$f(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \quad (1.6)$$

gdzie:

- $e$  – jest podstawa logarytmu naturalnego,
- $k$  – liczbą wystąpień zdarzenia,
- $\lambda$  – dodatnią liczbą rzeczywistą równą oczekiwanej liczbie zdarzeń.

Specyficznym algorymem regresyjnym uczenia maszynowego jest *regresja porządkowa* (ang. *ordinal regression*). Jest to funkcja, która może jedynie być kolejnym krokiem wybranego algorytmu regresyjnego. Można z niej korzystać tylko w przypadku danych składających się z liczb możliwych do reprezentowania w formie rankingu. Przykładowo, aby przewidzieć wyniki z egzaminu studentów bazując na poprzednich rezultatach, można by skorzystać z funkcji regresji liniowej. Aczkolwiek w celach predykcji rankingu tychże uczniów na podstawie ich ocen z egzaminu, należałoby uzyskać wynik przeanalizować z wykorzystaniem algorytmu regresji porządkowej [5, 9].

Sztuczne sieci neuronowe są bardzo ważnym i jednym z najstarszych mechanizmów sztucznej inteligencji. Ich konstrukcja jest zbudowana na podstawie obserwacji ludzkiego mózgu, a dokładniej rzecz biorąc *neuronu*. Jest to komórka przetwarzająca i przewodząca informacje w postaci sygnału elektrycznego. Neuron posiada wiele

wejścia (*dendryty*), ciało (*perikarion*) oraz jedno wyjście (*akson*). Akson jednej komórki łączy się z dendrytami innej poprzez biochemiczne złącza (*synapsy*), które modyfikują sygnał i są odpowiedzialne za tzw. *zaplon neuronu*, a więc pobudzenie neuronu do działania. Cała ta konstrukcja i mechanizm działania zainspirowała do odwzorowania takiego układu w dziedzinie sztucznej inteligencji. Każdy z opisanych elementów posiada swoje odwzorowanie w neuronie używanym w działach informatycznych. Dendryty są sygnałami wejściowymi takiego systemu, perikarion zawiera główną funkcję, synapsy opisane są jako wagę, a akson przedstawia sygnał wyjściowy. Pojedynczy neuron nie jest w stanie rozwiązać wiele zadań, dlatego tworzy się sieci neuronowe, czyli połączenie wielu takich komórek. W uczeniu maszynowym korzysta się z algorytmów opartych na sieciach neuronowych w celu regresji, klasyfikacji binarnej, jak i wieloklasowej. Dodatkowo do tej grupy można zaliczyć algorytm perceptronu, który jest najprostszą formą neuronu binarnego. Zasada działania klasyfikacji binarnej z użyciem algorytmu uśrednionego perceptronu składa się z dwóch kroków: odczytywanie przypadku treninowego i sprawdzanie wyniku. W przypadku, gdy końcowy wynik jest równoważny wartości zmiennej wyjściowej, to wagi nie są zmieniane. Jeżeli wynik jest równy „0”, a powinien wynosić „1”, to wartości wektora danych wejściowych są dodawane do wektora wag (np.  $w_n x_n$  gdzie  $w_n$  jest wagą, a  $x_n$  daną wejściową). W odwrotnym przypadku do wartości wektora danych wejściowych są odejmowane od wektora wag. Tak powtarzany cykl umożliwia przypisanie każdej danej do jednej z dwóch kategorii. Jest to bardzo prosty algorytm, radzący sobie tylko z przypadkami, gdzie odpowiedzi są liniowo separowalne, aczkolwiek charakteryzuje się on krótkim czasem obliczeń. Do bardziej skomplikowanych zadań wykorzystuje się algorymy sieci neuronowych. W odróżnieniu od algorytmu uśrednionego perceptronu, sieć neuronowa zmienia wagę w taki sposób, aby końcowy wynik był jak najbliższy wartości zmiennej wyjściowej, a nie w celu najlepszego dopasowania do przypadków treningowych. Sygnał wyjściowy neuronów liniowych jest sumą ważoną ich sygnałów wejściowych. Głównym celem algorytmu jest, więc minimalizacja sumy błędów, najczęściej metodą najmniejszych kwadratów. Algorytmy sieci neuronowej składają się z paru etapów [4, 5, 7, 9]:

- ustalenie topologii sieci,
- losowe inicjowanie wag,
- wyznaczenie odpowiedzi dla danego przypadku treningowego,
- obliczenie błędu kolejnych neuronów wyjściowych na podstawie otrzymanych i rzeczywistych odpowiedzi oraz zapis tych błędów,
- pobieranie kolejnego przypadku treningowego i powtórzenie wszystkich czynności,
- po przeanalizowaniu całego zbioru danych treningowych, modyfikacja wag wszystkich neuronów,
- losowa zmiana kolejności przypadków i rozpoczęcie kolejnej iteracji [5].

Algorytm zakończy swoje obliczenia w momencie, gdy błąd zmniejszy się poniżej ustalonego minimum, ostatnie zmiany wag będą mniejsze od ustalonego minimum lub kiedy zostanie przekroczena maksymalna ilość iteracji. Jednorazową zmianę wag w takim systemie można przedstawić wzorem:

$$\Delta W_i = \varepsilon \cdot x_i(t - y) \quad (1.7)$$

gdzie:

$\varepsilon$  – współczynnik uczenia (odwrotność wartości błędu),

$x_i$  – wartość parametru wejściowego,

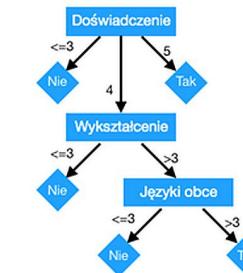
$t$  – poprawna odpowiedź,

$y$  – obliczona odpowiedź.

Główną zaletą algorytmów sieci neuronowych jest duża trafność predykcji, jednak charakteryzują się one dużym ryzykiem przeuczenia lub niedouczenia modelu [5].

W uczeniu maszynowym istnieje wiele algorytmów bazujących na zasadzie działania drzewa decyzyjnego. Jest to prosty model graficzny, który polega na dzieleniu danych na podzbiory bazując na obserwacji wpływu tejże decyzji na stan wyjściowy. W uczeniu maszynowym popularnymi algorytmami w celach klasyfikacji są: wzmacniające drzewo decyzyjne, las decyzyjny i dżungla decyzyjna. Przez duże podobieństwa w ich metodach działania określa się je mianem drzew klasyfikacyjnych. Konstrukcja taka składa się z: **korzenia** – początek modelu, którym jest zmienna mająca największy wpływ na wynik końcowy, **gałęzi** – połączenia zależne od wybranej decyzji, prowadzące do kolejnych **węzłów** – atrybuty danych, **liści** – ostateczne decyzje. W celu opisania działania drzewa decyzyjnego, Autor posłuży się rysunkiem 1.18 [5].

Wykształcenie	Języki obce	Doświadczenie	Przyjęty
2	4	1	Nie
4	3	4	Nie
4	5	5	Tak
1	3	2	Nie



Rysunek 1.18. Przykład tabeli decyzyjnej i drzewa decyzyjnego

Źródło: opracowanie własne

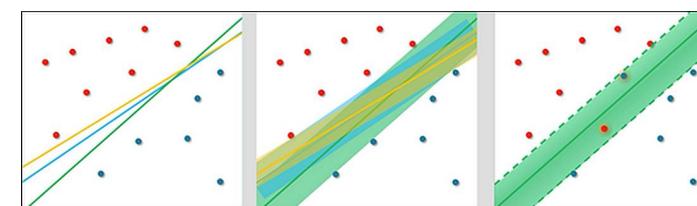
Część tabelaryczna rysunku 1.18 jest to tzw. *tabela decyzyjna*, w której zawarte są informacje na podstawie, których podejmowana jest decyzja o przyjęciu nowego

pracownika. Skonstruowane drzewo decyzyjne odzwierciedla, sposób podejmowania kolejnych decyzji w oparciu o poszczególne atrybuty. Algorytm rozpatruje indywidualnie każdy węzeł drzewa, decydując o tym, czy dany węzeł będzie zakończony liściem wg założonych kryteriów stopu algorytmu, czy też będzie rozgałęziała się wg kryteriów wyboru kolejnego atrybutu. Pojedyncze drzewo decyzyjne posiada wiele wad. Rozbudowa takiego modelu metodą najlepszego w danym momencie podziału często prowadzi do nieuwzględnienia wszystkich zmiennych wejściowych (w przykładzie przedstawionym w rysunku 1.18 algorytm mógł skończyć już przy podziale na *Doświadczenie* twierdząc, że na podstawie tylko tej jednej zmiennej są wybierani pracownicy) [5, 7].

Drzewa decyzyjne charakteryzują się także nadmiernym dopasowaniem do przypadków treningowych. Aby zniwelować te i inne błędy, w uczeniu maszynowym zastosowano tzw. modele złożone, czyli składające się z wielu algorytmów bazowych. W tym przypadku mowa o *lesie decyzyjnym* i *dżungli decyzyjnej*, która jest optymalizacją tego pierwszego. Pierwszym etapem tychże algorytmów jest wylosowanie *próby bootstrap* (metoda polegająca na wielokrotnym losowaniu ze zwracaniem przypadku do ogólnego zbioru, powodując możliwość ponownego jego wylosowania). Wybrana próba jest wykorzystywana w celu uczenia drzewa decyzyjnego, do którego, także losowo, wybierane są atrybuty. Stwarza to możliwość przeanalizowania bardzo dużej ilości możliwych zależności między wartościami wejściowymi i odpowiedzią wyjściową. W ostatniej kolejności wnioskuje się predykcję na podstawie głosowania większościowego wyników wielu drzew decyzyjnych. Należy nadmienić, że algorytmy lasów i dżungli decyzyjnych, mogą być używane w celach regresji, gdzie jedyną różnicą jest wyliczanie finalnego rezultatu poprzez uśrednienie wyników uzyskanych z każdego drzewa decyzyjnego. Głównymi zaletami algorytmu lasu drzew decyzyjnych jest skalowalność, odporność na wartości nietypowe oraz duża dokładność. Niemniej jednak przy bardzo dużych bazach danych należy mieć zarezerwowane ogromne ilości pamięci w celu przechowywania i przeszukiwania wszystkich tworzonych w drodze obliczeń drzew decyzyjnych. Dział Microsoft Research opracował algorytm dżungli drzew decyzyjnych, który, jak wyżej wspomniano, jest optymalizacją lasów decyzyjnych. Poprzez usunięcie duplikatów drzew decyzyjnych model ten zmniejsza zajmowaną przez algorytm pamięć. Innym algorytmem wykorzystywanym w celach klasyfikacji i regresji jest tzw. *drzewo decyzyjne ze wzmacnianiem*. Jego idea opiera się na myśl, że wynik złożony z wielu słabszych klasyfikatorów jest, co najmniej tak dobry (a najczęściej o wiele lepszy), jak wniosek z najlepszego drzewa decyzyjnego. Specyficznym aspektem tego algorytmu jest wybór danych treningowych faworyzując te przypadki, które były błędnie sklasyfikowane we wcześniejszym modelu bazowym, a nie w drodze losowej, jak to ma zwyczaj m.in. w lesie drzew decyzyjnych. Najpopularniejszym algorytmem używanym w drzewach decyzyjnych ze wzmacnianiem jest *AdaBoost (Adaptive Boosting)*. W każdej iteracji oceniana jest ważność klasyfikatorów analizując ich błąd klasyfikacji. Jeżeli błąd przekroczy określona granicę (najczęściej 50% błędów), klasyfikator ten jest wtenczas usuwany, a przypisanie nowych wag do przypadków zależy jest od określonej ważności danego klasyfikatora. Kolejnym przykładem wykorzystania konstrukcji lasów decyzyjnych jest *algorytm kwantylowy* (ang. *fast forest quantile regression*), który jest szczególnym

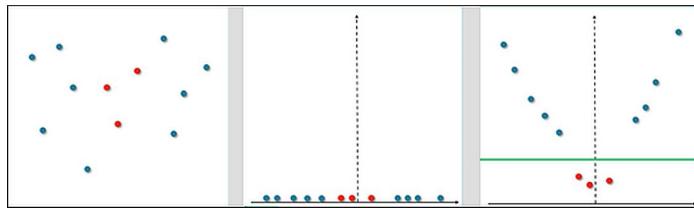
przypadkiem metody regresji. *Kwantyl* jest statystycznym parametrem służącym do opisu pewnego rozkładu populacji, do których można zaliczyć m.in. wyżej wspomniane już *kwartyle* (dzielące populację na cztery części), czy też *kwintyle* (dzielące populację na pięć części). W związku z tym, algorytm kwantylowy nie zwraca odpowiedzi w formie wartości średniej lub porządkowej, a przedstawia rozkład zmiennej wyjściowej. Przykładowo, taki model może dać odpowiedź czy cena roweru znajduje się wśród 25% najwyższych (w pierwszym kwantylu), czy cenowo należy do typowych (drugi kwantyl), czy też jest z tej półki najwyższych 25% (trzeci kwantyl) [5, 7, 9].

Maszyna wektorów nośnych (SVM, ang. *Support Vector Machine*) jest liniowym binarnym klasyfikatorem. Uczenie tego algorytmu polega na znalezieniu granicy z jak najszerzym marginesem rozdzielającym wartości zbioru treningowego na dwa podzbiory. Klasyfikacja kolejnych, nowych danych opiera się wówczas na decyzji, w której klasie powinny się one znaleźć. Poglądowy przykład działania SVM przedstawia rysunek 1.19 [5, 7].



Rysunek 1.19. Etapy działania SVM  
 Źródło: na podstawie [7]

Jak ilustruje rysunek 1.19 może istnieć wiele granic dzielących zbiór na dwie klasy. W celu wybrania optymalnego podziału, wyznacza się marginesy, których granice opisane są przez pierwsze punkty każdej z klas (czerwone i niebieskie kółka). Im szerszy margines tym mniejsze ryzyko nadmiernego dopasowania i lepsza jego generalizacja. Jak przedstawia trzecia część rysunku 1.19 często może dojść do sytuacji, gdzie kategorie nie są liniowo separowalne i mogą znaleźć się obserwacje w niewłaściwym podzbiorze. Algorytmy SVM unikają takich problemów poprzez bazowanie na koncepcji płynnych marginesów, która polega na niewprowadzaniu sztywnie ustalonej granicy, kierując się myślą „wyjątek nie czyni reguły”. Im bardziej restrykcyjne granice, tym większe ryzyko *overfittingu*. W większości przypadków badanych w Data Science, napotyka się kategorie nielinowo separowalne i jedną z metod rozwiązywania takich problemów przedstawiono na rysunku 1.20 [7].



Rysunek 1.20. Przykład działania SVM

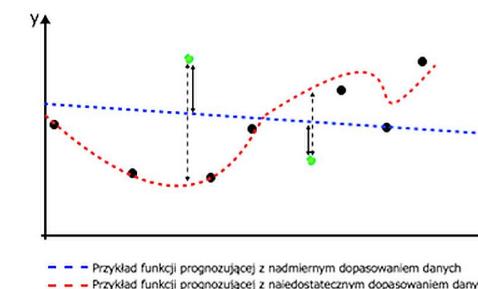
Źródło: na podstawie [7]

Przykład na rysunku 1.20 przedstawia sytuację, gdzie nie ma możliwości rozdzielić badanych kategorii (niebieskie i czerwone kółka) jedną linią. W takim przypadku jedna z metod SVM polega na przetransformowaniu danych i wyznaczenie granicy korzystając z dodatkowych wymiarów. Druga część rysunku 1.20 ilustruje przykładowe rzutowanie badanych wartości na oś x, gdzie następnie należy znaleźć odpowiednie położenie osi y tak, aby uzyskać lepszą separację dwóch zbiorów. Funkcje transformacyjne mogą być prostymi wzorami jak np.  $x^2$ , co jest przedstawione w ostatniej części rysunku 1.20. Tak przekształcony zbiór można w łatwy sposób rozdzielić granicą na dwie pożądane klasy. Podstawowym problemem w opisany przypadku jest wyznaczenie odpowiedniego miejsca osi y. Takie transformacje są jednak rozwijane w sposób automatyczny za pomocą funkcji jądrowych (ang. *kernel functions*), którymi są najczęściej wielomiany, funkcje sigmoidalne, funkcje radialne symetryczne, czy też wielomiany sklejane. Algorytm SVM w celach klasyfikacji binarnej charakteryzuje się dużą dokładnością oraz relatywnie szybkim wykrywaniem liniowo separowalnych klas. Niemniej jednak przy skomplikowanych sytuacjach, jak np. przy badaniu przestrzeni rzadkich (gdzie punkty są od siebie znacznie oddalone) czas nauki algorytmu SVM znacznie się wydłuża. Zespół Microsoft Research opracował algorytm lokalnie głębokiej maszyny wektorów nośnych (ang. *Two-class locally deep SVM*), który znacząco skracą czas obliczeń. Zasada działania tego algorytmu sprowadza się do porównywania odległości między punktami bazując na ich podobieństwach lokalnych, wyznaczonych za pomocą lokalnej funkcji jądrowej. Istnieją jeszcze inne obszary uczenia maszynowego, poza klasyfikacją, w której wykorzystywana jest konstrukcja SVM. Autor w dalszej części pracy zbadał działanie algorytmu z ang. *One-class SVM*. Jak sama nazwa wskazuje, funkcja ta bada w obszarze jednej klasy, co umożliwia detekcję anomalii. Przykładowo, można wyznaczyć algorytm klasyfikacji binarnej, w celu wyróżnienia, jakie działania są oszustwem, a które nie, dla zwykłego posiadacza konta bankowego. Należy nadmienić jednak, że taki algorytm nie będzie wystarczająco wrażliwy, aby poprawnie oznaczyć np. kradzież, która jest wykonana po raz pierwszy, w nowy sposób. W tym celu, umieszczanie przypadków w jednym zbiorze ułatwia detekcję anomalii tj. predykcja tornada w Polsce, oszustw bankowych, opadów śniegu w Afryce [5, 7, 9].

Kolejnym algorytmem uczenia maszynowego z rodziny detekcji anomalii i jednocześnie ostatnim badanym w tej pracy magisterskiej, jest algorytm bazujący na metodzie analizy głównych składowych (ang. *PCA-Based Anomaly Detection*). Podstawowym problemem w detekcji wartości odstających poza normę jest mała ilość danych reprezentujących takie anomalie, np. w celu wykrycia oszustw bankowych, algorytmy uczenia maszynowego mają za mało danych opisujących samą kradzież, a za to ogrom informacji przedstawiających prawidłowe operacje. Algorytm bazujący na metodzie PCA ma za zadanie określić, co jest normą w danym przypadku, analizując zależności i podobieństwa między danymi wejściowymi. W celu wykrycia anomalii wszystkie nowe informacje rzutowane są na obszar określony „normą”, a znormalizowany błąd jest wynikiem anomalii. Im większy błąd, tym większe odstępstwo od normalnego zbioru. Cały model algorytmu PCA opiera się na rachunku macierzowym. Poprzez odpowiednie transformacje wyznaczana jest macierz transformacji, dzięki której, wraz z macierzą danych wejściowych, wyliczania jest macierz głównych składowych, a jej wariancje wartości określają, które składowe są najbardziej znaczące [5, 7].

#### 1.4.3.3. Ewaluacja

Ewaluacja jest etapem systemu predykcyjnego, w którym stworzony model jest poddawany ocenie, co umożliwia ewentualną poprawę jego jakości. Określenie jakości modelu decyduje o poradzie bądź sukcesie projektu, ułatwia komunikację pomiędzy zespołami (np. technicznym, analitycznym i biznesowym). Miary te często zawierają wiele niezbędnych informacji pod postacią jednej wartości. Wyzwaniem dla projektanta takiego systemu jest wybór odpowiedniego typu ewaluacji spośród wielu możliwości, dobierając ją w zależności od kategorii badanego problemu. Wymaga to od takiej osoby znajomości kontekstu biznesowego, czy też branżowego dla danego modelu. Rodzaje ewaluacji modelu i interpretacje ich wyników zależne są także od wyboru kategorii algorytmu uczenia maszynowego. W celu opisania oceny jakości modeli regresyjnych połuży rysunek 1.21 [5, 7].



Rysunek 1.21. Ocena jakości modelu regresyjnego

Źródło: opracowanie własne na podstawie [7]

Ilustruje on przykład podejścia do predykcji cechujący się nadmiernym oraz niedostatecznym dopasowaniem. W takim przypadku, aby ocenić, który z zaprezentowanych modeli jest lepszy wystarczy wyznaczyć różnicę (błąd) pomiędzy predykcjami (zienione punkty), a wyznaczonymi wartościami. Istnieje szereg możliwych do wyznaczenia rodzajów błędów w modelach regresyjnych.

- Do najpopularniejszych z nich należą:
- średni błąd bezwzględny – MAE, z ang. *mean average error*,
  - średni bezwzględny błąd procentowy – MAPE, z ang. *mean average percentage error*,
  - pierwiastek błędu średniokwadratowego – RMSE, z ang. *root mean squared error*,
  - współczynnik determinacji – R<sup>2</sup>, ang. *coefficient of determination* [7].

Zasadę obliczania średniego błędu bezwzględnego przedstawiono na rysunku 1.22.

MAE									
Waga rzeczywista	58	62	65	71	75	81	92	95	
Waga przewidywana	57	62	67	70	78	81	93	91	
Błąd	-1	0	2	-1	3	0	1	-4	
Wartość bezwzględna błędu	1	0	2	1	3	0	1	4	
↓									
Średni błąd bezwzględny = MAE = 1,5									

Rysunek 1.22. Etapy wyliczania średniego błędu bezwzględnego  
Źródło: na podstawie [7]

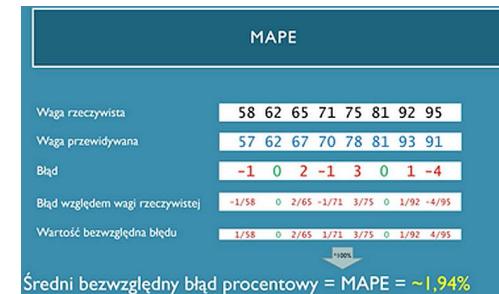
Jak pokazuje rysunek 1.22 wyliczanie MAE nie jest wysoce skomplikowaną czynnością. W pierwszej kolejności wyznacza się różnicę pomiędzy danymi uzyskanymi, a danymi przewidywanymi. Z perspektywy samej funkcji nie ma znaczenia, czy wyznaczony błąd jest na korzyść bądź na niekorzyść tej wartości tak, więc następnie wyliczana jest wartość bezwzględna obliczonej różnicy. Finalnie wylicza się średnią, co jest ostatecznym wynikiem. Średnią błędu bezwzględnego wyraża się ogólnym wzorem [7]:

$$MAE = \frac{1}{n} \sum_{i=1}^n |e_i| \quad (1.8)$$

gdzie:

- $i$  – jest kolejną obserwacją,
- $n$  – liczbą obserwacji,
- $e_i$  – błędem estymacji.

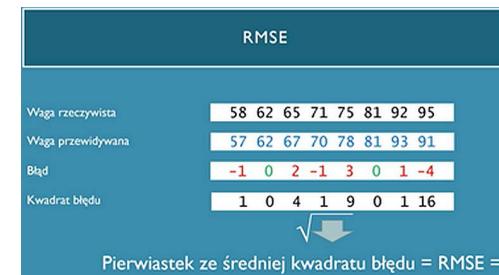
Celem projektowania modelu predykyjnego opartego na algorytmach regresyjnych jest minimalizacja średniego błędu bezwzględnego (im wartość bliżej zera, tym lepiej). Bardzo zbliżoną metodą do średniego błędu bezwzględnego jest metoda MAPE, której etapy wyznaczania przedstawiono na rysunku 1.23 [7].



Rysunek 1.23. Etapy wyliczania średniego bezwzględnego błędu procentowego  
Źródło: na podstawie [7]

Rysunek 1.23 przedstawia różnicę między wyliczaniem MAE, a MAPE. Po wyznaczeniu różnicy, między wartościami uzyskanymi a przewidywanymi, opisuje się ją w stosunku do danej rzeczywistej, wskazując na proporcje błędu. Wynik ostateczny uzyskuje się po przedstawieniu wartości bezwzględnej błędu, wyliczeniu jej średniej oraz przemnożeniu przez 100% [7].

Przykład wyznaczania wartości błędu metodą pierwiastka błędu średniokwadratowego zilustrowano na rysunku 1.24.



Rysunek 1.24. Etapy wyliczania pierwiastka błędu średniokwadratowego  
Źródło: na postawie [7]

Jak na nim widać pierwszy etap, czyli wyznaczenie różnicy między wartościami, jest identyczny do poprzednich dwóch opisanych metod. Następnie oblicza się kwadrat

błędu. Uzyskuje się dzięki temu podobny efekt, a więc wszystkie wyniki stają się wartością dodatnimi, ale dodatkowo uwypukla się te wartości, które odstają najbardziej [7].

Po obliczeniu średniej i jej spierwiastkowaniu, otrzymuje się ostateczny wynik RMSE. Ogólny wzór opisanej metody przedstawia się następująco [7]:

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (p_i - a_i)^2}{n}} \quad (1.9)$$

gdzie:

$n$  – jest liczbą obserwacji,

$i$  – kolejną obserwacją,

$p_i$  – oszacowaną wartością zmiennej wyjściowej  $i$ ,

$a_i$  – rzeczywistą wartością zmiennej wyjściowej  $i$ .

Metoda RMSE, MAE i MAPE mają wspólną zależność, że im mniejsza wartość wyniku, tym lepsza jakość modelu predykcyjnego. Główne różnice między metodami RMSE i MAE są takie, że ta pierwsza podkreśla rozpiętość błędów w predykcjach (poprzez kwadrat błędu), a średni błąd bezwzględny daje odpowiedź łatwą do interpretowania (w opisanym przykładzie, model przewiduje wynik z dokładnością  $\pm 1,5$  kg) [7].

Kolejnym popularnym sposobem oceny jakości modelu predykcyjnego jest współczynnik determinacji. Opiera się on o inne zasady wyznaczania niż opisane powyżej metody.  $R^2$  przedstawia wariancję wartości wyjściowej od zmiennych wejściowych. Wynik współczynnika determinacji przedstawiany jest w zakresie od 0 do 1 (lub procentowo) i im wyższy, tym lepsza jakość systemu predykcyjnego. Niski poziom  $R^2$  oznacza konieczność rozszerzenia bazy o dodatkowe dane. Współczynnik determinacji oblicza się na podstawie wzoru:

$$R^2 = 1 - \frac{SSE}{SST} \quad (1.10)$$

gdzie suma kwadratów błędów:

$$SSE = \sum (y - y')^2 \quad (1.11)$$

a różnica między wartościami rzeczywistymi a średnią:

$$SST = \sum (y - \bar{y})^2 \quad (1.12)$$

gdzie:

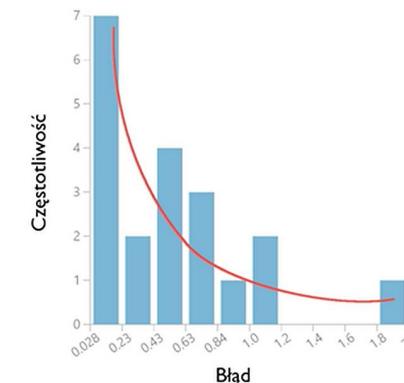
$y'$  – wartość rzeczywista,

$y$  – wartość przewidywana,

$\bar{y}$  – średnia wartość przewidywana [7].

Ważnym aspektem większości nowoczesnych platform do tworzenia modeli predykcyjnych jest możliwość wizualizacji wyników. Ułatwia to ich interpretację oraz

możliwość zwrócenia uwagi na błędy występujące w samym modelu, bądź w analizowanych danych. Microsoft Azure Machine Learning Studio jest platformą uczenia maszynowego, wykorzystaną przez autora w niniejszej pracy magisterskiej. Przykład wizualizacji przez ten program jakości modelu wykorzystującego algorytm regresyjny przedstawia rysunek 1.25 [7].



Rysunek 1.25. Histogram błędów dla modelu wykorzystującego algorytm regresji

Źródło: na podstawie [7]

Taka wizualizacja ilustruje rozkład błędów danego modelu. Dobry model charakteryzuje się małą wartością odchylen od wartości rzeczywistej, a im większe odchylenie tym mniejsza powinna być ich ilość. Dlatego też dodatkowo na rysunku 1.25 zaznaczono czerwoną linią pozwadany kształt histogramu [7].

Najbardziej popularną metodą przedstawiania jakości modeli wykorzystujących klasyfikatory binarne (i detekcji anomalii) jest macierz błędów (ang. confusion matrix), której przykład jest pokazany na rysunku 1.26 [7].

		Predykcja	
		+	-
Rzeczywistość	+	TP (true positive)	FN (false negative)
	-	FP (false positive)	TN (true negative)

Rysunek 1.26. Macierz błędów wykorzystywana do określenia jakości klasyfikatorów binarnych  
Źródło: na podstawie [7]

Rysunek 1.26 przedstawia przykład dwóch klas: „+” i „-”. W rzeczywistości mogą to być oczywiście jakieś kolwiek zbioru binarne jak np. duży i mały, czarny i biały. Jak pokazuje rysunek 1.26, macierz błędów posiada cztery możliwe sytuacje:

- *Prawdziwie pozytywna* – ang. *true positive* (TP), czyli prawidłowe wskazania pozytywne,
- *Prawdziwie negatywna* – ang. *true negative* (TN), czyli prawidłowe wskazania negatywne,
- *Nieprawdziwie pozytywna* – ang. *false positive* (FP), czyli nieprawidłowe wskazania pozytywne,
- *Nieprawdziwie negatywna* – ang. *false negative* (FN), czyli nieprawidłowe wskazania negatywne [5, 7].

Opisują one prawidłowość lub nieprawidłowość wyników predykcji przynależności lub nie do danych klas. Te cztery koncepcje są podstawą do wyliczania *miar oceny jakości* modeli wykorzystujących klasyfikatory dwuklasowe.

Pierwszą z takich miar jest tzw. *czułość* (ang. *sensitivity*, TPR – *true positive rate*) i określa ona zdolność do wykrycia kategorii pozytywnej. Przedstawia się następującym wzorem:

$$TPR = \frac{TP}{TP + FN} \quad (1.13)$$

Kolejną wartością jest *specyficzność* (ang. *specificity*, TNR – *true negative rate*) i wyznacza zdolność do wykrycia kategorii negatywnej. Specyficzność wyznaczana jest poprzez wzór:

$$TNR = \frac{TN}{TN + FP} \quad (1.14)$$

Wyniki czułości i specyficzności uzupełniają się i powinny być oceniane razem. Kolejną miarą oceny jakości modeli klasyfikatorów binarnych jest tzw. *precyzja* (ang. *precision*) i może zarówno dotyczyć pozytywnych kategorii (PPV) lub negatywnych kategorii (NPV). Wyznacza się ją następująco:

$$PPV = \frac{TP}{TP + FP} \quad (1.15)$$

$$NPV = \frac{TN}{TN + FN} \quad (1.16)$$

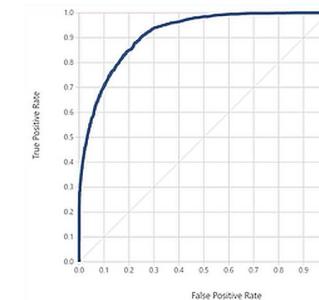
Jedną z ważniejszych metryk jest *dokładność* (ang. *accuracy*), która przedstawia proporcje właściwych predykcji do wszystkich przewidywań. Dokładność oblicza się z przedstawionego równania:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (1.17)$$

Dokładność jest często zbyt szybko uważana za najważniejszą (i jedyną) miarę jakości modelu, ze względu na jej łatwą interpretowalność. Jednakże analiza poszczególnych metryk zależy od scenariusza, na którym oparty jest zaprojektowany system predykcyjny. Ostatnią z najważniejszych miar oceny jakości modelu jest tzw. *współczynnik F1* (ang. *F1 score*). Określa on proporcję pomiędzy precyją i czułością. Wzór współczynnika F1 przedstawia się następująco:

$$F1 = 2 \cdot \frac{PPV \cdot TPR}{PPV + TPR} \quad (1.18)$$

Tak jak i w przypadku modeli regresyjnych, tak samo przy wykorzystaniu klasyfikatorów dwuklasowych istnieje możliwość wizualizacji wyników jakości. Szeroko polecanym typem wizualizacji jest *krzywa ROC* (ang. *receiver operating characteristics curve*). Ta funkcja przedstawia relację pomiędzy dobrze sklasyfikowanymi wartościami, a tymi obarczonymi błędami. Przykład krzywej ROC przedstawiono na rysunku 1.27 [5, 7].



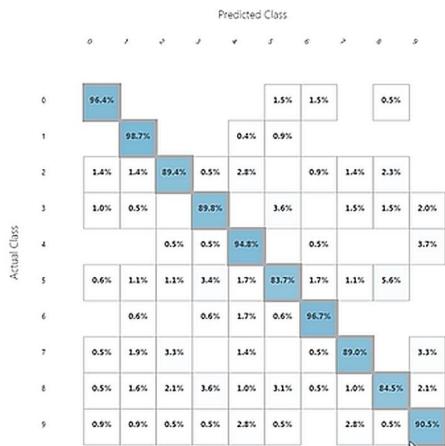
Rysunek 1.27. Przykład krzywej ROC  
Źródło: na podstawie [7]

Bardzo ogólnie określona reguła oceny jakości modelu na podstawie krzywej ROC brzmi następująco: im bardziej krzywa wygięta w kierunku lewego górnego rogu, tym lepsza jakość modelu predykcyjnego. W związku z tym, przykład na rysunku 1.27 przedstawia dobrą jakość badanego systemu. Ponieważ takie określenie może być bardzo subiektywne, to istnieje możliwość przedstawienia krzywej ROC w sposób numeryczny. Wskaźnik AUC (ang. *area under curve*) jest obliczany na podstawie powierzchni pola pod wyznaczoną krzywą ROC i powinien przyjąć wartość z przedziału od 0,5 do 1 [5, 7].

Ocena jakości modeli wykorzystujących klasyfikatory wieloklasowe także opisywana jest na podstawie macierzy błędów. Jest ona oczywiście odpowiednio większa, w zależności od ilości podzbiorów. Rysunek 1.28 przedstawia przykładową macierz błędów dla klasyfikatorów wieloklasowych [7].

## ROZDZIAŁ 2

### Konstrukcja urządzenia pomiarowego



Rysunek 1.28. Przykład macierzy błędów dla modeli wykorzystujących klasyfikatory wieloklasowe

Źródło: na podstawie [7]

Pierwszą rzeczą zauważalną na rysunku 1.28 jest diagonalna linia oznaczona na niebiesko, która w rzeczywistości określa prawidłową klasyfikację do odpowiednich podzbiorów. Dodatkowo, taka macierz pozwala dokładnie przeanalizować każdą z klas, jako osobny przypadek. Z tego względu, miary przedstawione w opisie oceny jakości klasyfikatorów binarnych, w tym przypadku mogą być wyliczane na różne sposoby: w skali micro oraz w skali makro. Średnia w skali mikro zakłada, że każda obserwacja ma taki sam wpływ na wynik, np. liczona jest całkowita wrażliwość dla całego zbioru. Średnia w skali makro, opiera się na tym, że każda z klas ma taki sam wpływ na wynik, np. wyliczana jest wrażliwość dla każdej z klas, po czym wyciągana jest średnia. Ta sama teoria tyczy się wyliczeń odnoszących się do dokładności. Wzory na wyliczenie poszczególnych metryk są analitycznie skonstruowane, jak w przypadku opisanych wcześniej klasyfikatorów binarnych, z uwzględnieniem odpowiedniej ilości klas [5, 7].

Możliwość zbadania wybranych algorytmów sztucznej inteligencji wymaga odpowiednio przygotowanych informacji. Jednym z zadań w ramach pracy dyplomowej, jakie Autor postawił przed sobą było stworzenie takiej bazy danych poprzez skonstruowanie urządzenia pomiarowego, które rejestruje żądane wartości. Jako że wieloletnim hobby Autora jest wędkowanie, zrodziło to pytanie odnośnie jakości połówów w zależności od zalistnialnych warunków pogodowych. W celu otrzymania odpowiedzi, w pierwszej kolejności postanowiono skonstruować urządzenie, które wykonuje pomiary warunków pogodowych w trakcie wędkowania. Należy nadmienić, że duże doświadczenie w konstrukcji podobnego urządzenia Autor nabył w trakcie tworzenia swojej pracy dyplomowej inżynierskiej pt. „Projekt wodoszczelnego przenośnego systemu do pomiaru parametrów pogodowych”.

#### 2.1 Projekt elektroniki, konstrukcja płyty PCB oraz obudowy

System pomiarowy wymaga monitorowania następujących wartości: temperatury powietrza, ciśnienia atmosferycznego, wilgotności powietrza, temperatury wody, położenia geograficznego, czasu i daty. Te same parametry były mierzone systemem skonstruowanym w poprzedniej pracy dyplomowej, której efekt końcowy przedstawiony jest na rysunku 2.1.



Rysunek 2.1. System do pomiaru parametrów pogodowych skonstruowany w ramach pracy inżynierskiej Autora  
Źródło: opracowanie własne

Moduły pomiarowe z poprzedniego urządzenia zostały wykorzystane ponownie, ze względu na ich wysoką precyzyję pomiarową, która została zbadana w pracy inżynierii Autora. Priorytetem było dodanie możliwości rejestrowania badanych wartości w celu przeniesienia ich na komputer i stworzenia bazy danych, a także poprawienie funkcjonalności, zmieniając ekran LCD sterowany przyciskami, na ekran dotykowy rezystancyjny. Jednym z najtrudniejszych etapów konstrukcji urządzenia był projekt płyty PCB tak, aby dopasować ją do obudowy. Aby ominąć te trudności, postanowiono zaprojektować i wykonać obudowę systemu metodą druku 3D. Zmieniono także układ zasilający urządzenie, ułatwiając przy tym budowę płyty PCB oraz zmniejszając wymagane miejsce w obudowie. Po długich poszukiwaniach odpowiednich elementów oraz dokładnej analizie możliwości konstrukcyjnych, postanowiono zaprojektować system pomiarowy oparty na elementach przedstawionych na rysunku 2.2.

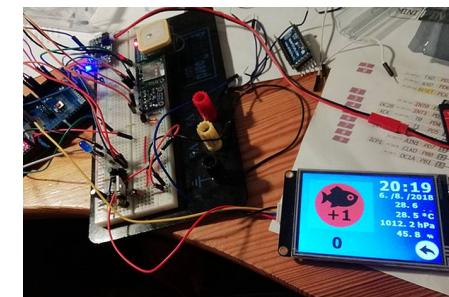


Rysunek 2.2. Elementy elektroniczne wchodzące w skład urządzenia pomiarowego  
Źródło: opracowanie własne

Moduł zasilający oparto na akumulatorze litowo-polimerowym AKYGA L3867100, którego napięcie wyjściowe wynosi 3,7 V, a pojemność 2400 mAh. Bateria ta posiada własny układ zabezpieczający przed nadmiernym rozładowaniem oraz uszkodzeniem baterii. Akumulator połączony jest z resztą układu poprzez moduł Adafruit PowerBoost 1000C, oparty na chipie MCP73871 (używanym w poprzedniej wersji systemu). Główną zaletą takiej konstrukcji jest możliwość jednoczesnego korzystania z urządzenia i ładowania nie uszkadzając przy tym akumulatora. Moduł ten posiada także szereg wyprowadzeń umożliwiających m.in. monitorowanie stanu baterii, a także konwersję napięcia na 5 V. W celu wykonywania pomiarów parametrów pogodowych, użyto sensora Bosch BME280 wmontowanego w moduł Adafruit BME280, który cechuje się możliwością jednoczesnego pomiaru: temperatury, ciśnienia atmosferycznego

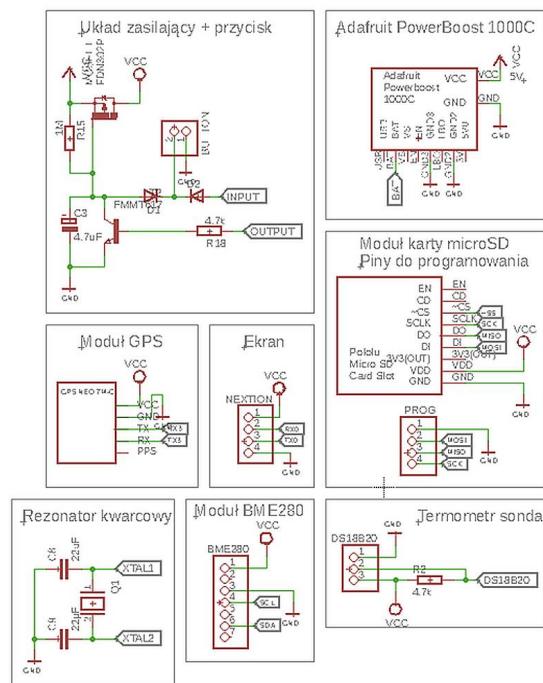
oraz wilgotności powietrza. Jest to element wykorzystany w poprzedniej wersji systemu, który charakteryzuje się wysoką dokładnością pomiarową i prostotą obsługi. Kolejnym modulem, który został uprzednio sprawdzony jest NEO-7M-C. Połączyszy się z systemem GPS, rejestruje on wiadomości m.in. o: położeniu geograficznym, liczbie satelitów, z którymi posiada łączność, aktualnej dacie i godzinie. Wartości wielkości, które są mierzone opisanymi elementami elektronicznymi, zapisywane są na karcie micro SD poprzez połączenie z modelem firmy Pololu. Posiada on wbudowany regulator napięcia minimalizując tym samym możliwość przypadkowych błędów w trakcie jego pracy. Po przetestowaniu szeregu ekranów dotykowych, zdecydowano się wykorzystać model Nextion Enhanced NX4832K035. Jest to ekran o rozmiarze 3,5", z wyświetlaczem TFT oraz panelem dotykowym rezystancyjnym. Głównym powodem wyboru tego elementu jest oprogramowanie firmy Nextion, które umożliwia zaprojektowanie i zaprogramowanie interfejsu w sposób bardzo prosty i intuicyjny. W konstrukcji całości urządzenia, zdecydowano się na użycie jednego przycisku, umożliwiającego włączanie oraz wyłączenie systemu, inspirując się konstrukcją nowoczesnych telefonów komórkowych. Całość urządzenia sterowana jest za pomocą mikrokontrolera firmy AVR model ATmega, który został wybrany ze względu na doświadczenie Autora w programowaniu tego typu modułów. W pierwszej kolejności programowano mikrokontroler AVR ATmega328, aczkolwiek ze względu na wymaganą dużo większą pamięć Flash, ostatecznie system opiera się na AVR ATmega2560. Charakteryzuje się on pamięcią Flash 256 KB, czterema interfejsami UART, pięcioma SPI oraz jednym I<sup>2</sup>C, co umożliwia działanie wszystkich modułów jednocześnie bez zacinania się systemu.

Konstrukcja prototypu systemu pomiarowego oraz jego wstępne programowanie pochłonęło najwięcej czasu w trakcie powstawania pracy magisterskiej. Zasilanie układu, funkcje przycisku oraz jednoczesna praca modułów przy najmniejszym zużyciu prądu – to przede wszystkim te etapy wymagały wielu modyfikacji. Rysunek 2.3. przedstawia fragment prac przeprowadzanych w trakcie powstawania prototypu.

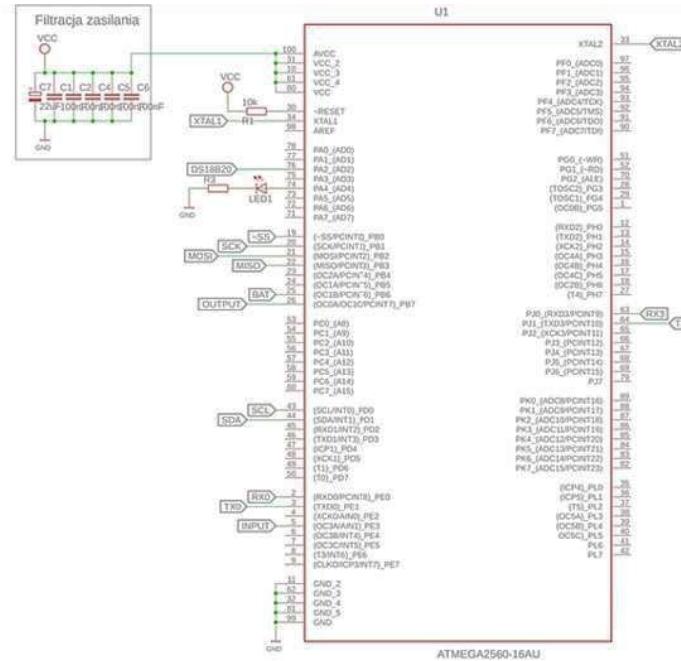


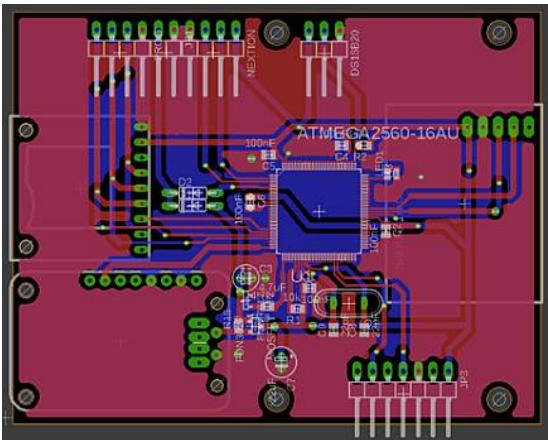
Rysunek 2.3. Etap powstawania prototypu urządzenia pomiarowego  
Źródło: opracowanie własne

Kolejnym etapem było zaprojektowanie oraz wykonanie płyty PCB opierając się na już przetestowanym prototypie. W tym celu skorzystano także z programu Autodesk Eagle. Zdecydowano się na tą platformę, прежде wszystkim ze względu na możliwość automatycznego wygenerowania modelu 3D stworzonej płytki, co poprzez program Autodesk Fusion 360, znacznie ułatwio zaprojektowanie obudowy dla urządzenia. Etap ten charakteryzował się jednocośnym korzystaniem z obu wyżej wymienionych programów ze względu na chęć zminimalizowania wielkości urządzenia pomiarowego, tym samym optymalizując rozmieszczenie poszczególnych elementów elektronicznych. W pierwszej kolejności zaprojektowano układ elektryczny, którego schemat przedstawiono na rysunkach 2.4 i 2.5.

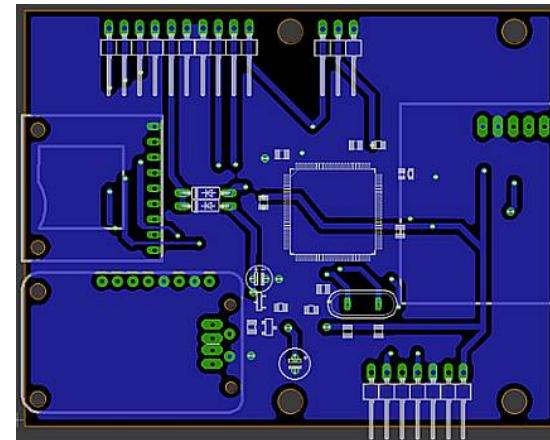


Rysunek 2.4. Schemat elektryczny cz.1 – poszczególne moduły  
 Źródło: opracowanie własne

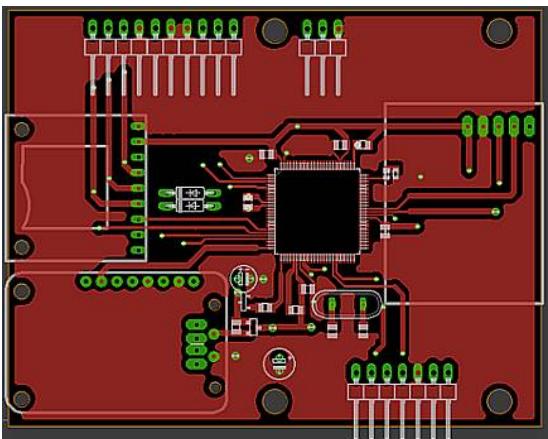




Rysunek 2.6. Projekt płyty PCB skonstruowanego urządzenia – widok całościowy  
 Źródło: opracowanie własne

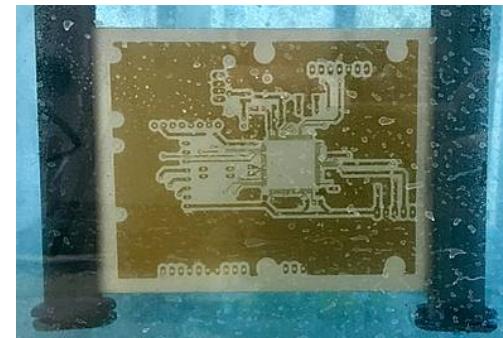


Rysunek 2.8. Projekt płyty PCB skonstruowanego urządzenia – widok warstwy Bottom  
 Źródło: opracowanie własne



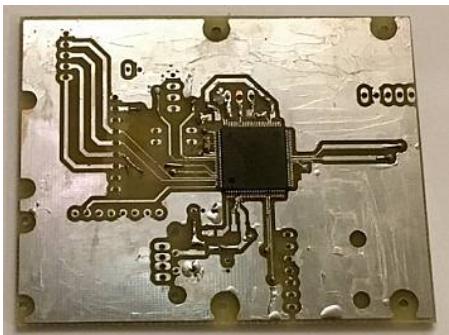
Rysunek 2.7. Projekt płyty PCB skonstruowanego urządzenia – widok warstwy Top  
 Źródło: opracowanie własne

Płytkę PCB wykonano w warsztacie Wydziału Elektrycznego metodą fotochemiczną. Sposób ten polega na wydrukowaniu schematu na folii, który następnie jest naświetlany promieniami UV na laminacie (w tym przypadku dwustronnym). Po prawidłowym naświetleniu następuje wywoływanie rysunków poprzez zamoczenie płytki w odpowiednim roztworze chemicznym. Następnie tak przygotowany element zanurza się w wytrawiaczu, który powoduje usunięcie warstwy metalicznej laminatu w miejscach, gdzie nie są zaznaczone ścieżki. Ten etap przedstawia rysunek 2.9.



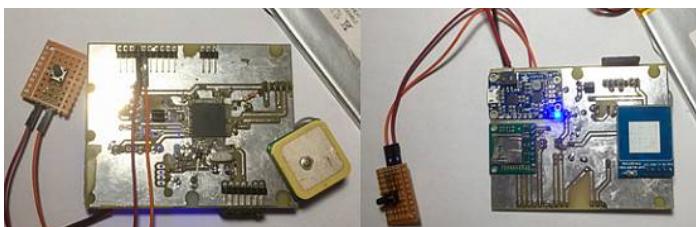
Rysunek 2.9. Etap wytrawiania płyty PCB  
 Źródło: opracowanie własne

W celu zwiększenia wytrzymałości ścieżek oraz polepszenia przewodności prądowej, całość płytki ocynowano. Po wykonaniu opisanego kroku oraz przylutowaniu mikrokontrolera, płytka PCB wygląda tak, jak przedstawiono na rysunku 2.10.



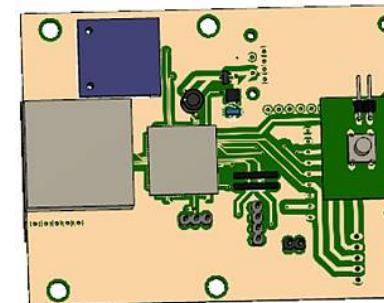
Rysunek 2.10. Płyta PCB z mikrokontrolerem  
 Źródło: opracowanie własne

Finalną wersję płytki PCB wraz z przylutowanymi wszystkimi elementami przedstawiono na rysunku 2.11.



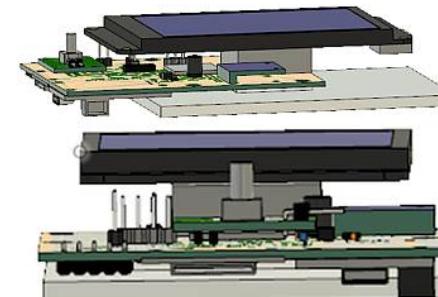
Rysunek 2.11. Płyta PCB- po lewej strona Top, po prawej strona Bottom  
 Źródło: opracowanie własne

Obudowę urządzenia zaprojektowano w programie Autodesk Fusion 360. W pierwszej kolejności eksportowano zaprojektowaną płytę PCB z programu Autodesk Eagle. Po wprowadzeniu kilku modyfikacji w budowie trójwymiarowej poszczególnych modułów elektronicznych uzyskano projekt 3D płytka PCB, co przedstawia rysunek 2.12.



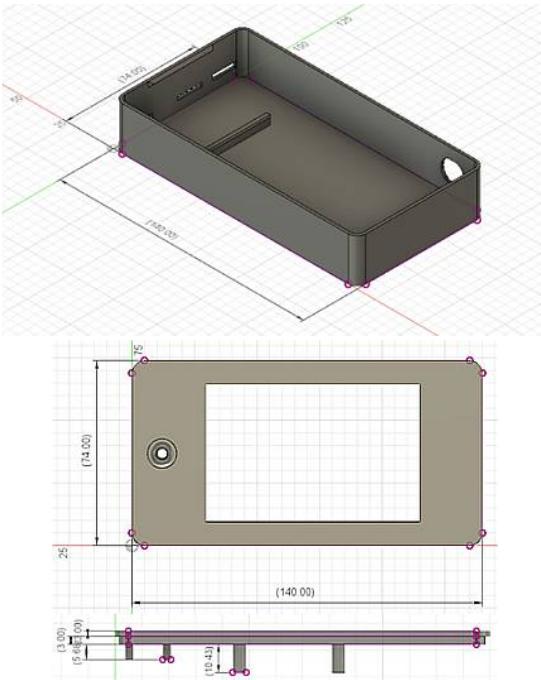
Rysunek 2.12. Projekt płyty PCB w 3D  
 Źródło: opracowanie własne

Następnie narysowano kolejne części elektroniczne, tj. ekran, akumulator i przycisk. Opisane elementy wraz z płytą PCB przedstawia (w dwóch rzutach) rysunek 2.13.



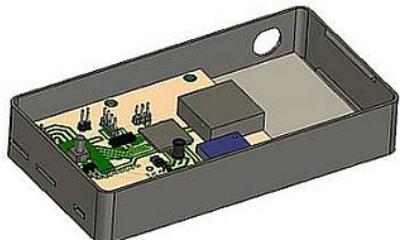
Rysunek 2.13. Płyta PCB, przycisk, ekran oraz akumulator w projekcie 3D  
 Źródło: opracowanie własne

Po narysowaniu wszystkich elementów wykonawczych rozpoczęto projektowanie obudowy. Aby zminimalizować rozmiar całości urządzenia pomiarowego zdecydowano się m.in. na zastosowanie zamknięcia na zatrzask. Podobną konstrukcję zaprojektowano dla miejsca na akumulator, aby unieruchomić element w obudowie. Zaplanowano także odpowiednio otwory na: gniazdo micro USB (ładowanie), wyrowadzenia umożliwiające programowanie, slot na kartę micro SD, przycisk oraz gniazdo dla sondy DS18B20. Projekt obudowy przedstawiono na rysunku 2.14.



Rysunek 2.14. Projekt obudowy w 3D  
 Źródło: opracowanie własne

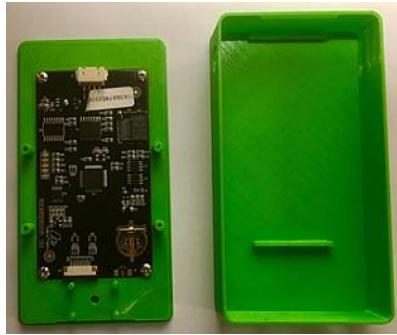
Rysunek 2.15. przedstawia różne przekroje całościowego projektu urządzenia rejestrującego parametry pogodowe.



Rysunek 2.15. Wizualizacja projektu urządzenia pomiarowego  
 Źródło: opracowanie własne

Zaprojektowaną obudowę wykonano metodą druku 3D. Finalny efekt, z już zamontowanym ekranem, przedstawiono na rysunku 2.16.





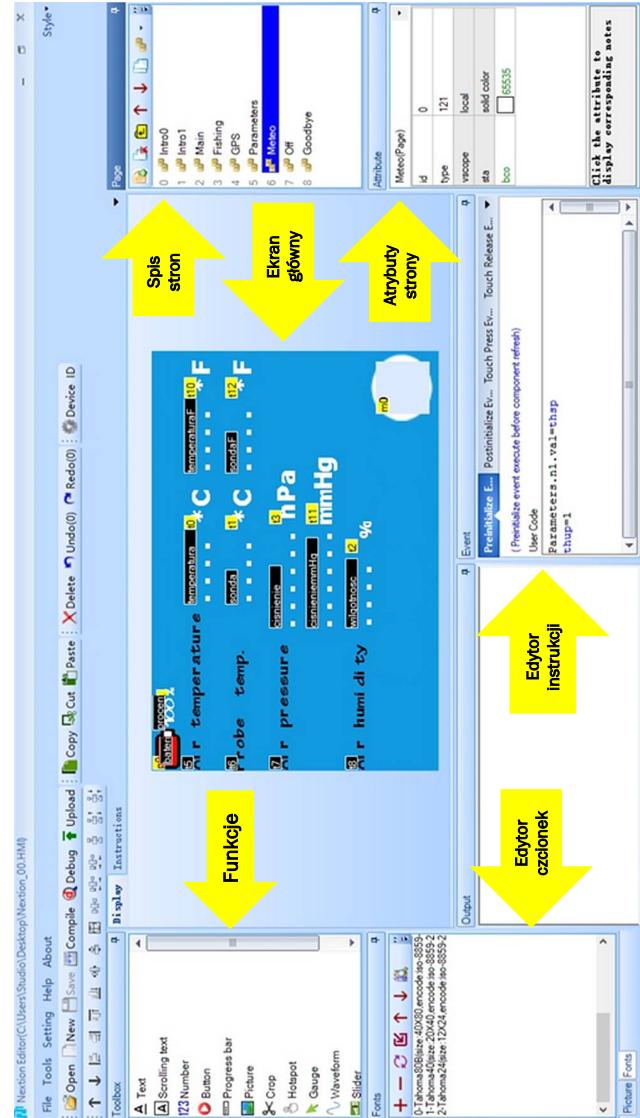
Rysunek 2.16. Obudowa urządzenia pomiarowego z zamontowanym ekranem  
 Źródło: opracowanie własne

## 2.2 Oprogramowanie systemu pomiarowego

Program do systemu rejestrującego parametry pogodowe został napisany na platformie *Arduino IDE*. W pierwszej kolejności, przy próbach napisania kodu na mikrokontroler *ATmega328* korzystano z *Arduino Uno*. Ze względu na niewystarczalną pojemność pamięci zamieniono go na mikrokontroler *ATmega2560*. Decyzja ta została podjęta w oparciu o parametry elementu, ale także ze względu na to, że taki sam układ wykorzystywany jest w popularnym urządzeniu *Arduino Mega*. Skutkowało to tym, że przy budowie prototypu korzystano właśnie z tego narzędzia, a po sprawdzeniu kodu został on ostatecznie zgrany bezpośrednio na mikrokontroler. *ATmega2560* występuje tylko w obudowie SMD, a więc możliwość jego zaprogramowania występowała dopiero po skonstruowaniu w całości płyty PCB.

W pierwszej kolejności opracowano system włączania i wyłączania urządzenia. Autor w tej części inspirował się działaniem współczesnych telefonów komórkowych, gdzie do włączenia urządzenia wystarczy przytrzymać przez ok. 3 sekundy przycisk, który powoduje uruchomienie się systemu. Podobna czynność jest wymagana w przypadku chęci wyłączenia urządzenia, gdzie dodatkowo pojawia się ekran wymuszający potwierdzenie, eliminując tym samym przypadkowe wyłączanie systemu. Aby istniała możliwość interakcji człowieka z urządzeniem pomiarowym, zaprojektowano i zaprogramowano interfejs ekranu dotykowego.

Ekrany firmy *Nextron* charakteryzują się wyjątkowo prostą obsługą poprzez ogólnodostępny edytor. Widok interfejsu programu wraz z ogólnym opisem przedstawiono na rysunku 2.17.



Rysunek 2.17. Interfejs Nextion Editor  
 Źródło: opracowanie własne

W sumie utworzono 9 kart ekranu. Zakładki *Intro0*, *Intro1* oraz *Goodbye* wykonyują jedynie funkcję wizualną podczas włączania i wyłączania urządzenia. Dłuższy czas pozwala na „spokojne” uruchomienie systemu oraz jego wyłączanie unikając tym samym utratę danych na karcie micro SD, które mogą być spowodowane właśnie m.in. nagłym odłączeniem zasilania. Karta *Main* jest główną zakładką systemu pomiarowego, z której można przejść do kolejnych ekranów. Pierwszą z opcji jest ekran *Fishing*, na której wyświetlana jest aktualna godzina, data, temperatura powietrza, temperatura zarejestrowana z sondy pomiarowej, ciśnienie atmosferyczne oraz wilgotność. Dodatkowo zaprojektowano przycisk, którego naciśnięcie powoduje wzrost licznika o „1”, a także zapis aktualnych parametrów na kartę micro SD. Strona GPS wyświetla aktualne wartości uzyskane z modułu GPS, tj. godzina i data, długość i szerokość geograficzna, prędkość poruszania się, wysokość w m n.p.m., liczbę satelitów, z którymi posiada łączność oraz jakość połączenia, wyznaczaną z wartości HDOP (ang. *Horizontal Dilution Of Precision* – parametr opisujący wpływ geometrii konstelacji satelitów dla współrzędnych płaskich na wyznaczanie pozycji). Karta *Meteo* wyświetla kolejno parametry: temperaturę powietrza i temperaturę z sondy w skali Celsjusza i Farenheita, ciśnienie powietrza w hektopaskalach oraz w milimetrach słupa rtęci, a także bezwzględną wilgotność powietrza wyrażaną w procentach. Zakładkę *Parameters* zaprojektowano w celu możliwości doboru ustawień jasności ekranu oraz czasu tzw. *sleep mode*, czyli trybu, w którym ekran jest wyłączony, ale wszystkie pomiary są nadal dokonywane. Takie opcje dodano ze względu na możliwość zmniejszenia zużycia baterii. Zadaniem strony *Off* jest, jak wyżej wspomniano, potwierdzenie lub zanegowanie chęci wyłączenia urządzenia. Dodatkowo przygotowano przycisk (z grafiką lupy), którego zadaniem miało być wyświetlanie historii pomiarów parametrów pogodowych oraz danych z modułu GPS. Ostatecznie zrezygnowano z tej opcji, ze względu na wymaganą dużą moc obliczeniową, która negatywnie wpływała na ogólną pracę urządzenia pomiarowego.

Dzięki wybraniu platformy *Arduino IDE* oraz popularnych modułów elektronicznych, całość kodu oparto na gotowych bibliotekach przypisanych odpowiednim opcjom. Umożliwiło to pobieranie parametrów pogodowych oraz z systemu GPS wykorzystując do tego gotowe funkcje. Największą trudnością w trakcie programowania okazało się przesyłanie oraz odczytywanie komend z ekranu. Biblioteka firmy *Nextion* jest bardzo rozbudowana, ale posiada jeszcze wiele nie do końca łatwo czytelnych funkcji. Spowodowało to awarię dwóch mikrokontrolerów z nieznanymi powodów oraz ostatecznie zmusiło do zmiany na mikrokontroler o większej mocy obliczeniowej i pamięci. Innym problematycznym etapem był odpowiedni dobór czasów pobierania parametrów z modułów oraz ich rejestrowanie na karcie micro SD. Element, na który były zapisywane dane jest bardzo wrażliwym urządzeniem i w przypadku, gdy program nie jest odpowiednio „uporządkowany”, bardzo często zdarza się dublowanie rejestrów, bądź ich utracenie.

Konstruowanie bazy danych odbywa się poprzez rejestrowanie wartości parametrów pogodowych w pliku z rozszerzeniem CSV. Przy włączeniu urządzenia w pier-

szej kolejności do pliku dodawany był nagłówek „*Date, Time, Latitude, Longitude, Water, Air, Humidity, Pressure, Activity*”. Umożliwiło to w kolejnym etapie rozróżnianie każdego okresu dokonywania pomiarów oraz ułatwiało to odczyt danych bezpośrednio z pliku. W celu wzbogacenia konstruowanej bazy danych zdecydowano się na zapis wartości parametrów pogodowych w odstępie 5 minut, z opisem „0” przy kolumnie opisującej aktywność oraz dodatkowo w trakcie przyciśnięcia na ekranie odpowiedniego przycisku, gdzie dla rozróżnienia w kolumnie *Activity* zapisywana była wartość „1”. Należy nadmienić, że w trakcie testowania prototypu zdecydowano się na opcję monitorowania stanu naładowania baterii, której wartość była wyświetlana na każdym ekranie w lewym górnym rogu. Niestety po zbudowaniu płyty PCB zdecydowano się zrezygnować z wyżej opisanej możliwości ze względu na wystąpienie zjawiska tzw. zasilania pasożytniczego. Wyjście modułu zasilającego, którego zadaniem jest przesyłanie napęcia odniesienia do mikrokontrolera powodowało zasilanie całego systemu poprzez jedno z wejść. W etapie konstrukcji prototypu problem ten nie był zauważony, ze względu na wykorzystanie zamiast mikrokontrolera platformy *Arduino Mega*, która posiada własne zabezpieczenia przed tego typu zasilaniem. Jednakże, pozostała możliwość na płycie PCB, w programie oraz w projekcie ekranu na powrót do opcji monitorowania stanu naładowania baterii w przypadku znalezienia innej metody na jego zbadanie. Całość programu dla systemu rejestrującego parametry pogodowe przedstawiono w załączniku B.

## ROZDZIAŁ 3

# Projekt aplikacji systemu do analizy algorytmów uczenia maszynowego w celu predykcji zdarzeń

Dział uczenia maszynowego w ostatnich latach tak się rozwinął, że spowodowało to powstanie wielu programów i platform umożliwiających projektowanie i badanie algorytmów sztucznej inteligencji. Zdecydowano się na skorzystanie z ogólnodostępnego środowiska Microsoft Azure Machine Learning Studio, którego funkcje jak i działanie przedstawiono w niniejszym rozdziale.

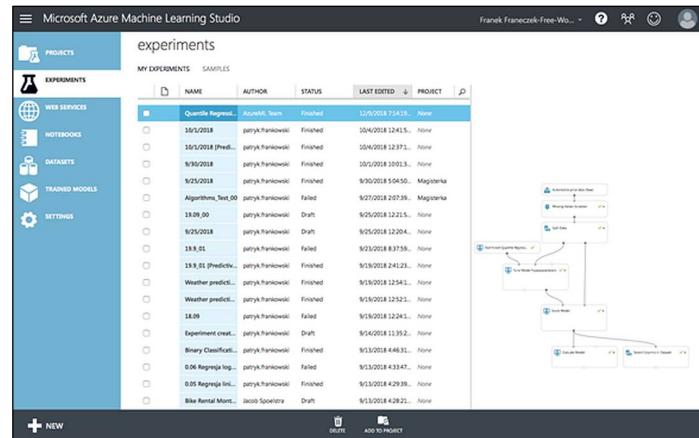
## 3.1. Microsoft Azure Machine Learning Studio

Microsoft Azure Machine Learning Studio jest narzędziem dostępnym z poziomu przeglądarki internetowej, a więc nie posiada wymagań sprzętowych ani nie wymaga żadnych instalacji. Istnieją różne wersje oprogramowania, m.in. bezpłatna, która posiada te same funkcje, co wersja płatna, jednakże charakteryzuje się mniejszą mocą obliczeniową, ograniczoną dostępną przestrzenią analizowanych baz danych (do 10 GB danych) oraz brakiem możliwości udostępniania swojej aplikacji na szeroką skalę. Zdecydowano się na przeprowadzenie badań właśnie w tej wersji programu.

Po zalogowaniu się na koncie Microsoft i włączeniu narzędzia Microsoft Azure Machine Learning Studio, wyświetla się panel kontrolny, który przedstawiono na rysunku 3.1.

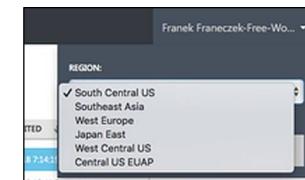
Po lewej stronie ekranu znajdują się kolejno zakładki:

- *Projects* – daje możliwość zarządzania całą grupą modeli w ramach danego projektu,
- *Experiments* – z tego poziomu można tworzyć i zarządzać modelami analitycznymi,
- *Web Services* – umożliwia udostępnianie modeli,
- *Notebooks* – narzędzie, które umożliwia dokładniejszą eksplorację danych,
- *Datasets* – spis zbiorów danych, które są dostarczone platformie,
- *Trained Models* – spis skompilowanych modeli do postaci binarnej,
- *Settings* – ustawienia programu.



Rysunek 3.1. Główny ekran narzędzia Microsoft Azure Machine Learning Studio  
Źródło: opracowanie własne

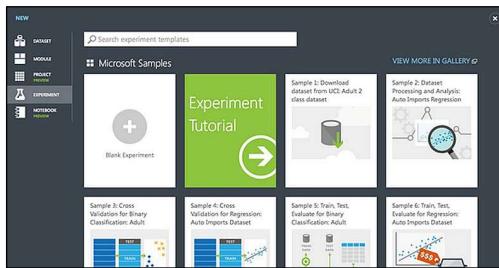
Należy nadmienić, że platforma Microsoft Azure Machine Learning Studio posiada funkcję pracy w różnych przestrzeniach roboczych, co przedstawiono na rysunku 3.2.



Rysunek 3.2. Przestrzenie robocze w Microsoft Azure Machine Learning Studio  
Źródło: opracowanie własne

Funkcja ta umożliwia pracę w tym samym czasie na kilku różnych projektach, czy też w różnych zespołach. Istnieje możliwość wyboru regionu jak i odpowiedniego Workspace. Przestrzeń robocza dla darmowej wersji platformy znajduje się w ośrodkach obliczeniowych w Stanach Zjednoczonych.

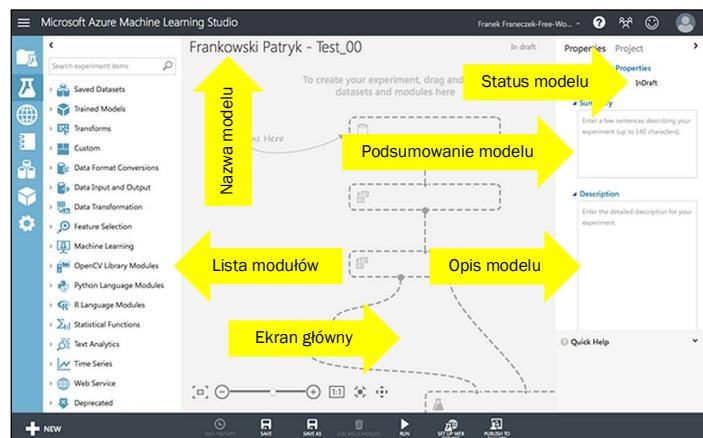
W celu utworzenia nowego modelu analitycznego (zwany dalej także eksperymentem) należy wejść w opcję „New”, gdzie ukazuje się ekran przedstawiony na rysunku 3.3.



Rysunek 3.3. Ekran wyboru możliwości tworzenia nowych eksperymentów  
 Źródło: opracowanie własne

Z tego poziomu programu można m.in. dodać nową bazę danych, czy też stworzyć projekt. Jak przedstawiono na rysunku 3.3 poza możliwością stworzenia nowego modelu analitycznego, firma Microsoft oferuje bardzo rozbudowaną galerię gotowych, przetestowanych modeli. Jest to bardzo przydatna funkcja programu, która umożliwia m.in. prostszą naukę posługiwania się odpowiednimi narzędziami, zrozumienie danych działań uczenia maszynowego, a także może dostarczyć inspiracji w tworzeniu własnego eksperymentu.

Po wejściu w opcję tworzenia nowego eksperymentu wyświetla się ekran przedstawiony na rysunku 3.4 wraz z opisanymi opcjami.



Rysunek 3.4. Ekran tworzenia nowego eksperymentu  
 Źródło: opracowanie własne

Jak przedstawia rysunek 3.4 platforma *Microsoft Azure Machine Learning Studio* dostarcza funkcje, które umożliwiają dokładny opis całości tworzonego projektu, co jest wygodnym narzędziem przy projektowaniu w grupie bądź konstruowaniu skomplikowanych modeli analitycznych.

Rozbudowany dział wsparcia oraz dostarczane opcje programu wręcz zachęcają do badań w sferze uczenia maszynowego, czyniąc zdobywanie umiejętności w tej dziedzinie o wiele prostszym i czytelnym. W niniejszym podrozdziale ukazano tylko podstawowe możliwości *Microsoft Azure Machine Learning Studio*. Tworzenie modelu analitycznego oraz dodatkowe możliwości programu omówiono w kolejnych etapach pracy magisterskiej.

### 3.2. Integracja z urządzeniem pomiarowym oraz baza danych

Zapis mierzonych parametrów na pamięci przenośnej umożliwia odczyt danych z poziomu komputera. Fragment jednego z pliku z zapisanymi wartościami przedstawiono na rysunku 3.5. Konstrukcja zapisu uwzględnia wymogi w tworzeniu bazy danych dla platformy *Microsoft Azure Machine Learning*.

Date	Time	Latitude	Longitude	Water	Air	Humidity	Pressure	Activity
23-08-2018	14:32:56	52.984584	15.236624	25.62	33.7	39.7	1005.7	0
23-08-2018	14:38:00	52.984573	15.236633	25.62	34.1	37.7	1005.7	0
23-08-2018	14:43:08	52.984577	15.236643	25.69	34.1	37.9	1005.6	0
23-08-2018	14:47:32	52.984584	15.236637	25.69	34.4	37.2	1005.6	1
23-08-2018	14:52:00	52.984584	15.236642	25.69	34.4	37.2	1005.6	0
23-08-2018	14:53:00	52.984596	15.236642	25.69	34.2	36.7	1005.5	0
23-08-2018	14:58:00	52.984607	15.236632	25.75	34.4	37.1	1005.5	0
23-08-2018	15:03:20	52.984603	15.236631	25.81	34.5	35.6	1005.5	0
23-08-2018	15:07:20	52.984592	15.236633	25.81	34.6	35.6	1005.5	1
23-08-2018	15:12:00	52.984584	15.236637	25.81	34.6	35.6	1005.5	0
23-08-2018	15:10:20	52.984588	15.236639	25.81	34.6	35.7	1005.5	0
23-08-2018	15:12:20	52.984584	15.236641	25.81	34.6	35.3	1005.2	1
23-08-2018	15:13:20	52.984580	15.236640	25.81	34.5	37.6	1005.2	0
23-08-2018	15:18:20	52.984573	15.236632	25.87	34.2	35.7	1005.2	0
23-08-2018	15:23:20	52.984573	15.236637	25.87	34.2	35.7	1005.2	0
23-08-2018	15:25:22	52.984584	15.236642	25.87	33.2	39.3	1005.1	0
23-08-2018	15:33:32	52.984594	15.236647	25.87	33.4	38.9	1005.1	0
23-08-2018	15:38:32	52.984580	15.236657	25.94	34.0	36.6	1005.0	0
23-08-2018	15:43:32	52.984588	15.236632	25.94	34.3	36.5	1005.0	0
23-08-2018	15:48:32	52.984584	15.236651	25.94	34.3	36.5	1004.9	0
23-08-2018	15:53:44	52.984582	15.236626	25.94	34.4	36.4	1004.8	0
23-08-2018	15:58:44	52.984584	15.236635	25.97	35.0	36.5	1004.7	0
23-08-2018	16:03:44	52.984569	15.236655	25.97	35.3	37.9	1004.6	0
23-08-2018	16:08:44	52.984569	15.236647	25.94	35.7	43.7	1004.6	0
23-08-2018	16:13:44	52.984569	15.236647	25.94	35.7	43.7	1004.6	0
23-08-2018	16:18:56	52.984592	15.236663	26.01	36.6	43.8	1004.5	0
23-08-2018	16:23:56	52.984596	15.236663	26.08	36.0	43.4	1004.5	0
23-08-2018	16:28:56	52.984573	15.236654	26.08	34.0	40.2	1004.4	0
23-08-2018	16:33:56	52.984573	15.236654	26.08	35.4	40.4	1004.3	0
23-08-2018	16:38:56	52.984580	15.236641	26.08	35.4	40.8	1004.3	0
23-08-2018	16:43:56	52.984580	15.236646	26.08	35.5	40.9	1004.3	0
23-08-2018	16:48:56	52.984589	15.236641	26.06	35.3	43.7	1004.3	1
23-08-2018	16:49:00	52.984554	15.236648	26.06	34.5	35.8	1004.2	0
23-08-2018	17:00:00	52.984554	15.236648	26.06	34.5	35.8	1004.2	0
23-08-2018	17:23:57	52.985283	15.235722	25.62	28.6	43.7	1003.7	0
23-08-2018	17:42:51	52.985267	15.235714	25.69	28.3	42.0	1003.6	0
23-08-2018	17:47:51	52.985263	15.235722	25.69	28.0	44.8	1003.6	0
23-08-2018	17:51:51	52.985260	15.235719	25.75	27.9	46.2	1003.6	1

Rysunek 3.5. Fragment pliku z zarejestrowanymi wartościami z urządzenia pomiarowego  
 Źródło: opracowanie własne

Zdecydowano się na zapis pliku w formacie CSV (ang. *Comma-Separated Values*). Jak wspomniano w poprzednim rozdziale, każde uruchomienie urządzenia powoduje powstanie linii tytułowej. Jest to fragment, który przede wszystkim ułatwia odczyt

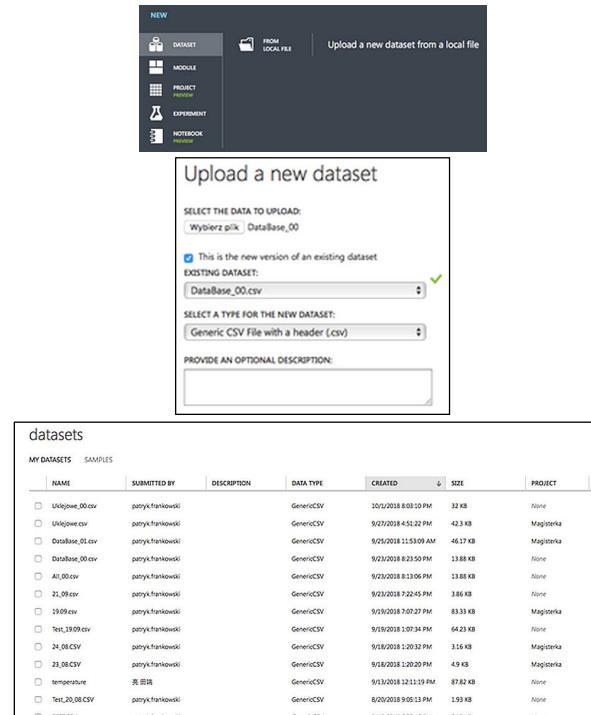
parametrów, a także umożliwia platformie *Microsoft Azure Machine Learning Studio* odpowiednie przypisanie nazw kolumnom. W każdej z linii zapisane są kolejno wartości: data, czas, szerokość geograficzna, długość geograficzna, temperatura wody, temperatura powietrza, ciśnienie atmosferyczne oraz aktywność. Ten ostatni parametr przyjmuje jedynie dwie wartości: „0”, które oznacza dokonanie pomiaru bez ingerencji użytkownika urządzenia (częstotliwość takiego pomiaru ustawiono na 5 minut) oraz „1”, które oznacza zapis pomiarów po wcisnięciu odpowiedniego przycisku na skonstruowanym systemie. W badanych warunkach (połowa ryb) wartość „1” w kolumnie *Activity* oznaczało „branie” ryby. Postanowiono wprowadzić oba stany ze względu na możliwość dokładniejszego rozróżnienia parametrów, w których ta aktywność była większa lub mniejsza. Ponadto stwarza to możliwość wyznaczenia konkretnego parametru, który ma największy wpływ na badaną aktywność.

Pomimo wielu prób utworzenia możliwości dodawania nowych danych do już gotowego modelu analitycznego (na poziomie wstępnego przetwarzania danych), zdecydowano się na tworzenie jednej bazy danych z kilku dni pomiarów, łącząc pomiary w arkuszu kalkulacyjnym. Aplikacja *Microsoft Azure Machine Learning Studio* umożliwia dodawanie kolejnych baz danych do gotowej aplikacji, aczkolwiek wymaga to wysokiej znajomości języka programistycznego *Python* oraz *R*. W trakcie łączenia rejestrów nie wprowadzano żadnych zmian w danych, w celu weryfikacji opcji programowych poprawiania błędów powstałych w trakcie dokonywania pomiarów. Należy wspomnieć, że dodano jedynie kolumnę („*DateTime*”), która łączy wartości daty i czasu w jedną wartość, aby sprawdzić niektóre funkcje platformy *Microsoft Azure Machine Learning Studio*, wymagające przedstawienia właśnie tych parametrów w innym formacie. Przygotowaną bazę danych w arkuszu kalkulacyjnym zilustrowano na rysunku 3.6.

2018-08-24 18:35:06	2018-08-24 18:35:06	52.858436	15.239754	23.56	22.2	58.3	1002.9	0
2018-08-24 18:40:06	2018-08-24 18:40:06	52.858436	15.239754	23.56	22	60.7	1000	0
2018-08-24 18:46:06	2018-08-24 18:46:06	52.858436	15.239754	23.56	21.9	61.4	1003.1	0
2018-08-24 19:00:06	2018-08-24 19:00:06	52.858436	15.239754	23.56	21.7	61.4	1003.1	0
2018-08-24 19:15:06	2018-08-24 19:15:06	52.858436	15.239754	23.56	21.5	61.4	1003.1	0
2018-08-24 19:30:06	2018-08-24 19:30:06	52.858436	15.239754	23.56	21.3	61.4	1003.1	0
2018-08-24 19:45:06	2018-08-24 19:45:06	52.858436	15.239754	23.56	21.1	61.4	1003.1	0
2018-08-24 20:00:06	2018-08-24 20:00:06	52.858436	15.239754	23.56	20.9	61.4	1003.1	0
2018-08-24 20:15:06	2018-08-24 20:15:06	52.858436	15.239754	23.56	20.7	61.4	1003.1	0
2018-08-24 20:30:06	2018-08-24 20:30:06	52.858436	15.239754	23.56	20.5	61.4	1003.1	0
2018-08-24 20:45:06	2018-08-24 20:45:06	52.858436	15.239754	23.56	20.3	61.4	1003.1	0
2018-08-24 21:00:06	2018-08-24 21:00:06	52.858436	15.239754	23.56	20.1	61.4	1003.1	0
2018-08-24 21:15:06	2018-08-24 21:15:06	52.858436	15.239754	23.56	19.9	61.4	1003.1	0
2018-08-24 21:30:06	2018-08-24 21:30:06	52.858436	15.239754	23.56	19.7	61.4	1003.1	0
2018-08-24 21:45:06	2018-08-24 21:45:06	52.858436	15.239754	23.56	19.5	61.4	1003.1	0
2018-08-24 22:00:06	2018-08-24 22:00:06	52.858436	15.239754	23.56	19.3	61.4	1003.1	0
2018-08-24 22:15:06	2018-08-24 22:15:06	52.858436	15.239754	23.56	19.1	61.4	1003.1	0
2018-08-24 22:30:06	2018-08-24 22:30:06	52.858436	15.239754	23.56	18.9	61.4	1003.1	0
2018-08-24 22:45:06	2018-08-24 22:45:06	52.858436	15.239754	23.56	18.7	61.4	1003.1	0
2018-08-24 22:59:06	2018-08-24 22:59:06	52.858436	15.239754	23.56	18.5	61.4	1003.1	0
2018-08-25 00:00:06	2018-08-25 00:00:06	52.858436	15.239754	23.56	18.3	61.4	1003.1	0
2018-08-25 00:15:06	2018-08-25 00:15:06	52.858436	15.239754	23.56	18.1	61.4	1003.1	0
2018-08-25 00:30:06	2018-08-25 00:30:06	52.858436	15.239754	23.56	17.9	61.4	1003.1	0
2018-08-25 00:45:06	2018-08-25 00:45:06	52.858436	15.239754	23.56	17.7	61.4	1003.1	0
2018-08-25 01:00:06	2018-08-25 01:00:06	52.858436	15.239754	23.56	17.5	61.4	1003.1	0
2018-08-25 01:15:06	2018-08-25 01:15:06	52.858436	15.239754	23.56	17.3	61.4	1003.1	0
2018-08-25 01:30:06	2018-08-25 01:30:06	52.858436	15.239754	23.56	17.1	61.4	1003.1	0
2018-08-25 01:45:06	2018-08-25 01:45:06	52.858436	15.239754	23.56	16.9	61.4	1003.1	0
2018-08-25 02:00:06	2018-08-25 02:00:06	52.858436	15.239754	23.56	16.7	61.4	1003.1	0
2018-08-25 02:15:06	2018-08-25 02:15:06	52.858436	15.239754	23.56	16.5	61.4	1003.1	0
2018-08-25 02:30:06	2018-08-25 02:30:06	52.858436	15.239754	23.56	16.3	61.4	1003.1	0
2018-08-25 02:45:06	2018-08-25 02:45:06	52.858436	15.239754	23.56	16.1	61.4	1003.1	0
2018-08-25 03:00:06	2018-08-25 03:00:06	52.858436	15.239754	23.56	15.9	61.4	1003.1	0
2018-08-25 03:15:06	2018-08-25 03:15:06	52.858436	15.239754	23.56	15.7	61.4	1003.1	0
2018-08-25 03:30:06	2018-08-25 03:30:06	52.858436	15.239754	23.56	15.5	61.4	1003.1	0
2018-08-25 03:45:06	2018-08-25 03:45:06	52.858436	15.239754	23.56	15.3	61.4	1003.1	0
2018-08-25 04:00:06	2018-08-25 04:00:06	52.858436	15.239754	23.56	15.1	61.4	1003.1	0
2018-08-25 04:15:06	2018-08-25 04:15:06	52.858436	15.239754	23.56	14.9	61.4	1003.1	0
2018-08-25 04:30:06	2018-08-25 04:30:06	52.858436	15.239754	23.56	14.7	61.4	1003.1	0
2018-08-25 04:45:06	2018-08-25 04:45:06	52.858436	15.239754	23.56	14.5	61.4	1003.1	0
2018-08-25 05:00:06	2018-08-25 05:00:06	52.858436	15.239754	23.56	14.3	61.4	1003.1	0
2018-08-25 05:15:06	2018-08-25 05:15:06	52.858436	15.239754	23.56	14.1	61.4	1003.1	0
2018-08-25 05:30:06	2018-08-25 05:30:06	52.858436	15.239754	23.56	13.9	61.4	1003.1	0
2018-08-25 05:45:06	2018-08-25 05:45:06	52.858436	15.239754	23.56	13.7	61.4	1003.1	0
2018-08-25 06:00:06	2018-08-25 06:00:06	52.858436	15.239754	23.56	13.5	61.4	1003.1	0
2018-08-25 06:15:06	2018-08-25 06:15:06	52.858436	15.239754	23.56	13.3	61.4	1003.1	0
2018-08-25 06:30:06	2018-08-25 06:30:06	52.858436	15.239754	23.56	13.1	61.4	1003.1	0
2018-08-25 06:45:06	2018-08-25 06:45:06	52.858436	15.239754	23.56	12.9	61.4	1003.1	0
2018-08-25 07:00:06	2018-08-25 07:00:06	52.858436	15.239754	23.56	12.7	61.4	1003.1	0
2018-08-25 07:15:06	2018-08-25 07:15:06	52.858436	15.239754	23.56	12.5	61.4	1003.1	0
2018-08-25 07:30:06	2018-08-25 07:30:06	52.858436	15.239754	23.56	12.3	61.4	1003.1	0
2018-08-25 07:45:06	2018-08-25 07:45:06	52.858436	15.239754	23.56	12.1	61.4	1003.1	0
2018-08-25 08:00:06	2018-08-25 08:00:06	52.858436	15.239754	23.56	11.9	61.4	1003.1	0
2018-08-25 08:15:06	2018-08-25 08:15:06	52.858436	15.239754	23.56	11.7	61.4	1003.1	0
2018-08-25 08:30:06	2018-08-25 08:30:06	52.858436	15.239754	23.56	11.5	61.4	1003.1	0
2018-08-25 08:45:06	2018-08-25 08:45:06	52.858436	15.239754	23.56	11.3	61.4	1003.1	0
2018-08-25 09:00:06	2018-08-25 09:00:06	52.858436	15.239754	23.56	11.1	61.4	1003.1	0
2018-08-25 09:15:06	2018-08-25 09:15:06	52.858436	15.239754	23.56	10.9	61.4	1003.1	0
2018-08-25 09:30:06	2018-08-25 09:30:06	52.858436	15.239754	23.56	10.7	61.4	1003.1	0
2018-08-25 09:45:06	2018-08-25 09:45:06	52.858436	15.239754	23.56	10.5	61.4	1003.1	0
2018-08-25 10:00:06	2018-08-25 10:00:06	52.858436	15.239754	23.56	10.3	61.4	1003.1	0
2018-08-25 10:15:06	2018-08-25 10:15:06	52.858436	15.239754	23.56	10.1	61.4	1003.1	0
2018-08-25 10:30:06	2018-08-25 10:30:06	52.858436	15.239754	23.56	9.9	61.4	1003.1	0
2018-08-25 10:45:06	2018-08-25 10:45:06	52.858436	15.239754	23.56	9.7	61.4	1003.1	0
2018-08-25 10:59:59	2018-08-25 10:59:59	52.858436	15.239754	23.56	9.5	61.4	1003.1	0

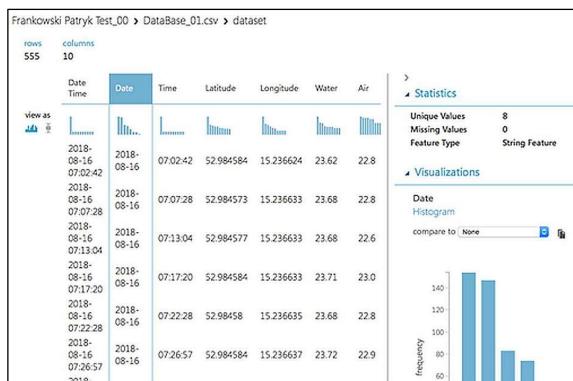
Rysunek 3.6. Fragment pliku z zarejestrowanymi wartościami z urządzenia pomiarowego  
Źródło: opracowanie własne

Dodarcie bazy danych do platformy *Microsoft Azure Machine Learning Studio* odbywa się poprzez wcisnięcie w głównym ekranie przycisku *New* oraz wybranie odpowiedniego pliku. Zapisane bazy danych są widoczne w zakładce *Datasets*. Etapy dodawania do programu bazy danych przedstawiono na rysunku 3.7.



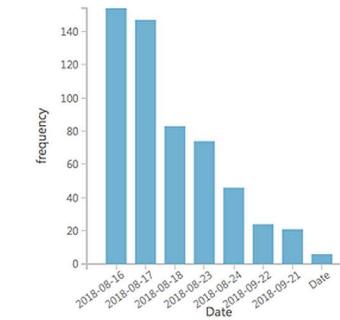
Aby móc przetworzyć dane do postaci, która umożliwia trenowanie algorytmów, w pierwszej kolejności przeanalizowano dane, które uzyskano w procesie rejestracji pomiarów skonstruowanym urządzeniem. Po dodaniu żądanej bazy danych do modelu analitycznego, zwizualizowano jej wartości poprzez udostępnioną funkcję na platformie

Microsoft Azure Machine Learning Studio. Widok takiego ekranu przedstawiono na rysunku 3.8.



Rysunek 3.8. Wizualizacja bazy danych w Microsoft Azure Machine Learning Studio  
Źródło: opracowanie własne

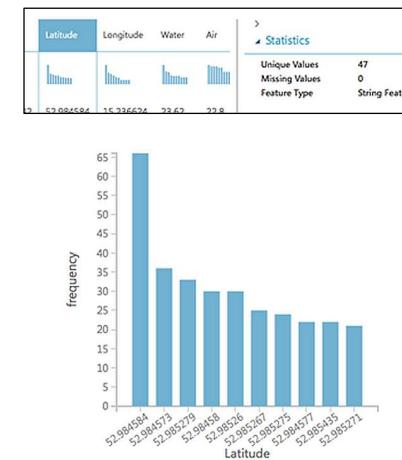
Główny ekran przedstawia: w kolumnach nazwy poszczególnych parametrów, a w wierszach kolejne pomiary. W lewej górnej stronie ekranu przedstawiona jest ilość wierszy (liczba pomiarów) oraz kolumn (liczba parametrów). Każdą kolumnę można osobno przeanalizować, do czego służą opcje znajdujące się po prawej stronie ekranu. *Unique Values* opisuje, ile znajduje się wartości danej kategorii, *Missing Values* określa liczbę brakujących wartości, a *Feature Type* przedstawia typ danych. W przypadku rysunku 3.8, analizowany jest parametr daty i można odczytać, że pomiary były wykonywane przez 8 różnych dni (różne daty), nie ma brakujących wartości oraz że jest to typ *String*. Poniżej znajduje się histogram, który przedstawiono na rysunku 3.9. Dotyczy on omawianego wyżej parametru, jakim jest data.



Rysunek 3.9. Histogram opisujący parametr daty  
Źródło: opracowanie własne

Załączony histogram w pierwszej kolejności daje projektantowi sygnał, że badania były wykonywane jednak przez 7 dni, a ósmą wartością jest błędnie zinterpretowana napis „Date”, który powstaje przy każdym uruchomieniu urządzenia. Kolejną informacją jest ilość zarejestrowanych próbek danego dnia.

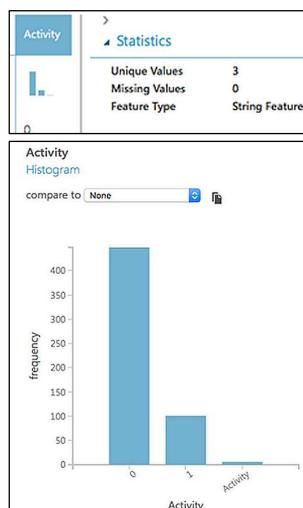
Następny przykład możliwości analizowania danych przedstawiono na rysunku 3.10.



Rysunek 3.10. Wizualizacja parametru szerokości geograficznej  
Źródło: opracowanie własne

Na załączonym rysunku 3.10 analizowane są pomiary szerokości geograficznej. Przy około 555 rejestrów zarejestrowano jedynie 47 różnych wartości tego parametru. Dogłębniej analizując histogram, na podstawie małych różnic pomiarów, można stwierdzić, że badanie było przeprowadzone w tej samej okolicy (wyniki długości geograficznej także oscylują w małym zakresie wartości). Dla osoby, która nie przeprowadzała badań, a otrzymuje tylko dane do przeanalizowania może być to kluczowa informacja podczas przeprowadzania kolejnych eksperymentów.

Najważniejszym parametrem z punktu widzenia niniejszej pracy, jest wartość aktywności. Wizualizację danych *Activity* przedstawiono na rysunku 3.11.



Rysunek 3.11. Wizualizacja parametru aktywności

Źródło: opracowanie własne

Pierwszym spostrzeżeniem analizując parametr *Activity* jest błędne wskazanie, że istnieją 3 odpowiedzi. Badając histogram można zauważać proporcje ilości wykonanych pomiarów w momencie „brania” ryby oraz w przypadku braku aktywności, a także błędna interpretację wyrażenia „*Activity*”.

Bardzo przydatną funkcją w tym etapie jest możliwość zestawienia i porównania wybranych parametrów. Przykładowo na rysunku 3.12 przedstawiono porównanie wartości aktywności do dat wykonywanych pomiarów.

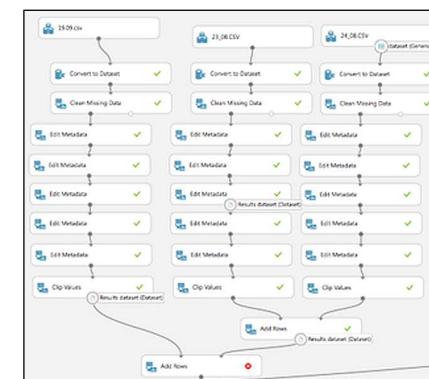


Rysunek 3.12. Zestawienie wartości aktywności do dat wykonywanych pomiarów  
Źródło: opracowanie własne

Takie zestawienie jak wskazano na rysunku 3.12 pozwala m.in. na pierwsze uwagi odnośnie wpływu danego parametru na aktywność „brania” ryb.

Po wstępnej analizie bazy danych wywnioskowano kilka spostrzeżeń. Zadowalającym jest, że w żadnym parametrze nie pojawiła się pusta wartość, co świadczy nie tylko o poprawności rejestrowanych pomiarów przez urządzenie, a o prawidłowym samym procesie pomiaru i zapisu. Istotnym także jest fakt występowania wielokrotnie w miejscowości nazewnictwa poszczególnych kolumn oraz błędne przypisanie typów danych, co może negatywnie wpływać na wyniki dalszych eksperymentów.

Jak wspomniano w poprzednim podrozdziale w pierwszej kolejności wykonywano próbę połączenia poszczególnych baz danych bezpośrednio w programie Microsoft Azure Machine Learning Studio. Jedno z takich podejść przedstawiono na rysunku 3.13.



Rysunek 3.13. Próba połączenia trzech baz danych w jedną  
Źródło: opracowanie własne

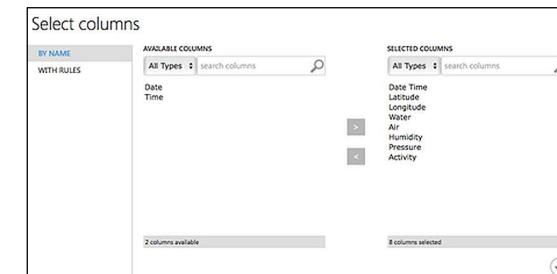
Ostatecznie zdecydowano się na wersję, gdzie jest dodawana jedna baza danych, która jestłączona z wielu pomiarów w arkuszu kalkulacyjnym.

Wstępne przetwarzanie danych jest etapem tworzenia modelu, gdzie istnieje bardzo duża możliwość wyboru sposobu podejścia do danych, wpływając jednocześnie na ostateczny wynik. Pierwszorzędnym celem, niezależnie od wybranej w dalszej części taktyki przeprowadzania badania, jest wyczyszczenie niechcianych wartości z bazy danych oraz ustawnie odpowiednich typów danych. Opisany etap przedstawiono na rysunku 3.14.



Rysunek 3.14. Model wstępnego przetwarzania danych  
Źródło: opracowanie własne

Pierwszy bloczek od góry na rysunku 3.14 jest dołączoną bazą danych, a każda z kolejnych funkcji jest wewnętrzna opcją Microsoft Azure Machine Learning Studio. W pierwszej kolejności wybrano parametry, które mają być brane pod uwagę w dalszej części modelowania, co przedstawiono na rysunku 3.15.



Rysunek 3.15. Funkcja Select Columns  
Źródło: opracowanie własne

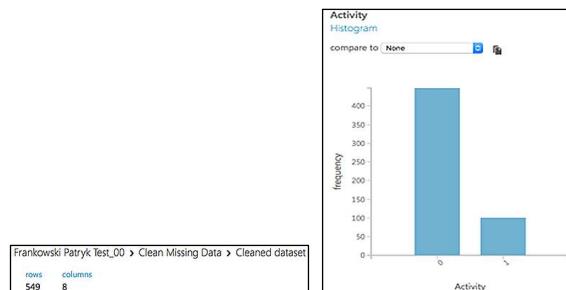
Przedstawiona na rysunku 3.15. funkcja Select Columns wykonuje odpowiednie operacje, aby z bazy danych pozbyć się niepożądanych parametrów. W tym przypadku usuwane są wartości daty i czasu, które zostały wcześniej, z poziomu arkusza kalkulacyjnego, złączone w jedną kolumnę. Taka decyzja zostanie wyjaśniona w dalszej części pracy.

W celu pozbycia się wszystkich wierszy zawierających nazwy kolumn, a nie konkretnych wartości pomiarowych skorzystano z dwóch funkcji: Convert to Dataset oraz Clean Missing Data. Etap ten przedstawiono na rysunku 3.16.

Rysunek 3.16. Funkcja Convert to Dataset i Clean Missing Data  
Źródło: opracowanie własne

Powyzsza operacja powoduje w pierwszej kolejności wyszukanie komórek, które zaczynają się od wyrażenia „Date Time” i usunięcie ich (zastąpienie pustymi komórkami). W kolejnym kroku, funkcja Clean Missing Data wyszukuje wszystkie puste komórki i usuwa całe linie danych powiązanych z tymi komórkami. W ten sposób po-

zbyto się zbędnych linijek zawierających powtarzającą się linię tytułową. Na rysunku 3.17 przedstawiono dowód na poprawne usunięcie opisywanych wartości.

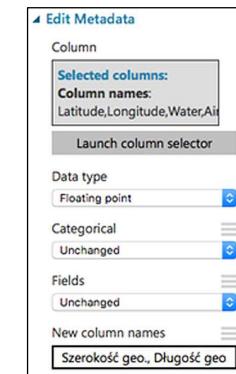


Rysunek 3.17. Baza danych po usunięciu zbytnich linii

Źródło: opracowanie własne

Jak widać na rysunku 3.17 liczba wierszy (badan) zmniejszyła się z 555 do 549, co sugeruje, że istniało 6 linii zawierających zbędne informacje. Jak wspomniano powyżej, badania przeprowadzano przez 7 dni, ale jedna linia z opisem parametrów pozostała jako ta, która widnieje na samej górze bazy danych i poprawnie spełnia swoją rolę. Dodatkowo, prawa część rysunku 3.17 ukazuje ponownie histogram *Activity*, z tymże można już przyjąć tylko dwie poprawne wartości.

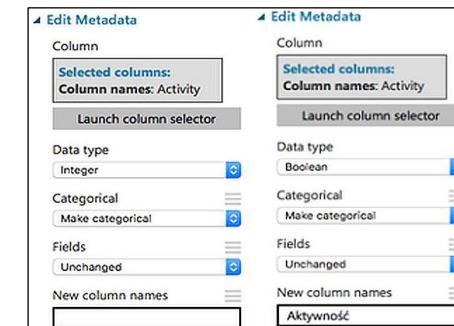
Kolejną ważną rzeczą w etapie wstępnego przetwarzania danych jest nadanie odpowiednich typów danych poszczególnym wartościom. Platforma Azure Machine Learning Studio posiada wbudowaną funkcję, która automatycznie analizuje informacje i przypisuje im odpowiednie typy danych. Niemniej jednak, po załadowaniu bazy danych wszystkie dane były błędnie określone przez program, jako *String*. Było to spowodowane wcześniej już opisanymi pojedynczymi liniami tytułowymi, które były „wpłatane” pomiędzy wartości. Większość mierzonych informacji to dane numeryczne. Rysunek 3.18 przedstawia ustawienie parametrów opcji *Edit Metadata* w celu zmiany typu danych ze *String* na *Float*.



Rysunek 3.18. Zmiana typów danych na *Float*

Źródło: opracowanie własne

Jak widać na rysunku 3.18 zdecydowano się dodatkowo przy tej operacji na zmianę nazewnictwa z języka angielskiego na język polski. W ten sposób zmieniono dane tj. szerokości i długości geograficznej, temperatury wody i powietrza, ciśnienia atmosferycznego i wilgotności. Parametr *Activity* przyjmuje jedynie dwie wartości: prawda lub fałsz. Zmusiło to zasugerowanie modelowi, iż jest to typ danych *Boolean*. Tę operację przedstawiono na rysunku 3.19.



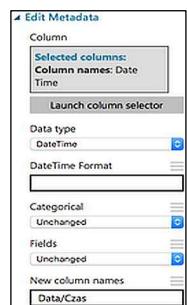
Rysunek 3.19. Zmiana parametru *Activity* na typ danych *Boolean*

Źródło: opracowanie własne

Ustalenie typu danych dla wartości *Activity* musiała odbyć się dwuetapowo. W pierwszej kolejności zmieniono na *Integer*, aby następnie była możliwość zmiany ty-

pu na *Boolean*. Podobnie jak w poprzednim przypadku zmieniono nazwę na nazewnicę w języku polskim, a dodatkowo ustawiono kategoryczny typ danych.

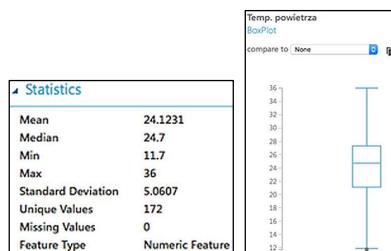
Aby skorzystać z pełni możliwości platformy zmieniono także typ danych opisujących datę i czas. Te dwie nietypowe wartości posiadają pośród cyfr inne znaki, np. dla daty „16-01-1992”, gdzie myślniki nie mogą być określone, jako typ *Float*, czy też *Integer*. Microsoft Azure Machine Learning Studio może określić taki typ danych, jako *DateTime*. Wymaga to jedynie umieszczenia obu wartości w jednej kolumnie, stąd wspomniano o tej operacji w poprzednim podrozdziale. Ustalenie tego typu danych odbyło się, tak samo jak w poprzednich przykładach, poprzez użycie funkcji *Edit Metadata*. Etap ten przedstawiono na rysunku 3.20.



Rysunek 3.20. Zmiana parametrów daty i czasu na typ danych *DateTime*

Źródło: opracowanie własne

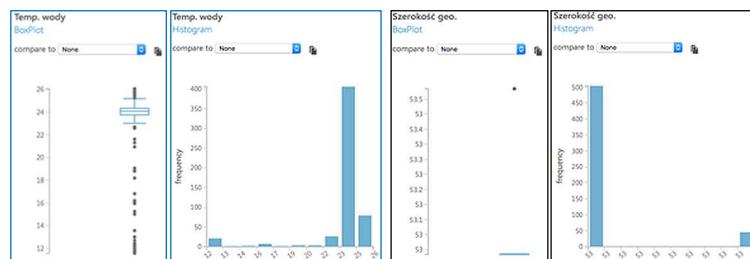
Zmiany opisane powyżej niosą za sobą, co najmniej trzy korzyści. Ustalenie konkretnych typów danych umożliwia: wykorzystanie większej ilości dostępnych funkcji, czytelniejsze manewrowanie w modelu analitycznym oraz dostarcza dodatkowych informacji odnośnie danych. Rysunek 3.21 przedstawia nowe statystyki, jakie można uzyskać po nadaniu numerycznych typów danych odpowiednim parametrom.



Rysunek 3.21. Przykład danych statystycznych pomiarów temperatury powietrza

Źródło: opracowanie własne

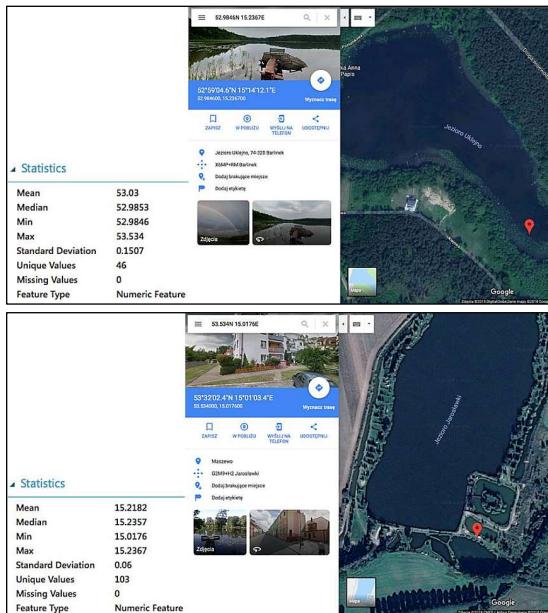
Lewa część rysunku 3.21 przedstawia przykład danych statystycznych, gdzie wyznaczona jest mediana, wartość średnia, minimalna oraz maksymalna, standardowe odchylenie i ilość poszczególnych informacji. Wartości numeryczne umożliwiają dodatkowo wyznaczenie wykresów pudełkowych. Na przykładzie rysunku 3.21 widać, że jest jedna wartość (zaznaczona punktem) odstojąca poza wyliczoną normę. Istnieją funkcje (m.in. *Clip Values*), których zadaniem jest automatyczne wyczyszczenie wartości odstających od normy. Jednakże mimo takich możliwości, w *data science* najważniejszą cechą jest analizowanie danych i odpowiedni dobór narzędzi. Jako przykład postawionej tezy posłużą wykresy przedstawione na rysunku 3.22.



Rysunek 3.22. Wykresy pomiarów temperatury wody (po lewej) oraz szerokości geograficznej (po prawej)

Źródło: opracowanie własne

Na rysunku 3.22 przedstawiono wykresy pudełkowe oraz histogramy zmierzonych wartości temperatury wody i szerokości geograficznej. Pierwszy wykres po lewej stronie może wskazywać na poważne błędy pomiarowe. Analizując dodatkowo histogram można jednak stwierdzić, że wykres pudełkowy jest tak charakterystyczny ze względu na bardzo dużą częstotliwość zarejestrowania wartości temperatury z zakresu 23°C – 25°C. Jednakże, posiadając informacje o formie przeprowadzanych rejestrów wiadomo jest, że taka różnica wynika z liczby pomiarów dokonywanych w sierpniu w stosunku do tych wykonywanych we wrześniu, kiedy to występują znaczne różnice temperatury i ilości opadów atmosferycznych. Po prawej stronie rysunku 3.22 przedstawiono zupełnie odmienny przypadek, gdzie pomiary szerokości geograficznej miejsca przeprowadzania badań skupią się mniej więcej w jednym punkcie, a wartość odstojąca jest wyraźnie poza zakresem wykresu pudełkowego. Analizując dodatkowo histogram tego samego parametru stwierdza się, że wyjątkowa wartość wystąpiła aż w ok. 10% wszystkich pomiarów. Jako że w wartościach długości geograficznej pojawiły się podobne odstępstwa i oba parametry opisują ten sam stan, to sprawdzono bezpośrednio wyznaczone „wyjątki” obu zbiorów informacji i wstępne wyniki przedstawiono na rysunku 3.23.

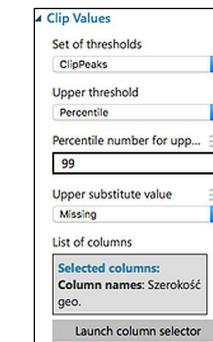


Rysunek 3.23. Wyznaczone miejsca przez moduł GPS  
 Źródło: opracowanie własne

Lewa część rysunku 3.23 przedstawia statystyki zarejestrowanych wartości szerokości geograficznej (u góry) i długości geograficznej (poniżej). Korzystając z aplikacji Google Maps sprawdzono wyznaczone lokalizacje wpisując odpowiednie wartości maksymalne i minimalne. Taka analiza pozwala stwierdzić, że 90% pomiarów była wykonywana na Jeziorze Uklejno (powiat Myśliborski) a 10% wartości odnosi się do Jeziora Jarosławki (powiat Goleniowski). Eksperyment ten został wykonany specjalnie, celem chcąc sprawdzenia możliwości wykrycia szumów wśród pomiarów.

Opisane powyżej przypadki obrazują ogólne problemy, jakie można napotkać w trakcie przeprowadzania badań z zakresu *data science* i jak ważnym jego aspektem jest analityczne podejście do każdego badanego punktu projektu. Funkcja automatycznie usuwająca odstające wartości danego parametru jest na początku, można by rzec, kusząca, gdyż może bez większego trudu usunąć wszystkie zbędne informacje. W praktyce jednak okazuje się, że czasami wielkości określane, jako „szумy pomiarowe”, są całkowicie prawidłowo zmierzone i logicznie pasują do innych informacji (przykład pomiarów temperatury wody). Innym razem takie dane skrywają dodatkowe wiedomości, co pozwala na odpowiednią interpretację i podjęcie decyzji przez projektanta modelu analitycznego (przykład położenia geograficznego).

Stwierdzono, że pozostawienie dwóch różnych miejsc badań spowodowałoby błędne wyznaczenie ostatecznego wyniku. Za pomocą funkcji *Clip Values* i *Clean Missing Data* usunięto linie, których pomiary dotyczyły Jeziora Jarosławki (10% badań). Wybór opcji funkcji *Clip Values* przedstawiono na rysunku 3.24.



Rysunek 3.24. Wybór opcji w funkcji Clip Values  
 Źródło: opracowanie własne

W funkcji *Clip Values* istnieje możliwość wybrania wartości odstających, które obserwowane są powyżej, czy też poniżej wąsów wykresu pułapkowego. Wybrano także próg odcięcia ustalając wartość procentową, która przekracza 99% wszystkich obserwowalnych wyników pomiarowych szerokości geograficznej. W przypadku wychwycenia outlierów zastąpiono je pustymi komórkami. Czyszczenie zbędnych linii odbywa się tak jak w poprzednim przypadku, czyli wykorzystując funkcję *Clean Missing Data*. Fragment finalnej wersji bazy danych przedstawiono na rysunku 3.25.

rows	columns					
504	8					
view as	Data/Czas	Szerokość geo.	Długość geo.	Temp. wody	Temp. powietrza	Wilgotność
	2018-08-16T07:02:42	52.984584	15.236624	23.62	22.8	68.1
	2018-08-16T07:07:28	52.984573	15.236633	23.68	22.8	68
	2018-08-16T07:13:04	52.984577	15.236633	23.68	22.6	68.4

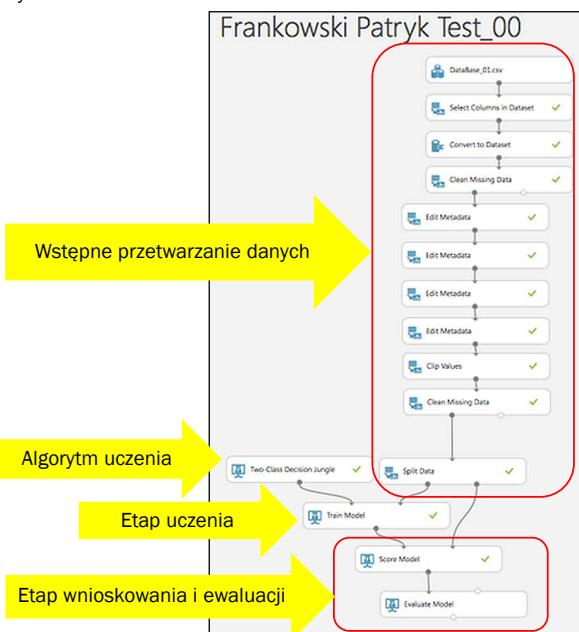
Rysunek 3.25. Finalna wersja bazy danych  
 Źródło: opracowanie własne

Jak przedstawia rysunek 3.25 liczba badań została zmniejszona o 51 pomiarów. To, że zostały usunięte żadane wartości zapewnia wynik odchylenia standardowego, który wynosi zaledwie 0,0004. Wraz z podobnymi rezultatami parametru długości geograficznej wywnioskowano, że baza danych jest skonstruowana tylko z wyników badań wykonanych jedynie na Jeziorze Uklejno.

Ostatnim krokiem podczas wstępnego przetwarzania danych jest podział informacji na zbiór uczących i testowych. Do tej operacji wykorzystano opcję *Split Data*, która oferuje m.in. możliwość wyboru współczynnika podziału oraz czy podział ma korzystać ze stratyfikacji.

### 3.4. Etap uczenia i ewaluacji modelu analitycznego

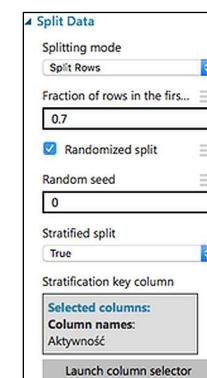
Projektowanie części uczenia i ewaluacji modelu analitycznego jest bardzo proste w porównaniu do etapu wstępnego przetwarzania danych na platformie Microsoft Azure Machine Learning Studio. Ogólny zarys podstawowej konstrukcji przedstawiono na rysunku 3.26.



Rysunek 3.26. Opis podstawowej konstrukcji modelu analitycznego w Microsoft Azure Machine Learning Studio  
 Źródło: opracowanie własne

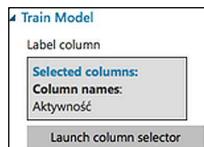
Celem przeprowadzanych eksperymentów jest wyznaczenie funkcji, która najlepiej przewiduje wartość aktywności bazując na innych parametrach. Postanowiono przeprowadzić badania skuteczności wybranych algorytmów uczenia maszynowego, które są dostępne w platformie Microsoft Azure Machine Learning Studio. W tym celu utworzono wiele modeli treningowych zmieniając jedynie algorytm uczenia.

Ze względu na dużą różnicę liczby pomiarów, które zostały wykonane z lub bez udziału użytkownika urządzenia rejestrującego, zdecydowano, że w każdym badanym eksperymencie w trakcie podziału danych na zbiór uczący i treningowy skorzysta się z opcji stratyfikacji. Powoduje to podział na podzbiory w taki sposób, aby zachować proporcję wartości „1” i „0” w parametrze opisującym aktywność. Bazując na obserwacjach innych modeli dostępnych w galerii Microsoft Azure Machine Learning Studio, stwierdzono, że zdecydowanie najczęściej korzysta się z proporcji podziału, gdzie 70% danych trafia do zbioru danych uczących, a pozostałe 30% tworzy zbiór danych treningowych. Opisane parametry opcji *Split Data* przedstawiono na rysunku 3.27.



Rysunek 3.27. Ustawione parametry opcji Split Data  
 Źródło: opracowanie własne

Blok Train Model nie wymaga dużych ingerencji projektanta modelu analitycznego. Do wejść podłączana jest funkcja wybranego algorytmu oraz baza danych. Jedyną opcją, jaką należy ustawić, jest wybór parametru na podstawie, którego trenowany będzie model, co przedstawiono na rysunku 3.28.



Rysunek 3.28. Ustawiony parametr opcji Train Model

Źródło: opracowanie własne

W celu odczytania ostatecznych wniosków skorzystano z dwóch opcji: *Score Model* oraz *Evaluate Model*. Obie funkcje nie wymagają ingerencji projektanta modelu w ustawianiu jakichkolwiek dodatkowych parametrów. Zadaniem opcji *Score Model* jest porównanie uzyskanego „wytrenowanego” modelu do wcześniej wydzielonego zbioru danych testowych. Zestawienie te polega na przeliczeniu danych ze zbioru testowego według wyznaczonej funkcji. Przykładowe wyniki przedstawiono na rysunku 3.29.

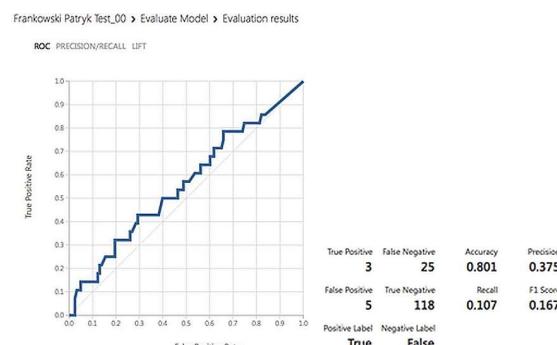
Aktywność	Scored Labels	Scored Probabilities
false	false	0.348639
true	false	0
false	false	0.3125
true	false	0.244792
true	true	0.51131

Rysunek 3.29. Fragment przykładowych wyników uzyskanych z opcji Score Model

Źródło: opracowanie własne

Jak przedstawiono na rysunku 3.29 funkcja *Score Model* powoduje dodanie kolejnych kolumn: *Scored Labels* i *Scored Probabilities*. W tej pierwszej wyznaczone są ostateczne odpowiedzi wytrenowanego modelu, które można porównać z wynikami z poprzedniej kolumny, a więc tymi uzyskanymi na drodze przeprowadzonych pomiarów. Kolumna *Scored Probabilities* wyznacza dokładną wartość wyliczoną na podstawie wytrenowanej funkcji, gdzie dodatkowo można stwierdzić, że punkt progowy w celu wyznaczenia jednej z dwóch odpowiedzi wynosi 0,5. Należy nadmienić, że dla każdej z rodzin algorytmów sztucznej inteligencji ta wizualizacja wygląda inaczej i może przedstawiać wartości innych parametrów.

*Evaluate Model* jest narzędziem, które wizualizuje uzyskane wnioski w bardzo czytelny sposób, wyliczając i przedstawiając m.in. błędy, czy też wykresy. Na rysunku 3.30 przedstawiono przykładowy wygląd wyniku opisywanej opcji.

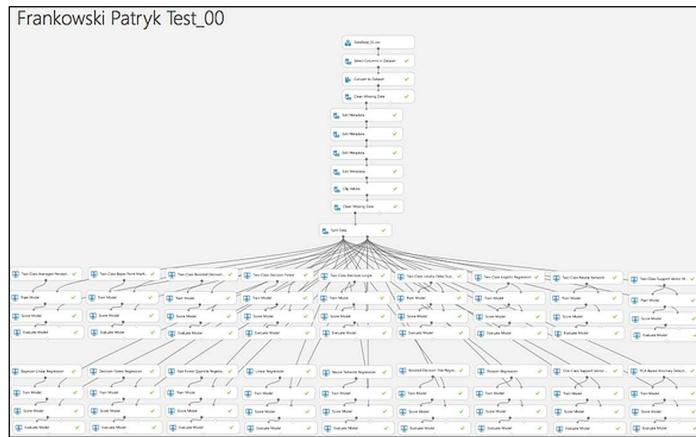


Rysunek 3.30. Fragment przykładowych wyników uzyskanych z opcji Evaluate Model

Źródło: opracowanie własne

Tak samo jak w przypadku omawiania funkcji *Score Model*, przykład wizualizacji opcji *Evaluate Model* na rysunku 3.30 różni się w zależności od wyboru rodziny algorytmów uczenia maszynowego.

Na podstawie stworzonej i opracowanej bazie danych, wykorzystując opisane powyżej opcje stworzono pierwszy model, którego zadaniem było przeanalizowanie wybranych dostępnych algorytmów platformy *Microsoft Azure Machine Learning Studio*. Zrezygnowano z przebadania algorytmów sztucznej inteligencji z rodziny klasyfikatorów wieloklasowych, ze względu na występowanie jedynie dwóch odpowiedzi („0” lub „1”) i podobieństwa podstaw matematycznych do tych z rodziny klasyfikatorów binarnych. Przeprowadzane badanie także nie mogło dotyczyć klasyfikacji rankingowej, więc zrezygnowano z funkcji *Ordinal Regression*. Wygląd tak zaprojektowanego modelu analitycznego przedstawiono na rysunku 3.31.



Rysunek 3.31. Projekt badanego modelu analitycznego

Źródło: opracowanie własne

Uzyskane wyniki z przeprowadzonego eksperymentu przedstawiono w kolejnym podrozdziale.

W następnej kolejności tworzone modele, wprowadzając nieznaczne modyfikacje do tego przedstawionego na rysunku 3.31 w celu przeprowadzenia bardziej szczegółowych badań. Wyniki z przeprowadzonych eksperymentów wraz z zestawieniem ich wartości opisano w kolejnym podrozdziale.

### 3.5. Zestawienie wyników i ich analiza

W pierwszej kolejności postanowiono wykonać jednakowe badania dla każdego z wybranych algorytmów dostępnych na platformie *Microsoft Azure Machine Learning*. W tym celu stworzono model przedstawiony na rysunku 3.29 i rozpoczęto działanie eksperymentu poprzez wcisnięcie przycisku *Run* znajdującego się w dolnej części ekranu projektowania. Większość z algorytmów posiada parametry, które można zmieniać. Postanowiono jednak w pierwszym badaniu pozostawić domyślne ustawienia, ponieważ zauważono, że algorytmy w takiej postaci są najczęściej używane w innych projektach tego typu. Wszystkie obliczenia zajęły około 30 minut, a uzyskane wyniki zapisano do jednej tabeli stworzonej w arkuszu kalkulacyjnym, którą przedstawiono poniżej.

**Tabela 3.1.** Wyniki przeprowadzonego eksperymentu działania wybranych algorytmów w ich podstawowych ustawieniach  
Źródło: opracowanie własne na podstawie [6]

Classification Algorithms							
Name of algorithm	Accuracy	Precision	TP (Poprawne True)	TN (Poprawne False)	FP	FN	Normalized
Two-Class Average Perceptron	0,815	1,000	0	123	0	28	NO
Two-Class Bayes Point Machine	0,815	1,000	0	123	0	28	NO
Two-Class Boosted Decision Tree	0,768	0,316	6	110	13	22	NO
Two-Class Decision Forest	0,808	0,444	4	118	5	24	NO
Two-Class Decision Jungle	0,801	0,375	3	118	5	25	NO
Two-Class Locally-Deep Support Vector Machine	0,821	0,667	2	122	1	26	NO
Two-Class Logistic Regression	0,815	1,000	0	123	0	28	NO
Two-Class Neural Network	0,815	1,000	0	123	0	28	NO
Two-Class Support Vector Machine	0,815	1,000	0	123	0	28	NO
Anomaly Detection Algorithms							
Name of algorithm	Accuracy	Precision	TP	TN	FP	FN	Normalized
One-Class Support Vector Machine	0,185	0,185	28	0	123	0	NO
PCA-Based Anomaly Detection	0,656	0,214	9	90	33	19	NO
Regression Algorithms							
Name of algorithm	Coefficient of Determination	Relative Squared Error	Mean Absolute Error	Root Mean Squared Error	Normalized		
Bayesian Linear Regression	0,02186	0,978814	0,297966	0,384507	NO		
Decision Forest Regression	-0,13562	1,13562	0,295875	0,414163	NO		
Boosted Decision Tree Regression	-0,370172	1,107728	0,334636	0,454928	NO		
Linear Regression	0,020279	0,979721	0,297858	0,384685	NO		
Neural Network Regression	-0,131635	1,131635	0,382261	0,413463	NO		
Poisson Regression	-0,007824	1,007824	0,299143	0,390164	NO		

W pierwszej kolejności zauważono, że w zależności od typu algorytmów uzyskuje się różne wyznaczniki określające jakość, które szczegółowo opisano w rozdziale 1. Kolejnym ważnym aspektem wnioskowania w oparciu o końcowe wyniki jest analizowanie wszystkich wartości jednocześnie, a nie każdej z osobna. Przykładowo, wyniki dokładności precyzyji algorytmu *Two-Class Average Perceptron* wskazywały na świetne działanie tej funkcji. Dokładniej analizując macierz błędów, widać, że tak wysoki wskaźnik *Accuracy* i *Precision* skutkuje dobrą predykcją wartości „0”. Ażkolwiek problem tego algorytmu tkwi w tym, że przypisał wszystkie dane, jako „0” i nie jest w sta-

nie przewidzieć sytuacji, w których zaistniało badane zdarzenie. W rodzinie klasyfikatorów binarnych zlokalizowano jedynie 4 algorytmy (zaznaczone na zielono), które są w stanie rozróżnić oba stany odpowiedzi.

Zaskakujące wyniki uzyskano z użyciem algorytmów służących do detekcji anomalii. W przypadku *One-Class Support Vector Machine* funkcja przypisała wszystkie wartości pomiarowe, jako te, które wpływają na zaistnienie badanego zdarzenia. *PCA-Based Anomaly Detection* jest w stanie rozróżnić stany na wyjściu uzyskując przy tym całkiem satysfakcyjną (porównując do pozostałych algorytmów) dokładność.

Najbardziej zaskakujące wyniki, w negatywnym tego słowa znaczeniu, uzyskano za pomocą algorytmów regresyjnych. Ujemne wartości współczynnika determinacji nie są spotykane często i świadczą o braku jakiegokolwiek zależności pomiędzy badanymi parametrami z bazy danych. Przyczyn takiego stanu może być wiele, m.in. za mała lub za duża liczba parametrów (kolumn), czy też niewłaściwa liczba przeprowadzonych badań (wierszy). Chęć znalezienia przyczyny takiego stanu nasunęła pomysł na przeprowadzenie kolejnego eksperymentu.

Teoretycznie, normalizację danych wykonuje się w przypadku wykorzystania w modelu algorytmów regresyjnych. Niemniej jednak postanowiono zarejestrować wyniki wszystkich badanych algorytmów z poprzedniego eksperymentu, które przedstawiono w tabeli 3.2.

**Tabela 3.2.** Wyniki przeprowadzonego eksperymentu badania wpływu normalizacji danych

Źródło: opracowanie własne na podstawie [6]

Classification Algorithms							
Name of algorithm	Accuracy	Precision	TP	TN	FP	FN	Normalized
Two-Class Average Perceptron	0,815	1,000	0	123	0	28	YES
Two-Class Bayes Point Machine	0,815	1,000	0	123	0	28	YES
Two-Class Boosted Decision Tree	0,722	0,320	8	106	17	20	YES
Two-Class Decision Forest	0,808	0,444	4	118	5	24	YES
Two-Class Decision Jungle	0,801	0,375	3	118	5	25	YES
Two-Class Locally_Deep Support Vector Machine	0,821	0,667	2	122	1	26	YES
Two-Class Logistic Regression	0,815	1,000	0	123	0	28	YES
Two-Class Neural Network	0,815	1,000	0	123	0	28	YES
Two-Class Support Vector Machine	0,815	1,000	0	123	0	28	YES
Anomaly Detection Algorithms							
Name of algorithm	Accuracy	Precision	TP	TN	FP	FN	Normalized
One-Class Support Vector Machine	0,185	0,185	28	0	123	0	YES
PCA-Based Anomaly Detection	0,656	0,214	9	90	33	19	YES
Regression Algorithms							

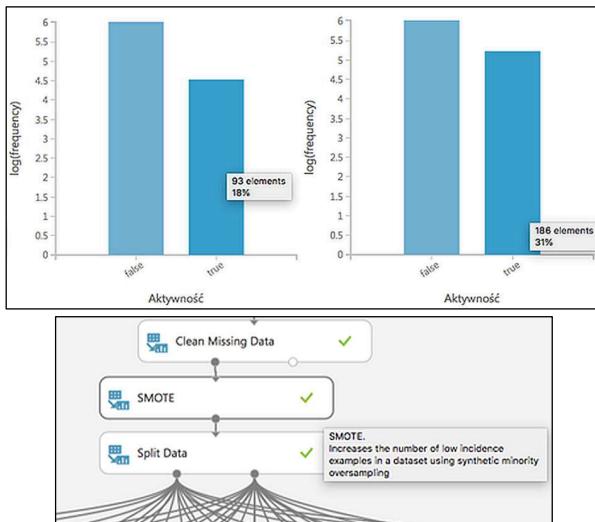
Name of algorithm	Coefficient of Determination	Relative Squared Error	Mean Absolute Error	Root Mean Squared Error	Normalized
Bayesian Linear Regression	-0,021898	1,021898	0,296887	0,392879	YES
Decision Forest Regression	-0,13562	1,13562	0,295875	0,414163	YES
Boosted Decision Tree Regression	-0,333048	1,333048	0,3368	0,448722	YES
Linear Regression	-0,023861	1,023861	0,29687	0,393256	YES
Neural Network Regression	-0,129861	1,129861	0,38167	0,41111	YES
Poisson Regression	-0,009544	1,00944	0,29987	0,390497	YES

W pierwszej kolejności, po porównaniu wyników z tabeli 3.2. i tabeli 3.1. stwierdzono, że funkcja normalizująca nie miała wpływu na algorytmy klasyfikujące i detekcji anomalii. Wyjątkiem jest *Two Class Boosted Decision Tree*, którego wyniki minimalnie się pogorszyły w drugim eksperymencie.

Badanie wpływu normalizacji danych było przeprowadzone przede wszystkim w celu wyjaśnienia bardzo złych wyników uzyskanych z wykorzystaniem algorytmów regresyjnych. Analizując uzyskane wyniki należy stwierdzić, że niestety dodatkowa funkcja, która teoretycznie powinna polepszyć rezultaty nie wpłynęła znacząco na poprawę jakości działania algorytmów regresyjnych. Zatem wszystko wskazuje na to, że badana baza danych zawiera za mało informacji, aby możliwym było znalezienie jakiegokolwiek korelacji pomiędzy parametrami a szukaną odpowiedzią.

Szukając odpowiedzi na uzyskanie bardzo złych wyników algorytmów regresyjnych, napotkano opinię w nieoficjalnych źródłach (na forach związanych ze statystyką), że algorytmy regresyjne w platformie *Microsoft Azure Machine Learning Studio* nie radzą sobie dobrze z danymi posiadającymi informacje o dacie i czasie. Wykonano, zatem badanie korzystając ponownie z funkcji stratyfikacji danych usuwając uprzednio kolumnę *Data/Czas* za pomocą wcześniej już wspomnianej opcji *Select Columns In Dataset*. Jako że wyniki końcowe nie zmieniły się w porównaniu do rezultatów poprzedniego eksperymentu oraz z powodu braku znalezienia jakichkolwiek informacji w oficjalnych źródłach mogących potwierdzić panującą opinię na forach zrezygnowano z bardziej szczegółowego przedstawienia końcowych efektów.

W trzecim eksperymencie postanowiono wypróbować funkcję *SMOTE*. Jest to opcja stworzona przez firmę *Microsoft*, która polega na zwiększeniu ilości danych. Nie polega jednak ona na prostym powielaniu informacji poprzez ich kopianie, a zawiera metody, których zadaniem jest powiększenie bazy danych o kolejne wartości w inteligentny sposób. *SMOTE* powoduje zminimalizowanie dysproporcji pomiędzy odpowiedziami wyjściowymi. W pierwotnej wersji bazy danych, 18% informacji dotyczyło pomiarów wykonanych w trakcie zarejestrowania danego zdarzenia, a 82% wartości odnosiło się do wartości „0” w parametrze *Aktywność*. Opisana proporcja danych przed i po użyciu funkcji *SMOTE* oraz jej umiejscowienie przedstawiono na rysunku 3.32.



Rysunek 3.32. Zestawienie proporcji odpowiedzi przed i po użyciu funkcji SMOTE  
Źródło: opracowanie własne

W tabeli 3.3 przedstawiono wyniki uzyskane z opisanego eksperymentu.

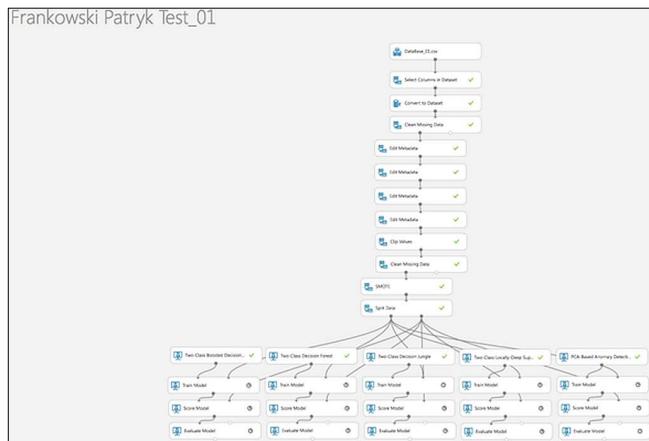
Tabela 3.3. Wyniki przeprowadzonego eksperymentu badania wpływu funkcji SMOTE  
Źródło: opracowanie własne na podstawie [6]

Classification Algorithms							
Name of algorithm	Accuracy	Precision	TP	TN	FP	FN	SMOTE
Two-Class Average Perceptron	0,687	1,000	0	123	0	56	YES
Two-Class Bayes Point Machine	0,687	1,000	0	123	0	56	YES
Two-Class Boosted Decision Tree	0,693	0,512	22	102	21	34	YES
Two-Class Decision Forest	0,737	0,629	22	110	13	34	YES
Two-Class Decision Jungle	0,682	0,486	18	104	19	38	YES
Two-Class Locally_Deep Support Vector Machine	0,743	0,778	14	119	4	42	YES
Two-Class Logistic Regression	0,687	1,000	0	123	0	56	YES
Two-Class Neural Network	0,687	1,000	0	123	0	56	YES
Two-Class Support Vector Machine	0,687	1,000	0	123	0	56	YES
Anomaly Detection Algorithms							
Name of algorithm	Accuracy	Precision	TP	TN	FP	FN	SMOTE

One-Class Support Vector Machine	0,313	0,313	56	0	123	0	YES
PCA-Based Anomaly Detection	0,587	0,304	14	91	32	42	YES
Regression Algorithms							
Name of algorithm	Coefficient of Determination	Relative Squared Error	Mean Absolute Error	Root Mean Squared Error	SMOTE		
Bayesian Linear Regression	0,019882	0,98018	0,422156	0,459021	YES		
Decision Forest Regression	-0,028741	1,028741	0,356413	0,470269	YES		
Boosted Decision Tree Regression	-0,370172	1,107728	0,334636	0,454928	YES		
Linear Regression	0,020499	0,979501	0,42202	0,458877	YES		
Neural Network Regression	-0,049093	1,049093	0,385592	0,474898	YES		
Poisson Regression	0,006893	0,993107	0,426164	0,462053	YES		

Funkcja SMOTE spowodowaławiększą „chęć” przypisywanie przez algorytmy wartości „1” dla parametru Aktywność. Tyczy się to zarówno prawidłowego przypisania tej odpowiedzi (True Positive), ale także błędnego odczytania zestawienia informacji (True Negative). Tym samym zmniejszylo to poprawne przewidywanie odpowiedzi „0” (False Positive) i zwiększenie predykcji błędnej dla tej samej wartości (False Negative). Porównując do poprzednich wyników, te uzyskane w tym eksperymencie Autor określa, jako najlepsze dla niektórych algorytmów klasyfikujących. Nadal dość zaskakującym jest fakt całkiem poprawnych wyników dla PCA-Based Anomaly Detection. Rezultaty uzyskane z wykorzystaniem algorytmów regresyjnych nadal są bardzo niesatysfakcjonujące. Jednakże wskazały one nieznaczna poprawę wyników w odniesieniu do poprzednich badań. Taka obserwacja potwierdza kilka możliwości określania głównego problemu wyznaczenia odpowiedniej funkcji predykcyjnej przez algorytmy regresyjne. Przyczyną tak złych wyników może być: za mała ilość danych lub parametrów, zbyt duża dysproporcja informacji uzyskanych z przyczyn aktywności użytkownika urządzenia („1”) bądź jej braku („0”), czy też brak możliwości wyznaczenia zależności pomiędzy zarejestrowanymi wartościami, a aktywnością badanego zdarzenia.

W dalszej kolejności postanowiono przeprowadzać eksperymenty na wybranych algorytmach z poprzednich badań, które wykazały najlepsze odpowiedzi (zaznaczone w tabeli 3.3 zielonym kolorem). Do badań wybrano: Two-Class Decision Tree, Two-Class Boosted Decision Forest, Two-Class Decision Jungle, Two-Class Locally-Deep Support Vector Machine oraz PCA-Based Anomaly Detection. Zmodyfikowano model, który przedstawiono na rysunku 3.33, tak aby przeprowadzać eksperymenty tylko na pięciu wybranych algorytmach zmniejszając przy tym czas obliczeń.



Rysunek 3.33. Projekt modelu analitycznego do przeprowadzania kolejnych badań  
 Źródło: opracowanie własne

Celem następnego badania było sprawdzenie wpływu zmiany proporcji podziału bazy danych na zbiór uczący i zbiór testowy. W każdej kolejnej próbie zmieniano ten współczynnik, a wyniki przedstawiono na poniższych rysunkach przedstawiających wartości poszczególnych miar jakości modeli i krzywą ROC. (od rysunku 3.34 do 3.38).

Podział	Krzywa ROC	Wyniki			
		True Positive	False Negative	Accuracy	Precision
0,1		67	100	0.615	0.385
		False Positive	True Negative	Recall	F1 Score
		107	263	0.401	0.393
0,2		67	82	0.676	0.479
		False Positive	True Negative	Recall	F1 Score
		73	256	0.450	0.464
0,3		51	79	0.660	0.447
		False Positive	True Negative	Recall	F1 Score
		63	225	0.392	0.418
0,4		47	65	0.671	0.470
		False Positive	True Negative	Recall	F1 Score
		53	194	0.420	0.443
0,5		38	55	0.681	0.487
		False Positive	True Negative	Recall	F1 Score
		40	165	0.409	0.444
0,6		38	36	0.723	0.559
		False Positive	True Negative	Recall	F1 Score
		30	134	0.514	0.535
0,7		22	34	0.693	0.512
		False Positive	True Negative	Recall	F1 Score
		21	102	0.393	0.444
0,8		12	25	0.681	0.480
		False Positive	True Negative	Recall	F1 Score
		13	69	0.324	0.387
0,9		6	13	0.667	0.462
		False Positive	True Negative	Recall	F1 Score
		7	34	0.316	0.375

Rysunek 3.34. Wyniki eksperymentu dla algorytmu Two-Class Boosted Decision Tree  
 Źródło: opracowanie własne

Podział	Krzywa ROC	Wyniki			
		Two- Class Decision Forest			
0,1		True Positive <b>61</b>	False Negative <b>106</b>	Accuracy <b>0.657</b>	Precision <b>0.439</b>
		False Positive <b>78</b>	True Negative <b>292</b>	Recall <b>0.365</b>	F1 Score <b>0.399</b>
0,2		True Positive <b>51</b>	False Negative <b>98</b>	Accuracy <b>0.661</b>	Precision <b>0.443</b>
		False Positive <b>64</b>	True Negative <b>265</b>	Recall <b>0.342</b>	F1 Score <b>0.386</b>
0,3		True Positive <b>36</b>	False Negative <b>94</b>	Accuracy <b>0.679</b>	Precision <b>0.474</b>
		False Positive <b>40</b>	True Negative <b>248</b>	Recall <b>0.277</b>	F1 Score <b>0.350</b>
0,4		True Positive <b>39</b>	False Negative <b>73</b>	Accuracy <b>0.688</b>	Precision <b>0.500</b>
		False Positive <b>39</b>	True Negative <b>208</b>	Recall <b>0.348</b>	F1 Score <b>0.411</b>
0,5		True Positive <b>35</b>	False Negative <b>58</b>	Accuracy <b>0.718</b>	Precision <b>0.574</b>
		False Positive <b>26</b>	True Negative <b>179</b>	Recall <b>0.376</b>	F1 Score <b>0.455</b>
0,6		True Positive <b>29</b>	False Negative <b>45</b>	Accuracy <b>0.727</b>	Precision <b>0.592</b>
		False Positive <b>20</b>	True Negative <b>144</b>	Recall <b>0.392</b>	F1 Score <b>0.472</b>
0,7		True Positive <b>22</b>	False Negative <b>34</b>	Accuracy <b>0.737</b>	Precision <b>0.629</b>
		False Positive <b>13</b>	True Negative <b>110</b>	Recall <b>0.393</b>	F1 Score <b>0.484</b>
0,8		True Positive <b>10</b>	False Negative <b>27</b>	Accuracy <b>0.689</b>	Precision <b>0.500</b>
		False Positive <b>10</b>	True Negative <b>72</b>	Recall <b>0.270</b>	F1 Score <b>0.351</b>
0,9		True Positive <b>6</b>	False Negative <b>13</b>	Accuracy <b>0.700</b>	Precision <b>0.545</b>
		False Positive <b>5</b>	True Negative <b>36</b>	Recall <b>0.316</b>	F1 Score <b>0.400</b>

Rysunek 3.35. Wyniki eksperymentu dla algorytmu Two-Class Decision Forest  
Źródło: opracowanie własne

Podział	Krzywa ROC	Wyniki			
		Two- Class Decision Jungle			
0,1		True Positive <b>50</b>	False Negative <b>117</b>	Accuracy <b>0.598</b>	Precision <b>0.336</b>
		False Positive <b>99</b>	True Negative <b>271</b>	Recall <b>0.299</b>	F1 Score <b>0.316</b>
0,2		True Positive <b>61</b>	False Negative <b>88</b>	Accuracy <b>0.651</b>	Precision <b>0.436</b>
		False Positive <b>79</b>	True Negative <b>250</b>	Recall <b>0.409</b>	F1 Score <b>0.422</b>
0,3		True Positive <b>49</b>	False Negative <b>81</b>	Accuracy <b>0.679</b>	Precision <b>0.480</b>
		False Positive <b>53</b>	True Negative <b>235</b>	Recall <b>0.377</b>	F1 Score <b>0.422</b>
0,4		True Positive <b>38</b>	False Negative <b>74</b>	Accuracy <b>0.674</b>	Precision <b>0.469</b>
		False Positive <b>43</b>	True Negative <b>204</b>	Recall <b>0.339</b>	F1 Score <b>0.394</b>
0,5		True Positive <b>32</b>	False Negative <b>61</b>	Accuracy <b>0.721</b>	Precision <b>0.593</b>
		False Positive <b>22</b>	True Negative <b>183</b>	Recall <b>0.344</b>	F1 Score <b>0.435</b>
0,6		True Positive <b>28</b>	False Negative <b>46</b>	Accuracy <b>0.744</b>	Precision <b>0.651</b>
		False Positive <b>15</b>	True Negative <b>149</b>	Recall <b>0.378</b>	F1 Score <b>0.479</b>
0,7		True Positive <b>18</b>	False Negative <b>38</b>	Accuracy <b>0.682</b>	Precision <b>0.486</b>
		False Positive <b>19</b>	True Negative <b>104</b>	Recall <b>0.321</b>	F1 Score <b>0.387</b>
0,8		True Positive <b>11</b>	False Negative <b>26</b>	Accuracy <b>0.664</b>	Precision <b>0.440</b>
		False Positive <b>14</b>	True Negative <b>68</b>	Recall <b>0.297</b>	F1 Score <b>0.355</b>
0,9		True Positive <b>10</b>	False Negative <b>9</b>	Accuracy <b>0.750</b>	Precision <b>0.625</b>
		False Positive <b>6</b>	True Negative <b>35</b>	Recall <b>0.526</b>	F1 Score <b>0.571</b>

Rysunek 3.36. Wyniki eksperymentu dla algorytmu Two-Class Decision Jungle  
Źródło: opracowanie własne

Podział	Krzywa ROC	Wyniki			
		Two- Class Locally- Deep Suport Vector Machine			
0,1		True Positive <b>56</b>	False Negative <b>111</b>	Accuracy <b>0.607</b>	Precision <b>0.359</b>
		False Positive <b>100</b>	True Negative <b>270</b>	Recall <b>0.335</b>	F1 Score <b>0.347</b>
0,2		True Positive <b>31</b>	False Negative <b>118</b>	Accuracy <b>0.669</b>	Precision <b>0.437</b>
		False Positive <b>40</b>	True Negative <b>289</b>	Recall <b>0.208</b>	F1 Score <b>0.282</b>
0,3		True Positive <b>32</b>	False Negative <b>98</b>	Accuracy <b>0.675</b>	Precision <b>0.457</b>
		False Positive <b>38</b>	True Negative <b>250</b>	Recall <b>0.246</b>	F1 Score <b>0.320</b>
0,4		True Positive <b>14</b>	False Negative <b>98</b>	Accuracy <b>0.721</b>	Precision <b>0.875</b>
		False Positive <b>2</b>	True Negative <b>245</b>	Recall <b>0.125</b>	F1 Score <b>0.219</b>
0,5		True Positive <b>17</b>	False Negative <b>76</b>	Accuracy <b>0.728</b>	Precision <b>0.773</b>
		False Positive <b>5</b>	True Negative <b>200</b>	Recall <b>0.183</b>	F1 Score <b>0.296</b>
0,6		True Positive <b>17</b>	False Negative <b>57</b>	Accuracy <b>0.748</b>	Precision <b>0.850</b>
		False Positive <b>3</b>	True Negative <b>161</b>	Recall <b>0.230</b>	F1 Score <b>0.362</b>
0,7		True Positive <b>14</b>	False Negative <b>42</b>	Accuracy <b>0.743</b>	Precision <b>0.778</b>
		False Positive <b>4</b>	True Negative <b>119</b>	Recall <b>0.250</b>	F1 Score <b>0.378</b>
0,8		True Positive <b>11</b>	False Negative <b>26</b>	Accuracy <b>0.739</b>	Precision <b>0.688</b>
		False Positive <b>5</b>	True Negative <b>77</b>	Recall <b>0.297</b>	F1 Score <b>0.415</b>
0,9		True Positive <b>6</b>	False Negative <b>13</b>	Accuracy <b>0.750</b>	Precision <b>0.750</b>
		False Positive <b>2</b>	True Negative <b>39</b>	Recall <b>0.316</b>	F1 Score <b>0.444</b>

Rysunek 3.37. Wyniki eksperymentu dla algorytmu Two-Class Locally-Deep Support Vector Machine

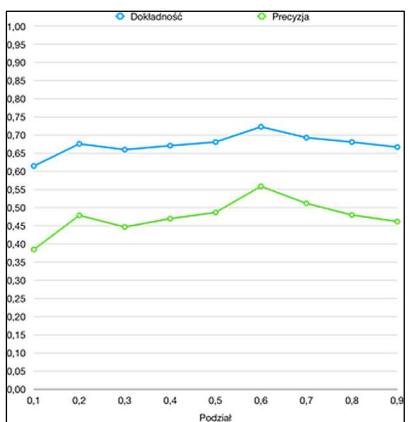
Źródło: opracowanie własne

Podział	Krzywa ROC	Wyniki			
		PCA- Based Anomaly Detection			
0,1		True Positive <b>27</b>	False Negative <b>140</b>	Accuracy <b>0.547</b>	Precision <b>0.208</b>
		False Positive <b>103</b>	True Negative <b>267</b>	Recall <b>0.162</b>	F1 Score <b>0.182</b>
0,2		True Positive <b>15</b>	False Negative <b>134</b>	Accuracy <b>0.621</b>	Precision <b>0.242</b>
		False Positive <b>47</b>	True Negative <b>282</b>	Recall <b>0.101</b>	F1 Score <b>0.142</b>
0,3		True Positive <b>13</b>	False Negative <b>117</b>	Accuracy <b>0.641</b>	Precision <b>0.283</b>
		False Positive <b>33</b>	True Negative <b>255</b>	Recall <b>0.100</b>	F1 Score <b>0.148</b>
0,4		True Positive <b>11</b>	False Negative <b>101</b>	Accuracy <b>0.646</b>	Precision <b>0.297</b>
		False Positive <b>26</b>	True Negative <b>221</b>	Recall <b>0.098</b>	F1 Score <b>0.148</b>
0,5		True Positive <b>11</b>	False Negative <b>82</b>	Accuracy <b>0.648</b>	Precision <b>0.324</b>
		False Positive <b>23</b>	True Negative <b>182</b>	Recall <b>0.118</b>	F1 Score <b>0.173</b>
0,6		True Positive <b>7</b>	False Negative <b>67</b>	Accuracy <b>0.643</b>	Precision <b>0.280</b>
		False Positive <b>18</b>	True Negative <b>146</b>	Recall <b>0.095</b>	F1 Score <b>0.141</b>
0,7		True Positive <b>4</b>	False Negative <b>52</b>	Accuracy <b>0.620</b>	Precision <b>0.200</b>
		False Positive <b>16</b>	True Negative <b>107</b>	Recall <b>0.071</b>	F1 Score <b>0.105</b>
0,8		True Positive <b>6</b>	False Negative <b>31</b>	Accuracy <b>0.647</b>	Precision <b>0.353</b>
		False Positive <b>11</b>	True Negative <b>71</b>	Recall <b>0.162</b>	F1 Score <b>0.222</b>
0,9		True Positive <b>5</b>	False Negative <b>14</b>	Accuracy <b>0.683</b>	Precision <b>0.500</b>
		False Positive <b>5</b>	True Negative <b>36</b>	Recall <b>0.263</b>	F1 Score <b>0.345</b>

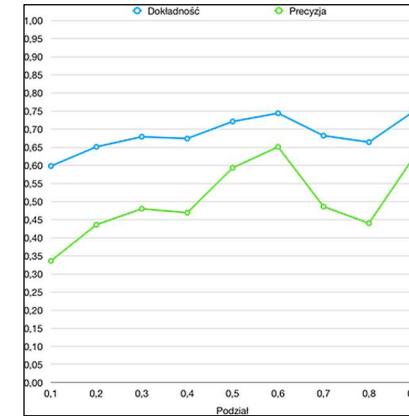
Rysunek 3.38. Wyniki eksperymentu dla algorytmu PCA-Based Anomaly Detection

Źródło: opracowanie własne

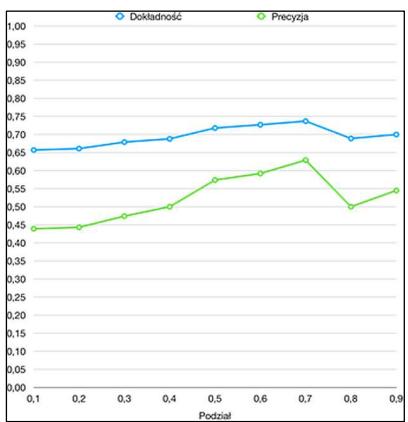
W celu czytelniejszego przedstawienia uzyskanych wyników, wyznaczono wykresy (rysunki od 3.39 do 3.43) zestawiające wartości dokładności i precyzji w zależności od stosunku podziału danych na zbiór uczący i zbiór testowy.



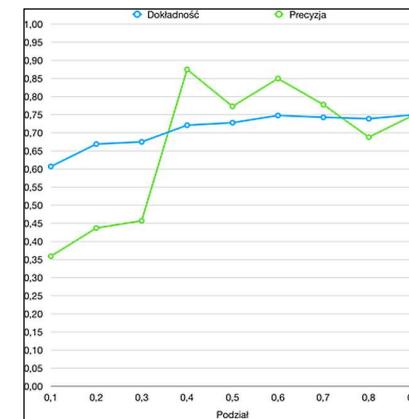
Rysunek 3.39. Wykres zestawiający wyniki dla algorytmu Two-Class Boosted Decision Tree  
 Źródło: opracowanie własne



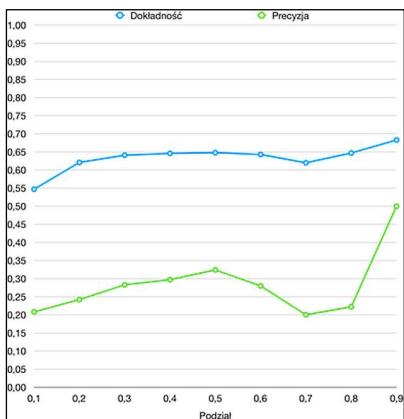
Rysunek 3.41. Wykres zestawiający wyniki dla algorytmu Two-Class Decision Jungle  
 Źródło: opracowanie własne



Rysunek 3.40. Wykres zestawiający wyniki dla algorytmu Two-Class Decision Forest  
 Źródło: opracowanie własne



Rysunek 3.42. Wykres zestawiający wyniki dla algorytmu Two-Class Locally-Deep Support Vector Machine  
 Źródło: opracowanie własne



Rysunek 3.43. Wykres zestawiający wyniki dla algorytmu PCA-Based Anomaly Detection  
Źródło: opracowanie własne

Algorytm *Two-Class Boosted Decision Tree* najlepsze wyniki osiąga w przypadku podziału danych w stosunku 60% na zbiór uczący i 40% zbiór testowy. Charakterystycznym dla tej funkcji jest fakt, że zmiany wartości dokładności i precyzji są niemalże jednakowe.

Dla algorytmu *Two-Class Decision Forest* najlepsze predykcje wartości uzyskuje się w przypadku, gdy 70% wszystkich informacji tworzy zbiór danych uczących. Funkcja ta wskazuje także w miarę przybliżone zmiany wartości współczynnika dokładności i precyzji.

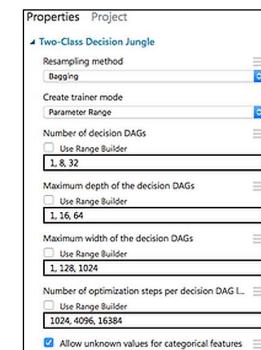
Algorytm *Two-Class Decision Jungle* podobnie jak w przypadku pierwszej z bahanich funkcji najskuteczniej wnioskuje w przypadku podziału w stosunku 0,6. Zaskakującym może być fakt bardzo dobrego wyniku także dla próby, w której 90% danych tworzyło zbiór uczący. Przy tak wysokich proporcjach, należy być jednak ostrożnym w wyciąganiu pozytywnych wniosków. Należy mieć na uwadze, że badany zbiór danych nie posiada ogromnej ilości informacji, a przebadanie stworzonej przez algorytm funkcji odbyło się tylko na ok. 50 przypadkach. Stwierdzenie to można potwierdzić obserwując trend wyznaczonego wykresu i zaskakujący nagły wzrost przy ostatniej próbie. Dlatego też postanowiono pominąć opisywany przykład w trakcie analizy wyników wszystkich badań dla tego algorytmu.

Funkcja *Two-Class Locally-Deep Support Vector Machine* charakteryzuje się małymi zmianami współczynnika dokładności, w odróżnieniu od wartości precyzji. Uzyskane wyniki utrudniają określenie najkorzystniejszego podziału danych dla tej funkcji. Można jedynie stwierdzić, że najlepsze wyniki uzyskiwane są dla przypadku, gdy zbiór uczący stanowi ilość informacji z przedziału 40% + 60% wszystkich danych.

Uzyskane wyniki algorytmu *PCA-Based Anomaly Detection* wskazują, że zmiana współczynnika podziału danych nie wpływa satysfakcyjnie lepiej na końcowy efekt. Wartości współczynnika precyzji są najniższe w porównaniu do innych algorytmów analizowanych w tym eksperymencie. Dla żadnego przypadku, ilość poprawnie przypisanych odpowiedzi „1” nie przewyższa liczby tych źle przewidywanych. O tym, że opisywany algorytm wypadł najsłabiej w eksperymencie świadczą także wizualizacje krzywej ROC.

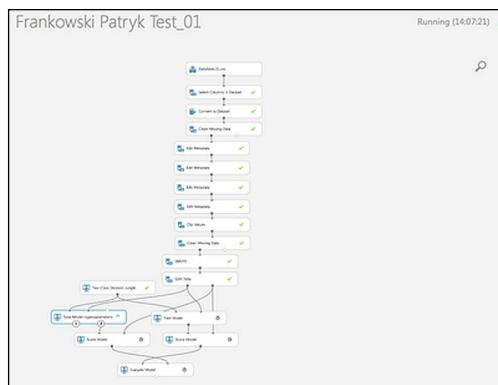
Najlepsze wyniki funkcji predykcyjnej uzyskano z wykorzystaniem algorytmów klasyfikujących bazujących na logice działania drzewa decyzyjnego. *Two-Class Boosted Decision Tree*, *Two-Class Decision Forest* i *Two-Class Decision Jungle* wykazały satysfakcyjne i zbliżone do siebie wyniki. Ze względu na bardzo zbliżoną konstrukcję tych algorytmów, często w przypadku poprawnego działania jednego z nich, inne z tej samej rodziny także wskazują pozytywne predykcje. Wybór pomiędzy nimi zależy od czasu trwania symulacji modelu analitycznego. Kolejno: drzewo decyzyjne, las decyzyjny i dżungla decyzyjna charakteryzują się odpowiednio od najkrótszego do najdłuższego czasu trwania symulacji kosztem jakości modelu predykcyjnego. W badanym przykładzie baza danych nie jest bardzo rozbudowana i proces symulacji dla pojedynczego algorytmu przebiega stosunkowo szybko. Dlatego też zdecydowano się przeprowadzić ostatni eksperyment korzystając w nim z algorytmu *Two-Class Decision Jungle*.

Każdy algorytm posiada własne parametry, które w programie *Microsoft Azure Machine Learning Studio* można ustawić. Bardzo często, są to opcje, które ciężko zrozumieć i poprawnie ustawić bez posiadania specjalistycznej wiedzy. Dla algorytmu *Two-Class Decision Jungle*, fragment możliwych do modyfikacji parametrów przedstawiono na rysunku 3.44.



Rysunek 3.44. Fragment opcji ustawień dla algorytmu *Two-Class Decision Jungle*  
Źródło: opracowanie własne

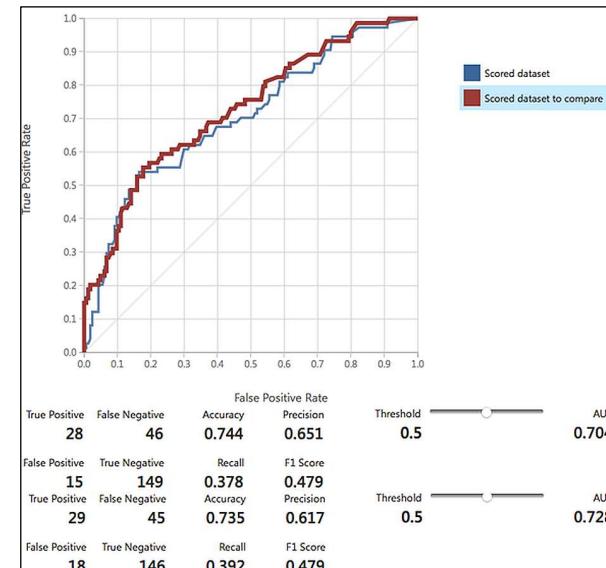
Firma Microsoft stworzyła funkcję, która przeprowadza eksperyment dla wybranego zakresu możliwych ustawień poszczególnych parametrów algorytmu, finalnie wybierając optymalne wartości opcji. *Tune Model Hyperparameters* jest bardzo wygodnym narzędziem w uczeniu maszynowym, aczkolwiek jego użycie znacznie przedłuża czas trwania symulacji całego modelu predykcyjnego. Postanowiono w ramach kolejnego eksperymentu skorzystać z dostępnej opcji, optymalizując funkcję predykcyjną dla badanej bazy danych. Skonstruowany model analityczny przedstawiono na rysunku 3.45.



**Rysunek 3.45.** Projekt modelu analitycznego do optymalizacji funkcji predykcyjnej  
 Źródło: opracowanie własne

W pierwszej kolejności przeprowadzono próbę optymalizacyjną dla całości zakresu wartości wszystkich możliwych parametrów ustawień (maksymalne ustawienia optymalizacyjne). Po nieco ponad 14 godzinach przeprowadzania symulacji (na rysunku 3.45 prawy górny róg) postanowiono zaprzestać dalszego poszukiwania optymalnej funkcji z maksymalnymi ustawieniami opcji *Tune Model Hyperparameters*.

Do kolejnego badania ustawiiono dla algorytmu *Two-Class Decision Jungle* możliwość wyboru optymalnej funkcji dla parametrów: ilość drzew decyzyjnych i głębokość decyzyjna z zakresu 1–100. Rysunek 3.46 przedstawia uzyskany wynik wraz z porównaniem do modelu niewykorzystującego opcji *Tune Model Hyperparameters*.



**Rysunek 3.46.** Zestawienie wyników po użyciu funkcji optymalizującej parametry algorytmu  
 Źródło: opracowanie własne

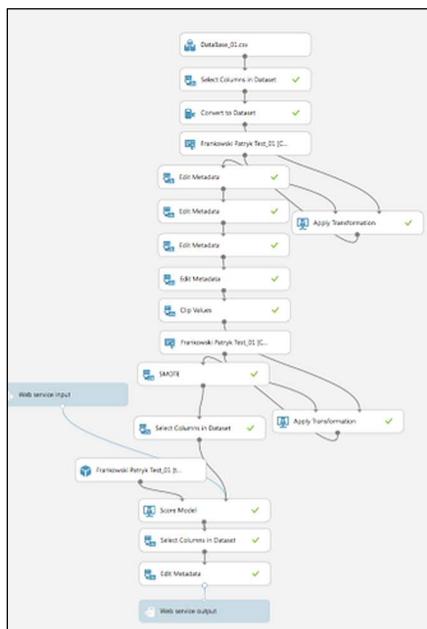
Na rysunku 3.46 przedstawiono wykres, gdzie czerwonym kolorem oznaczono krzywą ROC dla modelu wykorzystującego funkcję optymalizującą wartości parametrów algorytmu, a niebieskim wyniki modelu analitycznego bez tej funkcji. Poniższe wyniki współczynnika dokładności i precyji sugerują, że nieznacznie lepszą funkcję predykcyjną uzyskano w przypadku symulacji modelu niewykorzystującego opcji *Tune Model Hyperparameters*. Dokładniej analizując uzyskane wartości można jednak stwierdzić, że dodatkowa funkcja w modelu powoduje dążenie do zwiększenia możliwych odpowiedzi „1”. Poświadczają to także wyższy współczynnik AUC określający wielkość pola pod krzywą ROC. Wnioski z przeprowadzonego badania sugerują, że funkcja *Tune Model Hyperparameters* jest narzędziem bardzo przydatnym i ułatwiającym pracę z algorytmami uczenia maszynowego. Negatywnym aspektem tego jest bardzo duże zwiększenie potrzebnego czasu symulacji.

Finalnie na podstawie wykonanych badań można stwierdzić, że uzyskano funkcję predyktacyjną charakteryzującą się zadowalającymi wynikami. Prawdopodobnie to zbyt uboga w informacje baza danych uniemożliwiła poprawne wnioskowanie z wykorzystaniem algorytmów regresyjnych.

### 3.6. Operacyjonalizacja modelu predykcyjnego

Operacyjonalizacja jest dodatkowym etapem, który polega na wdrożeniu skonstruowanego modelu. Często zawierają się w tym kroku takie zadania jak: przepisanie modelu na żądaną platformę, walidacja poprawności, czy też wdrożenie ogólnodostępnej aplikacji. *Microsoft Azure Machine Learning* w bezpłatnej wersji umożliwia stworzenie podstawowej aplikacji umożliwiającej przeprowadzenie testu, co opisano w niniejszym podrozdziale.

W pierwszej kolejności, po skonstruowaniu żądanego modelu analitycznego, zaznaczono funkcję *Set Up Web Service* z opcją stworzenia *Predictive Web Service*. Spowodowało to automatyczną kopię modelu z wprowadzonymi niezbędnymi modyfikacjami tj. skompresowanie niektórych części modelu oraz dodanie funkcji umożliwiających wprowadzanie danych wejściowych i wyświetlenie danych wyjściowych. Tak skonstruowany we wstępny etapie operacyjonalizacji model przedstawiono na rysunku 3.47.



Rysunek 3.47. Wygląd modelu po pierwszym etapie operacyjonalizacji  
Źródło: opracowanie własne

Na rysunku 3.48 przedstawiono fragment wyników, jakie uzyskuje się poprzez skorzystanie ze skonstruowanego modelu.

Wilgotność	Ciśnienie	Scored Labels	Scored Probabilities
68.1	1024.1	false	0.034144
68	1024.1	false	0.017857
68.4	1024	false	0.181878
68.1	1024.1	false	0.069858
68	1024.1	false	0.021164
66.9	1024	true	0.624504

Rysunek 3.48. Wyniki uzyskiwane za pomocą skonstruowanego modelu  
Źródło: opracowanie własne

Po wyborze opcji *Deploy Web Service* ukazuje się okno, które przedstawia szeroką możliwości skonstruowania modelu w zależności od potrzeb. Można m.in. wygenerować model analityczny w kodzie Python lub C#, czy też skompilować go, aby móc korzystać z poziomu programu Microsoft Excel. Istnieje także opcja wyświetlenia wersji testowej takiego programu w przeglądarce internetowej, z której skorzystano w przypadku zaprojektowanego modelu predykcyjnego i wizualizację dwóch różnych wyników przedstawiono na rysunku 3.49.

input1		output1	
Data/Czas	18-08-2018T13:00	Przewidywanie (skala 0 - 1)	0.244331065759637
Szerokość geo.	52.985		
Długość geo.	15.236		
Temp. wody	20		
Temp. powietrza	25		
Wilgotność	40		
Ciąnienie	1012.2		
<b>Test Request-Response</b>			
Data/Czas	18-08-2018T13:00	Przewidywanie (skala 0 - 1)	0.672902494331066
Szerokość geo.	52.985		
Długość geo.	15.236		
Temp. wody	24		
Temp. powietrza	25		
Wilgotność	45		
Ciąnienie	1002		
<b>Test Request-Response</b>			

Rysunek 3.49. Wizualizacja wygenerowanej aplikacji w wersji testowej  
 Źródło: opracowanie własne

W lewej części ekranu wpisuje się dane wejściowe do modelu analitycznego. Po wciśnięciu przycisku *Test Request-Response* po prawej stronie wyświetla się wynik predykcji w skali od „0” do „1”. Z punktu środowiska, w którym badano zdarzenia, rezultat ten można odczytać, jako współczynnik wskazujący prawdopodobieństwo żeru ryb dla danych parametrów.

## Wnioski

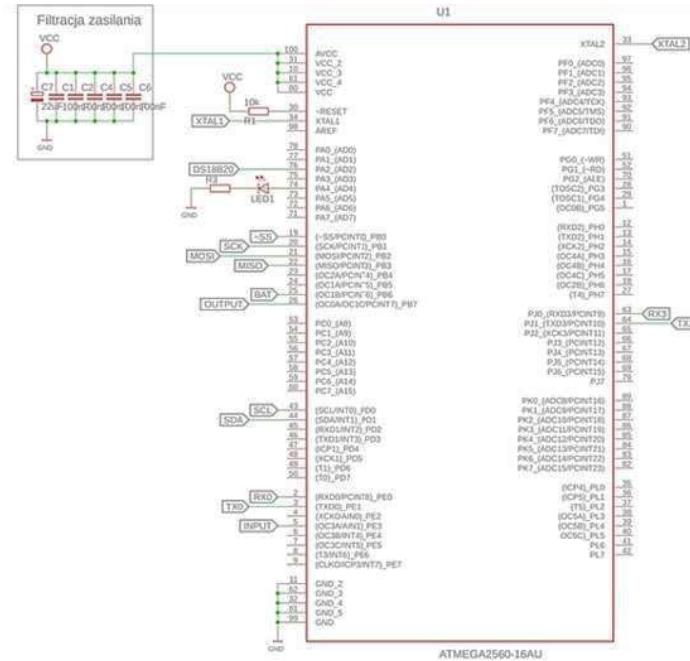
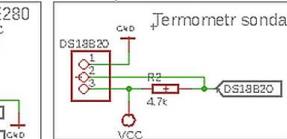
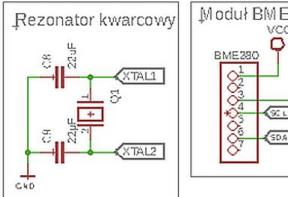
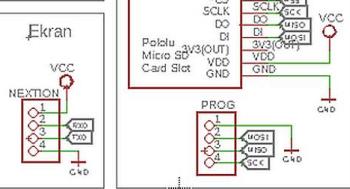
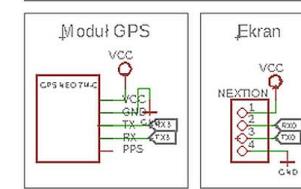
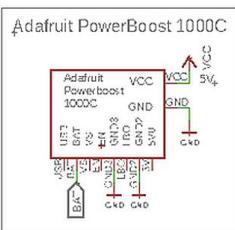
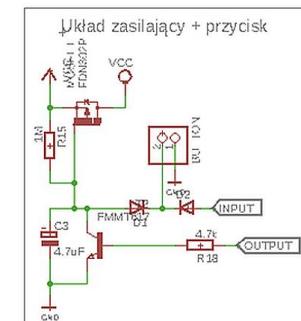
Celem badania pierwotnie było wyznaczenie najlepszego algorytmu, który przewidawałby aktywność żeru ryb na podstawie wpisanych prognoz pogody na kolejne dni. Wszystkie przeprowadzone eksperymenty zmierzały ku temu, aczkolwiek Autor z góry założył, że z dużym prawdopodobieństwem, wywnioskowana funkcja nie będzie „najlepszą z możliwych”. Już na poziomie zapoznawania się z teorią uczenia maszynowego, można stwierdzić, że żeby osiągnąć wysoki poziom w tej dziedzinie niezbędnym jest posiadanie nie tylko ogromnej wiedzy, ale także doświadczenia w projektowaniu modeli analitycznych.

Pierwsze wnioski sugerowały już, że wybrana tematyka przeprowadzanych pomiarów i część zbadania aktywności żeru ryb jest bardzo „niestabilnym gruntem” do wykonywania tego typu eksperymentów. Mimo dużego doświadczenia Autora w uprawianym hobby, istnieje wiele czynników (m.in. kierunek wiatru i opady deszczu), które nie zostały zmierzone i sprawdzone z punktu niniejszej pracy, a które mogą mieć znaczący wpływ na badane zdarzenia. W tym celu wymagana jest poszerzona wiedza z dziedziny ichtiologii. Należy wspomnieć także o fakcie, że żeby wnioski były optymalne potrzebny jest ogrom danych przeprowadzonych w skali, co najmniej całego roku, a wręcz gromadzonych latami, ponieważ każdy parametr może wpływać inaczej w danej porze roku. Można się spodziewać, że różnice także by się pojawiały w zależności od badanego akwenu. Reasumując, w badanym obszarze istnieje ogrom czynników, które mają znaczący wpływ na końcowe rezultaty.

Uzyskane ostateczne wyniki badań algorytmów bazujących na logice działania drzewa decyzyjnego okazały się w pełni zadowalające. Umożliwiają one z satysfakcjonującą dokładnością przewidywanie występowania danego zdarzenia. Rezultaty pozostałycych badanych algorytmów wskazują na zbyt małą ilość informacji zawartych w bazie danych, aby była możliwość wyznaczenia zależności między nimi, a szukaną odpowiedzią.

Całość przeprowadzonego eksperymentu tworzy bazę do dalszych działań w celu zoptymalizowania i usprawnienia modelu predykcyjnego oraz ostatecznej odpowiedzi na pytania postawione na początku pracy. Jeżeli badania miałyby przebiegać w tym samym środowisku, w pierwszej kolejności należałoby dodać kolejne moduły pomiarowe do skonstruowanego urządzenia. Najważniejszym aspektem byłoby znaczne zwiększenie obszaru i ilości zarejestrowanych zdarzeń wraz z parametrami mu towarzyszącymi w danej chwili. Finalnie należałoby stworzyć rozbudowaną aplikację, która posiada opcje przeglądania historii pomiarów i zestawienie ich z wynikami uzyskanymi dzięki skonstruowanemu modelowi predykcyjnemu.

## Załącznik A. Schemat elektryczny urządzenia



## Załącznik B. Listing programu

```
#include <Wire.h> // Libraries for BME280
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <OneWire.h> // Libraries for DS18B20 probe
#include <DallasTemperature.h>
#include <TinyGPS++.h> // Library for GPS NEO-7M
#include <SPI.h> // Libraries for SD card
#include <SD.h>
#include <Nextion.h>

#define On 13 // Power On/Off
#define Off 5
#define Sonda 24 // Probe DS18B20
#define CS 53 // CS for SD card

static const uint32_t GPSBaud = 9600;
static const uint32_t DisplayBaud = 115200;
unsigned long timer = 5000;
unsigned long timer2 = 300000;
int powerOn = 0;
float temperatura = 0;
float cisnienie = 0;
float wilgotnosc = 0;
float sonda = 0;
float temperaturaF = 0;
float cisnieniemmHg = 0;
float sondaf = 0;
String godzina;
String minuta;
String sekunda;
String rok;
String miesiac;
String dzien;
String szerokoscgeo;
String dlugoscgeo;
String predkosc;
String wysokosc;
String sateleta;
int temporaryHour = 0;
int temporaryMinute = 0;
int temporarySecond = 0;
int temporaryMonth = 0;
int temporaryDay = 0;
float temporaryHdop = 0;
unsigned long temporaryLng = 0;
unsigned long temporaryLat = 0;
int bateria = 0;
```

```
int counterButton = 0;
bool pushButton = 0;

NexHotspot m1 = NexHotspot(3, 23, "m1");
NexHotspot m0 = NexHotspot(7, 2, "m0");
NexTouch *nex_listen_list[] =
{
    &m1,
    &m0,
    NULL
};

File file;
OneWire oneWire(Sonda);
DallasTemperature sensors(&oneWire);
Adafruit_BME280 bme;
TinyGPSPlus gps;

void setup()
{
    deviceOn();
    Serial.begin(DisplayBaud);
    Serial3.begin(GPSBaud);
    m1.attachPush(m1PushCallback);
    m0.attachPush(m0PushCallback);
    bme.begin();
    sensors.begin();
    SD.begin(CS);
    file = SD.open("TEST.csv", FILE_WRITE);
    file.print("Date,Time,Latitude,Longitude,Water,Air,Humidity,Pressure,Activity");
    file.close();
}

void loop()
{
    checkPowerOff();
    nexLoop(nex_listen_list);
    if (Serial3.available() > 0)
        if (timer < millis() && gps.encode(Serial3.read())) // Updating and sending data
            displayParam();
            displayDateTime();
    }
    if (timer2 < millis()) // Saving data on SD card & Check Battery
        // displayBattery();
        saveParam();
        timer2 = millis() + 300000;
}
```

```

/* Power On/Off the device */
void deviceOn() {
pinMode(On, OUTPUT);
digitalWrite(On, HIGH);
pinMode(Off, INPUT);
pinMode(Off, INPUT_PULLUP);
//delay(3000);
}

void checkPowerOff() {
if(digitalRead(Off) == LOW) {
Serial.print("sleep=0");
endCommand();
delay(3000);
powerOn = 1;
if(digitalRead(Off) == LOW && powerOn == 1) {
Serial.print("page 7");
endCommand();
}
}
else {
powerOn = 0;
}
}

/* Actions for buttons */
void m1PushCallback(void *ptr)
{
pushButton = 1;
counterButton = counterButton + 1;
saveParam();
Serial.print("licznik.val=");
Serial.print(counterButton);
Serial.write(0xff);
Serial.write(0xff);
Serial.write(0xff);
pushButton = 0;
}

void m0PushCallback(void *ptr)
{
delay(1000);
digitalWrite(On, LOW);
}

/* Update & Send Parameters*/
void displayParam(){
updateParam();
sendParam();
timer = millis() + 5000;
}

void displayDateTime(){
updateTime();
sendTime();
updateDate();
sendDate();
updateLocation();
sendLocation();
}

void updateParam(){
temperatura = bme.readTemperature();
temperaturaF = temperatura * 1.8 + 32;
cisenenie = bme.readPressure();
cisenenie = cisenenie/100;
ciseniemmHg = cisenenie * 0.75;
wilgotnosc = bme.readHumidity();
sensors.requestTemperatures();
sonda = sensors.getTempCByIndex(0);
sondaF = sonda * 1.8 + 32;
}

void updateTime(){
temporaryHour = gps.time.hour() + 2;
if (temporaryHour < 10){
godzina = "0" + String(temporaryHour);
}else if (temporaryHour == 24){
godzina = "00";
}else if (temporaryHour == 25){
godzina = "01";
}else{
godzina = String(temporaryHour);}

temporaryMinute = gps.time.minute();
if (temporaryMinute < 10){
minuta = "0" + String(temporaryMinute);
}else{
minuta = String(temporaryMinute);}

temporarySecond = gps.time.second();
if (temporarySecond < 10){
sekunda = "0" + String(temporarySecond);
}else{
sekunda = String(temporarySecond);}
}

void updateDate(){
temporaryDay = gps.date.day();
if (temporaryDay < 10){
dzien = "0" + String(temporaryDay);
}else{
dzien = String(temporaryDay);}

temporaryMonth = gps.date.month();
}

```

```

if (temporaryMonth < 10){
miesiac = "0" + String(temporaryMonth);
}else{
miesiac = String(temporaryMonth);}

rok = gps.date.year();
}

void updateLocation(){
temporaryLng = gps.location.lng();
temporaryLat = gps.location.lat();
szerokoscgeo = String(gps.location.lat(), 6);
dlugoscgeo = String(gps.location.lng(), 6);
predkosc = String(gps.speed.kmph());
wysokosc = String(gps.altitude.meters());
satelita = String(gps.satellites.value());
temporaryHdop = gps.hdop.hdop();
/*Serial.println(temporaryLng);
Serial.println(temporaryLat);
Serial.println(szerokoscgeo);
Serial.println(dlugoscgeo);
Serial.println(predkosc);
Serial.println(gps.speed.kmph());
Serial.println(wysokosc);
Serial.println(gps.altitude.meters());
Serial.println(satelita);
Serial.println(gps.satellites.value());
Serial.println(temporaryHdop);
Serial.println(gps.hdop.hdop());*/
}

void endCommand()
{
Serial.write(0xff);
Serial.write(0xff);
Serial.write(0xff);
}
void sendParam(){
String command = "temperatura.txt=\\""+String(temperatura,1)+"\\";
Serial.print(command);
endCommand();
String command1 = "cinsnienie.txt=\\""+String(cinsnienie,1)+"\\";
Serial.print(command1);
endCommand();
String command2 = "wilgotnosc.txt=\\""+String(wilgotnosc,1)+"\\";
Serial.print(command2);
endCommand();
String command3 = "sonda.txt=\\""+String(sonda,1)+"\\";
Serial.print(command3);
endCommand();
String command4 = "temperaturaF.txt=\\""+String(temperaturaF,1)+"\\";
Serial.print(command4);
}

endCommand();
String command5 = "sondaF.txt=\\""+String(sondaF,1)+"\\";
Serial.print(command5);
endCommand();
String command6 = "cinsnieniemmHg.txt=\\""+String(cinsnieniemmHg,1)+"\\";
Serial.print(command6);
endCommand();
}

void sendTime(){
String command = "godzina.txt=\\""+ godzina +"\\";
Serial.print(command);
endCommand();
String command1 = "minuta.txt=\\""+ minuta +"\\";
Serial.print(command1);
endCommand();
}

void sendDate(){
String command = "rok.txt=\\""+ rok +"\\";
Serial.print(command);
endCommand();
String command1 = "miesiac.txt=\\""+ miesiac +"\\";
Serial.print(command1);
endCommand();
String command2 = "dzien.txt=\\""+ dzien +"\\";
Serial.print(command2);
endCommand();
}

void sendLocation(){
String command = "szerokoscgeo.txt=\\""+ szerokoscgeo +"\\";
Serial.print(command);
endCommand();
String command1 = "dlugoscgeo.txt=\\""+ dlugoscgeo +"\\";
Serial.print(command1);
endCommand();
String command2 = "predkosc.txt=\\""+ predkosc +"\\";
Serial.print(command2);
endCommand();
String command3 = "wysokosc.txt=\\""+ wysokosc +"\\";
Serial.print(command3);
endCommand();
String command4 = "satelity.txt=\\""+ satelita +"\\";
Serial.print(command4);
endCommand();
if (temporaryHdop > 20.0 || temporaryHdop==0.0){
Serial.print("hdop.txt=\\"v.bad\\"");
endCommand();
if (temporaryHdop >= 9.0 && temporaryHdop <= 20.0){
Serial.print("hdop.txt=\\"bad\\"");
endCommand();
if (temporaryHdop >= 7.0 && temporaryHdop <= 8.0){
Serial.print("hdop.txt=\\"medium\\"");
}
}
}

```

```

endCommand();
if (temporaryHdop >= 4.0 && temporaryHdop <= 6.0){
Serial.print("hdop.txt=\\"good\\\"");
endCommand();
if (temporaryHdop >= 2.0 && temporaryHdop <= 3.0){
Serial.print("hdop.txt=\\"v.good\\\"");
endCommand();
if (temporaryHdop > 0.0 && temporaryHdop <= 1.0){
Serial.print("hdop.txt=\\"great\\\"");
endCommand();
if (temporaryLat < 0.00){
Serial.print("ns.txt=\\"S\\\"");
endCommand();
}else{
Serial.print("ns.txt=\\"N\\\"");
endCommand();
}
if (temporaryLng < 0.00){
Serial.print("ew.txt=\\"W\\\"");
endCommand();
}else{
Serial.print("ew.txt=\\"E\\\"");
endCommand();
}
}
/* Check battery voltage */
/* void displayBattery(){
int batValue = analogRead(VoltageTest);
float voltage = batValue * (4.2 / 859.32);
voltage = voltage - 3;
bateria = (voltage * 100)/1.2;
Serial.println(batValue);
Serial.println(voltage);
Serial.println(bateria);
Serial.print("bateria.val=");
Serial.print(bateria);
endCommand();
Serial.print("procent.val=");
Serial.print(bateria);
endCommand(); } */

/* Saving data on SD card */
void saveParam(){
file = SD.open("TEST.csv", FILE_WRITE);
file.println();
file.print(dzien);
file.print("-");
file.print(miesiac);
file.print("-");
file.print(rok);
file.print(",");
file.print(godzina);
file.print(":");
file.print(minuta);
file.print(":");
file.print(sekunda);
file.print(",");
file.print(gps.location.lat(), 6);
file.print(",");
file.print(gps.location.lng(), 6);
file.print(",");
file.print(sonda);
file.print(",");
file.print(temperatura ,1);
file.print(",");
file.print(wilgotnosc, 1);
file.print(",");
file.print(cisnienie, 1);
file.print(",");
file.print(pushButton);
file.close();
}

```

## Bibliografia

- [1] Wawryński P.: *Podstawy sztucznej inteligencji*, wydanie I, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, 2014.
- [2] Flasiński M.: *Wstęp do sztucznej inteligencji*, Wydawnictwo Naukowe PWN SA, Warszawa, 2011.
- [3] Walsh T.: *To żyje! Sztuczna inteligencja od logicznego fortepianu po zabójcze roboty*, Wydanie I, Wydawnictwo Naukowe PWN SA, Warszawa, 2018.
- [4] Krawiec K., Stefanowski J.: *Uczenie maszynowe i sieci neuronowe*, Wydanie I, Wydawnictwo Politechniki Poznańskiej, Poznań, 2003.
- [5] Szeliga M., *Data Science i Uczenie Maszynowe*, Wydanie I, Wydawnictwo Naukowe PWN SA, Warszawa, 2017.
- [6] Raschka S., *Python Uczenie Maszynowe*, thum. Sawka K., Wydawnictwo HELION, Gliwice, 2016.
- [7] Źyliński M., kurs internetowy: *Uczenie maszynowe od podstaw z Azure Machine Learning Studio*, Platforma <https://www.udemy.com>
- [8] Grzyb M., *Czym jest Data Science?*, 2016, URL: <https://mateuszgrzyb.pl/czym-jest-data-science/>
- [9] Microsoft Azure Machine Learning Studio, Dział Dokumentacji I Pomocy, [online] URL: <https://azure.microsoft.com/pl-pl/>

## Spis tabel

<b>Tabela 1.1.</b> Przykładowe wyniki procesu normalizacji i standaryzacji.....	22
<b>Tabela 1.2.</b> Przykład danych w celu opisu problemu prywatności danych.....	23
<b>Tabela 1.3.</b> Charakterystyki poszczególnych algorytmów .....	28
<b>Tabela 3.1.</b> Wyniki przeprowadzonego eksperymentu działania wybranych algorytmów w ich podstawowych ustawieniach.....	84
<b>Tabela 3.2.</b> Wyniki przeprowadzonego eksperymentu badania wpływu normalizacji danych ..	85
<b>Tabela 3.3.</b> Wyniki przeprowadzonego eksperymentu badania wpływu funkcji SMOTE.....	87

## Spis rysunków

Rysunek 1.1. Sztuczna inteligencja, Data Science i uczenie maszynowe – działy .....	14
Rysunek 1.2. Uczenie nadzorowane – podział na informacje.....	15
Rysunek 1.3. Uczenie nadzorowane – przykład podziału na informacje .....	16
Rysunek 1.4. Przykład klasifikacji binarnej .....	16
Rysunek 1.5. Przykład regresji liniowej .....	17
Rysunek 1.6. Uczenie nienadzorowane – podział na informacje.....	17
Rysunek 1.7. Przykład grupowania za pomocą klasteryzacji .....	18
Rysunek 1.8. Schemat oddziaływania w metodzie uczenia przez wzmacnianie.....	19
Rysunek 1.9. Schemat etapów modelu predykcyjnego.....	19
Rysunek 1.10. Schemat metodyki CRISP-DM .....	20
Rysunek 1.11. Etapy eksploracji danych oraz związane z nimi problemy.....	20
Rysunek 1.12. Przykładowa baza danych zapisana w formacie CSV.....	21
Rysunek 1.13. Tworzenie wykresów pudelkowych.....	24
Rysunek 1.14. Przykład zjawiska <i>underfittingu</i> i <i>overfittingu</i> .....	25
Rysunek 1.15. Microsoft Azure Machine Learning: Algorithm Cheat Sheet.....	27
Rysunek 1.16. Przykład regresji liniowej.....	30
Rysunek 1.17. Przykład regresji logistycznej.....	31
Rysunek 1.18. Przykład tabeli decyzyjnej i drzewa decyzyjnego.....	34
Rysunek 1.19. Etapy działania SVM.....	36
Rysunek 1.20. Przykład metody działania SVM .....	37
Rysunek 1.21. Ocena jakości modelu regresyjnego .....	38
Rysunek 1.22. Etapy wyliczania średniego błędu bezwzględnego.....	39
Rysunek 1.23. Etapy wyliczania średniego bezwzględnego błędu procentowego .....	40
Rysunek 1.24. Etapy wyliczania pierwiastka błędu średniokwadratowego .....	40
Rysunek 1.25. Histogram błędów dla modelu wykorzystującego algorytm regresji.....	42
Rysunek 1.26. Macierz błędów wykorzystywana do określenia jakości klasyfikatorów binarnych .....	42
Rysunek 1.27. Przykład krzywej ROC .....	44
Rysunek 1.28. Przykład macierzy błędów dla modeli wykorzystujących klasyfikatory wieloklasowe .....	45
Rysunek 2.1. System do pomiaru parametrów pogodowych skonstruowany w ramach pracy inżynierskiej Autora.....	46
Rysunek 2.2. Elementy elektroniczne wchodzące w skład urządzenia pomiarowego .....	47
Rysunek 2.3. Etap powstawania prototypu urządzenia pomiarowego.....	48
Rysunek 2.4. Schemat elektryczny cz.1 – poszczególne moduły.....	49
Rysunek 2.5. Schemat elektryczny cz. 2 - mikrokontroler .....	50
Rysunek 2.6. Projekt płyty PCB skonstruowanego urządzenia – widok całościowy.....	51
Rysunek 2.7. Projekt płyty PCB skonstruowanego urządzenia – widok warstwy Top .....	51
Rysunek 2.8. Projekt płyty PCB skonstruowanego urządzenia – widok warstwy Bottom .....	52
Rysunek 2.9. Etap wytrawiania płyty PCB .....	52
Rysunek 2.10. Płyta PCB z mikrokontrolerem.....	53
Rysunek 2.11. Płyta PCB- po lewej strona Top, po prawej strona Bottom .....	53
Rysunek 2.12. Projekt płyty PCB w 3D .....	54
Rysunek 2.13. Płyta PCB, przycisk, ekran oraz akumulator w projekcie 3D.....	54
Rysunek 2.14. Projekt obudowy w 3D.....	55
Rysunek 2.15. Wizualizacja projektu urządzenia pomiarowego .....	56
Rysunek 2.16. Obudowa urządzenia pomiarowego z zamontowanym ekranem .....	57
Rysunek 2.17. Interfejs Nextion Editor .....	58
Rysunek 3.1. Główny ekran narzędzia Microsoft Azure Machine Learning Studio .....	62
Rysunek 3.2. Przestrzenie robocze w Microsoft Azure Machine Learning Studio.....	62
Rysunek 3.3. Ekran wyboru możliwości tworzenia nowych eksperymentów.....	63
Rysunek 3.4. Ekran tworzenia nowego eksperymentu .....	63
Rysunek 3.5. Fragment pliku z zarejestrowanymi wartościami z urządzenia pomiarowego ....	64
Rysunek 3.6. Fragment pliku z zarejestrowanymi wartościami z urządzenia pomiarowego .....	65
Rysunek 3.7. Dodawanie bazy danych.....	66
Rysunek 3.8. Wizualizacja bazy danych w Microsoft Azure Machine Learning Studio .....	67
Rysunek 3.9. Histogram opisujący parametr daty .....	68
Rysunek 3.10. Wizualizacja parametru szerokości geograficznej .....	68
Rysunek 3.11. Wizualizacja parametru aktywności .....	69
Rysunek 3.12. Zestawienie wartości aktywności do dat wykonywanych pomiarów .....	70
Rysunek 3.13. Próba połączenia trzech baz danych w jedną .....	70
Rysunek 3.14. Model wstępnego przetwarzania danych .....	71
Rysunek 3.15. Funkcja Select Columns .....	72
Rysunek 3.16. Funkcja Convert to Dataset i Clean Missing Data.....	72
Rysunek 3.17. Baza danych po usunięciu zbędnych linii .....	73
Rysunek 3.18. Zmiana typów danych na Float .....	74
Rysunek 3.19. Zmiana parametru <i>Activity</i> na typ danych Boolean .....	74
Rysunek 3.20. Zmiana parametrów daty i czasu na typ danych <i>DateTime</i> .....	75
Rysunek 3.21. Przykład danych statystycznych pomiarów temperatury powietrza .....	75
Rysunek 3.22. Wykresy pomiarów temperatury wody (po lewej) oraz szerokości geograficznej (po prawej) .....	76
Rysunek 3.23. Wyznaczone miejsca przez moduł GPS .....	77
Rysunek 3.24. Wybór opcji w funkcji Clip Values .....	78
Rysunek 3.25. Finalna wersja bazy danych .....	78
Rysunek 3.26. Opis podstawowej konstrukcji modelu analitycznego w Microsoft Azure Machine Learning Studio .....	79
Rysunek 3.27. Ustawione parametry opcji Split Data .....	80
Rysunek 3.28. Ustawiony parametr opcji Train Model .....	81
Rysunek 3.29. Fragment przykładowych wyników uzyskanych z opcji Score Model .....	81
Rysunek 3.30. Fragment przykładowych wyników uzyskanych z opcji Evaluate Model .....	82
Rysunek 3.31. Projekt badanego modelu analitycznego .....	83
Rysunek 3.32. Zestawienie proporcji odpowiedzi przed i po użyciu funkcji SMOTE .....	87
Rysunek 3.33. Projekt modelu analitycznego do przeprowadzania kolejnych badań .....	89
Rysunek 3.34. Wyniki eksperymentu dla algorytmu Two-Class Boosted Decision Tree .....	90
Rysunek 3.35. Wyniki eksperymentu dla algorytmu Two-Class Decision Forest .....	91
Rysunek 3.36. Wyniki eksperymentu dla algorytmu Two-Class Decision Jungle .....	92
Rysunek 3.37. Wyniki eksperymentu dla algorytmu Two-Class Locally-Deep Support Vector Machine .....	93
Rysunek 3.38. Wyniki eksperymentu dla algorytmu PCA-Based Anomaly Detection .....	94
Rysunek 3.39. Wykres zestawiający wyniki dla algorytmu Two-Class Boosted Decision Tree .....	95
Rysunek 3.40. Wykres zestawiający wyniki dla algorytmu Two-Class Decision Forest .....	95
Rysunek 3.41. Wykres zestawiający wyniki dla algorytmu Two-Class Decision Jungle .....	96
Rysunek 3.42. Wykres zestawiający wyniki dla algorytmu Two-Class Locally-Deep Support Vector Machine .....	96
Rysunek 3.43. Wykres zestawiający wyniki dla algorytmu PCA-Based Anomaly Detection .....	97
Rysunek 3.44. Fragment opcji ustawień dla algorytmu Two- Class Decision Jungle .....	98
Rysunek 3.45. Projekt modelu analitycznego do optymalizacji funkcji predykcyjnej .....	99
Rysunek 3.46. Zestawienie wyników po użyciu funkcji optymalizującej parametry algorytmu .....	100
Rysunek 3.47. Wygląd modelu po pierwszym etapie operacyjonalizacji .....	101
Rysunek 3.48. Rezultaty uzyskiwane za pomocą skonstruowanego modelu .....	102
Rysunek 3.49. Wizualizacja wygenerowanej aplikacji w wersji testowej .....	103