SZKOŁA GŁÓWNA HANDLOWA W WARSZAWIE
WARSAW SCHOOL OF ECONOMICS

Studium Magisterskie

Kierunek Analiza Danych - Big Data

Karol Szerszeń
Nr albumu 71273

Optymalizacja marketingu bezpośredniego z wykorzystaniem genetycznych sieci neuronowych

Praca magisterska
napisana w Instytucie Ekonometrii

pod kierunkiem naukowym
Dr. Michała Bernardellego

Warszawa 2018

Spis treści

Streszczenie

Celem pracy jest zbadanie skuteczności genetycznych sieci neuronowych w problemie optymalizacji kampanii marketingu bezpośredniego. Weryfikacji została poddana hipoteza, mówiąca o większej skuteczności tego modelu aniżeli modelu referencyjnego jakim jest regresja logistyczna.

Praca składa się z dwóch części: teoretycznej i praktycznej. W części teoretycznej opisane zostało zagadnienie marketingu bezpośredniego oraz rola jaką odgrywa w niej analiza danych oraz modele statystyczne. W drugim rozdziale przedstawiony został klasyczny algorytm genetyczny oraz szczegóły implementacji w poniższej pracy. Rozdział trzeci obejmuje przedstawienie modelu sieci neuronowych, przedstawienie problemu topologii oraz powiązanie jej z zastosowanym algorytmem genetycznym.

W części praktycznej przedstawiono zbiór danych programu lojalnościowego PAYBACK, który posłużył do weryfikacji postawionej hipotezy. Przedstawiono zawarte w nim zmienne, podzielono go na zbiory testowy oraz treningowy jak również dokonano niezbędnych przekształceń ze względu na niezbilansowanie zbioru. Dokonano analizy topologii sieci neuronowych, uzyskanych w wyniku działania algorytmu genetycznego oraz porównano najlepsze z nich z modelem referencyjnym co pozwoliło stwierdzić, iż zastosowany model ma pozytywny wpływ na wyniki kampanii marketingu bezpośredniego i jego implementacja posiada biznesowe uzasadnienie.

Wstęp

Celem pracy było porównanie efektywności działania genetycznej sieci neuronowej z klasycznymi metodami ekonometrycznymi wykorzystywanymi w problematyce marketingu bezpośredniego. Przeprowadzone zostało badanie empiryczne mające na celu porównanie mocy predykcyjnej modeli oraz skuteczności klasyfikacji maksymalizującej zysk w omawianym zagadnieniu marketingowym.

Formy promocji, dzielące się na marketing masowy oraz bezpośredni, są głównymi źródłami kontaktu przedsiębiorstwa z klientami w celu poinformowania o dostępnym asortymencie produktowym. Głównymi środkami komunikacji w marketingu bezpośrednim są poczta, telefon oraz różnorodne powiadomienia internetowe, takie jak wiadomości e-mail czy reklamy umieszczane na witrynach. Skuteczność danego działania marketingowego, mierzona za pomocą wskaźnika konwersji, jest proporcjonalna do stosunku poniesionego kosztu i odnotowanego zysku. Przewaga marketingu bezpośredniego nad marketingiem masowym polega na możliwości personalizacji oferty oraz selekcji grupy klientów, do której trafi (ang. *targeting*[1]). Pozwala to na ograniczenie wolumenu danego sposobu komunikacji, czyli zmniejszenie kosztu, przy jednoczesnym zwiększeniu prawdopodobieństwa sprzedaży przez wskazanie osób z największą skłonnością do transakcji, co tym samym zwiększa zysk.

Problem selekcji klientów jest zagadnieniem, do którego rozwiązania wykorzystywane są metody ilościowe – modele statystyczne czy uczenie maszynowe – na podstawie zebranych danych historycznych o klientach. Ze względu na swoją interpretowalność i zadowalające wyniki często wykorzystywanym modelem jest regresja logistyczna. Poświęcając możliwość interpretacji wyników możliwe jest zastosowanie również innych metod, takich jak drzewa decyzyjne, lasy losowe czy sieci neuronowe. Te ostatnie z powodzeniem zastosował Microsoft do optymalizacji zagadnienia marketingu bezpośredniego wskazując na obniżenie kosztów o 35%[2] . Ten model zastosowany został również w poniższej pracy, z wykorzystaniem algorytmu genetycznego do poszukiwania jak najwydajniejszej architektury sieci. Dane użyte w

[1] Thomas B., Housden M., *Direct and Digital Marketing in Practice,* Bloomsbury Publishing Plc, 2017

[2] https://marketinginsidergroup.com/content-marketing/artificial-neural-networks-every-marketer-know/, dostęp: 20 sierpień 2018

testowaniu metody pochodzą z bazy danych programu lojalnościowego PAYBACK i informują o odpowiedzi na jedną z kampanii marketingowych.

W rozdziale pierwszym przedstawione zostało pojęcie marketingu bezpośredniego, zastosowanie metod analizy danych w tym zagadnieniu oraz podstawowe miary oceny ich efektywności. W rozdziałach drugim oraz trzecim zaprezentowane zostały teoretyczne podstawy odpowiednio algorytmu genetycznego oraz sieci neuronowych. W czwartym rozdziale został scharakteryzowany zbiór danych oraz opisane zostały przekształcenia zmiennych. W rozdziale piątym zostały przedstawione wyniki modelowania, implementacji algorytmu genetycznego i sieci neuronowej oraz porównanie zastosowanego rozwiązania z tradycyjnym podejściem.

1. Marketing bezpośredni

1.1 Pojęcie marketingu bezpośredniego

Współcześnie przeciętny konsument wystawiony jest na kilkaset do kilku tysięcy ofert marketingowych dziennie[3], nie zdając sobie nawet z tego faktu sprawy a co za tym idzie ignorując je. Z tego powodu przyciągnięcie klienta za pomocą tradycyjnych, masowych kanałów promocji staje się coraz mniej popularne. Mianem nowej koncepcji marketingu został określony[4] marketing bezpośredni, który w latach 90 XX. wieku zaczął się wyodrębniać jako autonomiczna dyscyplina teoretyczna i praktyczna spośród szerokiego grona technik promocji. Jak podaje Szymoniuk[5] przez marketing bezpośredni określamy „system wzajemnego oddziaływania, który stosuje jeden lub wiele nośników reklamowych w celu skłonienia klienta do obserwowalnej reakcji, prowadzącej do zawarcia transakcji w dowolnym miejscu. Obejmuje całokształt działań marketingowych wykorzystujących wielofunkcyjne media reklamy bezpośredniej i bazę danych do zbudowania długotrwałej, indywidualnej i obopólnie korzystnej więzi z klientem". Kanałami, które służą do kontaktu z klientem za pomocą tej techniki są coraz rzadziej poczta i ulotki a coraz częściej spersonalizowane wiadomości e-mail, powiadomienia *push* na stronach internetowych czy wiadomości sms. Wprost z powyższej definicji wynika, że główne cechy marketingu bezpośredniego to:[6]

1. zorientowanie na potrzeby indywidualnych klientów a nie ich grup,
2. znaczenie bazy danych o klientach,
3. budowanie relacji z najatrakcyjniejszymi klientami,
4. kierowanie nacisku na bezpośrednią komunikację z indywidualnym klientem.

Staje się więc oczywiste, że ten rodzaj sposobu na docieranie do klienta jest preferowany jako przynoszący największą szansę na konwersję[7]. Dodatkowo kanałem, który cieszy się największą popularnością jest wiadomość e-mail. Jednym z powodów

[3] https://sjinsights.net/2014/09/29/new-research-sheds-light-on-daily-ad-exposures/, dostęp: 10 luty 2018
[4] Trojanowski M., *Marketing bezpośredni. Koncepcja – zarządzanie – instrumenty*, PWE, Warszawa 2010
[5] Szymoniuk B., *Komunikacja marketingowa. Instrumenty i metody*, PWE, Warszawa 2006
[6] Dobski P., Szuman-Dobska M., *Marketing bezpośredni*. Wydawnictwo Prawno-Ekonomiczne, Warszawa 1999
[7] https://mediaclick.pl/optymalizacja-wspolczynnika-konwersji-cr-ang-ang-conversion-rate/, dostęp: 20 sierpień 2018

tego stanu jest powszechny dostęp Internetu – jak podaje GUS[8] w 2016 roku 80,4% polskich gospodarstw domowych posiadało dostęp do Internetu, a aż 97% internautów to aktywni użytkownicy skrzynek pocztowych. Jeśli weźmiemy pod uwagę fakt, iż jest to też jeden z najtańszych sposobów na dotarcie klienta ze zwrotem z inwestycji około 4000%[9] nie powinna dziwić jego popularność wśród marketingowców.

1.2 Analiza danych w marketingu

Analiza danych w marketingu spełnia trzy zasadnicze role[10]:

1. zwiększenie efektywności działań marketingowych,
2. rozwój relacji z klientem,
3. zwiększanie przewagi konkurencyjnej na rynku.

Analityka odgrywa kluczową rolę w pierwszym punkcie, na którym skupia się również niniejsza praca. Pozwala ona na identyfikację nieefektywnych działań i wydatków marketingowych. Miarą efektywności danego sposobu komunikacji z użytkownikiem jest współczynnik konwersji (CR – ang. *conversion rate*)[11]:

$$CR = \frac{A}{C} * 100\%,$$

gdzie:

- A – liczba akcji podjętych przez odbiorców,
- C – sumaryczna liczba odbiorców.

W przypadku komunikacji skłaniającej do zakupu, za akcję uznaje się dokonanie transakcji, zaś liczba odbiorców to wolumen danej wysyłki. Wskaźnik ten można maksymalizować, gdy ofertę otrzymują osoby, u których prawdopodobieństwo podjęcia akcji jest największe – czyli zastosowanie tzw. *selekcji klientów*. W Tabela 1 przedstawiono ekonomię przykładowej wysyłki masowej o wolumenie 250 tys. Zysk wypracowany przez to działanie marketingowe bez zastosowania selekcji to 5 tys. zł, przy jednostkowym zysku 20 zł oraz koszcie 0,2 zł. Możliwa jest jego znaczna poprawa poprzez wyselekcjonowanie pierwszych 6 decyli najlepszych klientów.

---

[8]https://stat.gov.pl/obszary-tematyczne/nauka-i-technika-spoleczenstwo-informacyjne/spoleczenstwo-informacyjne/spoleczenstwo-informacyjne-w-polsce-w-2016-roku,2,6.html, dostęp: 13 listopad 2017

[9] Błażewicz G., *Rewolucja z marketing automation. Jak wykorzystać potencjał Big Data*, PWN, Warszawa 2016, str. 72.

[10] Blattberg R. C., Kim Byung-Do, Neslin S.A, *Database Marketing*, Springer, 2008

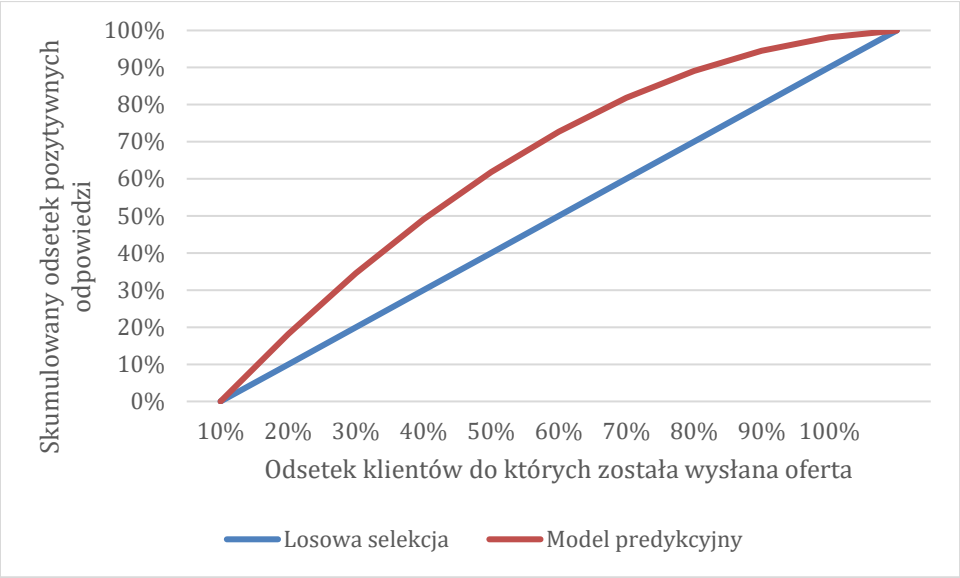
[11] https://mediaclick.pl/optymalizacja-wspolczynnika-konwersji-cr-ang-ang-conversion-rate/, dostęp: 20 sierpień 2018

Tabela 1 Ekonomika akcji marketingowej

Decyl	**Liczba wysyłek**	**Konwersja**	**Zysk (zł)**	**Skumulowany Zysk (zł)**
1	25 tys.	2,00%	5000	5000
2	25 tys.	1,80%	4000	9000
3	25 tys.	1,60%	3000	12000
4	25 tys.	1,40%	2000	14000
5	25 tys.	1,20%	1000	15000
6	25 tys.	1,00%	0	15000
7	25 tys.	0,80%	-1000	14000
8	25 tys.	0,60%	-2000	12000
9	25 tys.	0,40%	-3000	9000
10	25 tys.	0,20%	-4000	5000
Ogółem	**250 tys.**	**1,10%**	**5000**	

Źródło: Opracowanie własne

Wykorzystując dane historyczne znajdujące się w bazie danych możliwe jest zbudowanie modelu predykcyjnego określającego prawdopodobieństwo konwersji danego użytkownika. W wyniku posortowania wszystkich potencjalnych odbiorców według tego prawdopodobieństwa, możliwy jest podział całego zbioru klientów na równoliczne grupy, które różnią się wskaźnikiem odpowiedzi. Pozwala to na wyselekcjonowanie grup, do których wysyłka jest najbardziej opłacalna – w podanym przykładzie są to osoby znajdujące się w decylach 1 do 6. Takie postępowanie pozwala zwiększyć zysk z 5 tys. zł do 15 tys. zł a współczynnik konwersji z 1,1% do 1,5%.



Rysunek 1 Wykres zysku dla losowej selekcji oraz modelu predykcyjnego dla przykładu z Tabeli 1.

Sposobem pozwalającym na wizualizację powyższego przykładu jest wykres zysku (ang. *gains chart*)[12], który dla powyższego przykładu zaprezentowano na Rysunek 1. Pozwala on na określenie zależności pomiędzy odsetkiem klientów, do których oferta została wysłana, a odsetkiem tych, którzy na nią faktycznie odpowiedzieli. Dla analizowanego przykładu wysyłka do 50% osób z zastosowaniem modelu losowego pozwala na trafienie do 50% wszystkich osób, które odpowiedzą na kampanię. Natomiast wykorzystanie modelu predykcyjnego pozwala na dotarcie do 60% osób, które potencjalnie odpowiedzą na kampanię.

W niniejszej pracy przeanalizowana zostanie przykładowa wysyłka jednego z czołowych programów lojalnościowych działających na terenie Polski. Ma ona formę wiadomości e-mail wysyłanej na osobiste konto użytkownika programu. Jej celem jest skłonienie do zakupu w jednym z kilku sklepów przedstawionych w wiadomości.

[12] Boire R., *Data Mining for Managers*, Palgrave Macmillan, 2014

2. Algorytmy genetyczne

2.1 Algorytmy ewolucyjne

Algorytmy ewolucyjne to techniki optymalizacyjne inspirowane analogiami biologicznymi. Oparte są na idei doboru naturalnego i ewolucji osobników danej populacji pod wpływem zmieniających się czynników środowiska[13]. Presja tegoż środowiska stymuluje proces selekcji naturalnej prowadzący do promowania osobników posiadających lepsze predyspozycje i eliminacji tych z mniejszymi, tym samym zwiększając szanse przetrwania całej populacji. Idea algorytmów ewolucyjnych sięga lat 50. XX wieku, ale matematyczne podstawy zostały opracowane przez Johna H. Hollanda[14] w 1975 roku. Przedstawił on modele pozwalające na opisanie nieliniowych interakcji z jakimi mamy do czynienia w procesie ewolucji i dostosowania do zmieniającego się środowiska. Techniki optymalizacyjne oparte na algorytmach ewolucyjnych używają podejścia opartego nie na pojedynczych rozwiązaniach ale na całych ich populacjach[15], gdzie w pojedynczej iteracji bierze udział zbiór rozwiązań i "ewoluuje" w nowy zbiór rozwiązań w kolejnej iteracji. Gwiazda[16] wyróżnia następujące zalety algorytmów ewolucyjnych:

- brak wymagań co do postaci rozwiązywanego problemu - nie jest wymagane podanie postaci funkcji celu, a jedynie jej istnienie lub bardziej ogólnie - istnienie miary, która pozwala na wybranie lepszego z dwóch dostępnych rozwiązań, gdzie termin *lepsze* jest definiowany przez konkretne zagadnienie, do którego algorytm jest wykorzystywany;
- zdolność opuszczania lokalnych ekstremów - dzięki rozpatrywaniu populacji rozwiązań oraz mechanizmom samych algorytmów, mimo znalezienia lokalnego ekstremum możliwe jest jego opuszczenie i kontynuowanie poszukiwania rozwiązania w innej przestrzeni;

---

[13] Figielska E., *Algorytmy ewolucyjne i ich zastosowanie*, w: http://zeszyty-naukowe.wwsi.edu.pl/zeszyty/zeszyt1/Algorytmy_Ewolucyjne_I_Ich_Zastosowania.pdf, dostęp: listopad 2017

[14] Holland J. H., *Adaptation in Natural and artificial systems*, MIT Press , Cambridge 1975

[15] Deb K., *Multi-Objective Optimization using Evolutionary Algorithms*, Wiley, 2001

[16] Gwiazda T. D., *Algorytmy ewolucyjne w rozwiązywaniu nieliniowych problemów decyzyjnych*, Wydawnictwa Naukowe Wydziału Zarządzania Uniwersytetu Warszawskiego, 2002

- uniwersalność stosowania - brak istnienia dedykowanego rozwiązania danego problemu nie jest przeszkodą do zastosowania algorytmu w jego klasycznej postaci.

Dodatkowo algorytmy ewolucyjne operując na populacjach rozwiązań, a nie na pojedynczym, punktowym rozwiązaniu, pozwalają na łatwe zastosowanie przetwarzania wielowątkowego, co często prowadzi do krótszego czasu obliczeń w porównaniu z klasycznymi metodami optymalizacji.

Natomiast do głównych wad algorytmów ewolucyjnych możemy zaliczyć[17]:

- uniwersalność prowadząca do mniejszej skuteczności niż w przypadku algorytmów dedykowanych;
- większa złożoność obliczeniowa - algorytmy ewolucyjne są zwykle wolniejsze od metod zachłannych (obliczenia prowadzone na całych populacjach);
- niedeterministyczność - z założenia algorytm jest losowy, więc odtworzenie rozwiązania bywa niemożliwe;
- postać funkcji celu ma duży wpływ na jakość rozwiązania.

Algorytmy ewolucyjne są określeniem stosowanym do opisu klasy zagadnień bazujących na zbliżonej metodologii, obejmującej m.in.:

- algorytmy genetyczne,
- programowanie genetyczne,
- strategie ewolucyjne.

W niniejszej pracy zdecydowano się na zastosowanie algorytmów genetycznych, głównie ze względu na ich możliwość przeszukiwania przestrzeni decyzyjnej w kilku punktach jednocześnie.

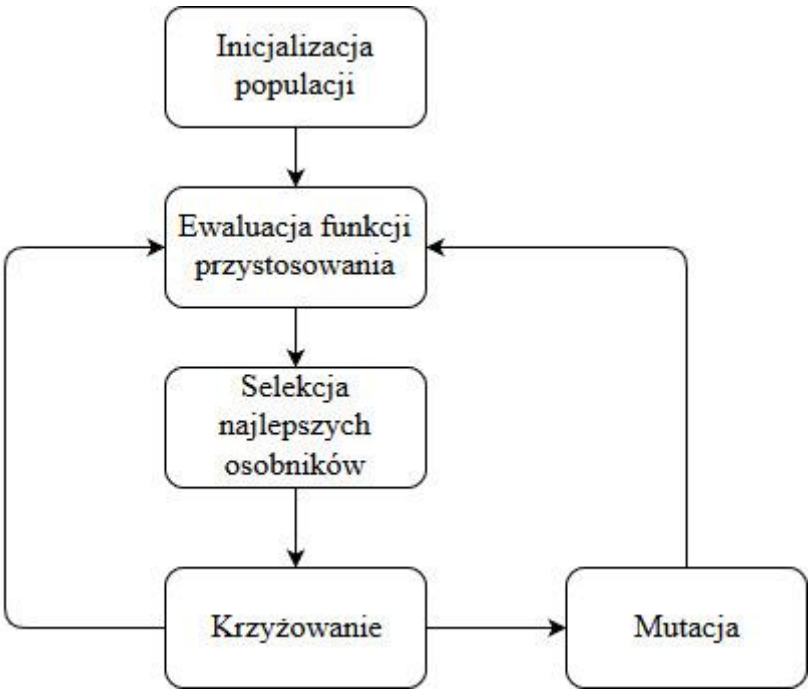
2.2 Klasyczny algorytm genetyczny

Algorytmy ewolucyjne charakteryzują się specyficzną nomenklaturą zaczerpniętą z pojęć stosowanych w dziedzinie biologii - genetyce. W klasycznym algorytmie genetycznym[18] możemy wyróżnić struktury, zwane chromosomami (ang. *chromosomes*), będące elementami dziedziny optymalizowanej funkcji, a składające się z mniejszych jednostek, to jest genów. Zbiór struktur w poszczególnych etapach przetwarzania (w

[17] http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad10/w10.htm, dostęp: 20 luty 2018

[18] Trzaskalik T., *Algorytmy genetyczne, ewolucyjne i metaheurystyki.* Katowice, Wydawnictwo Akademii Ekonomicznej w Katowicach, 2005, s.13-14

danej iteracji) tworzy populację. Określamy również funkcję przystosowania (ang. *fitness function*), która pozwala na mierzenie jakości dopasowania tych struktur podczas tworzenia nowej populacji. Najlepsze z nich "przeżywają" i uczestniczą w procesie reprodukcji, przeprowadzanym przez mechanizm selekcji, natomiast pozostałe "wymierają" i nie są uwzględniane w przyszłych populacjach. Geny struktur wyselekcjonowanych do reprodukcji są wybierane przez operator krzyżowania i wymieniane pomiędzy rozwiązaniami rodzicielskimi. Z niewielkim prawdopodobieństwem poszczególne geny mogą zostać poddane mutacji, czyli wymienione na inne. Dzięki powyższym mechanizmom przeszukiwanie przestrzeni rozwiązań w kolejnych iteracjach zostaje zawężone do potencjalnie najbardziej obiecujących obszarów przestrzeni decyzyjnej, a w konsekwencji zwiększa szansę na znalezienie rozwiązania optymalnego. Zastosowanie algorytmu genetycznego sprowadza się do dbania o odpowiedni stopień zróżnicowania populacji, co można osiągnąć dzięki zastosowaniu operatora mutacji. W konsekwencji pozwala to na ominięcie lub wyjście z ekstremum lokalnego. Innym sposobem osiągnięcia tego efektu jest maksymalne zróżnicowanie populacji początkowej - maksymalizacja entropii.



Rysunek 2. Schemat klasycznego algorytmu genetycznego,

Źródło: opracowanie własne

Na Rysunek 2 zostało przedstawione działanie klasycznego algorytmu genetycznego - mutacja oznaczona została jako opcjonalny krok w wyznaczaniu populacji potomków. Inicjalizacja populacji początkowej to tworzenie losowego zbioru,

na którym wykonywane będą dalsze kroki. Ważne jest, aby charakteryzowała się dużym zróżnicowaniem, gdyż efektywność początkowego algorytmu jest uzależniona właśnie od tego zbioru. Wielokrotne występowanie ciągów genów o wysokiej wartości funkcji przystosowania może prowadzić do całkowitego zdominowania rozwiązania przez właśnie te geny.

2.3 Kodowanie chromosomów

Jednym z najistotniejszych problemów w algorytmach genetycznych jest sposób kodowania chromosomów oraz pojedynczych genów. Zależy on głównie od optymalizowanego zagadnienia, co sprawia, że istnieje wiele sposobów kodowania chromosomów. Najpopularniejszym, a jednocześnie najprostszym sposobem kodowania jest kodowanie binarne, polegające na zapisaniu chromosomu za pomocą ciągu zer i jedynek[19], jak na przykład dla dwóch poniższych chromosomów A i B:

A = [1 0 0 1 0 1 0 0 1 1 1]

B = [1 0 0 0 1 1 1 0 1 0 1]

Taki sposób kodowania został opracowany po raz pierwszy przez Hollanda[20] oraz jego współpracowników, zatem i dla niego zostały przewidziane wszystkie operatory – selekcji, krzyżowania i mutacji.

Jednym z częściej stosowanych, a podobnym do binarnego, jest kodowanie za pomocą liczb rzeczywistych lub całkowitych[21]. Chromosomy również są ciągami liczb o równej długości, ale zakres liczbowy obejmuje więcej niż tylko dwie wartości:

A = [2 0 14 5 7 9 230]

B = [3 11 9 5 83 1 39]

Dla wielu zastosowań jest to bardziej naturalny sposób reprezentacji rozwiązań niż kodowanie binarne – gdy cechy reprezentowane przez pojedyncze geny przyjmują wartości z pewnego zakresu, np. od 1 do 256, możliwe jest zakodowanie całego ich zestawu w pojedynczym chromosomie. Ten sposób kodowania został użyty w niniejszej pracy, m.in. ze względu na możliwość reprezentacji wszystkich parametrów sieci neuronowej jako pojedynczego chromosomu (por. rozdział 5.2 Implementacja algorytmu genetycznego), co ułatwiło implementację rozwiązania. Wydajność algorytmu

[19] Goldberg D.E., *Algorytmy genetyczne i ich zastosowania***,** Warszawa, WNT, 1995.
[20] Holland J. H., *Adaptation in Natural and artificial systems*, Cambridge, MIT Press, 1975
[21] Mitchell M., *An Introduction to Genetic Algorithm,* Cambridge, MIT Press, 1996

genetycznego może zależeć od sposobu reprezentacji genów[22] jednak jest ona w dużej mierze zależna od danego problemu oraz pozostałych składowych algorytmu. Poza tymi dwoma sposobami kodowań istnieje jeszcze wiele innych, jednak nie zostały one użyte w pracy.

2.4 Operatory

2.4.1 Selekcja

Operator selekcji pozwala na zdefiniowanie procesu określającego, które rozwiązania (chromosomy) przetrwają i będą miały szansę na przekazanie genów populacji potomnej, a które wymrą". Głównym zadaniem tego operatora jest stawianie nacisku na wybór dobrze przystosowanych rozwiązań i eliminacja źle przystosowanych, przy jednoczesnym utrzymywaniu wielkości populacji na stałym poziomie. Istnieje wiele rodzajów operatorów selekcji, ale ich wspólną charakterystyką jest próba zbalansowania przystosowania oraz różnorodności osobników w danej populacji[23]. Faworyzowanie przystosowania względem różnorodności – nacisku selektywnego (ang. *selective pressure*) – prowadzi do całkowitej dominacji jednego chromosomu nad innymi – zjawiska stłoczenia (ang. *crowding*). Metody eliminujące problem stłoczenia zaproponowali m.in. De Jong[24] sugerując, że nowe osobniki powinny zastępować te najbardziej do nich podobne oraz Goldberg i Richardson[25], którzy przyjęli funkcję podziału przystosowania, zmniejszającą wartość przystosowania danego chromosomu wprost proporcjonalnie do skali podobieństwa do innych chromosomów, tym samym nagradzając różnorodność.

Podstawowym i najczęściej stosowanymi operatorami selekcji są:

- metoda ruletki,
- selekcja rankingowa,

---

[22] Janikow C.Z., Michalewicz Z., *An experimental comparison of binary and floating point representations in genetic algorithms*, w: *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, 1991

[23] Larose D.T., *Metody i modele eksploracji danych*, Wiley 2006

[24] De Jong K., *An analysis of the behavior of a class of genetic adaptive systems, praca doktorska*, University of Michigan, Ann Arbor, 1975

[25] Goldberg D., Richardson J., *Genetic algorithms with sharing for multi-modal function optimization*, w: *Genetic Algorithms and Their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms*, Greffenstette J. (red.), Lawrence Erlbaum Associates, Hillsdale, 1987

- selekcja turniejowa.

W niniejszej pracy zastosowana została metoda ruletki, która swą nazwę zawdzięcza analogii do losowania za pomocą koła kasynowej ruletki. Ogólny schemat tej metody wygląda następująco:

1. Obliczenie sumy wartości funkcji celu (przystosowania):
   $F(x) = \sum_{i=1}^{n} f(x_i)$,
2. Obliczenie wkładu każdego osobnika w sumę: $p(x_i) = f(x_i) / F(x)$
3. Traktujemy wartości $p(x_i)$ jako rozkład prawdopodobieństwa i *n*-krotnie losujemy osobniki zgodnie z tym rozkładem,

gdzie:

n – liczebność populacji,

x_i – *i*-ty osobnik populacji,

$f(x_i)$ – wartość funkcji przystosowania dla *i*-tego osobnika,

$p(x_i)$ – prawdopodobieństwo wylosowania *i*-tego osobnika.

Selekcja chromosomu może być postrzegana jako obrót koła ruletki, dzięki czemu zostaje wybrany chromosom, należący do wybranego fragmentu koła. Im większy jest fragment koła, tym prawdopodobieństwo wyboru jest większe a tym samym wprost proporcjonalne do wartości funkcji przystosowania danego chromosomu – im lepsze dane rozwiązanie tym częściej będzie wybierane.

2.4.2 Krzyżowanie

Krzyżowanie to proces wymiany informacji zapisanej w genach poszczególnych genotypów pochodzących od różnych rozwiązań rodzicielskich. Pozwala on na otrzymywanie nowych kombinacji cech – zarówno łącznie lepszych niż oba rozwiązania rodzicielskie, jak i z możliwością bycia gorszym od każdego z nich. Krzyżowanie jest dominującym operatorem w algorytmach genetycznych – decyduje o konieczności utrzymania różnorodnej oraz licznej populacji. W pracy zastosowany został operator krzyżowania prostego, którego schemat został przedstawiony poniżej.

Krzyżowanie proste[26] polega na wybraniu losowej liczby *m* – punktu krzyżowania. W kolejnym kroku zastosowana jest następująca reguła:

- pierwszy potomek otrzymuje pierwsze *m* genów od pierwszego rozwiązania rodzicielskiego, natomiast pozostałe od drugiego,
- drugi potomek otrzymuje pozostałe geny obu rozwiązań rodzicielskich, czyli pierwsze *n* od drugiego oraz pozostałe od pierwszego.

Na rysunku 3 zaprezentowane zostały podziały chromosomów rozwiązań rodzicielskich oraz tworzenie nowych chromosomów przy pomocy operatora krzyżowania, z wartością *m* = 3:

Rodzic A	0	0	0	0	0	0	0	0
Rodzic B	1	1	1	1	1	1	1	1
Potomek A	0	0	0	1	1	1	1	1
Potomek B	1	1	1	0	0	0	0	0

Rysunek 3 Przykład tworzenia nowych chromosomów w wyniku operatora krzyżowania dla n=3

Źródło: opracowanie własne

2.4.3 Mutacja

Operator mutacji polega na zmianie wartości jednego z genów danego chromosomu na inny, np. dla chromosomu [1 1 0 0 1] mutacji może ulec gen trzeci, w wyniku czego powstanie chromosom [1 1 1 0 1]. Mutacja zachodzi z – ustalanym przy inicjalizacji algorytmu – niewielkim prawdopodobieństwem *p*, najczęściej nieprzekraczającym kilku procent. Jedną z metod wyboru genów do mutacji jest wylosowanie liczby z przedziału [0, 1] dla każdego genu i zastosowanie mutacji dla tych, gdzie wylosowana liczba jest mniejsza od prawdopodobieństwa p.

2.5 Funkcja przystosowania

Funkcja przystosowania służy do oceny jakości przystosowania danego chromosomu – określa jak dobrze dany chromosom rozwiązuje dany problem. Pozwala

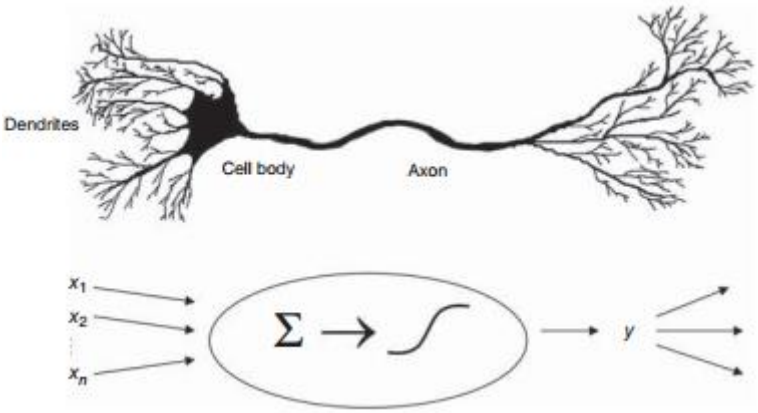
[26] Trzaskalik T., *Algorytmy genetyczne, ewolucyjne i metaheurystyki*, Katowice, Wydawnictwo Akademii Ekonomicznej w Katowicach, 2005, s.18

ona na porównywanie poszczególnych rozwiązań pomiędzy sobą, a jej zdefiniowanie jest niezbędne do działania operatora selekcji. Funkcja przystosowania jest wybierana do zagadnienia, do którego rozwiązania jest stosowany algorytm – będzie ona inna przy problemie komiwojażera oraz inna przy optymalizacji wag sieci neuronowej. Funkcja przystosowania w problemie optymalizacji topologii sieci neuronowej będzie odzwierciedlać poprawność klasyfikacji przez sieć o danej topologii. Zostanie ona omówiona w dalszej części pracy.

3. Sieci Neuronowe

3.1 Idea

Sztuczne sieci neuronowe zostały stworzone tak, aby odzwierciedlać strukturę ludzkiego mózgu. Celem było stworzenie tzw. sztucznej inteligencji. Ludzki umysł składa się z około 10^{11} neuronów, z których każdy jest połączony do średnio 10.000 innych neuronów, co daje około 10^{15} połączeń. Sieci neuronowe są próbą odtworzenia nieliniowego sposobu uczenia obecnego w sieciach neuronowych występujących w przyrodzie. Na Rysunku 4 przedstawiony został schemat biologicznego neuronu, który używa dendrytów do zbierania bodźców od innych neuronów i łączy informacje wejściowe, generując nieliniową odpowiedź i przesyłając ją przez akson do innych neuronów[27].



Rysunek 4 Schemat ludzkiego neuronu oraz prostej sieci neuronowej,

Źródło: Data Mining and Predictive Analytics, Larose, D. T. , s.340

[27] Larose D. T., Larose C., *Data Mining and Predictive Analytics*, Hoboken, Wiley, 2015

Na rysunku przedstawiona została również postać sztucznego neuronu, który obecny jest w sieciach neuronowych. Informacje wejściowe (x_i) są przechwytywane z nadrzędnych warstw neuronów (lub zbioru danych w przypadku pierwszej warstwy), następnie zbierane przez funkcję kombinacji oraz przekształcane przez nieliniową funkcję aktywacji w celu stworzenia informacji wyjściowej, która jest przekazywana do kolejnej warstwy neuronów. Pierwszy prosty model neuronu został zaproponowany przez McCullocha i Pittsa[28] w 1943 roku. Udało im się matematycznie opisać pojedynczy neuron w sposób, który pozwalał na zrozumienie podstawowych procesów uczenia zachodzących w mózgu. Sukces tego podejścia polegał na prostocie rozwiązania, które wprawdzie nie było specjalnie potężne, ale dało narzędzie, które w połączeniu w sieć stało się osobną dziedziną nauki, zaś jego możliwości przerosły oczekiwania twórców.

Do zalet sieci neuronowych możemy zaliczyć przede wszystkim ich ogólny charakter – nie ma dużego znaczenia jaki problem próbujemy rozwiązać, sieć będzie się sprawdzać tak samo dobrze. Dzięki tej ogólności można je stosować do rozwiązywania problemów, dla których nie istnieją wyspecjalizowane sposoby radzenia sobie z nimi lub są one zbyt trudne do rozwiązania metodami analitycznymi, np. zagadnienia modelowania cen na rynkach papierów wartościowych czy też zachowania silników indukcyjnych[29]. Dodatkowo sieci pozwalają na pominięcie długiego i często bardzo skomplikowanego procesu programowania danego rozwiązania zastępując je procesem uczenia sieci.

Podstawową wadą sieci neuronowych jest niewielka interpretowalność wyników. W przeciwieństwie do klasycznych metod regresji nie jesteśmy w stanie powiedzieć o ile zmieni się odpowiedź klasyfikacji, jeśli zmienimy wartość jednego z atrybutów. Nieco pomaga w interpretacji analiza wrażliwości, jednak daje ona tylko informacje o tym, który z parametrów wejściowych jest najbardziej istotny, natomiast nie pozwala na określenie jak bardzo zmieni się wartość wyjściowa.

Sieci neuronowe, znalazły zastosowania w wielu dziedzinach nauki. Potharst, Kaymak i Pijls[30] zastosowali sieci neuronowe do doboru grupy docelowej dla akcji

---

[28] Winkowska-Nowak K., *Modelowanie matematyczne i symulacje komputerowe w naukach społecznych,* Warszawa, Wydawnictwo SWPS Academica, 2007

[29] Tadeusiewicz R., *O celowości zastosowania sieci neuronowych w problemach związanych z elektrotechniką,* w: *Przegląd Elektrotechniczny,* R. 85 NR 2/2009

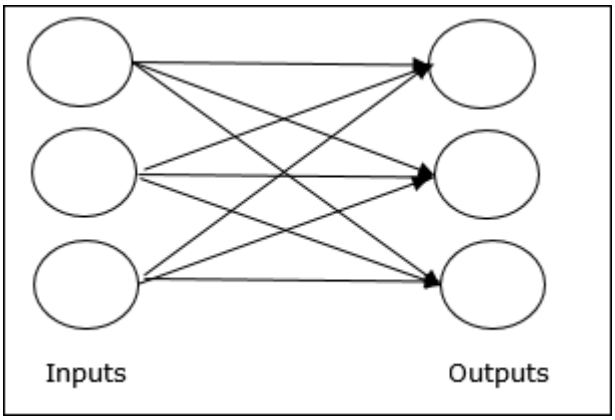
[30] Potharst R., Kaymak U., Pijls W.H.L.M., *Neural Networks for Target Selection in Direct Marketing,* w: *ERIM Report Series Research in Management*, na: http://hdl.handle.net/1765/83, data dostępu: styczeń 2018

charytatywnej. Natomiast Yang, Liu i Coid[31] użyli ich jako jednej z technik do klasyfikacji poważnych przestępstw oraz określania prawdopodobieństwa recydywy wśród skazanych. Jednym z najpopularniejszych zastosowań sieci neuronowych jest przewidywanie cen instrumentów finansowych, ze względu na nieprzewidywalność rynków oraz ciężkie do określenia zależności. Jako przykład można podać pracę Dunisa[32], w której zastosował on kilka hybrydowych rozwiązań łączących sieci neuronowe z algorytmami genetycznymi, uzyskując dobre rezultaty.

3.2 Topologia sieci

Topologia sieci neuronowych (architektura) to określenie obrazujące strukturę danej sieci, czyli sposób połączenia poszczególnych neuronów oraz miejsc, w których wykonywane są obliczenia. Istnieją trzy podstawowe klasy architektury sieci[33].

Najprostszą architekturą jest jednowarstwowa, jednokierunkowa sieć (ang. *single-layer feedforward network*) gdzie neurony ułożone są w warstwy. W tej najprostszej formie możemy wyróżnić warstwę wejściową, która przesyła informacje bezpośrednio do warstwy wyjściowej, ale nigdy na odwrót. Nazwa „jednowarstwowa" odnosi się do jednej tylko warstwy wykonującej obliczenia – wyjściowej.

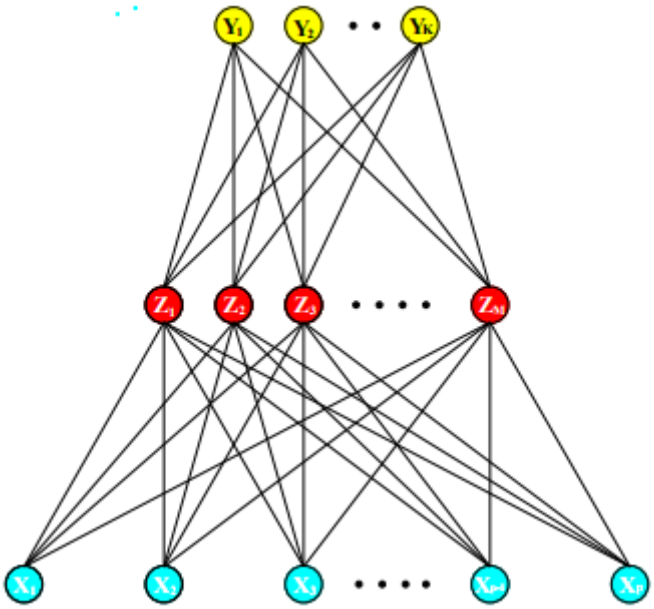


Rysunek 5 Schemat jednowarstwowej sieci neuronowej

Źródło: https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_quick_guide.htm, dostęp: sierpień 2018

[31] Yang M., Liu Y., Coid J., *Applying Neural Networks and other statistical models to the classification of serious offenders and the prediction of recidivism,* w: *Ministry of Justice Research Series,* nr 6/10, 2010
[32] Dunis CH., *Computation Intelligence Techniques for Trading and Investment*, New York, Routledge, 2014
[33] Haykin S., *Neural Networks and Learning Machines,* Pearson Prentice Hall, Upper Saddle River, 2009

Drugą klasą sieci neuronowych, którą odróżnia od poprzedniej obecność warstwy *ukrytej* jest wielowarstwowa jednokierunkowa sieć (ang. *multilayer feedforward network*, lub równoznacznie *Multi Layer Perceptron – MLP*), która została użyta w tej pracy. W tej architekturze warstwa ukryta, czy też neurony ukryte, odpowiada za większość obliczeń. Określenie *warstwa ukryta* odnosi się do faktu, iż neurony tej warstwy nie są widziane ani od strony wejściowej sieci, ani od warstwy wyjściowej. Główną funkcją jaką spełniają neurony warstwy ukrytej to przechwytywanie informacji z warstwy wejściowej oraz przekształcanie ich w sygnał wyjściowy za pomocą nieliniowych funkcji, zwanych funkcjami aktywacji. Sygnały wejściowe w warstwie wejściowej sieci dostarczają informacji o wektorze wejściowym, który jest przekształcany przez funkcję aktywacji i przekazywany do drugiej warstwy, którą w tym przypadku jest pierwsza warstwa ukryta. Sygnały wyjściowe z tej warstwy przekazywane są do kolejnej, przekształcane przez funkcję aktywacji i tak dalej, dla całej struktury sieci, aż do końcowej warstwy wyjściowej. Zestaw danych wyjściowych z ostatniej warstwy – warstwy wyjściowej – to ostateczny wynik, odpowiedź sieci na problem zadany przez dane wejściowe. W ogólnym przypadku sieć mająca m źródeł danych, h_1 neuronów pierwszej warstwy ukrytej, h_2 neuronów drugiej warstwy ukrytej oraz q neuronów w warstwie wyjściowej definiujemy jako sieć m - h_1 - h_2 - q. Z reguły takie sieci są w pełni połączone, tzn. każdy neuron jest połączony z każdym innym neuronem warstwy poprzedzającej oraz następnej. Na Rysunek 6 została zaprezentowana właśnie taka architektura, gdzie przez X_p oznaczone zostały neurony warstwy wejściowej, Z_m neurony warstwy ukrytej zaś Y_k to neurony warstwy wyjściowej.



Rysunek 6 Jednowarstwowa jednokierunkowa sieć neuronowa z warstwą ukrytą

Źródło: The Elements of Statistical Learning, Data Mining, Inference, and Prediction, *T. Hastie, R. Tibshirani, J. Friedman, Springer, New York, 2009, s.393*

Sieci rekurencyjne (ang. *recurrent networks*) różnią się od pozostałych architektur sposobem przepływu informacji. W takich sieciach istnieje co najmniej jedna pętla przekazująca informacje między warstwą wyjściową a wejściową. Jeśli sieć posiada jedną warstwę wejściową i jedną wyjściową to sygnał z każdego neuronu warstwy wyjściowej jest przekazywany do pozostałych neuronów warstwy wejściowej. Pętla informacyjna (ang. *feedback loop)* może zawierać również przepływy informacji wyjściowej danego neuronu do niego samego (ang. *self-feedback loop)*. Obecność pętli informacyjnych ma znaczący wpływ na sposób uczenia się sieci oraz na jej wydajność i dokładność. Co więcej, sieci rekurencyjne korzystają z gałęzi złożonych z elementów opóźnionych w czasie, co skutkuje nieliniowym zachowaniem, zakładając, że sieć zawiera nieliniowe elementy.

W związku z ogromnym zainteresowaniem sieciami neuronowymi w latach 90. XX wieku powstało wiele innych topologii, jednakże ich zastosowanie z reguły ograniczało się do rozwiązywania jednego, konkretnego problemu. Większość sieci neuronowych współcześnie stosowanych opiera się na przedstawionych w tym podrozdziale trzech schematach, różniąc się nieznacznie od pierwowzoru takimi meta-parametrami jak rodzaj funkcji aktywacji (sieci o radialnej funkcji aktywacji – ang. *radial*

basis function networks[34]), liczbą neuronów w warstwie ukrytej, czy też liczbą warstw ukrytych.

Topologia sieci neuronowej z reguły jest wynikiem manualnego dostosowywania parametrów, określania liczebności warstw ukrytych oraz neuronów w każdej z nich. W związku z ogromną liczbą możliwych kombinacji połączenia już najprostszej sieci MLP, proces ten jest żmudny i czasochłonny. Jednym z popularniejszych i często używanych rozwiązań tego problemu jest zastosowanie algorytmu genetycznego do optymalizacji topologii. Takie właśnie podejście zostało wykorzystane w niniejszej pracy.

3.3 Funkcje aktywacji

Funkcje aktywacji to funkcje służące do przekształcenia sygnału wejściowego neuronu do postaci nieliniowej. Jako informację wejściową dla neuronu *j* traktuje się sumę sygnałów wejściowych oraz przypisanych im wag[35]:

$$z_j = \sum_{i=0}^{n} w_{j,i} x_i,$$

gdzie:

- n – liczba zmiennych (cech) wejściowych
- x_i – wartość dla *i*-tej zmiennej wejściowej, gdzie $i = 0, 1, ..., n$
- $w_{j,i}$ – waga *i*-tej cechy dla neuronu *j*

Wobec wartości z_j zostaje zastosowana funkcja aktywacji f co w efekcie daje wartość wyjściową *o* dla neuronu *j:*

$$o_j = f(z_j)$$

Dla każdej sieci można przypisać inne funkcje aktywacji dla poszczególnych warstw a nawet neuronów, natomiast w większości wypadków stosuje się jedną funkcję aktywacji dla całej sieci. Funkcja aktywacji musi spełniać kilka założeń[36]:

- nie może być stała,

---

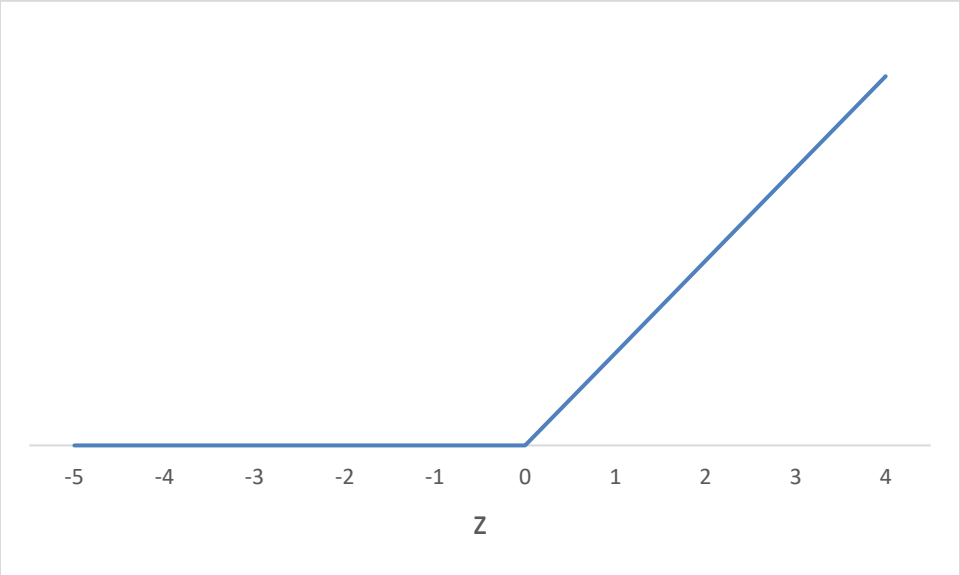
[34] Orr M. J. L., *Introduction to Radial Basis Function Networks,* na: https://www.cc.gatech.edu/~isbell/tutorials/rbf-intro.pdf, dostęp: 13 styczeń 2018
[35] Ye N., *Data Mining – Theories, Algorithms and Examples,* New York, CRC Press, 2014, s. 63
[36] Haykin S., *Neural Networks and Learning Machines,* Pearson Prentice Hall, Upper Saddle River, 2009

- musi być ograniczona,
- musi być monotonicznie rosnąca,
- musi być różniczkowalna.

Powinna ona również odzwierciedlać rodzaj odpowiedzi czy też danych wyjściowych, których model ma dostarczać – jeśli zjawisko modelowane jest binarne takiej też odpowiedzi powinna dostarczyć funkcja. W przypadku klasyfikacji z kilkoma kategoriami, musi ona móc zwrócić każdą z możliwych odpowiedzi. Biorąc pod uwagę powyższe własności, współcześnie najczęściej stosowanymi funkcjami aktywacji są[37]:

- funkcja sigmoidalna:

$$f(z) = \frac{1}{1 + e^{-z}},$$



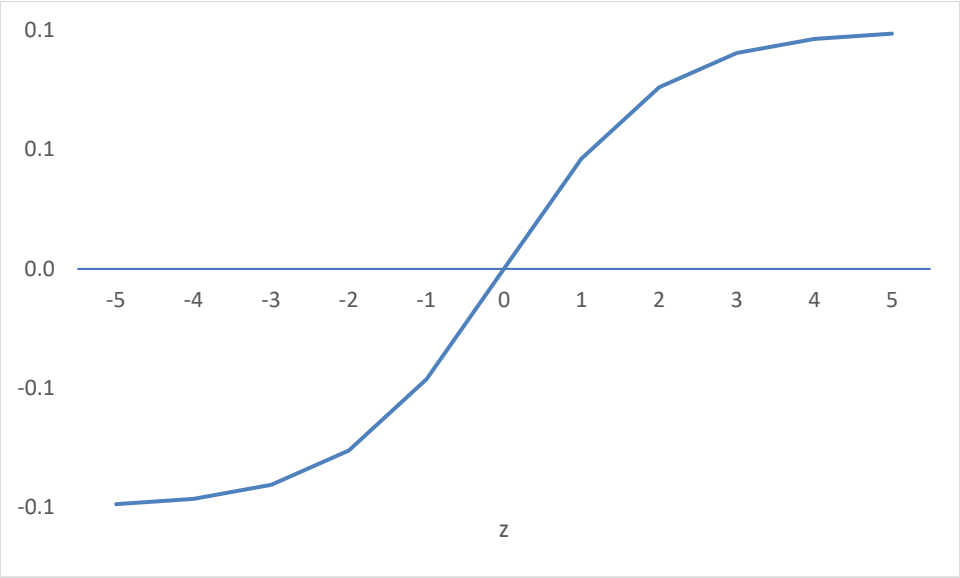
Rysunek 7 Przykład funkcji aktywacji: funkcja sigmoidalna

Źródło: opracowanie własne

- tangens hiperboliczny:

$$f(z) = \tanh(z) = \frac{1 - e^{-z}}{1 + e^{-z}},$$

[37] Buduma N., *Fundamentals of Deep Learning,* Sebastopol, O'Reilly Media, 2017

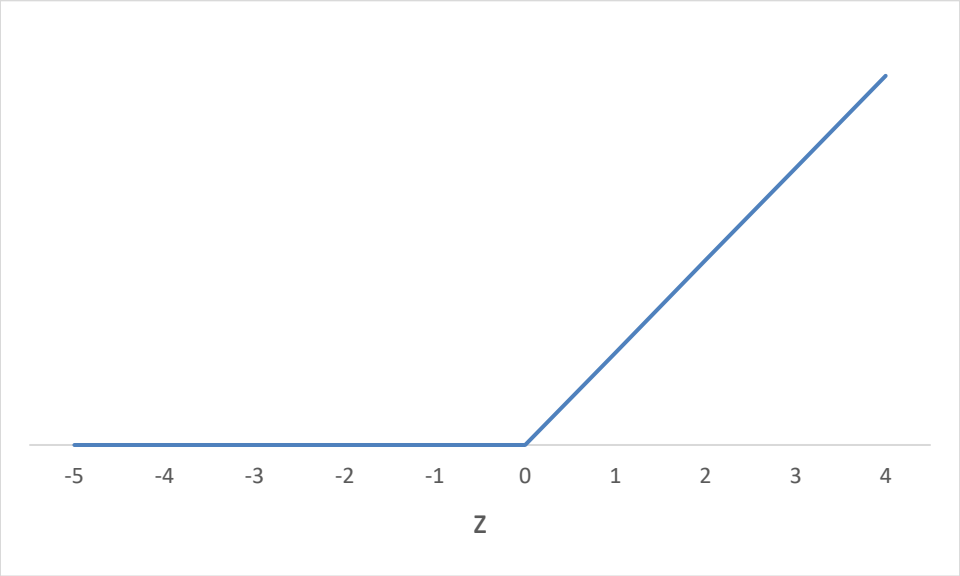


Rysunek 8 Funkcja tangens hiperboliczny

Źródło: opracowanie własne

- ReLU (ang. *restricted linear unit*):

$$f(z) = \max(0, z)$$



Rysunek 9 Funkcja ReLU

Źródło: opracowanie własne

Poza faktem spełniania wyżej wymienionych założeń powyższe funkcje mają też inne zalety. Funkcja sigmoidalna dla *z* bliskiego zeru wykazuje się liniowością, natomiast im *z* bardziej oddala się od zera tym wartości są bliższe odpowiednio 0 lub 1, stając się prawie funkcją stałą. Funkcja tangens hiperboliczny zachowuje się w bardzo podobny sposób, z tym że wartości skrajne to -1 i 1. Funkcja ReLU znajduje zastosowanie m.in. w

przetwarzaniach obrazu, głównie ze względu na wydajność obliczeń. Jest ona też standardową funkcją w części zastosowań (np. w pakiecie scikit-learn[38] języka Python).

3.4 Uczenie sieci

W zadaniach klasyfikacji, przez model dobrze dopasowany do danych rozumie się model, który minimalizuje funkcję szacującą błąd klasyfikacji. Jest to zatem przykład zagadnienia optymalizacyjnego. W przypadku sieci neuronowej błąd ten definiuje się jako funkcję zależną od macierzy wag dla neuronów *j*[39]:

$$E(W) = \frac{1}{2}\sum_j \sum_d (t_{j,d} - o_{j,d})^2,$$

gdzie:

- d – liczba obserwacji w zbiorze
- $t_{j,d}$ – wartość oczekiwana dla obserwacji *d* neuronu *j*,
- $o_{j,d}$ – wartość dla obserwacji *d* oszacowana przez neuron *j* z wagami *W*.

Metodą, która służy do optymalizacji macierzy wag w sieci neuronowej w celu uzyskania minimalnej wartości błędu zdefiniowanego powyżej jest algorytm propagacji wstecznej (ang. *backpropagation algorithm*). Został on przedstawiony po raz pierwszy w 1986 na łamach magazynu *Nature*[40]. Podstawą algorytmu jest informacja o tym, w jaki sposób zmienia się błąd E(*W*) przy zmianie wartości jednej wagi o jednostkę. Na tej podstawie możliwe jest minimalizowanie błędu w danych wyjściowych[41]. Metoda propagacji wstecznej przeszukuje przestrzeń rozwiązań i ewaluuje zbiór wag w danym kroku na podstawie wartości funkcji E(*W)* metodą spadku gradientowego (ang. *gradient descent*), który można przedstawić jako zmianę Δ wag neuronu *j* dla cechy *i*:

$$\Delta w_{j,i} = \alpha \delta_j \bar{o}_{j,i},$$

[38] http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, dostęp: 18 styczeń 2018
[39] Ye N., *Data Mining – Theories, Algorithms and Examples*, New York, CRC Press, 2014, s. 80
[40] Rumelhart D .E., Hinton G. E., Williams R. J., *Learning Representations by BackPropagating Errors*, w: *Nature* nr 323, 1986
[41] McIlwrath D., Marmanis H., Babenko D., *Inteligentna Sieć, Algorytmy przyszłości,* Gliwice, Helion, 2017

gdzie:

- $\delta_j = -E'(z_j)$, czyli wartość pochodnej cząstkowej funkcji błędu E względem wag i wartości wejściowych
- α – stała wartość współczynnika szybkości uczenia, $\alpha \in [0, 1]$,
- $\bar{o}_{j,i}$ – sygnał wejściowy *i* do neuronu *j*

Wartość współczynnika uczenia określa szybkość zmiany wag – dla wartości równej zero wagi nie zmieniają się pomiędzy iteracjami, natomiast wartość bliska jeden może spowodować rozbieżność algorytmu[42]. Jeśli neuron *j* należy do warstwy wejściowej to $\bar{o}_{j,i} = x_i$, w przeciwnym wypadku jest to wartość przekazana z poprzedzającej warstwy. Jeśli neuron *j* należy do warstwy wyjściowej to:

$$\delta_j = \left(t_{j,d} - o_{j,d}\right) f'_j(z_j),$$

gdzie:

- f'_j – pochodna cząstkowa funkcji aktywacji *f* względem sumy wag z_j.

Jeśli neuron *j* należy do warstwy ukrytej to:

$$\delta_j = \left(\sum_n \delta_n w_{n,j}\right) f'_j(net_j),$$

gdzie δ_n to gradient wag warstwy wyjściowej *n*.

Ze względu na fakt, że do obliczeń jako pierwsza wymagana jest wartość δ_n, zmiana wag sieci zaczynana jest od ostatniej warstwy, przekazując tę wartość do warstw poprzednich w celu wyliczenia δ_j – czyli dokonywana jest jej propagacja wsteczna – stąd nazwa algorytmu. Na podstawie powyższych równań aktualizowane są wagi $w_{j,i}$ dla sieci w iteracji k poprzez ich zmianę $\Delta w_{j,i}$, która powoduje zmniejszenie funkcji błędu:

$$w_{j,i}(\mathrm{k}+1) = w_{j,i}(\mathrm{k}) + \Delta w_{j,i}$$

[42] Buduma N., *Fundamentals of Deep Learning,* Sebastopol, O'Reilly Media, 2017, s. 21

Kryterium stopu algorytmu jest osiągnięcie ustalonej z góry liczby iteracji lub ustalonego poziomu błędu. Oczywistą wadą algorytmu jest złożoność obliczeniowa związana z koniecznością obliczania pochodnych cząstkowych oraz wymagania, przedstawione we wcześniejszym podrozdziale, co do funkcji aktywacji.

4. Zbiór danych

4.1 Charakterystyka danych

Zbiór danych został udostępniony przez firmę Loyalty Partner Polska Sp. z o.o., której głównym produktem jest program lojalnościowy PAYBACK. Dane pochodzą z akcji marketingowej obejmującej wysyłkę e-mail, informującą o dostępnych ofertach wyróżnionych sklepów oraz związanych z nimi promocjach. Głównym celem akcji było skłonienie do zakupu w jednym ze sklepów promowanych przez program.

Zmienne występujące w zbiorze opisują uczestników programu w dniu poprzedzającym rozpoczęcie akcji marketingowej jaką było wysyłanie wiadomości e-mail. Każdy wiersz odpowiada jednemu potencjalnemu odbiorcy.

Zbiór danych składa się z 226.272 obserwacji, z czego 1.772 obserwacje zostały oznaczone jako zdarzenia "pozytywne" czyli zaledwie 0,7%. Definicja zdarzenia pozytywnego to otrzymanie wiadomości e-mail, otworzenie jej oraz dokonanie zakupu w ciągu 14 dni w jednym ze sklepów komunikowanych w tej wysyłce. W dalszej części tego podrozdziału przedstawiona zostanie analiza eksploracyjna wybranych zmiennych.

W zbiorze występuje łącznie 96 zmiennych, z czego 5 jest typu tekstowego, natomiast 91 numerycznego. Zmienna *event* to zmienna celu – oznacza wystąpienie badanego zdarzenia, czyli dokonanie zakupu w jednym ze sklepów w 14 dni po otrzymaniu wiadomości e-mail. W załączniku nr 1 przedstawiono listę wszystkich zmiennych, wraz z ich typem oraz opisem. Tutaj ograniczono się do przedstawienia krótkiej charakterystyki kluczowych dla badania zmiennych.

Ogół zmiennych można podzielić na kilka kategorii. Te oznaczone numerami 1 do 10 charakteryzują transakcyjność danego uczestnika – liczbę punktów zebranych w programie (*TRX_aff_L12M_COL_PTS*) czy kwotę wydaną na zakupy w sklepach online w ciągu ostatnich 12 miesięcy (*TRX_aff_L12M_PUR_AMT*). Wszystkie one mają charakter numeryczny. Nie występują braki danych, natomiast kilka kolumn przyjmuje tylko wartości 0. Zmienne te zostały zaprezentowane w Tabela 2, jak również liczba obserwacji, które przyjmują wartość różną od zera. Można dostrzec, że zmienne związane z wydawaniem (tzw. *redempcją)* punktów w sklepach online (*TRX_aff_L12M_RED_CNT, TRX_aff_L12M_RED_PTS, TRX_aff_L3M_RED_CNT,*

TRX_aff_L3M_RED_PTS) dla wszystkich obserwacji są wypełnione zerami, więc zostaną one usunięte ze zbioru danych, jako nie wnoszące żadnej informacji.

Tabela 2 Zmienne 1-10 występujące w zbiorze danych wraz z liczebnością obserwacji różnych od zera

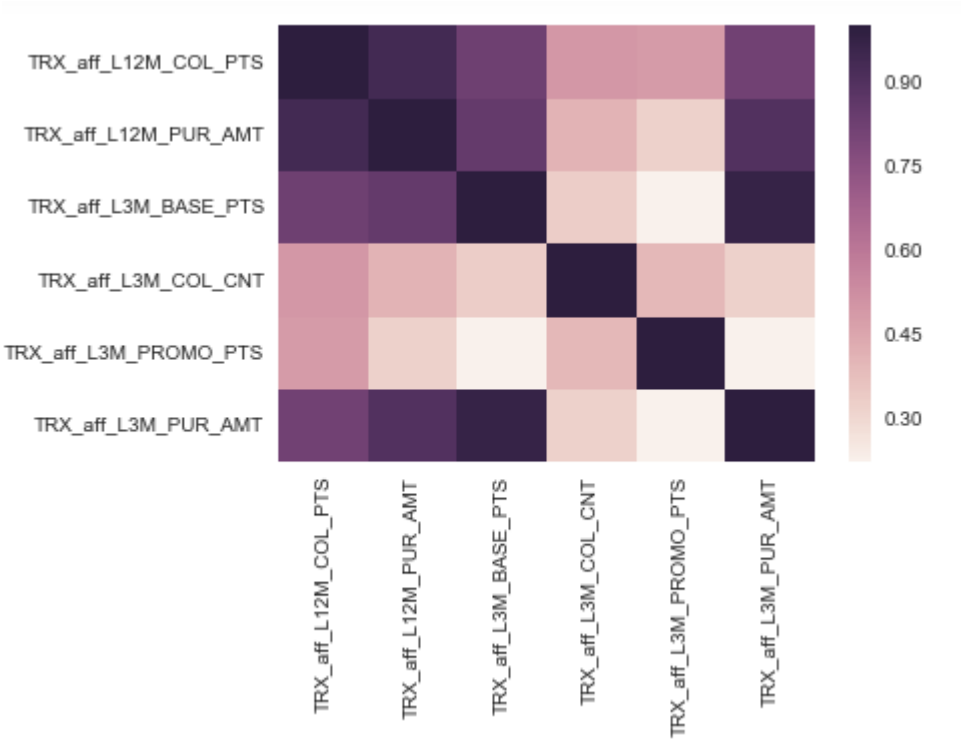
Nazwa zmiennej	Liczebność
TRX_aff_L12M_COL_PTS	4340
TRX_aff_L12M_PUR_AMT	3843
TRX_aff_L12M_RED_CNT	0
TRX_aff_L12M_RED_PTS	0
TRX_aff_L3M_BASE_PTS	3469
TRX_aff_L3M_COL_CNT	3997
TRX_aff_L3M_PROMO_PTS	3010
TRX_aff_L3M_PUR_AMT	3490
TRX_aff_L3M_RED_CNT	0
TRX_aff_L3M_RED_PTS	0

Źródło: opracowanie własne

Powyższe zmienne charakteryzują się również wysoką skośnością, co zostało zaprezentowane w Tabela 3, zawierającej podstawowe statystyki opisowe zmiennych transakcyjnych. Zawarty został również 95. percentyl dla odpowiedniej ilustracji asymetryczności.

Tabela 3 Statystyki opisowe zmiennych transakcyjnych

Nazwa Zmiennej	Średnia	Odchylenie standardowe	Minimum	50%	95%	Maksimum
TRX_aff_L12M_COL_PTS	62	1568	0	0	0	381560
TRX_aff_L12M_PUR_AMT	51	2491	0	0	0	762614
TRX_aff_L3M_BASE_PTS	14	707	0	0	0	213603
TRX_aff_L3M_COL_CNT	0	2	0	0	0	542
TRX_aff_L3M_PROMO_PTS	11	183	0	0	0	23400
TRX_aff_L3M_PUR_AMT	19	1265	0	0	0	394954



Rysunek 10 Wykres korelacji Pearsona pomiędzy zmiennymi opisującymi transakcyjność uczestników

Na Rysunek 10 zostały przedstawione współczynniki korelacji Pearsona pomiędzy zmiennymi opisującymi transakcyjność uczestników programu. Można zauważyć, że zmienna opisująca wartość zebranych punktów podstawowych (*TRX_aff_L3M_BASE_PTS)* jest silnie skorelowana z wydaną kwotą (*TRX_aff_L3M_PUR_AMT),* natomiast słabo skorelowana z wartością zebranych punktów promocyjnych (*TRX_aff_L3M_PROMO_PTS)*

Druga grupa zmiennych, od 7 do 15, opisuje stan wirtualnego konta danego uczestnika programu – stan salda punktowego (*ACCT_TOT_COL_AMT*) czy liczba kart płatniczych przypisanych do konta (*CARDA_PAYM_CNT*). Wszystkie zmienne z tej kategorii są typu numerycznego, nie występują również braki danych. W Tabela 4 zostały przedstawione dla nich podstawowe statystyki opisowe.

Tabela 4 Statystyki opisowe zmiennych charakteryzujących konto uczestnika

Nazwa zmiennej	Średnia	Odchylenie	Min	50%	75%	Max
ACCT_BAL_MOD_CNT	457	572	0	298	645	56182
ACCT_TOT_COL_AMT	31317	55256	0	14968	39132	4455410
ACCT_UNBLOCKED_POINTS	4619	8757	0	1759	5266	831637

TIME_FROM_ENRL	1885	831	31	2105	2696	2755
TIME_FROM_FIRST_DT	1999	799	32	2304	2718	2758
TIME_FROM_LAST_DT	174	375	1	22	137	2737
CARDA_PAYM_CNT	0	1	0	0	0	33
ACCT_PTS_EXPIRED	-330	1605	-99039	0	0	0
ACCT_PTS_TO_EXPIRE	124	681	0	0	0	88779

Kolejną grupę stanowią zmienne demograficzne i geo-lokalizacyjne – są to na przykład wiek uczestnika (*CUST_AGE)*, jego płeć (*CUST_GENDER)* oraz liczba ludności miejscowości zamieszkania (*CUST_GFK_POPULATION)*. W tej kategorii znajdują się zmienne deklaratywne – są to wspomniane już wiek i płeć oraz wykonywany zawód (*CUST_PROFESSION_CD)*. Dekodowanie wartości poszczególnych zmiennych przedstawione zostało w Tabela 5.

Tabela 5 Zmienne deklaratywne nominalne i ich dekodowanie

Nazwa zmiennej	Dekodowane wartości
CUST_PROFESSION_CD	1 - Manager, 2 - Samozatrudniony, 3 - Pracownik biurowy, 4 - Pracownik fizyczny, 5 - Pracuje w domu, 6 - Nie pracuje, 7 - Student, 99 - brak danych
CUST_GENDER	1 - mężczyzna, 2 - kobieta, 0 – nieokreślona

Niestety zmienna określająca zawód uczestnika, jako deklaratywna, posiada znaczną liczbę braków danych (oznaczonych liczbą 99), co zostało zaprezentowane w Tabela 6. W związku z tym faktem zmienna ta również nie została uwzględniona w badaniu.

Tabela 6 Kategorie zmiennej CUST_PROFESSION_CD

Kategoria	Liczebność	Procent zbioru
99	154576	68%
3	33829	15%
2	11529	5%
4	9463	4%

1	7728	3%
7	4417	2%
6	3077	1%
5	1653	1%

Zmienne geo-lokalizacyjne obejmują zmienne binarne informujące o położeniu miejsca aktywności uczestnika w obszarze aktywności danego sklepu oraz odległość od centrum takiego obszaru, co zostało zaprezentowane w Tabela 7.

Tabela 7 Zmienne geolokalizacyjne

Nazwa zmiennej	Średnia	Odchylenie standardowe	Min	50%	75%	Max
CUST_CA_AAA_FLG	0,603451	0,489182	0	1	1	1
CUST_CA_BBB_FLG	0,967822	0,176473	0	1	1	1
CUST_CA_CCC_FLG	0,630719	0,482611	0	1	1	1
CUST_CA_BBB_KM_CNT	5,735212	32,123715	0	1,99	5,14	999
CUST_CA_AAA_KM_CNT	17,17591	37,780421	0	6,34	19,81	999
CUST_CA_CCC_KM_CNT	23,45034	43,725027	0	6,61	34,41	999

Ostatnią kategorię stanowią zmienne charakteryzujące aktywność uczestnika na stronie internetowej programu PAYBACK, w kilku jej sekcjach, w różnych przedziałach czasowych. Stanowią one najliczniejszą grupę – aż 50 zmiennych. W Tabela 8 zostały przedstawione zmienne reprezentujące wszystkie sekcje w przedziale czasowym ostatnich 6 miesięcy.

Tabela 8 Zmienne opisujące aktywność uczestnika na stronie internetowej programu w ciągu ostatnich 6 miesięcy

Nazwa zmiennej	Średnia	Odchylenie standardowe	Min	25%	50%	75%	Max
WEB_ALL_TIME_SPEND_L6M	3286	15182	0	0	298	3111	3149094
WEB_AFF_TIME_SPEND_L6M	711	4070	0	0	0	151	731304
WEB_REW_TIME_SPEND_L6M	1079	4156	0	0	0	799	1302762

WEB_OCC_TIME_SPEND_L6M	463	2756	0	0	2	297	867372
WEB_HIS_TIME_SPEND_L6M	227	2453	0	0	0	60	811922

W tej kategorii nie występują braki danych. Można zauważyć, że i w tej kategorii występuje wysoka skośność a drugi kwartyl stanowią uczestnicy mało aktywni – nie będący ani razu na żadnej z sekcji strony internetowej Programu. Ostatnie dwie zmienne w zbiorze to numer oferty, na którą odpowiedziała dana osoba oraz kategoria tej oferty. Ze względu na to, iż są one przypisane tylko do osób, które odpowiedziały na mailing nie będą one użyte w modelu.

4.2 Przekształcenia zmiennych

Sieci neuronowe to zbiór modeli, wymagający zbioru danych w postaci numerycznej – takie też są wszystkie zmienne w zbiorze. Jak wynika z porównania statystyk opisowych zakres wartości dla wszystkich zmiennych jest bardzo duży – pomiędzy 0 i 1 dla jednych, aż po kilkadziesiąt tysięcy poniżej lub powyżej zera dla innych. Aby ograniczyć wpływ tak wysokich wartości na modele za pomocą modułu *preprocessing* z pakietu *sklearn* przeprowadzona została standaryzacja Z w celu uzyskania rozkładu o średniej 0 i odchyleniu standardowym 1, według poniższego wzoru[43]:

$$Z = \frac{x - \mu}{s},$$

gdzie:

- x – wartość zmiennej
- μ – wartość średnia danej zmiennej
- s – odchylenie standardowe zmiennej

Takie przeskalowanie pomaga zapewnić równe traktowanie danych wejściowych względem wag w procesie uczenia oraz bardziej jednorodny wybór ich początkowego zakresu[44]. W zbiorze nie występują zmienne kategoryczne, więc nie ma potrzeby

[43] https://github.com/scikit-learn/scikit-learn/blob/a24c8b46/sklearn/preprocessing/data.py#L461, dostęp: 5 luty 2018

[44] Hastie T., Tibshirani R., Friedman J., *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*, Springer, New York, 2009, s.400

przeprowadzenia kodowania takich zmiennych, co jest zwyczajową procedurą w przygotowywaniu zbioru dla sieci neuronowych.

Analizowany zbiór składa się z zaledwie 0,7% pozytywnych zdarzeń. Taki rozkład zmiennej celu oznacza, że model klasyfikujący wszystkie zdarzenia jako 0 osiągnie ponad 99% skuteczność. W celu zniwelowania tej niepożądanej właściwości zastosowane zostało zbilansowanie zbioru za pomocą próbkowania losowego[45]. W tym celu wykorzystany został algorytm *RandomUnderSampler* z pakietu *imblearn*[46]. Po tym przekształceniu liczebność zbioru to 5746, gdzie liczba obserwacji o kategorii 1 wynosi 1326, czyli około 30% liczebności nowego zbioru. Natomiast liczebność obserwacji o kategorii 0 to 4420, czyli około 70% zbioru.

[45] Frątczak E., *Zaawansowane metody analiz statystycznych*, Oficyna Wydawnicza Szkoły Głównej Handlowej w Warszawie, 2013, Warszawa, s.597

[46]http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/imblearn.under_sampling.RandomUnderSampler.html, dostęp: 3 styczeń 2018

5. Modelowanie

5.1 Parametry optymalizacji topologii sieci neuronowej

W niniejszej pracy algorytmy genetyczne zostały użyte do optymalizacji topologii sieci neuronowej. Implementacja rozwiązania została stworzona w języku programowania Python. Do stworzenia sieci został wykorzystany moduł *scikit-learn* z implementacją perceptronu wielowarstwowego (*MLPClassifier*)[47]. Implementacja ta pozwala na określenie parametrów sieci neuronowej, z których te najważniejsze stały się przedmiotem optymalizacji. Są to:

- funkcja aktywacji,
- rodzaj współczynnika uczenia sieci,
- wartość startowego współczynnika uczenia,
- liczba warstw ukrytych,
- liczba neuronów w warstwie ukrytej.

Wśród optymalizowanych funkcji aktywacji znalazły się funkcja logistyczna, tangens hiperboliczny oraz ReLU, które zostały omówione w podrozdziale 3.3. Rodzaj współczynnika uczenia określony został jako stały (ang. *constant)* lub zmniejszający się wraz z przebiegiem procesu uczenia (ang. *invscaling*). Trzecim parametrem jest wartość startowego współczynnika uczenia, który może przyjmować wartości od 5e-4 do 5e-1. Liczba warstw ukrytych przyjmuje wartości od 1 do 5, natomiast liczba neuronów w warstwie od 1 do 128. Wartości te zostały przyjęte po części na podstawie ogólnej praktyki[48] mówiącej o całkowitej liczbie neuronów w warstwie ukrytej sieci pomiędzy 5 a 100, a po części w celu sprawdzenia, czy dużo wyższa liczba neuronów może prowadzić do wyższej mocy predykcyjnej. Powyższe wartości parametrów pozwalają na osiągnięcie od 1 do 640 neuronów w warstwie ukrytej.

5.2 Implementacja algorytmu genetycznego

Podstawową kwestią związaną z implementacją algorytmu genetycznego było kodowanie parametrów sieci – czyli takie ich przedstawienie, jako pojedynczego

[47]http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, dostęp: 3 styczeń 2018

[48] Hastie T., Tibshirani R., Friedman J., *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*, Springer, New York, 2009, s.400

chromosomu, aby możliwe było zastosowanie wszystkich operatorów algorytmu. Ze względu na liczbę parametrów sieci oraz ich postać (liczby od 0 do 1001) przyjęte zostało kodowanie za pomocą liczb naturalnych. Za gen została uznana liczba reprezentująca dany parametr sieci, przyjmująca odpowiedni zakres. Przykładowy chromosom wygląda następująco:

```
[ 1,   0, 714,   3,  80 ]
```

Chromosom reprezentowany jest przez listę zawierającą liczby w zakresie zależnym od danego parametru. Pierwszy element to rodzaj funkcji aktywacji – gdzie 1 oznacza funkcję logistyczną, 2 – tangens hiperboliczny, 3 – ReLU. Drugi element listy to rodzaj współczynnika uczenia – 0 reprezentuje stały współczynnik, natomiast 1 – zmniejszający się w trakcie uczenia. Trzeci element to początkowa wartość współczynnika uczenia – tutaj reprezentowana przez liczby od 1 do 1000. Ta wartość zostaje przekształcona przez poniższe równanie w celu otrzymania pożądanych wartości z przedziału 5e-4 do 5e-1:

$$współczynnik\ \ uczenia = \frac{1}{2*A},$$

gdzie A to trzeci element chromosomu.
Czwarty element listy to liczba warstw ukrytych, natomiast piąty – liczba neuronów w każdej warstwie.

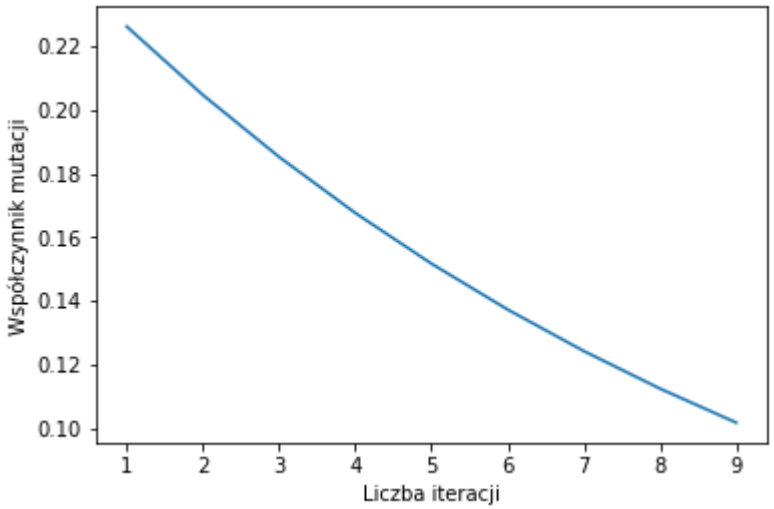
Mutacja została zaimplementowana na poziomie chromosomu, nie na poziomie pojedynczego genu tzn. dla konkretnego chromosomu sprawdzano czy należy dokonać mutacji. Jeśli tak, to został zastosowany operator mutacji na jednym z genów danego chromosomu. Współczynnik mutacji został określony następująco:

$$prawdopodobieństwo\ mutacji = \frac{0{,}25}{\exp\left(\frac{i}{10}\right)},$$

gdzie *i* to numer generacji.
Taka implementacja została zapożyczona z algorytmu wyżarzania symulowanego i stosowanego tam współczynnika temperatury[49], który maleje wraz z liczbą iteracji. Współczynnik mutacji pełni rolę sposobu na wychodzenie z minimum lokalnego, ale może też powodować uzyskiwanie nieoptymalnych rozwiązań, jeśli obecne optymalne rozwiązanie będzie zmieniać się zbyt często.

[49] Rere L.M. R., Fanany M. I., Arymurthy A. M., *Simulated Annealing Algorithm for Deep Learning*, *w: Procedia Computer Science*, nr 72, s. 137-144, 2015

Dzięki takiej definicji prawdopodobieństwo mutacji maleje wraz z liczbą iteracji algorytmu co zostało przedstawione na Rysunek 11:



Rysunek 11 Wykres prawdopodobieństwa mutacji wraz ze wzrostem liczby iteracji algorytmu

Kolejną istotną częścią algorytmu genetycznego jest funkcja przystosowania. W przypadku modelowania zagadnienia związanego z marketingiem bezpośrednim, czy innym opartym o zbiór z niską liczbą odpowiedzi pozytywnych, określenie sukcesu jako klasyfikacja największej liczby obserwacji poprawnie nie miałaby sensu, gdyż model przypisałby wszystkie obserwacje jako zdarzenie negatywne i osiągnął 99,2% dokładność. Dlatego też funkcją przystosowania stała się funkcja zysku (straty) zdefiniowana jako iloczyn wartości danej kategorii zaklasyfikowania i liczby obserwacji zaklasyfikowanych do tej kategorii a reprezentowana przez macierz kosztu z Tabeli 9.

Tabela 9Macierz kosztu dla klasyfikacji odpowiedzi na kampanię marketingową

Wynik	**Klasyfikacja**	**Rzeczywista odpowiedź**	**Zysk**	**Uzasadnienie**
True Positive	Odpowiedź	Odpowiedź	19,9	Zysk minus koszt wysyłki
False Positive	Odpowiedź	Brak odpowiedzi	0,1	Koszt wysyłki
True Negative	Brak odpowiedzi	Brak odpowiedzi	0	Brak kontaktu, brak straconego zysku
False Negative	Brak odpowiedzi	Odpowiedź	-20	Stracony zysk

Wartości przypisane do poszczególnych kategorii nie odzwierciedlają prawdziwych kosztów wysyłki w firmie PAYBACK. Są one umowne i mają na celu reprezentowanie faktu, że zysk z odpowiedzi na wysyłkę jest znacznie większy niż koszt wysłania mailingu. Tak więc funkcja przystosowania będzie mieć następującą postać:

$$Funkcja\ przystosowania = \boldsymbol{Macierz\ kosztu}\ x\ \boldsymbol{Macierz\ trafności}$$

Taka postać pozwala na promowanie topologii sieci, które poprawnie klasyfikują zdarzenia pozytywne oraz optymalizuje zysk z kampanii marketingowej.

W zastosowanym podejściu przeprowadzono 10 iteracji algorytmu genetycznego (czyli 10 generacji), gdzie w każdej iteracji populacja składała się z 10 chromosomów – sieci neuronowych – co daje łącznie uczenie 100 modeli.

5.3 Wyniki

W tym podrozdziale przedstawione zostaną wyniki modelowania odpowiedzi na kampanię marketingową, które jednocześnie będą stanowić weryfikację hipotezy o przydatności ewolucyjnych sieci neuronowych w badanym zjawisku.

5.3.1 Model referencyjny

Standardowym modelem używanym do optymalizacji kampanii marketingowych jest model regresji logistycznej[50], głównie ze względu na swoją prostotę, niewielkie zapotrzebowanie na moc obliczeniową oraz łatwą możliwość interpretacji uzyskanych wyników. Do trenowania i testowania modelu użyto tych samych danych co w przypadku sieci neuronowej, również poddanych standaryzacji oraz próbkowaniu losowemu. Podobnie jak w przypadku sieci zbiór treningowy składał się z 5756 obserwacji, natomiast testowy z 56 568. Podstawową miarą jakości modelu jest wartość funkcji przystosowania, bezpośrednio określająca przychód generowany przez model, uzyskana na podstawie macierzy trafności.

Tabela 10 Macierz trafności dla modelu regresji logistycznej

		Klasyfikacja	
		0	1
Odpowiedź	0	49 501	6 621

[50] Stanisz A., *Modele regresji logistycznej : zastosowania w medycynie, naukach przyrodniczych i społecznych,* Kraków, 2016

	1	109	337

W Tabeli 10 przedstawione zostały wyniki klasyfikacji uzyskane przez model regresji logistycznej. Jedną ze statystyk wykorzystywanych w ocenie jakości modelu jest zliczeniowy współczynnik R^2, dany wzorem:

$$R^2 = \frac{TP + TN}{N},$$

gdzie:

- TP – liczba klasyfikacji prawdziwie pozytywnych,
- TN – liczba klasyfikacji prawdziwie negatywnych,
- N – liczba wszystkich obserwacji.

W przypadku regresji logistycznej współczynnik ten wynosi 0,881. W oparciu o macierz trafności obliczono również czułość klasyfikatora (ang. *sensitivity)*, która jest miarą informującą o stosunku liczby obserwacji poprawnie klasyfikowanych jako pozytywne do liczby naprawdę pozytywnych obserwacji (określana również jako ang. *true positive rate)*:

$$TPR = \frac{337}{109 + 337} = 0{,}7556 = 75{,}56\%$$

Model regresji zatem uzyskał całkiem zadowalający wynik – niemalże 76% obserwacji pozytywnych jest przez niego klasyfikowanych jako pozytywne. Wartość funkcji przystosowania dla tego modelu wyniosła 3853 z maksymalnej możliwej 8875 dla modelu idealnie klasyfikującego co stanowi 43% tej wartości.

5.3.2 Modele sieci neuronowych

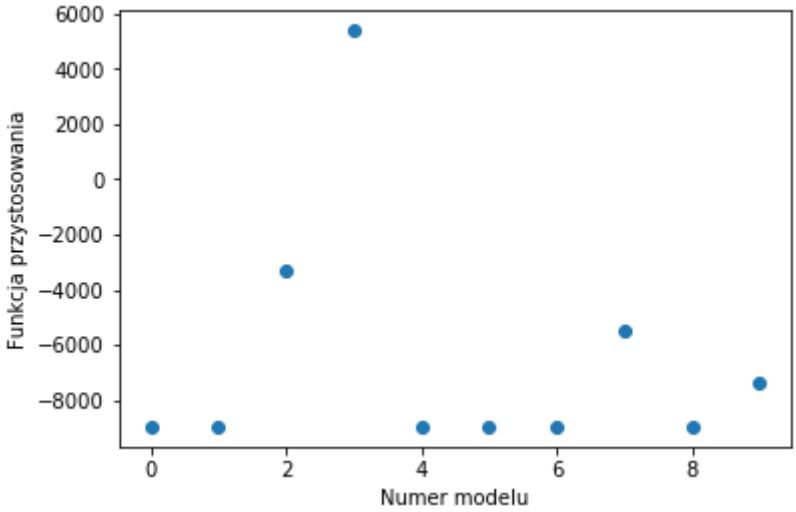
W trakcie 10 iteracji algorytmu genetycznego treningowi oraz ewaluacji poddanych zostało łącznie 100 sieci. Pierwszą populację stanowiły modele o zupełnie losowych topologiach, co przedstawione zostało w Tabela 11.

Tabela 11 Parametry populacji sieci dla pierwszej iteracji algorytmu

	Wartości parametrów				
Numer sieci	1	2	3	4	5
1	1	0	985	4	21
2	1	0	970	3	54
3	3	1	753	3	81
4	3	0	392	2	22
5	1	1	895	3	11
6	1	0	469	5	90
7	1	0	512	3	98

8	3	1	711	3	44
9	1	1	126	4	103
10	3	1	335	4	114

Niestety w pierwszej iteracji do populacji nie został wybrany gen reprezentujący funkcję aktywacji tangens hiperboliczny (parametr 1), jak również liczba warstw ukrytych jest w zakresie od 2 do 5 a nie jak zakładano od 1 do 5, co nieco ogranicza przestrzeń dostępnych rozwiązań. Na Rysunek 12 została przedstawiona wartość funkcji przystosowania dla poszczególnych modeli w tej iteracji. Możemy zaobserwować, że dla 9 spośród 10 modeli wartość funkcji przystosowania jest poniżej 0 a aż 6 z nich przyjmuje wartość poniżej -8000. Jeśli przyjrzymy się wartościom macierzy trafności okazuje się, że modele z ujemną wartością tejże funkcji klasyfikują wszystkie obserwacje jako jedną z kategorii, zupełnie ignorując drugą czyli są równoważne modelom naiwnym.



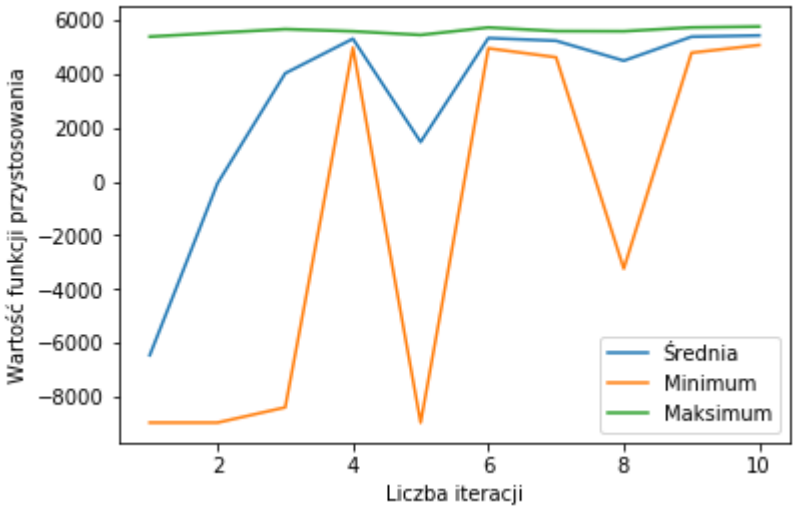
Rysunek 12 Wykres wartości funkcji przystosowania z pierwszej iteracji dla wybranych 10 modeli sieci neuronowych

Tabela 12 Macierz trafności po pierwszej iteracji algorytmu dla wybranych 10 modeli sieci neuronowych

	Klasyfikacja			
Numer sieci	TN	FN	FP	TP
1	56122	0	446	0
2	56122	0	446	0
3	47175	8947	282	164
4	45290	10832	60	386
5	56122	0	446	0
6	56122	0	446	0
7	56122	0	446	0
8	55166	956	357	89
9	56122	0	446	0
10	55752	370	405	41

Jeden z modeli osiągnął wartość funkcji przystosowania ponad 5 tysięcy – co jest już całkiem dobrym wynikiem. Można się spodziewać, że w kolejnych iteracjach jego topologia będzie najbardziej dominującą w całej populacji.

Na Rysunek 13 przedstawiona została wartości funkcji przystosowania dla wszystkich iteracji algorytmu, w podziale na średnią, minimum oraz maksimum. Wartość średnia, podobnie jak minimalna, dla pierwszej populacji jest najniższa zgodnie z tym czego oczekiwalibyśmy po losowym doborze parametrów. Już w drugiej iteracji średnia osiąga wartość bliską zeru, a w trzeciej i kolejnych ponad 4 tysiące, czyli lepiej niż model referencyjny. W badanym przypadku już od czwartej iteracji średnia nie ulega znacznemu polepszeniu. Jeśli chodzi o minimalną wartość funkcji przystosowania, widać że mutacja w pokoleniach 5 i 8 sprawiła, że jest ona niezwykle niska, natomiast w pozostałych nie różni się wiele od średniej.



Rysunek 13 Wykres wartości funkcji przystosowania w kolejnych iteracjach

Wartość maksymalna funkcji przystosowania wydaje się być stałą wartością, co świadczy o tym, że losowy sposób doboru parametrów może czasem zaowocować znalezieniem rozwiązania bliskiego optymalnemu lub przynajmniej bardzo dobrego.

Ostatecznie najlepszym modelem okazała się sieć neuronowa z iteracji numer 10, której wartość funkcji przystosowania wynosi 5768 a jej czułość to 0,886, czyli poprawnie klasyfikuje 88,6% obserwacji pozytywnych. W Tabela 13 przedstawiona została macierz trafności dla tego modelu. Zliczeniowy współczynnik R^2 wynosi 0,81.

Tabela 13 Macierz trafności dla najlepszego modelu sieci neuronowej.

Klasyfikacja	
0	1

Odpowiedź	0	45 445	10 677
	1	41	395

Topologia tej sieci składa się z funkcji ReLU, stałego współczynnika uczenia z wartością początkową 392, trzema warstwami ukrytymi, w których znajdują się po 22 neurony. Jest ona niemalże identyczna jak najlepsza sieć w pierwszej generacji – różni się od niej jedynie większą liczbą warstw ukrytych.

5.4. Wnioski

W celu weryfikacji hipotezy o przewadze sieci neuronowych nad regresją logistyczną w poniższym podrozdziale dokonano porównania obu modeli. Ze względu na zbiór danych charakteryzujący się silnym niezbilansowaniem nie zostały użyte tradycyjne metryki, takie jak błąd średniokwadratowy czy klasyczny współczynnik determinacji R^2.

Wartości pierwszej z użytych miar, zliczeniowego współczynnika R^2, dla porównywanych modeli wynoszą 0,88 i 0,81, odpowiednio dla regresji logistycznej oraz sieci neuronowej. Oznacza to iż regresja klasyfikuje poprawnie o 7 punktów procentowych więcej obserwacji prawdziwie pozytywnych oraz prawdziwie negatywnych niż sieć neuronowa.

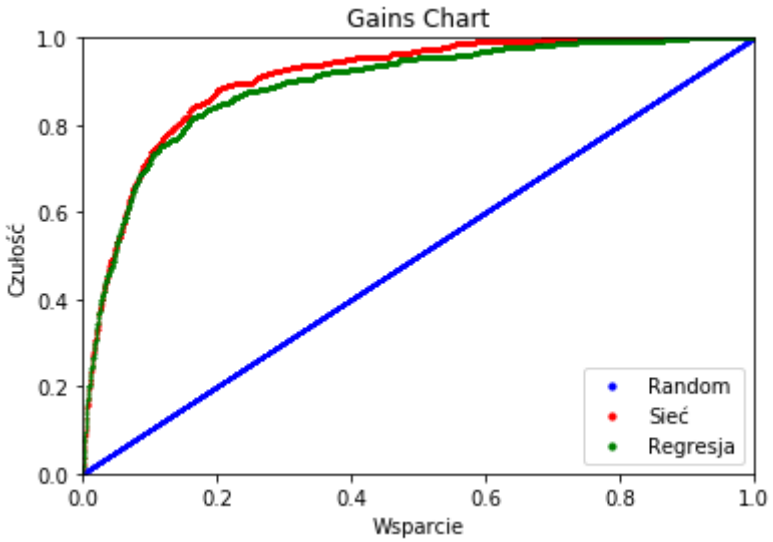
Drugą ze statystyk, która definiuje przydatność modelu w omawianym zagadnieniu jest wartość funkcji zysku (będąca również funkcją przystosowania w algorytmie genetycznym), która przypisuje wartości zysku dla poszczególnych rodzajów klasyfikacji na podstawie macierzy trafności, co przedstawione zostało w Tabela 14.

Tabela 14 Porównanie wartości funkcji zysku badanych modeli

Model	Wartość funkcji zysku	Procent wartości maksymalnej
Regresja logistyczna	3853	43,4%
Sieć neuronowa	5768	65,0%
Idealny	8875	100,0%

Maksymalna wartość jaką może uzyskać model idealnie klasyfikujący to 8875 – co wynika z zysku 19,9 w przypadku klasyfikacji prawdziwie pozytywnej. Regresja logistyczna uzyskała wynik 3853 co stanowi 43,4% tej wartości. Najlepszy z przetestowanych modeli sieci neuronowych uzyskał wynik 5768 czyli 65% maksymalnej

wartości. Oznacza to, że zastosowanie sieci do modelowania zjawiska marketingu bezpośredniego przynosi o 49,7% większy zysk niż zastosowanie regresji logistycznej, co w kontekście biznesowym oznacza znaczne zwiększenie konwersji z tego medium.



Rysunek 14 Krzywa zysku modeli: losowego, regresji logistycznej oraz sieci neuronowej

Dodatkowo na Rysunek 14 został przedstawiony wykres zysku dla omawianych modeli w porównaniu z modelem losowym. Można dostrzec wyraźną przewagę sieci neuronowej już od drugiego decyla, aż do siódmego czyli istnieje istotna podstawa do użycia tegoż modelu do selekcji odbiorców oferty marketingu bezpośredniego.

Zakończenie

Zastosowanie metod statystycznych w problemie selekcji odbiorców wiadomości marketingowych przynosi przedsiębiorstwom zdecydowany zysk, już przy wykorzystaniu prostych, popularnych modeli, co jest szczególnie widoczne przy porównaniu ich do metody wysyłania komunikacji do wszystkich potencjalnych zainteresowanych.

Hipotezą postawioną w poniższej pracy była celowość zastosowania modelu genetycznych sieci neuronowych do powyższego zagadnienia, która została zweryfikowana przy pomocy badania empirycznego. Wprawdzie metoda ta nie okazała się wyjątkowo skuteczna jeśli chodzi o samą poprawność klasyfikacji, co zostało uwidocznione przez nieco niższy współczynnik zliczeniowego R^2 niż w przypadku modelu referencyjnego, jednakże wykazała się dużo większą precyzją w klasyfikacji obserwacji prawdziwie pozytywnych – czyli wskazywania użytkowników z największym prawdopodobieństwem na skorzystanie z proponowanej oferty. W efekcie model sieci neuronowej okazał się dużo bardziej przydatnym narzędziem do maksymalizacji zysku ze środka promocji jakim jest marketing bezpośredni – oferuje on prawie 50% poprawę rentowności w porównaniu z regresją logistyczną.

Ograniczenia, z którymi trzeba się liczyć podczas implementacji omawianej metody obejmują między innymi złożoność obliczeniową algorytmu uczenia sieci neuronowej, niezbilansowanie zbioru treningowego oraz wartości parametrów algorytmu genetycznego (współczynnik mutacji czy liczba populacji). Szybkość uczenia algorytmu ma mniej znaczący wpływ na implementację ze względu na stale rosnące możliwości obliczeniowe współczesnych komputerów oraz możliwości obliczeń w środowiskach rozproszonych. Na ostateczny wynik może poważnie wpłynąć postać zbioru uczącego oraz sposoby jego przekształceń – takie jak zastosowanie innej metody próbkowania w celu zbilansowania liczby obserwacji, czy zastosowanie przekształceń zmiennych.

Algorytm genetyczny posiada kilka różnych parametrów, które w badaniu empirycznym zostały ustalone na podstawie najlepszych praktyk, natomiast definitywnie możliwe jest ich modyfikowanie i weryfikacja. Są to przede wszystkim liczba iteracji algorytmu oraz liczba chromosomów (sieci neuronowych) w danej populacji. Również modyfikacji można poddać współczynnik mutacji, który nie miał aż tak dużego wpływu na obecność nowych cech jak wstępnie zakładano.

Podsumowując, genetyczne sieci neuronowe posiadają znaczny potencjał w optymalizacji marketingu bezpośredniego, jednak wymagają pewnych nakładów pracy w celu dopasowania parametrów do omawianego zagadnienia.

Bibliografia

1. Blattberg R. C., Kim Byung-Do, Neslin S.A, *Database Marketing*, Springer, 2008
2. Błażewicz G., *Rewolucja z marketing automation. Jak wykorzystać potencjał Big Data*, PWN, Warszawa 2016, str. 72.
3. Boire R., *Data Mining for Managers*, Palgrave Macmillan, 2014
4. Buduma N., *Fundamentals of Deep Learning*, Sebastopol, O'Reilly Media, 2017
5. De Jong K., *An analysis of the behavior of a class of genetic adaptive systems, praca doktorska*, University of Michigan, Ann Arbor, 1975
6. Deb K., *Multi-Objective Optimization using Evolutionary Algorithms*, Wiley, 2001
7. Dobski P., Szuman-Dobska M., *Marketing bezpośredni*. Wydawnictwo Prawno-Ekonomiczne, Warszawa 1999
8. Dunis CH., *Computation Intelligence Techniques for Trading and Investment*, New York, Routledge, 2014
9. Figielska E., *Algorytmy ewolucyjne i ich zastosowanie*, w: http://zeszyty-naukowe.wwsi.edu.pl/zeszyty/zeszyt1/Algorytmy_Ewolucyjne_I_Ich_Zastosowania.pdf, dostęp: listopad 2017
10. Frątczak E., *Zaawansowane metody analiz statystycznych*, Oficyna Wydawnicza Szkoły Głównej Handlowej w Warszawie, Warszawa 2013
11. Goldberg D.E., *Algorytmy genetyczne i ich zastosowania***,** Warszawa, WNT, 1995
12. Goldberg D., Richardson J., *Genetic algorithms with sharing for multi-modal function optimization*, w: *Genetic Algorithms and Their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms*, Greffenstette J. (red.), Lawrence Erlbaum Associates, Hillsdale, 1987
13. Gwiazda T. D., *Algorytmy ewolucyjne w rozwiązywaniu nieliniowych problemów decyzyjnych*, Wydawnictwa Naukowe Wydziału Zarządzania Uniwersytetu Warszawskiego, 2002
14. Hastie T., Tibshirani R., Friedman J., *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*, Springer, New York 2009
15. Haykin S., *Neural Networks and Learning Machines*, Pearson Prentice Hall, Upper Saddle River, 2009

16. Holland J. H., *Adaptation in Natural and artificial systems*, MIT Press, Cambridge 1975
17. Janikow C.Z., Michalewicz Z., *An experimental comparison of binary and floating point representations in genetic algorithms,* w: *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, 1991
18. Larose D.T., *Metody i modele eksploracji danych*, Wiley, 2006
19. Larose D. T., Larose C., *Data Mining and Predictive Analytics,* Wiley, Hoboken 2015
20. McIlwrath D., Marmanis H., Babenko D., *Inteligentna Sieć, Algorytmy przyszłości,* Helion, Gliwice 2017
21. Mitchell M., *An Introduction to Genetic Algorithm,* MIT Press, Cambridge 1996
22. Orr M. J. L., *Introduction to Radial Basis Function Networks,* na: https://www.cc.gatech.edu/~isbell/tutorials/rbf-intro.pdf, dostęp: styczeń 2018
23. Potharst R., Kaymak U., Pijls W.H.L.M., *Neural Networks for Target Selection in Direct Marketing,* w: *ERIM Report Series Research in Management*, na: http://hdl.handle.net/1765/83, data dostępu: styczeń 2018
24. Rere L.M. R., Fanany M. I., Arymurthy A. M., *Simulated Annealing Algorithm for Deep Learning, w: Procedia Computer Science*, nr 72, s. 137-144, 2015
25. Rumelhart D .E., Hinton G. E., Williams R. J., *Learning Representations by BackPropagating Errors*, w: *Nature* nr 323, 1986
26. Stanisz A., *Modele regresji logistycznej : zastosowania w medycynie, naukach przyrodniczych i społecznych*, StatSoft, Kraków 2016
27. Szymoniuk B., *Komunikacja marketingowa. Instrumenty i metody*., PWE, Warszawa 2006
28. Tadeusiewicz R., *O celowości zastosowania sieci neuronowych w problemach związanych z elektrotechniką*, w: *Przegląd Elektrotechniczny*, R. 85 NR 2/2009
29. Thomas B., Housden M., *Direct and Digital Marketing in Practice*, Bloomsbury Publishing Plc, 2017
30. Trojanowski M., *Marketing bezpośredni. Koncepcja – zarządzanie – instrumenty*. PWE, Warszawa 2010
31. Trzaskalik T., *Algorytmy genetyczne, ewolucyjne i metaheurystyki*. Katowice, Wydawnictwo Akademii Ekonomicznej w Katowicach, 2005, s.13-14

32. Winkowska-Nowak K., *Modelowanie matematyczne i symulacje komputerowe w naukach społecznych,* Warszawa, Wydawnictwo SWPS Academica, 2007
33. Yang M., Liu Y., Coid J., *Applying Neural Networks and other statistical models to the classification of serious offenders and the prediction of recidivism,* w: *Ministry of Justice Research Series,* nr 6/10, 2010
34. Ye N., *Data Mining – Theories, Algorithms and Examples,* New York, CRC Press, 2014, s. 63

Źródła Internetowe

1. https://sjinsights.net/2014/09/29/new-research-sheds-light-on-daily-ad-exposures/, dostęp: luty 2018
2. https://stat.gov.pl/obszary-tematyczne/nauka-i-technika-spoleczenstwo-informacyjne/spoleczenstwo-informacyjne/spoleczenstwo-informacyjne-w-polsce-w-2016-roku,2,6.html, dostęp: listopad 2017
3. http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad10/w10.htm, dostęp: luty 2018
4. http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, dostęp: styczeń 2018
5. https://github.com/scikit-learn/scikit-learn/blob/a24c8b46/sklearn/preprocessing/data.py#L461, dostęp: luty 2018
6. http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/imblearn.under_sampling.RandomUnderSampler.html, dostęp: styczeń 2018
7. http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, dostęp: styczeń 2018
8. https://marketinginsidergroup.com/content-marketing/artificial-neural-networks-every-marketer-know/, dostęp: sierpień 2018

Spis rysunków

Spis tabel

Załącznik 1 – lista i opis zmiennych w zbiorze

Lp.	Zmienna	Typ	Opis
1	TRX_aff_L12M_COL_PTS	Num	Liczba wszystkich punktów zebrana w ciągu ostatnich 12 miesięcy w Sklepach Online
2	TRX_aff_L12M_PUR_AMT	Num	Kwota wydana na transakcje zakupowe w ciągu ostatnich 12 miesięcy w Sklepach Online
3	TRX_aff_L12M_RED_CNT	Num	Liczba transakcji z płatnością punktami dokonana w ciągu ostatnich 12 miesięcy w Sklepach Online
4	TRX_aff_L12M_RED_PTS	Num	Liczba punktów wykorzystanych w zakupach w ciągu ostatnich 12 miesiecy w Sklepach Online
5	TRX_aff_L3M_BASE_PTS	Num	Liczba punktów podstawowych zebrana w ciągu ostatnich 3 miesiecy w Sklepach Online
6	TRX_aff_L3M_COL_CNT	Num	Liczba transakcji kolekcyjnych dokonanych w ciagu ostatnich 3 miesiecy w Sklepach Online
7	TRX_aff_L3M_PROMO_PTS	Num	Liczba punktów promocyjnych zebrana w ciągu ostatnich 3 miesięcy w Sklepach Online
8	TRX_aff_L3M_PUR_AMT	Num	Kwota wydana podczas transakcji zakupowych w ciagu ostatnich 3 miesiecy w Sklepach Online
9	TRX_aff_L3M_RED_CNT	Num	Liczba transakcji redempcyjnych dokonana w ciągu ostatnich 3 miesięcy w Sklepach Online
10	TRX_aff_L3M_RED_PTS	Num	Liczba punktów redempcyjnych wydanych w ciągu ostatnich 3 miesiecy w Sklepach Online
11	ACCT_BAL_MOD_CNT	Num	Liczba modyfikacji salda konta
12	ACCT_TOT_COL_AMT	Num	Liczba zebranych punktów (ever) w programie
13	ACCT_UNBLOCKED_POINTS	Num	Liczba niezablokowanych punktów na koncie
14	TIME_FROM_ENRL	Num	Czas od zarejestrowania konta
15	TIME_FROM_FIRST_DT	Num	Czas od pierwszej akcji w programie
16	TIME_FROM_LAST_DT	Num	Czas od ostatniej akcji w programie
17	CARDA_PAYM_CNT	Num	Liczba kart płatniczych przypisanych do konta
18	ACCT_PTS_EXPIRED	Num	Liczba wygaszonych punktów
19	ACCT_PTS_TO_EXPIRE	Num	Liczba punktów do wygaśnięcia
20	CUST_GFK_POPULATION	Num	Liczba mieszkańców w miejscu zamieszkania uczestnika

21	CUST_GFK_HOUSEHOLDS	Num	Liczba gospodarstw domowych w miejscu zamieszkania uczestnika
22	CUST_HH_SIZE	Num	Wielkość gospodarstwa domowego uczestnika
23	CUST_PROFESSION_CD	Num	Zawód uczestnika
24	CUST_AGE	Num	Wiek uczestnika
25	CUST_GENDER	Num	Płeć uczestnika
26	CUST_CA_AAA_FLG	Char	Czy uczestnik mieszka w pobliżu sieci partnera AAA
27	CUST_CA_BBB_FLG	Char	Czy uczestnik mieszka w pobliżu sieci partnera BBB
28	CUST_CA_CCC_FLG	Char	Czy uczestnik mieszka w pobliżu sieci partnera CCC
29	CUST_CA_AAA_KM_CNT	Num	Odległość (km) od najbliższego punktu sprzedaży partnera AAA
30	CUST_CA_BBB_KM_CNT	Num	Odległość (km) od najbliższego punktu sprzedaży partnera BBB
31	CUST_CA_CCC_KM_CNT	Num	Odległość (km) od najbliższego punktu sprzedaży partnera CCC
32	WEB_ALL_UNQ_SES_CNT_L2W	Num	Liczba unikalnych sesji w ciagu ostatnich 2 tygodni na stronie PAYBACK
33	WEB_ALL_TIME_SPEND_L2W	Num	Liczba minut spędzona w ostatnich 2 tygodniach na stronie PAYBACK
34	WEB_ALL_UNQ_SES_CNT_L4W	Num	Liczba unikalnych sesji w ciagu ostatnich 4 tygodni na stronie PAYBACK
35	WEB_ALL_TIME_SPEND_L4W	Num	Liczba minut spędzona w ostatnich 4 tygodniach na stronie PAYBACK
36	WEB_AFF_UNQ_SES_CNT_L2W	Num	Liczba unikalnych sesji w ciagu ostatnich 2 tygodni na stronie Sklepów Online
37	WEB_AFF_TIME_SPEND_L2W	Num	Liczba minut spędzona w ostatnich 2 tygodniach na stronie Sklepów Online

38	WEB_AFF_UNQ_SES_CNT_L4W	Num	Liczba unikalnych sesji w ciagu ostatnich 4 tygodni na stronie Sklepów Online
39	WEB_AFF_TIME_SPEND_L4W	Num	Liczba minut spędzona w ostatnich 4 tygodniach na stronie Sklepów Online
40	WEB_HIS_UNQ_SES_CNT_L2W	Num	Liczba unikalnych sesji w ciagu ostatnich 2 tygodni na stronie Historii Transakcji
41	WEB_HIS_TIME_SPEND_L2W	Num	Liczba minut spędzona w ostatnich 2 tygodniach na stronie Historii Transakcji
42	WEB_HIS_UNQ_SES_CNT_L4W	Num	Liczba unikalnych sesji w ciagu ostatnich 4 tygodni na stronie Historii Transakcji
43	WEB_HIS_TIME_SPEND_L4W	Num	Liczba minut spędzona w ostatnich 4 tygodniach na stronie Historii Transakcji
44	WEB_OCC_UNQ_SES_CNT_L2W	Num	Liczba unikalnych sesji w ciagu ostatnich 2 tygodni na stronie eKupony
45	WEB_OCC_TIME_SPEND_L2W	Num	Liczba minut spędzona w ostatnich 2 tygodniach na stronie eKupony
46	WEB_OCC_UNQ_SES_CNT_L4W	Num	Liczba unikalnych sesji w ciagu ostatnich 4 tygodni na stronie eKupony
47	WEB_OCC_TIME_SPEND_L4W	Num	Liczba minut spędzona w ostatnich 4 tygodniach na stronie eKupony
48	WEB_REW_UNQ_SES_CNT_L2W	Num	Liczba unikalnych sesji w ciagu ostatnich 2 tygodni na stronie Sklepu z Nagrodami
49	WEB_REW_TIME_SPEND_L2W	Num	Liczba minut spędzona w ostatnich 2 tygodniach na stronie Sklepu z Nagrodami
50	WEB_REW_UNQ_SES_CNT_L4W	Num	Liczba unikalnych sesji w ciagu ostatnich 4 tygodni na stronie Sklepu z Nagrodami
51	WEB_REW_TIME_SPEND_L4W	Num	Liczba minut spędzona w ostatnich 4 tygodniach na stronie Sklepu z Nagrodami
52	WEB_ALL_UNQ_SES_CNT_L3M	Num	Liczba unikalnych sesji w ciagu ostatnich 3 miesięcy na stronie PAYBACK
53	WEB_ALL_TIME_SPEND_L3M	Num	Liczba minut spędzona w ostatnich 3 miesiącach na stronie PAYBACK
54	WEB_ALL_UNQ_SES_CNT_L6M	Num	Liczba unikalnych sesji w ciagu ostatnich 6 miesięcy na stronie PAYBACK
55	WEB_ALL_TIME_SPEND_L6M	Num	Liczba minut spędzona w ostatnich 6 miesiącach na stronie PAYBACK

56	WEB_ALL_UNQ_SES_CNT_L12M	Num	Liczba unikalnych sesji w ciagu ostatnich 12 miesięcy na stronie PAYBACK
57	WEB_ALL_TIME_SPEND_L12M	Num	Liczba minut spędzona w ostatnich 12 miesiącach na stronie PAYBACK
58	WEB_AFF_UNQ_SES_CNT_L3M	Num	Liczba unikalnych sesji w ciagu ostatnich 3 miesięcy na stronie Sklepów Online
59	WEB_AFF_TIME_SPEND_L3M	Num	Liczba minut spędzona w ostatnich 3 miesiącach na stronie Sklepów Online
60	WEB_AFF_UNQ_SES_CNT_L6M	Num	Liczba unikalnych sesji w ciagu ostatnich 6 miesięcy na stronie Sklepów Online
61	WEB_AFF_TIME_SPEND_L6M	Num	Liczba minut spędzona w ostatnich 6 miesiącach na stronie Sklepów Online
62	WEB_AFF_UNQ_SES_CNT_L12M	Num	Liczba unikalnych sesji w ciagu ostatnich 12 miesięcy na stronie Sklepów Online
63	WEB_AFF_TIME_SPEND_L12M	Num	Liczba minut spędzona w ostatnich 12 miesiącach na stronie Sklepów Online
64	WEB_HIS_UNQ_SES_CNT_L3M	Num	Liczba unikalnych sesji w ciagu ostatnich 3 miesięcy na stronie Historii Transakcji
65	WEB_HIS_TIME_SPEND_L3M	Num	Liczba minut spędzona w ostatnich 3 miesiącach na stronie Historii Transakcji
66	WEB_HIS_UNQ_SES_CNT_L6M	Num	Liczba unikalnych sesji w ciagu ostatnich 6 miesięcy na stronie Historii Transakcji
67	WEB_HIS_TIME_SPEND_L6M	Num	Liczba minut spędzona w ostatnich 6 miesiącach na stronie Historii Transakcji
68	WEB_HIS_UNQ_SES_CNT_L12M	Num	Liczba unikalnych sesji w ciagu ostatnich 12 miesięcy na stronie Historii Transakcji
69	WEB_HIS_TIME_SPEND_L12M	Num	Liczba minut spędzona w ostatnich 12 miesiącach na stronie Historii Transakcji
70	WEB_OCC_UNQ_SES_CNT_L3M	Num	Liczba unikalnych sesji w ciagu ostatnich 3 miesięcy na stronie eKupony
71	WEB_OCC_TIME_SPEND_L3M	Num	Liczba minut spędzona w ostatnich 3 miesiącach na stronie eKupony
72	WEB_OCC_UNQ_SES_CNT_L6M	Num	Liczba unikalnych sesji w ciagu ostatnich 6 miesięcy na stronie eKupony
73	WEB_OCC_TIME_SPEND_L6M	Num	Liczba minut spędzona w ostatnich 6 miesiącach na stronie eKupony

74	WEB_OCC_UNQ_SES_CNT_L12M	Num	Liczba unikalnych sesji w ciagu ostatnich 12 miesięcy na stronie eKupony
75	WEB_OCC_TIME_SPEND_L12M	Num	Liczba minut spędzona w ostatnich 12 miesiącach na stronie eKupony
76	WEB_REW_UNQ_SES_CNT_L3M	Num	Liczba unikalnych sesji w ciagu ostatnich 3 miesięcy na stronie Sklepu z Nagrodami
77	WEB_REW_TIME_SPEND_L3M	Num	Liczba minut spędzona w ostatnich 3 miesiącach na stronie Sklepu z Nagrodami
78	WEB_REW_UNQ_SES_CNT_L6M	Num	Liczba unikalnych sesji w ciagu ostatnich 6 miesięcy na stronie Sklepu z Nagrodami
79	WEB_REW_TIME_SPEND_L6M	Num	Liczba minut spędzona w ostatnich 6 miesiącach na stronie Sklepu z Nagrodami
80	WEB_REW_UNQ_SES_CNT_L12M	Num	Liczba unikalnych sesji w ciagu ostatnich 12 miesięcy na stronie Sklepu z Nagrodami
81	WEB_REW_TIME_SPEND_L12M	Num	Liczba minut spędzona w ostatnich 12 miesiącach na stronie Sklepu z Nagrodami
82	event	Num	Zdarzenie
83	Id	Num	Identyfikator obserwacji
84	Oferta	Char	Numer oferty z danej wysyłki
85	Kategoria	Char	Kategoria sklepu

Załącznik 2 – kod implementacji algorytmu genetycznego i sieci neuronowej

```
1.  import numpy as np
2.  import pandas as pd
3.  from math import exp
4.  import matplotlib.pyplot as plt
5.  from sklearn.model_selection import train_test_split
6.  from sklearn.neural_network import MLPClassifier
7.  from sklearn.preprocessing import StandardScaler
8.  from sklearn.preprocessing import MinMaxScaler
9.  from sklearn.metrics import confusion_matrix, classification_report
10.
11.
12. def parametryTopologii():
13.     '''''
14.     Funkcja zwracająca optymalizowane parametry topologii sieci
15.
16.     Zwraca:
17.     parametry (dict) - słownik z listą parametrów
18.
19.     '''
20.
```

```
21.     fa = [1, 2, 3] #funkcja aktywacji
22.     lt = [0, 1] # współczynnik uczenia
23.     lr = list(range(1, 1001, 1)) # startowy współczynnik uczenia
24.     hl = list(range(1, 6)) # liczba warstw ukrytych
25.     nn = list(range(1, 129)) # liczba neuronów w warstwie
26.     parametry = {0:fa, 1:lt, 2:lr, 3:hl, 4:nn}
27. 
28.     return parametry
29. 
30. 
31. def startowaPopulacja(liczebnosc_pop):
32.     '''''
33.     Funkcja tworząca startową populacje
34. 
35.     Parametry:
36.     liczebnosc_pop (int) - liczba określająca liczebność populacji czyli liczb
    ę chromosomów
37. 
38.     Zwraca:
39.     populacja (dict) - słownik z listami chromosomow
40. 
41.     '''
42.     populacja = {}
43.     #liczebnosc_pop = 10
44. 
45.     for j in range(liczebnosc_pop):
46.         ch2 = np.ones(5, dtype=int)
47.         for i in range(5):
48.             ch2[i] = np.random.choice(parametryTopologii()[i])
49.         populacja[j] = ch2
50.     return populacja
51. 
52. 
53. def mutacja(chromosom):
54.     '''''
55. 
56.     Funkcja mutacji - zmieniajaca losowo jeden z parametrow chromosomu
57.     Wybiera losowy gen i zastepuje jego wartość dowolną wartość dozwoloną dla
    tego genu
58. 
59.     Parametry:
60.     - chromosom (list) - wektor reprezentujacy chromosom
61. 
62.     Zwraca:
63.     - m_chromosom (list) - lista reprezentująca zmutowany chromosom
64. 
65.     '''
66. 
67.     # parametr który ma zostać zmutowany
68.     p = np.random.randint(0, len(chromosom))
69.     m_chromosom = chromosom.copy()
70.     pT = parametryTopologii()[p]
71.     pT.pop(pT.index(m_chromosom[p]))
72.     m_chromosom[p] = np.random.choice(pT)
73. 
74.     return m_chromosom
75. 
76. 
77. def selekcja(populacja, przystosowanie):
78.     '''''
79.     Funkcja selekcji - wybierająca chromosomy do krzyżowania
80. 
81.     Parametry:
82.     - populacja (dict) - słownik z aktualną populacją
83.     - przystosowanie (dict) - słownik z wartościami funkcji przystosowania dla
     każdego chromosomu w populacji
```

```
84.
85.     Zwraca:
86.     - nowa_populacja (dict) - słownik z nową populacją
87.
88.     '''
89.     fitness = przystosowanie
90.     mmsc = MinMaxScaler(feature_range=(0,100)).fit_transform(fitness.reshape(-
    1, 1))
91.     s = sum(mmsc)
92.     mmsc = mmsc/s*100
93.     f_total = []
94.     for m, i in enumerate(mmsc):
95.         for j in range(int(i)):
96.             f_total.append(m)
97.
98.     parents_pool = np.copy(f_total)
99.
100.            for j in range(len(populacja)):
101.
102.                # wybieranie losowych rodziców metodą ruletki
103.                father = np.random.choice(parents_pool)
104.
105.                # drugi rodzic nie może być taki sam jak pierwszy
106.                mother = parents_pool[parents_pool!=father]
107.
108.                # jeśli pozostał tylko jeden rodzic w populacji to muszę go wyb
    rać
109.                if len(mother) == 0:
110.                    mother = father
111.                else:
112.                    mother = np.random.choice(mother)
113.
114.                father = populacja[father]
115.                mother = populacja[mother]
116.                c = np.random.randint(1, len(father))
117.                child = np.concatenate([father[:c], mother[c:]])
118.                # mutacja
119.                mutationPr = 0.25
120.                wsp_mutacji = mutationPr/np.exp(i/10)
121.                m = np.random.random()
122.                if m < wsp_mutacji:
123.                    child = mutacja(child)
124.                nowa_populacja[j] = child
125.
126.            return nowa_populacja
127.
128.        def siec(chromosom):
129.            '''''
130.
131.            Funkcja tworząca nową sieć na podstawie otrzymanego chromosomu
132.            Parametry:
133.             - chromosom (list) - lista parametrów topologii
134.
135.            Zwraca:
136.             - macierz (ndarray) - macierz błędu
137.
138.            '''
139.            f_aktywacji = {1:'logistic', 2:'tanh', 3:'relu'}
140.            wsp_uczenia = {0:'constant', 1:'invscaling'}
141.            warstwy = tuple(np.ones(chromosom[3], dtype=int)*chromosom[4])
142.            mlp = MLPClassifier(hidden_layer_sizes=warstwy, activation=f_aktywa
    cji[chromosom[0]],\
143.                                learning_rate=wsp_uczenia[chromosom[1]], learni
    ng_rate_init = 1/(2*chromosom[2]))
144.
145.            mlp.fit(X_train_sc, Y_train)
```

```
146.            prognoza = mlp.predict(X_test_sc)
147.            macierz = confusion_matrix(Y_test, prognoza)
148.
149.            return macierz
150.
151.
152.        # macierz kosztu = [TP, FP, TN, FN]
153.        def funkcjaZysku(k):
154.            '''''
155.            Funkcja obliczajaca wartosci zysku danej sieci na podstawie macierz
    y kontyngencji składającej się z:
156.            TruePositive, FalsePositive, TrueNegative, FalseNegative
157.
158.            Parametry:
159.            k (list) - lista zawierajaca liczbe wartosci każdej z czterech możl
    iwych kombinacji klasyfikacji
160.
161.            Zwraca:
162.            koszt (float) - liczba rzeczywista określająca zysk związany z dany
    m efektem klasyfikacji
163.
164.            '''
165.
166.            cost_matrix = np.array([19.9, -0.1, 0, -20.1])
167.            zysk = np.round(np.dot(k, cost_matrix), 5)
168.
169.            return zysk
170.
171.
172.        # Główna część wykonująca
173.
174.        # liczba iteracji algorytmu
175.        epochs = 10
176.
177.        # liczba chromosomów w populacji - liczba różnych topologii
178.        liczebnosc_pop = 10
179.
180.        # wartości funkcji przystosowania dla poszczególnych iteracji
181.        f_fit = {}
182.
183.        average_fitness = np.zeros(liczebnosc_pop)
184.        maximum_fitness = np.zeros(liczebnosc_pop)
185.        minimum_fitness = np.zeros(liczebnosc_pop)
186.
187.        # zbior populacji
188.        zb_pop = {}
189.
190.        populacja = startowaPopulacja(liczebnosc_pop)
191.        for i in range(epochs):
192.            zb_pop[i] = populacja
193.            f_fit[i] = np.zeros(liczebnosc_pop)
194.            fitness = np.zeros(len(populacja))
195.            for p in range(len(populacja)):
196.                y = siec(populacja[p])
197.                # wektor w przechowuje liczby: TP, FP, TN, FN
198.                w = np.array([y[0][0], y[0][1], y[1][1], y[1][0]])
199.                fitness[p] = funkcjaZysku(w)
200.            f_fit[i] = fitness
201.            average_fitness[i] = np.mean(fitness)
202.            maximum_fitness[i] = np.max(fitness)
203.            minimum_fitness[i] = np.min(fitness)
204.            if minimum_fitness[i] == maximum_fitness[i]:
205.                print('Najmniejsza i największa wartość funkcji przystosowania
    są takie same.')
206.                print('Osiągnięto minimum lokalne.')
207.                print('Iteracja nr {}'.format(i+1))
```

```
208.                 break
209.             nowa_populacja = selekcja(populacja, fitness)
210.             populacja = nowa_populacja
```

OŚWIADCZENIE AUTORA PRACY MAGISTERSKIEJ

pod tytułem: *Optymalizacja marketingu bezpośredniego z wykorzystaniem genetycznych sieci neuronowych*
napisanej przez: Karola Szerszenia, **nr albumu** 71273
pod kierunkiem dr. Michała Bernardellego

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca dyplomowa nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy dyplomowej jest identyczna z załączoną wersją elektroniczną.

Wyrażam zgodę na poddanie pracy dyplomowej kontroli, w tym za pomocą programu wychwytującego znamiona pracy niesamodzielnej, zwanego dalej programem, oraz na umieszczenie tekstu pracy dyplomowej w bazie porównawczej programu, w celu chronienia go przed nieuprawnionym wykorzystaniem, a także przekazanie pracy do Ogólnopolskiego Repozytorium Prac Dyplomowych.

Wyrażam także zgodę na przetwarzanie przez Szkołę Główną Handlową w Warszawie moich danych osobowych umieszczonych w pracy dyplomowej w zakresie niezbędnym do jej kontroli za pomocą programu oraz w zakresie niezbędnym do jej archiwizacji i nieodpłatnego udostępniania na zasadach określonych w zarządzeniu.

……. ……..
(data) (podpis autora)