NATALIA BERCZ

e-mail: 183913@student.ue.wroc.pl

ORCID: 0000-0001-6682-7551

Uniwersytet Ekonomiczny we Wrocławiu

ZASTOSOWANIE MODELOWANIA MATEMATYCZNEGO I UCZENIA MASZYNOWEGO W ZARZĄDZANIU RYZYKIEM KREDYTOWYM W BANKACH I INSTYTUCJACH FINANSOWYCH

JEL Classification: C88, Y80

Streszczenie: Udzielanie kredytów wiąże się z wysokim ryzykiem. Banki oraz instytucje finansowe poświęcają uwagę i zasoby na zarządzanie tym procesem w celu zwiększenia prawdopodobieństwa, że nie wystąpią niepożądane zdarzenia prowadzące do utraty należności. Modelowanie matematyczne i uczenie maszynowe umożliwiają prognozowanie prawdopodobieństwa niewywiązania się ze zobowiązań i ułatwiają podejmowanie decyzji o udzieleniu pożyczki lub odrzuceniu wniosku o kredyt. Celem artykułu jest analiza zastosowania modelowania matematycznego i uczenia maszynowego w zarządzaniu ryzykiem kredytowym. Istotny element stanowi zaprezentowanie sposobu umożliwiającego zbudowanie modelu w postaci karty scoringowej przydzielającego określoną liczbę punktów każdemu potencjalnemu klientowi. Zbadano także skuteczność takiego modelu, czyli zweryfikowano tezę, czy dzięki niemu ryzyko niespłacenia kredytu przez potencjalnych kredytobiorców może być minimalizowane. Zastosowane w artykule metody badawcze to między innymi: analiza literatury przedmiotu, obserwacja zjawisk, metody modelowania i metody ilościowe.

Słowa kluczowe: ryzyko kredytowe, zarządzanie ryzykiem kredytowym, modelowanie ryzyka kredytowego, *default*.

1. Wstęp

Udzielanie kredytów wiąże się z wysokim ryzykiem. Proces ten powinien zatem być przeprowadzany przejrzyście i kompleksowo. Banki oraz instytucje finansowe, czyli przedsiębiorstwa, dla których ryzyko kredytowe jest szczególnie istotne, poświęcają uwagę i zasoby na zarządzanie takim ryzykiem w celu zwiększenia prawdopodobieństwa, że nie wystąpią niepożądane zdarzenia prowadzące do utraty należności. Skuteczne zarządzanie ryzykiem kredytowym wymaga stosowania różnych technik i składa się z wielu etapów. Modelowanie matematyczne i uczenie maszynowe umożliwiają prognozowanie prawdopodobieństwa niewywiązania się ze zobowiązań przez potencjalnego klienta oraz ułatwiają podejmowanie decyzji o udzieleniu pożyczki lub odrzuceniu wniosku o kredyt. Ponadto zagadnienia z uczenia maszynowego i modelowania matematycznego są interesujące także ze względu na ich szerokie zastosowanie w życiu codziennym.

2. Istota modelowania matematycznego i uczenia maszynowego

Modelowanie matematyczne polega na zastosowaniu matematyki do opisu problemów świata rzeczywistego i badania aspektów, które z tego wynikają. Problem ze świata rzeczywistego tłumaczony jest na problem matematyczny za pomocą narzędzi matematycznych. Modelowanie matematyczne jest czynnością polegającą na przeprowadzaniu eksperymentów. Wyzwanie polega na stworzeniu możliwie najprostszego modelu zawierającego główne cechy interesującego zjawiska. Efektywny model matematyczny odzwierciedla zachowanie rzeczywistej sytuacji życiowej. Oznacza to, że analizując model przy użyciu odpowiednich narzędzi matematycznych, można lepiej zrozumieć określony system. W przypadku bardziej złożonych problemów przydatne okazuje się podejście numeryczne, jednak w wielu sytuacjach wskazane jest sformułowanie złożonego systemu za pomocą prostego modelu – o równaniu zapewniającym rozwiązanie analityczne (Banerjee, 2014).

Do stworzenia modelu matematycznego konieczne jest:

- wprowadzenie czasu oraz zmiennych przestrzennych (opis geometrii),
- stosowanie jednostki podczas rozwiązywania zadania matematycznego – skupienie się na jednostkach wymiarowych,
- wprowadzenie możliwie najprostszych założeń i warunków,
- sformułowanie prawa (fizycznego, ekonomicznego, biologicznego itd.),
- opracowanie możliwie najprostszego opisu matematycznego,
- rozwiązywanie problemów matematycznych – porównanie wyników otrzymanych w przypadku zastosowania różnych metod (Mityushev, Nawalaniec i Rylko, 2018).

W modelowaniu matematycznym przydatnym narzędziem są wykresy. Ważna jest jednak świadomość ich zastosowań i ograniczeń związanych z wykorzystywaniem wykresów. Wnioskowanie odbywa się za pomocą geometrycznej intuicji. Jednym z najłatwiejszych do zaobserwowania obiektów jest linia prosta. Wykresy przynoszą korzyści podczas analizowania przybliżonych danych lub zależności jakościowych obejmujących kilka zmiennych. Graficzne podejście do problemu przydaje się także, gdy dostępnych informacji jest niewiele lub gdy są one określone w niezbyt precyzyjnej formie. Jeżeli dostępne są bardziej dokładne informacje, użyteczniejsze okazują się często metody analityczne (Bender, 1978).

Sformułowanie modelu matematycznego jest zazwyczaj wyraźnie oddzielone od rozwiązania problemów matematycznych. Szukanie rozwiązania jest często skomplikowane i w wielu przypadkach musi być wykonywane przez odpowiednie oprogramowanie. Właśnie dlatego znalezienie rozwiązania większości problemów eksperymentalnych zajęłoby znacznie więcej czasu bez zastosowania modeli matematycznych. Matematykę można postrzegać jako ogromny zasób metod i instrumentów, które mogą być wykorzystywane do rozwiązywania problemów. Modele sprawiają, że metody te mają zastosowanie w niematematycznych problemach (Velten, 2009).

Uczenie maszynowe – z angielskiego *machine learning* – można zdefiniować jako metody obliczeniowe wykorzystujące doświadczenie. Stosuje się je w celu dokonywania dokładnych prognoz lub poprawy wydajności. Doświadczenie odnosi się do informacji z przeszłości, przybierających zazwyczaj formę danych elektronicznych, zebranych i przekazanych do analizy. Uczenie maszynowe polega na projektowaniu dokładnych i wydajnych algorytmów predykcji. Krytyczną miarą jakości takich algorytmów jest ich złożoność czasowa i przestrzenna, jednak w uczeniu maszynowym potrzebne jest także pojęcie złożoności próbki. Umożliwia ono ocenę wielkości próbki, która wymagana jest, żeby algorytm mógł nauczyć się rodziny pojęć. Sukces algorytmu uczenia zależy od użytych danych, dlatego uczenie maszynowe jest ściśle związane ze statystyką i analizą danych. Techniki uczenia to metody oparte na danych, łączące pojęcia informatyczne z pomysłami z optymalizacji, statystyki i prawdopodobieństwa (Mohri, Rostamizadeh i Talwalkar, 2012).

Uczenie maszynowe polega na stosowaniu komputerów do modyfikowania lub dostosowywania działań w celu poprawy ich dokładności. Dokładność mierzona jest tym, jak dobrze wybrane działania odzwierciedlają prawidłowe działania. Uczenie maszynowe jest wielodyscyplinarne. Eksploracja danych polega na wydobywaniu cennych informacji z dużych zbiorów danych, dlatego wymaga wydajnych algorytmów. Istotna jest tutaj również złożoność obliczeniowa metod uczenia maszynowego i utworzonych algorytmów (Zicari, 2018).

Zakres problemów w uczeniu maszynowym jest duży. Naukowcy identyfikują coraz więcej szablonów, które mogą być pomocne w rozwiązaniu pewnych sytuacji. Szablony te ułatwiają wdrażanie uczenia maszynowego w praktyce. Jednym z najczęściej badanych problemów jest klasyfikacja binarna, która sprowadza się do oszacowania, jaką wartość przyjmie zmienna losowa $y \in \{\pm 1\}$, jeśli x pochodzi z dziedziny X. Na przykład na podstawie zdjęć jabłek i pomarańczy dzięki uczeniu maszynowemu można stwierdzić, czym jest zadany obiekt. Podobne przewidywanie ma miejsce, gdy za pomocą danych o dochodach i historii kredytowej właściciela domu ustala się, czy istnieje prawdopodobieństwo, że nie spłaci pożyczki (Smola i Vishwanathan, 2008).

3. Modelowanie ryzyka kredytowego

Dla banków i instytucji finansowych jakość kredytów bankowych jest kluczowym wyznacznikiem konkurencji, rentowności i przetrwania. Ocena kredytowa jest zatem jednym z najważniejszych procesów w zarządzaniu ryzykiem kredytowym. Proces ten obejmuje zbieranie, analizowanie oraz klasyfikację różnych zmiennych kredytowych w celu oceny decyzji kredytowej. Scoring kredytowy polega na zastosowaniu modeli statystycznych do określenia prawdopodobieństwa niewywiązania się ze spłaty zobowiązania przez potencjalnego kredytobiorcę (Abdou i Pointon, 2011). Oznacza on również uszeregowanie klientów wnioskujących o kredyt według zaobserwowanej jakości, która pozwoli przydzielić ich do odrębnych grup – dobrych

i złych kredytobiorców. Jeśli szacowany wynik jest wysoki, klienta można postrzegać jako wiarygodnego. System scoringu kredytowego powinien być optymalny i przypisywać odpowiednią ocenę każdemu klientowi (Ochirsukh, 2016).

Jednym z elementów zarządzania ryzykiem kredytowym w bankach i instytucjach finansowych jest nadanie dłużnikowi statusu *default*. Akronim PD (*Probability of Default*) opisuje prawdopodobieństwo, że klient, który obecnie nie znajduje się w stanie *default*, nie wywiąże się ze swoich zobowiązań w ciągu 12 miesięcy. Jeśli zdarzenie *default* nie wystąpi przez cały okres obserwacji (rok), uważa się ją za dobrą. Jeśli natomiast zdarzenie to będzie miało miejsce przynajmniej raz w ciągu 12 miesięcy od daty rozpoczęcia analizy, obserwacja uważana jest za złą (Koulafetis, 2017).

Podczas tworzenia modeli kart scoringowych wartość WOE (*Weight of Evidence*) stosowana jest do rozróżniania dobrych i złych obserwacji. Istotną cechą WOE jest przekształcanie ryzyka związanego z decyzją o przyznaniu kredytu na łatwiejszą do oceny skalę liniową. Równanie można wyrazić jako:

$$WOE_i = ln\left(\left(\frac{N_i}{\sum N}\right) / \left(\frac{P_i}{\sum P}\right)\right).$$

Brak wystąpienia zdarzenia *default* oznaczono jako *N*, natomiast *P* opisuje obserwacje, w których nastąpiło niewywiązanie się ze zobowiązań (Ochirsukh, 2016).

Jednym ze sposobów wyboru zmiennych predykcyjnych podczas opracowywania kart scoringowych metodą binarnej regresji logistycznej jest obliczenie statystyki IV – wartości informacyjnej. Wielkość tej statystyki decyduje o sile każdej ze zmiennych:

- mniej niż 0,02 – brak przewidywania,
- 0,02 do 0,1 – słaby predyktor,
- 0,1 do 0,3 – średni predyktor,
- większe niż 0,3 – silny predyktor (Ochirsukh, 2016).

Wartość IV zmiennej zależy od sposobu, w jaki została skategoryzowana – im większa jest liczba koszyków, tym większa będzie IV. Wartość informacyjna odnosi się do wartości sumy WOE we wszystkich koszykach i opisana jest równaniem:

$$IV = \sum_{i=1}^{n}\left[\left(\frac{N_i}{\sum N} - \frac{P_i}{\sum P}\right) \cdot WOE_i\right]$$ (Ochirsukh, 2016).

Klasyfikacja zmiennych predykcyjnych jest częścią procesu przekształcania i selekcji zmiennych podczas tworzenia modelu regresji liniowej lub logistycznej. Cechy liczbowe i jakościowe zmodyfikować można tak, aby zmienną ciągłą przeobrazić w kilka dyskretnych przedziałów, a różne poziomy w zmiennych jakościowych podzielić na mniejszą liczbę klas. Większa liczba koszyków wiąże się z ryzykiem wystąpienia przesadnego dopasowania (*overfitting*) oraz z większym potencjałem

złożoności modelu, jednak liczba zbyt mała może prowadzić do utraty wartości informacyjnej. Celem klasyfikacji jest więc znalezienie minimalnej liczby koszyków, z zachowaniem maksymalnej możliwej wartości informacyjnej (Ochirsukh, 2016).

Budowa modelu karty scoringowej w języku R

Biblioteki R, które można wczytać za pomocą funkcji *library*(), umożliwiają korzystanie z dodatkowych opcji programu.

```
library(readxl)
library(rsample)
library(tidyverse)
library(scorecard)
library(reldist)
library(ROCR)
```

Jeśli pewne czynności wykonywane są więcej niż jeden raz, dobrym sposobem na zredukowanie powtarzających się linijek kodu jest zdefiniowanie dodatkowych funkcji poprzez komendę *function*().

Funkcja *plot_roc_curve*(), przyjmująca argument *data*, czyli ramkę danych, za pomocą funkcji *prediction*() z biblioteki *ROCR*, tworzy obiekt predykcji, przekształcając dane wejściowe do standardowego formatu. Polecenie *performance*() ocenia predyktor *pred*, a następnie funkcja *plot*() rysuje wykres krzywej na podstawie tak wyznaczonego obiektu *perf* – pozostałe argumenty tej funkcji odpowiadają za wygląd i etykiety wykresu. Komenda *ab-line*() szkicuje prostą $y = x$ przydatną do analizy krzywej.

```
plot_roc_curve = function(data) {
 pred = prediction(data$SCORE, data$DEFAULT, c(1, 0))
 perf = performance(pred, 'tpr', 'fpr')
 plot(perf, col='darklateblue', lwd=2, main='ROC CURVE',
      xlab='FALSE POSITIVE RATE', ylab='TRUE POSITIVE RATE')
 abline(a=0, b=1, lty=3) }
```

Funkcja *plot_score_dist*() przyjmuje argumenty: *data* – ramkę danych, *score* – wektor punktacji, *max* – maksymalną wielkość przedziału, oraz *step*, opisujący, w którym momencie powinien zaczynać się nowy przedział. Cel funkcji stanowi wyznaczenie wektora z przedziałami, obliczenie liczby zdarzeń *default* w każdym z przedziałów – za pomocą pętli *for*, a następnie sporządzenie wykresu rozkładu punktów w całej populacji. Funkcja *plot_sco-re_dist*() wykorzystuje pakiet *tidyverse*, w którym znajdują się polecenia *mutate*() – umożliwiające modyfikację kolumn w ramce danych, oraz *ggplot*() – odpowiadające za utworzenie finalnego wykresu.

```
plot_score_dist = function(data, score, max, step) {
 score_bins = seq(0, max, by=step)
 score_groups = integer(length(score_bins)-1)
 default_counter = integer(length(score_bins)-1)

 df = data.frame(seq(0, max-10, by=step), score_groups, default_counter)

 scores = c()
 for (i in 2:length(score_bins)-1) {
      counter = 0
      index = match(score_bins[i], score_bins)
      scores = c(scores, str_c('[ ', score_bins[i], ', ', score_bins[i+1], ' )'))
      levels = scores
      for (sc in score) {
           counter = counter + 1
           if (sc > score_bins[i] && sc <= score_bins[i+1]) {
              df$score_groups[index] = df$score_groups[index] + 1
           if (data$DEFAULT[counter] == 1) {
              df$default_counter[index] = df$default_counter[index] + 1 }}}}

 df = df %>%
      mutate(bad_rate=default_counter/score_groups) %>%
      mutate(score_dens=df$score_groups/length(score))

ggplot(df, aes(x=factor(scores, levels = scores))) +
      geom_col(aes(y=score_dens), color='black', fill='powderblue') +
      geom_point(aes(y=bad_rate), color='darkslateblue', size=3) +
      ggtitle('SCORE DISTRIBUTION IN THE ENTIRE POPULATION') +
      xlab('SCORE RANGES') + ylab('') + theme_bw() +
      theme(plot.title=element_text(hjust=0.5),
            panel.grid.major=element_blank(),
            panel.grid.minor=element_blank()) }
```

Funkcja *read_excel*(), dostępna w pakiecie *readxl*, umożliwia wczytanie danych, na których zostanie utworzony model karty scoringowej.

```
credit_data = read_excel(‚bank_data.xlsx’)
```

Jeżeli istnieje dostatecznie wiele obserwacji, ze zbioru danych można wydzielić pewną podpopulację, która będzie przydatna do przetestowania poprawności modelu. Wylosowana próba testowa może zawierać do 30% wszystkich danych. Na tym etapie istotne jest ustawienie ziarna, które skutkuje wyborem takich samych danych

testowych podczas każdego uruchomienia kodu i umożliwia zbudowanie stabilnego modelu.

```
set.seed(123)
```

Podział danych wykonać można z użyciem funkcji *initial_split*() z biblioteki *rsample*. Argument *prop* oznacza proporcję podziału danych, natomiast *strata* powoduje, że proporcja ta zachowana jest również w kolumnie *DEFAULT*.

```
data_split = initial_split(credit_data, prop=0.7, strata='DEFAULT')

train_data = training(data_split)
test_data = testing(data_split)
```

Funkcja *woebin*() generuje optymalne koszyki dla każdego predyktora. Argument *y* oznacza zmienną, względem której koszyki powinny być utworzone, natomiast *var_skip* umożliwia pominięcie zmiennych.

```
train_buckets = woebin(train_data, y='DEFAULT',
                                   var_skip=c(,ID_KLIENTA', ,DATA_WNIOSKU'))
```

W celu obliczenia IV dla wszystkich zmiennych można zastosować listę *train_buckets*. Funkcja *map_df*() umożliwia wywołanie komendy *pluck*() na każdym elemencie tej listy. Polecenie *pivot_longer*() transponuje tabelkę do pozycji pionowej oraz nadaje jej nagłówki.

```
IV = map_df(train_buckets, ~pluck(.x, 10, 1)) %>%
  pivot_longer(everything(), names_to='VAR', values_to='IV')
```

Na rysunku 1 przedstawiono ramkę danych IV, na której poprzez funkcję *arrange*(), dostępną w pakiecie *tidyverse*, zastosowane jest sortowanie rosnąco po wartościach IV.

```
IV %>% arrange(IV)
```

	VAR	IV
	<chr>	<dbl>
1	WIELKOSC_PRACODAWCY	0.0328
2	STATUS_MIESZKANIOWY	0.0521
3	WOJEWODZTWO	0.0538
4	DOCHOD	0.0617
5	WYKSZTALCENIE	0.107
6	WYKONYWANY_ZAWOD	0.127
7	STAN_CYWILNY	0.158
8	SEKTOR	0.168
9	WNIOSKOWANA_KWOTA	0.195
10	RODZAJ_ZATRUDNIENIA	0.203
11	BIK_LICZBA_WNIOSKOW_POPRZEDNI_MC	0.215
12	MIESIACE_ZATRUDNIENIA	0.216
13	TYP_PRACODAWCY	0.230
14	BIK_LICZBA_WNIOSKOW_BIEZACY_MC	0.244
15	PRODUKT	0.304

Rys. 1. Wartości informacyjne zmiennych

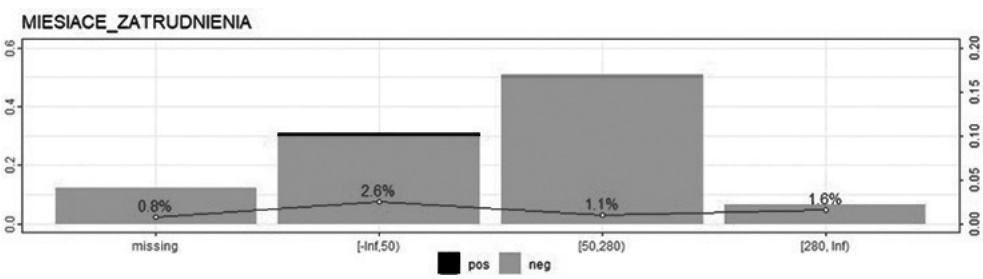
Źródło: opracowanie własne.

Zmienne ciągłe należy zweryfikować pod względem monotoniczności złych obserwacji, które w nich występują. Sprawdzenie to będzie dotyczyło jedynie czterech zmiennych: *MIESIACE_ZATRUDNIENIA*, *WNIOSKOWANA_KWOTA*, *BIK_WNIOSKI_POPRZEDNI_MC* oraz *BIK_WNIOSKI_BIEZACY_MC*, ponieważ *WIELKOSC_PRACODAWCY* oraz *DOCHOD* zostały odrzucone na poprzednim etapie.

Za pomocą funkcji *woebin_plot*(), z biblioteki *scorecard*, można wyświetlić wykres liczebności przedziałów oraz rozkład złych obserwacji, które w nich występują. Kolorem niebieskim oznaczone są obserwacje negatywne – nie odnotowano zdarzenia *default*, natomiast czerwonym pozytywne.

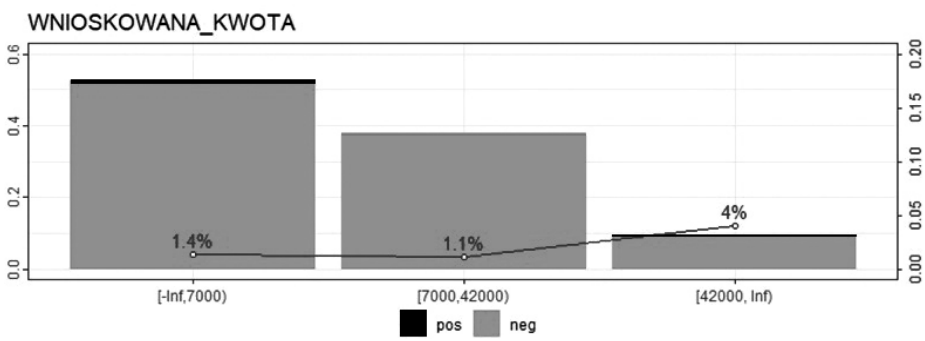
```
woebin_plot(train_buckets$MIESIACE_ZATRUDNIENIA)
woebin_plot(train_buckets$WNIOSKOWANA_KWOTA)
woebin_plot(train_buckets$BIK_WNIOSKI_BIEZACY_MC)
woebin_plot(train_buckets$BIK_WNIOSKI_POPRZEDNI_MC)
```

Na rysunku 2 przedstawiony jest procent złych obserwacji dla zmiennej *MIESIACE_ZATRUDNIENIA*. Monotoniczność nie występuje, ponieważ 2,6% > 11%, natomiast 1,1% < 1,6%.



Rys. 2. Rozkład defaultów – MIESIACE_ZATRUDNIENIA

Źródło: opracowanie własne.

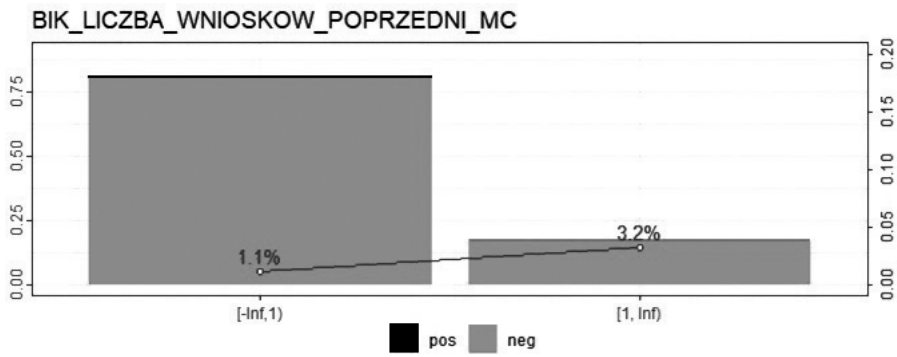


Rys. 3. Rozkład defaultów – WNIOSKOWANA_KWOTA

Źródło: opracowanie własne.

Rysunek 3 pokazuje, że dla kwot poniżej 7000 zł złe obserwacje stanowią 1,4%, w przedziale [7000, 42000) jest to 1,1%, natomiast dla kwot większych niż 42 000 – 4%. Można więc zauważyć, że procent obserwacji *default* nie jest monotoniczny dla zmiennej *WNIOSKOWANA_KWOTA*.

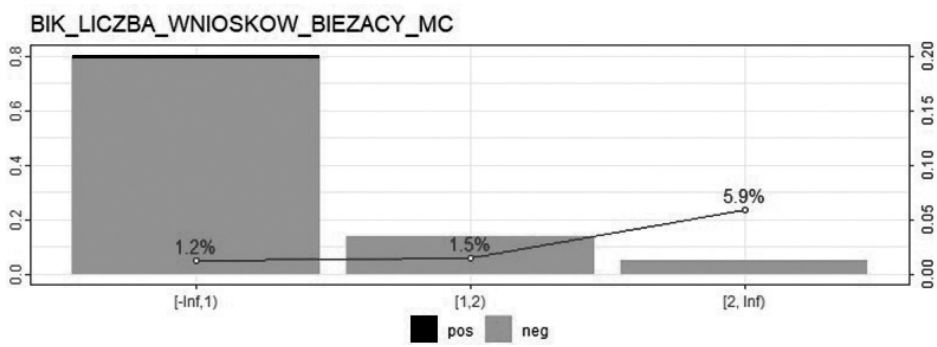
Na rysunku 4 przedstawiono monotoniczność rozkładu złych obserwacji dla zmiennej *BIK_LICZBA_WNIOSKOW_POPRZEDNI_MC*.



Rys. 4. Rozkład defaultów – BIK_LICZBA_WNIOSKOW_POPRZEDNI_MC

Źródło: opracowanie własne.

Na rysunku 5 można zaobserwować monotoniczność obserwacji *default* dla zmiennej *BIK_LICZBA_WNIOSKOW_BIEZACY_MC*.



Rys. 5. Rozkład defaultów – BIK_LICZBA_WNIOSKOW_BIEZACY_MC

Źródło: opracowanie własne.

Za pomocą funkcji *select*() z pakietu *tidyverse* oraz znaku „–” przed nazwą kolumny z modelowania można odrzucić zmienne o niskiej wartości informacyjnej lub niemonotonicznym rozkładzie złych obserwacji, a także kolumny *ID_KLIENTA* i *DATA_WNIOSKU*.

```
train_model_data = train_data %>%
 select(-WIELKOSC_PRACODAWCY, -STATUS_MIESZKANIOWY,
        -WOJEWODZTWO, -WNIOSKOWANA_KWOTA, -DOCHOD,
        -MIESIACE_ZATRUDNIENIA, -ID_KLIENTA, -DATA_WNIOSKU)
```

Funkcja *woebin_ply*(), znajdująca się w bibliotece *scorecard*, umożliwia przekonwertowanie każdej z kolumn na wyznaczone w nich wartości WOE.

```
train_data_woe1 = woebin_ply(train_model_data, train_fine_classing)
```

W celu utworzenia modelu można zastosować funkcję *glm*(), która służy do dopasowywania uogólnionych modeli liniowych. Funkcja ta przyjmuje argumenty:

- *formula* – symboliczny opis modelu, który ma zostać dopasowany,
- *family* – opis rozkładu błędów,
- *data* – ramka danych zawierająca zmienne do modelowania.

Parametr *link='logit'* jest domyślnym argumentem dla rodziny *binomial*(). Oznacza to, że podczas tworzenia modelu używana jest specyfikacja logitowa regresji logistycznej.

Model 1

W pierwszym modelu zostały użyte wszystkie dotychczas nieodrzucone zmienne. Rysunek 6 pokazuje wynik z funkcji *summary*(*model1*). Predyktor – określony symbolem *** – jest bardzo istotny, natomiast zmienna – oznaczona pustym polem – nie jest istotna.

```
model1 = glm(DEFAULT~., family=binomial(), data=train_data_woe1)
summary(model1)
```

```
Coefficients:
                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                      -4.17009    0.07486 -55.709  < 2e-16 ***
PRODUKT_woe                       1.01867    0.14010   7.271 3.57e-13 ***
WYKSZTALCENIE_woe                 0.96577    0.22726   4.250 2.14e-05 ***
WYKONYWANY_ZAWOD_woe             -0.11711    0.24581  -0.476   0.6338
RODZAJ_ZATRUDNIENIA_woe           0.64973    0.22383   2.903   0.0037 **
SEKTOR_woe                        0.47853    0.21248   2.252   0.0243 *
TYP_PRACODAWCY_woe                0.37059    0.22087   1.678   0.0934 .
STAN_CYWILNY_woe                  0.80431    0.15823   5.083 3.71e-07 ***
BIK_LICZBA_WNIOSKOW_BIEZACY_MC_woe  0.69347    0.11193   6.196 5.80e-10 ***
BIK_LICZBA_WNIOSKOW_POPRZEDNI_MC_woe  0.71440    0.13505   5.290 1.22e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Rys. 6. Podsumowanie pierwszego modelu

Źródło: opracowanie własne.

Model 2

Zmienna *WYKONYWANY_ZAWOD_woe* była najmniej istotna w pierwszym modelu. Można więc z niej zrezygnować i powtórzyć pozostałe kroki.

```
train_data_woe2 = train_data_woe1 %>%
 select(-WYKONYWANY_ZAWOD_woe)

model2 = glm(DEFAULT~., family=binomial(), data=train_data_woe2)
summary(model2)
```

Rysunek 7 przedstawia podsumowanie drugiego modelu.

```
Coefficients:
                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                       -4.16967    0.07482 -55.729  < 2e-16 ***
PRODUKT_woe                        1.01735    0.14016   7.259 3.91e-13 ***
WYKSZTALCENIE_woe                  0.94904    0.22448   4.228 2.36e-05 ***
RODZAJ_ZATRUDNIENIA_woe            0.60176    0.20058   3.000   0.0027 **
SEKTOR_woe                         0.48544    0.21235   2.286   0.0223 *
TYP_PRACODAWCY_woe                 0.34671    0.21595   1.606   0.1084
STAN_CYWILNY_woe                   0.80526    0.15811   5.093 3.53e-07 ***
BIK_LICZBA_WNIOSKOW_BIEZACY_MC_woe 0.69255    0.11191   6.189 6.07e-10 ***
BIK_LICZBA_WNIOSKOW_POPRZEDNI_MC_woe 0.71473  0.13505   5.292 1.21e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Rys. 7. Podsumowanie drugiego modelu

Źródło: opracowanie własne.

Model 3

W drugim modelu najmniej istotna była zmienna *TYP_PRACODAWCY_woe*. Nie zostanie więc ona użyta do zbudowania następnego modelu.

```
train_data_woe3 = train_data_woe2 %>%
 select(-TYP_PRACODAWCY_woe)

model3 = glm(DEFAULT~., family=binomial(), data=train_data_woe3)
summary(model3)
```

Na rysunku 8 przedstawiono podsumowanie trzeciego modelu.

```
Coefficients:
                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                      -4.16762    0.07459 -55.874  < 2e-16 ***
PRODUKT_woe                       1.01696    0.13981   7.274 3.49e-13 ***
WYKSZTALCENIE_woe                 0.93768    0.22376   4.191 2.78e-05 ***
RODZAJ_ZATRUDNIENIA_woe           0.78357    0.16564   4.730 2.24e-06 ***
SEKTOR_woe                        0.64320    0.19013   3.383 0.000717 ***
STAN_CYWILNY_woe                  0.81660    0.15834   5.157 2.51e-07 ***
BIK_LICZBA_WNIOSKOW_BIEZACY_MC_woe  0.69530    0.11185   6.217 5.08e-10 ***
BIK_LICZBA_WNIOSKOW_POPRZEDNI_MC_woe  0.71386    0.13505   5.286 1.25e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Rys. 8. Podsumowanie trzeciego modelu

Źródło: opracowanie własne.

Za pomocą funkcji *scorecard*(), z biblioteki *scorecard*, można utworzyć kartę scoringową. W tym przypadku parametry zostały dopasowane tak, aby liczba punktów nie była mniejsza od 0.

```
train_scorecard = scorecard(train_buckets, model3, points0=120, pdo=25)
```

Funkcja *scorecard_ply*() pozwala obliczyć wynik kredytowy dla każdego klienta z zastosowaniem karty scoringowej *train_scorecard*. Polecenie *unlist*() zmienia listę w wektor, który poprzez funkcję *mutate*() może zostać dodany do tabeli z danymi treningowymi w postaci kolumny SCORE.

```
train_score = unlist(scorecard_ply(train_data, train_scorecard))
train_data = train_data %>%
 mutate(SCORE=train_score)
```

Po wyświetleniu fragmentów kolumn *ID_KLIENTA* oraz *SCORE* można upewnić się, że punkty zostały przydzielone (rys. 9).

```
train_data %>% select(ID_KLIENTA, SCORE)
```

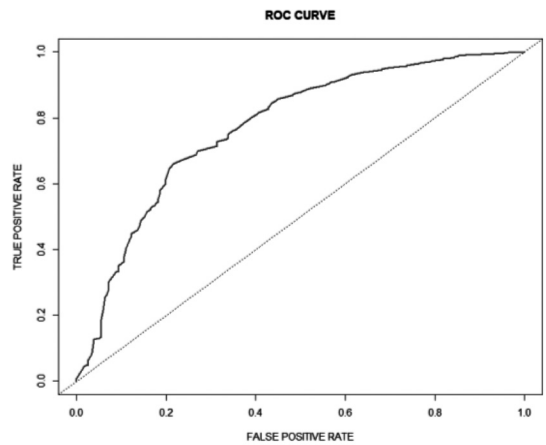
```
   ID_KLIENTA SCORE
        <dbl> <dbl>
 1 4651882957   158
 2 5955439230    89
 3 1828184461   193
 4 1511147183   146
 5 4323588145   142
 6 8559770598   243
 7 9787703856   204
 8 7045065620   204
 9 5631164959   249
10 2484475671   171
# ... with 15,432 more rows
```

Rys. 9. Fragment tabeli z przydzielonymi punktami

Źródło: opracowanie własne.

Na rysunku 10 przedstawiono krzywą ROC, która została wyświetlona za pomocą funkcji *plot_roc_curve*(). Model nie jest losowy, ponieważ krzywa znajduje się w pewnej odległości od prostej $y = x$.

```
plot_roc_curve(train_data)
```

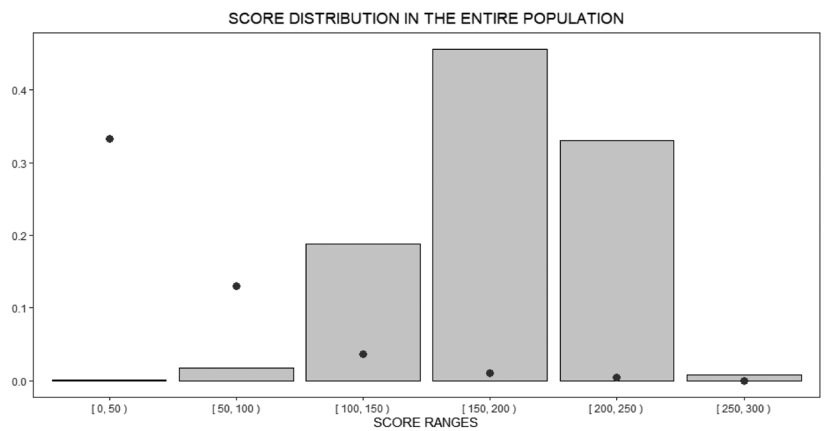


Rys. 10. Krzywa ROC dla populacji treningowej

Źródło: opracowanie własne.

Rysunek 11 prezentuje wykres rozkładu punktów w całej populacji wyświetlony poprzez funkcję *plot_score_dist*(). Granatowe punkty oznaczają wyznaczony w każdym z koszyków procent defaultów. Można zauważyć, że zależność ta jest malejąca. Model działa zatem prawidłowo – im więcej punktów otrzymał klient, tym mniejsze jest prawdopodobieństwo, że nie wywiązał się ze zobowiązania.

```
plot_score_dist(train_data, train_score, 300, 50)
```



Rys. 11. Rozkład SCORE w populacji treningowej

Źródło: opracowanie własne.

W celu sprawdzenia modelu na danych testowych należy odrzucić zmienne, które zostały pominięte podczas jego budowy, policzyć optymalne koszyki oraz przekonwertować kolumny ze zmiennymi na wartości WOE.

```
test_model_data = test_data %>%
 select(-WIELKOSC_PRACODAWCY, -STATUS_MIESZKANIOWY,
        -WOJEWODZTWO, -DOCHOD, -WNIOSKOWANA_KWOTA,
        -MIESIACE_ZATRUDNIENIA, -ID_KLIENTA, -DATA_WNIOSKU,
        -WYKONYWANY_ZAWOD, -TYP_PRACODAWCY)

test_buckets = woebin(test_model_data, y='DEFAULT')

test_data_woe = woebin_ply(test_model_data, test_buckets)
```

Ramka danych *test_data_woe* zawiera jedynie zmienne z modelu treningowego, dlatego funkcja *glm*(), zastosowana na danych testowych, może wyglądać następująco:

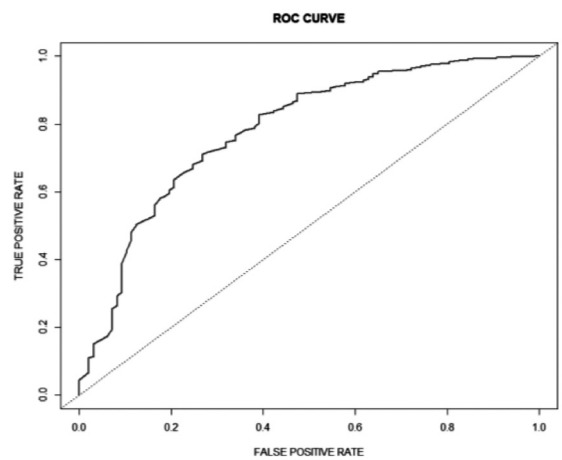
```
test_model = glm(DEFAULT~., family=binomial(), data=test_data_woe)
```

Na danych testowych można utworzyć kartę scoringową, stosując takie same parametry jak dla danych treningowych.

```
test_scorecard = scorecard(test_buckets, test_model, points0=120, pdo=25)
test_score = unlist(scorecard_ply(test_data, test_scorecard))
test_data = test_data %>%
 mutate(SCORE=test_score)
```

Rysunek 12 przedstawia wykres krzywej ROC. Można zauważyć, że model zachowuje się prawidłowo, także dla danych testowych.

```
plot_roc_curve(test_data)
```

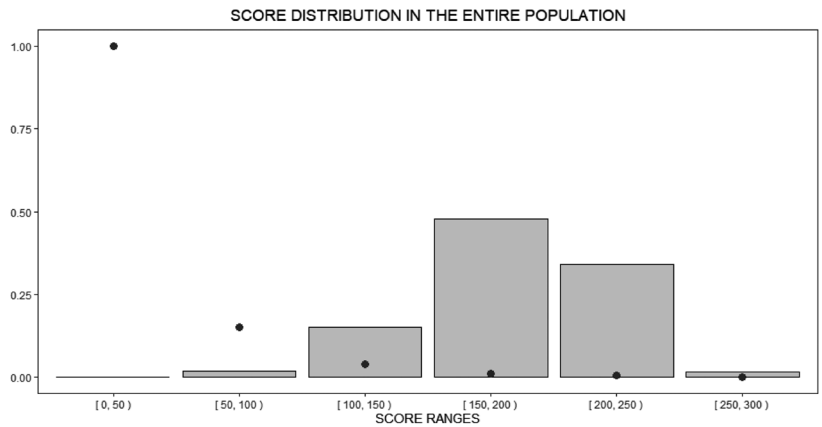


Rys. 12. Krzywa ROC dla populacji testowej

Źródło: opracowanie własne.

Tak samo jak w przypadku danych treningowych zależność odsetka defaultów do liczby przyznanych punktów jest malejąca (rys. 13). Działanie modelu jest zatem prawidłowe.

```
plot_score_dist(test_data, test_score, 300, 50)
```



Rys. 13. Rozkład SCORE w populacji testowej

Źródło: opracowanie własne.

4. Podsumowanie

Artykuł zawierał informacje dotyczące modelowania matematycznego i uczenia maszynowego. W wyniku przeprowadzonych badań stwierdzono, że utworzenie modelu ryzyka kredytowego w postaci karty scoringowej, przypisującego klientowi odpowiednią liczbę punktów, umożliwia zmniejszenie prawdopodobieństwa niespłacania podjętego przez niego kredytu, ponieważ model ten poprawnie szereguje klientów – im większy wynik uzyskuje kredytobiorca, tym mniejsze jest prawdopodobieństwo niewywiązania się przez niego ze zobowiązania.

Literatura

Abdou, H. i Pointon, J. (2011). *Credit scoring, statistical techniques and evaluation criteria*. Salford: Wiley-Blackwell.

Banerjee, S. (2014). *Mathematical modeling: Models, analysis and applications*. Boca Raton: CRC Press.

Bender, E. (1978). *An introduction to mathematical modeling*. Toronto: Wiley-Interscience.

Mityushev, V., Nawalaniec, W. i Rylko, N. (2018). *Introduction to mathematical modeling and computer simulations*. Kraków: CRC Press.

Mohri, M., Rostamizadeh, A. i Talwalkar, A. (2012). *Foundations of machine learning.* Cambridge: The MIT Press.
Ochirsukh, M. (2016). *Application scorecard modelling: Techniques and performance*. Amsterdam: VU University Amsterdam.
Smola, A. i Vishwanathan, S. (2008). *Introduction to machine learning*. Cambridge: Cambridge University Press.
Velten, K. (2009). *Mathematical modeling and simulation*: *Introduction for scientists and engineers.* Verlag: Wiley-Vch.
Zicari, R. (2018). *Explorations in artificial intelligence and machine learning*. Frankfurt: CRC Press.

THE APPLICATION OF MATHEMATICAL MODELLING AND MACHINE LEARNING IN BANKS AND FINANCIAL INSTITUTIONS

Abstract: The lending process is associated with high risk, therefore it should be carried out transparently and comprehensively. Banks and financial institutions devote their attention and resources to managing credit risk to increase the probability that undesirable events will not occur leading to loss of receivables. Mathematical modelling and machine learning make it possible to forecast the probability of a potential customer defaulting and facilitate loan granting or loan rejection decisions. Presenting a way to build a model that assigns score to each potential client and verification of that model is an important element of this article. The effectiveness of such a model is tested – whether on its basis the risk of loan default by potential borrowers can be minimized. The research methods used in the work include: analysis of the literature on the subject, observation of phenomena, modelling methods and quantitative methods.

Keywords: credit risk, credit risk management, credit risk models, default.