Code

Zastosowanie wybranych metod uczenia maszynowego w zagadnieniu scoringu kredytowego

Paweł Warchoł

30 maja 2019

1. Wprowadzenie

W ramach niniejszego projektu przedstawione zostanie zastosowanie wybranych metod uczenia maszynowego w zagadnieniu scoringu kredytowego. Celem projektu jest przegląd i zapoznanie się z charakterystyką wybranych technik oraz zbudowaniem modeli prognozujących wiarygodność kredytową indywidualnych klientów, a także wybór najlepszej z metod na podstawie otrzymanych wyników. W ramach projektu użyte zostaną dane umieszczone na stronie Analytics Vidhya w ramach jednego z otwartych przez nich konkursów, który można znaleźć pod tym linkiem.

Kolejne części projektu dotyczyć będą:

- przedstawienia danych oraz przeprowadzonych na nich operacji (takich jak przygotowanie danych do badania, tworzenie nowych zmiennych, sposób uzupełniania braków danych czy analiza obserwacji odstających),
- zagadnienia dzielenia zbioru na uczący oraz testowy,
- opisu wykorzystanych w badaniu technik,
- zbudowanie modeli opartych o owe techniki,
- dyskusji na otrzymanymi wynikami oraz porównania ich i wyboru najlepszej.

2.Przedstawienie i praca z danymi

Na początek krótkie zapoznanie się z danymi - opis zmiennych, struktura danych i podstawowe statystyki opisowe zbioru.
(W tej części kodu widoczna jest również lista użytych pakietów, wraz z komentarzami (jeśli potrzebne) do czego były użyte)

Code

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAm
LP001002	Male	No	0	Graduate	No	5849	0.00	
LP001003	Male	Yes	1	Graduate	No	4583	1508.00	
LP001005	Male	Yes	0	Graduate	Yes	3000	0.00	
LP001006	Male	Yes	0	Not Graduate	No	2583	2358.00	
LP001008	Male	No	0	Graduate	No	6000	0.00	
LP001011	Male	Yes	2	Graduate	Yes	5417	4196.00	
LP001013	Male	Yes	0	Not Graduate	No	2333	1516.00	
LP001014	Male	Yes	3+	Graduate	No	3036	2504.00	

2.1 Opis wczytanych zmiennych :

- Gender - zmienna informująca o płci wnioskodawcy,
- Loan_ID - unikalny numer kredytu,
- Married - czy wnioskodawca/wnioskodawczyni jest żonaty/wyszłą za mąż,
- Dependents - liczba osób na utrzymaniu wnioskodawcy,
- Education - zmienna opisująca czy wnioskodawca ma wykształcenie wyższe,
- Self_Employed - czy wnioskodawca pracuje na własny rachunek,

- ApplicantIncome - miesięczny przychód wnioskodawcy,
- CoapplicantIncome - miesięczny przychód współwnioskodawcy,
- LoanAmount - kwota kredytu wyrażona w tysiącach,
- Loan_Amount_Term - termin kredytu,
- Credit_History - czy historia kredytowa spełnia wytyczne,
- Property_Area - obszar zamieszkania,
- Loan_Status - decyzja czy udzielono kredytu wnioskodawcy.

2. Struktura danych i ich wstępne podsumowanie:

```
## 'data.frame':    614 obs. of  13 variables:
##  $ Loan_ID          : Factor w/ 614 levels "LP001002","LP001003",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Gender           : Factor w/ 3 levels "","Female","Male": 3 3 3 3 3 3 3 3 3 3 ...
##  $ Married          : Factor w/ 3 levels "","No","Yes": 2 3 3 3 2 3 3 3 3 3 ...
##  $ Dependents       : Factor w/ 5 levels "","0","1","2",..: 2 3 2 2 2 4 2 5 4 3 ...
##  $ Education        : Factor w/ 2 levels "Graduate","Not Graduate": 1 1 1 2 1 1 2 1 1 1 ...
##  $ Self_Employed    : Factor w/ 3 levels "","No","Yes": 2 2 3 2 2 3 2 2 2 2 ...
##  $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
##  $ CoapplicantIncome: num  0 1508 0 2358 0 ...
##  $ LoanAmount       : int  NA 128 66 120 141 267 95 158 168 349 ...
##  $ Loan_Amount_Term : int  360 360 360 360 360 360 360 360 360 360 ...
##  $ Credit_History   : int  1 1 1 1 1 1 1 0 1 1 ...
##  $ Property_Area    : Factor w/ 3 levels "Rural","Semiurban",..: 3 1 3 3 3 3 3 2 3 2 ...
##  $ Loan_Status      : Factor w/ 2 levels "N","Y": 2 1 2 2 2 2 2 1 2 1 ...
```

Jak widać zmienne są głównie kategoryczne, aczkolwiek np ‘Credit_History’ nie jest wczytana jako typ factor, mimo że powinna. Ponadto, w przypadku tych zmienny obserwacje brakujące są zawarte jako jeden z poziomów czynnika (“”), co nie jest pożądaną sytuacją. Mylące jest również wyrażanie zmiennej LoanAmount w tysiącach.
Po uporządkowaniu omawianych kwestii dane prezentują się następująco:

Code

```
## 'data.frame':    614 obs. of  13 variables:
##  $ Loan_ID          : Factor w/ 614 levels "LP001002","LP001003",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Gender           : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Married          : Factor w/ 2 levels "No","Yes": 1 2 2 2 1 2 2 2 2 2 ...
##  $ Dependents       : Factor w/ 4 levels "0","1","2","3+": 1 2 1 1 1 3 1 4 3 2 ...
##  $ Education        : Factor w/ 2 levels "Graduate","Not Graduate": 1 1 1 2 1 1 2 1 1 1 ...
##  $ Self_Employed    : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 2 1 1 1 1 ...
##  $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
##  $ CoapplicantIncome: num  0 1508 0 2358 0 ...
##  $ LoanAmount       : num  NA 128000 66000 120000 141000 267000 95000 158000 168000 349000 ...
##  $ Loan_Amount_Term : int  360 360 360 360 360 360 360 360 360 360 ...
##  $ Credit_History   : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 1 2 2 ...
##  $ Property_Area    : Factor w/ 3 levels "Rural","Semiurban",..: 3 1 3 3 3 3 3 2 3 2 ...
##  $ Loan_Status      : Factor w/ 2 levels "N","Y": 2 1 2 2 2 2 2 1 2 1 ...
```

Podsumowanie danych za pomocą funkcji `summary`:

```
##      Loan_ID       Gender    Married    Dependents        Education
##  LP001002:  1   Female:112   No  :213   0   :345   Graduate    :480
##  LP001003:  1   Male  :489   Yes :398   1   :102   Not Graduate:134
##  LP001005:  1   NA's  : 13   NA's:  3   2   :101
##  LP001006:  1                           3+  : 51
##  LP001008:  1                           NA's: 15
##  LP001011:  1
##  (Other) :608
##  Self_Employed ApplicantIncome CoapplicantIncome   LoanAmount
##  No  :500      Min.   :  150   Min.   :    0     Min.   :  9000
##  Yes : 82      1st Qu.: 2878   1st Qu.:    0     1st Qu.:100000
##  NA's: 32      Median : 3812   Median : 1188     Median :128000
##                Mean   : 5403   Mean   : 1621     Mean   :146412
##                3rd Qu.: 5795   3rd Qu.: 2297     3rd Qu.:168000
##                Max.   :81000   Max.   :41667     Max.   :700000
##                                                  NA's   :22
##  Loan_Amount_Term Credit_History   Property_Area Loan_Status
##  Min.   : 12      No  : 89       Rural    :179   N:192
##  1st Qu.:360      Yes :475       Semiurban:233   Y:422
##  Median :360      NA's: 50       Urban    :202
##  Mean   :342
##  3rd Qu.:360
##  Max.   :480
##  NA's   :14
```

2.3 Obserwacje brakujące

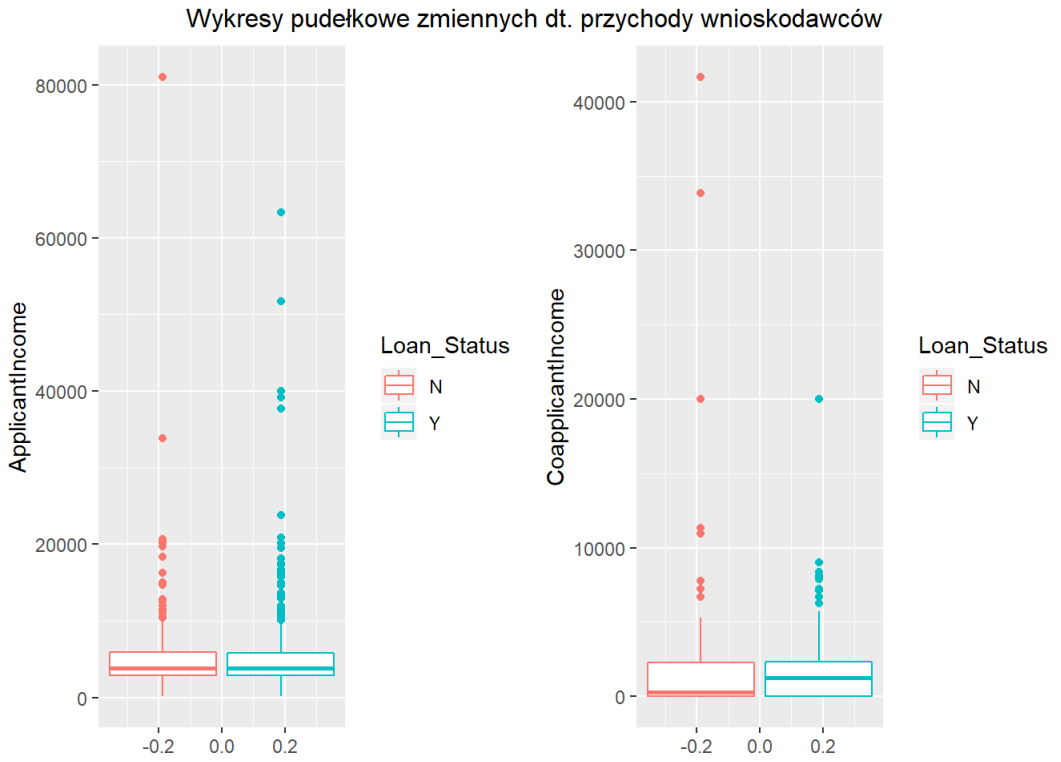
Widać, że kolumna zawierająca ID kredytu nie będzie potrzebna w badaniu, więc może zostać usunięta. W podsumowaniu zawarto również informację, że wśród danych znajdują się braki. Liczba NA nie jest bardzo duża, lecz ich uzupełnienie może w pozytywny sposób wpłynąć na wyniki modeli. W związku z tym, NA zostaną uzupełnione za pomocą funkcji `missForest` z pakietu o takiej samej nazwie. Funkcja ta, jak wskazuje jej nazwa, jest implementacją algorytmu lasów losowych. Dla każdej zmiennej funkcja ta buduje model lasów losowych (używając do tego pozostałych zmiennych) i używa go potem do predykcji wartości brakujących zmiennych. Sam algorytm lasów losowych jest następujący: ze zbioru uczącego (w tym wypadku wszystkich dostępnych obserwacji) losowane ze zwracaniem jest N próbek x-elementowych. Dla każdej z próbek losowane są zmienne (ich liczba najczęściej to wyznaczana jako pierwiastek z liczby wszystkich zmiennych), które wykorzystane zostaną do stworzenia drzewa na podstawie danej próbki. Dla każdej próbki tworzone jest drzewo decyzyjne. Później na podstawie wyników wszystkich drzew dokonuje się predykcji na zasadzie głosowania (zmienne kategoryczne) lub uśrednia się wyniki (zmienne ciągłe).
Po uzupełnieniu braków, dane przedstawiają się następująco:

```
##     Gender    Married   Dependents        Education   Self_Employed
##  Female:115   No :215   0 :351     Graduate    :480   No :528
##  Male  :499   Yes:399   1 :105     Not Graduate:134   Yes: 86
##                         2 :106
##                         3+: 52
##
##
##  ApplicantIncome CoapplicantIncome   LoanAmount     Loan_Amount_Term
##  Min.   :  150   Min.   :    0     Min.   :  9000   Min.   : 12.0
##  1st Qu.: 2878   1st Qu.:    0     1st Qu.:100000   1st Qu.:360.0
##  Median : 3812   Median : 1188     Median :128000   Median :360.0
##  Mean   : 5403   Mean   : 1621     Mean   :146505   Mean   :341.8
##  3rd Qu.: 5795   3rd Qu.: 2297     3rd Qu.:167750   3rd Qu.:360.0
##  Max.   :81000   Max.   :41667     Max.   :700000   Max.   :480.0
##  Credit_History   Property_Area Loan_Status
##  No : 89        Rural    :179   N:192
##  Yes:525        Semiurban:233   Y:422
##                 Urban    :202
##
##
##
```

2.4 Obserwacje odstające i modyfikacja/tworzenie nowych zmiennych

Kolejnym problemem jaki zauważyć można na podstawie podsumowania są tzw. outliery, czyli obserwacje odstające. Widać, że zmienne ‘ApplicantIncome’ i ‘CoapplicantIncome’ mają bardzo duży rozrzut - różnica między wartościami maksymalnymi, a 3 kwantylem jest bardzo duża. Dla lepszego rozpoznania sytuacji przedstawione zostaną odpowiednie wykresy pudełkowe.

Code



Widzimy dość dziwny przypadek, w którym pożyczkobiorca o największym dochodzie nie otrzymuje kredytu. Podobnie 2 współwnioskodawców o najwyższych dochodach. Dokładniej mowa o poniższych obserwacjach:

Code

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	L
410	Male	Yes	3+	Graduate	No	81000	0	360000	
582	Male	No	0	Graduate	No	1836	33837	90000	
601	Female	No	3+	Graduate	No	416	41667	350000	

1 przypadek dotyczy kredytu dla żonatego mężczyzny z terminem spłaty wynoszącym 360 miesięcy, na sumę 360 tysięcy. W przypadku 2 kolejnych wniosków, mimo dużych dochodów współwnioskodawców, kredyt również nie został przyznany. Mimo, że wydaje się to nieintuicyjne, to pokazywać może jednak ważne zależności - w 1 przypadku o nieprzyznaniu kredytu prawdopodobnie zadecydował brak pozytywnej historii kredytowej, podczas gdy w 2 kolejnych decydujące mogły być stosunkowo niskie dochody głównego wnioskodawcy i brak ślubu. Równoznaczne jest to z tym, że współwnioskodawcą w takim razie nie mógł być ani mąż (3 z przedstawionych obserwacji), ani żona (2ga), podczas gdy to właśnie taka struktura jest ‘najmilej’ widziana przez banki. Ze względu na potencjalne zależności jakie mogą być widoczne w tych obserwacjach, a także bardzo małą liczbę obserwacji (jedynie sześćset kilkanaście), decyduję się na pozostawienie tych obserwacji.

W zbiorze danych zawarto takie zmienne jak LoanAmount i Loan_Amount_Term. Na podstawie wcześniejszych statystyk można zauważyć, że różnica między 3 kwantylem, a wartością max LoanAmount to ponad pół miliona, gdzie średnia wartość kredytów oscyluje w granicach około 150 000. Widzimy więc podobny problem co wcześniej ze zmiennymi dt. przychodu wnioskodawców. Dobrym pomysłem może być więc sprawdzenie, ile wynosi wartość kredytu w przeliczeniu na 1 miesiąc jego spłaty. W celu zobrazowania tej cechy stworzę nową zmienną, która będzie oczywiście liczona jako LoanAmount / Loan_Amount_Term. Z tego względu zmienne te nie pozostaną w niezmienionej formie. Decyduję się jednak na ich modyfikację. Zmienna opisująca długość okresu spłaty posłuży do stworzenia 2 kolejnych zmiennych - Short_Term i Long_Term. Długość okresu spłaty jest kluczowym aspektem kredytów hipotecznych, który pociąga za sobą różne skutki. Kredyty długoterminowe są bardziej opłacalne dla banków, ponieważ dają większy zysk (wyższe raty). Ponadto, łatwiej spłacać niższe miesięczne raty, więc można zakładać, że kredytobiorca ma wtedy większe szanse na spłacenie kredytu. Wiąże się z tym jednak ryzyko wynikające np. z faktu możliwych zmian w życiu kredytobiorcy czy różnych wydarzeń losowych (narodziny dziecka, rozwód czy sytuacje losowe wpływające na majątek kredytobiorcy jak powodzie/pożary, itd.). Dłuższy okres kredytowy więc powinien pozwalać na uzyskanie stosunkowo łatwiej zdolności kredytowej, podczas gdy krótki przeciwnie - będzie go trudniej spłacić ze względu na większe raty. Stąd więc stworzę zmienne Short_Term - informującą czy okres spłaty jest mniejszy lub równy 15 lat (180 miesięcy) oraz Long_Term - czy okres spłaty jest większy niż 30 lat (360 miesięcy). Analogicznie dla LoanAmount: LowAmountCredit - czy kredyt na poniżej 100 000 (wartość 1 kwantyla) oraz czy kredyt na powyżej 200 000 (około 35 000 więcej niż wynosi 3 kwantyl).

Code

2.5 Zależności między X i Y.

Dla zmiennych objaśniających kategorycznych stworzone zostaną tabelki obrazujące ich rozkład na tle zmiennej objaśnianej, a dla zmiennych numerycznych przedstawione zostaną wykresy.

Zmienne kategoryczne:

Gender

	N	Y
Female	38	77
Male	154	345

Married

	N	Y
No	79	136
Yes	113	286

Education

	N	Y
Graduate	140	340
Not Graduate	52	82

Self_Employed

	N	Y
No	164	364
Yes	28	58

Credit_History

	N	Y
No	82	7
Yes	110	415

Dependents

	N	Y
0	109	242
1	38	67
2	26	80
3+	19	33

Property_Area

	N	Y
Rural	69	110
Semiurban	54	179
Urban	69	133

Short_Term

	N	Y
No	174	384
Yes	18	38

Long_Term

	N	Y
No	181	416
Yes	11	6

LowAmountCredit

	N	Y
No	147	324
Yes	45	98

HighAmountCredit

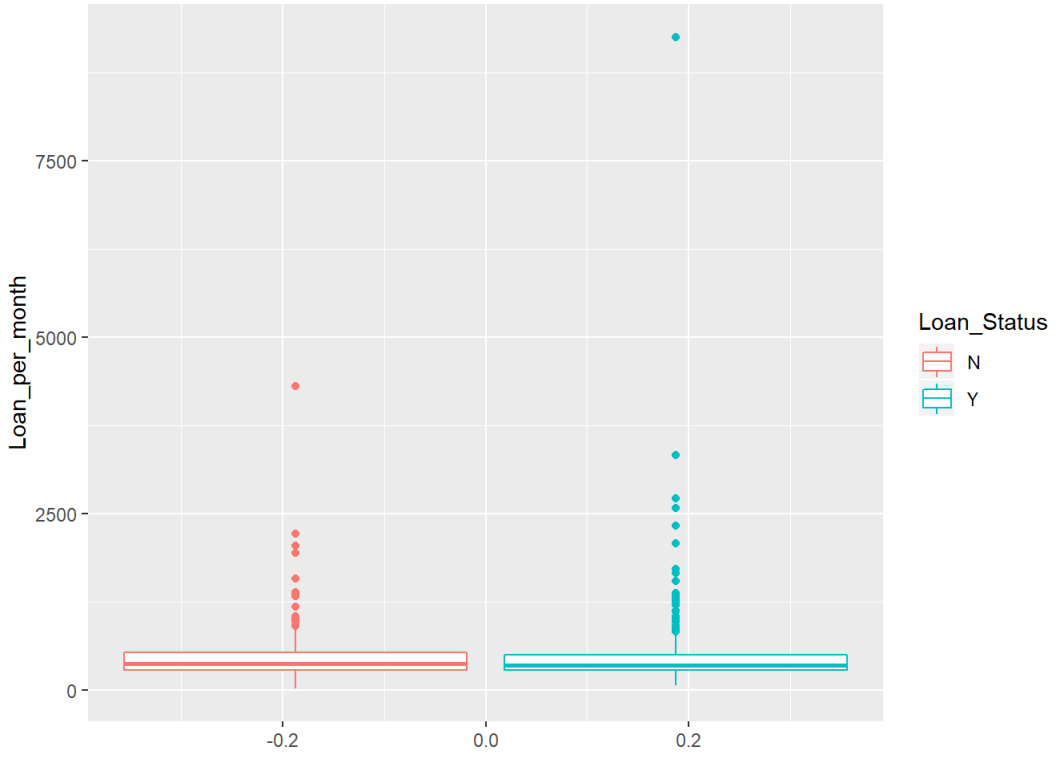
	N	Y
No	160	371
Yes	32	51

Rzut oka na tabelki i krótka analiza pozwala na wyciągnięcie paru wniosków. Najważniejszą zależnością jaką można zobaczyć na podstawie tabel jest fakt, że jeśli historia kredytowa wnioskodawcy nie spełnia wymagań, kredyt prawie na pewno nie zostanie przydzielony (stało sie tak tylko 7 razy na 89 przypadków). Zauważyć można również mniejsze szanse na otrzymanie kredytu dla osób które nie wzięły jeszcze ślubu (w porównaniu do tych które mają to już za sobą) czy też bez bez wykształcenia wyższego (w grupie osób z wykształceniem wyższym na 1 osobę, która nie otrzymała kredytu przypada ok. 2,5 osoby która go otrzymała, podczas gdy w grupie osób bez wykształcenia wyższego proporcje te są mniejsze - około 1 do 1,5). W porównaniu do pozostałych grup w zmiennej Dependents rzadziej kredyty otrzymują wnioskodawcy z obszarów półmiejskich (semiurban). Trochę zaskakuje fakt, że osoby wnioskujące o kredyt długoterminowy rzadziej go otrzymują niż w przypadku wniosków o kredyty krótko- i średnio-terminowe, co jednak można tłumaczyć chociażby większym ryzykiem wynikającym z dłuższego okresu. Podobnie w przypadku wysokości kredytu - widzimy, że trudniej uzyskać (32 osoby na 83) kredyt powyżej 200 000 (w grupie poniżej 160/531).

Zmienne liczbowe:

Wykresy pudełkowe zmiennych dt. przychodów wnioskodawców zostały zaprezentowane wcześniej. Wykres nowej zmiennej ‘Loan_per_month’ prezentuje się następująco:

Code



Między ‘pudełkami’ nie widać większych różnic, jednak zastanawiająca jest obserwacja o najwyższej wartości. Przyjrzyjmy się jej bliżej:

Code

	Gender	Married	Dependents	Education	Self_Employed	Short_Term	Long_Term	LowAmountCredit	HighAmo
498	Male	Yes	0	Graduate	No	Yes	No	No	No

Bardzo zastanawiające jest, że wartość pożyczki w przeliczeniu na miesiąc spłaty byłaby wyższa niż zsumowany dochód wnioskodawców, a kredyt nadal byłby przyznany! Zdecydowano więc o usunięciu tej obserwacji. Być może związane jest to z np. błędem predykcji podczas uzupełniania braków danych. Kolejnym argumentem przemawiającym za usunięciem tej obserwacji jest fakt, że w zbiorze danych jest (oprócz omawianej) tylko 1 obserwacja z podobną sytuacją (zsumowany dochód wnioskodawców < wartość kredytu/msc) i dla tej wnioskodawczyni kredytu nie udzielono.

Code

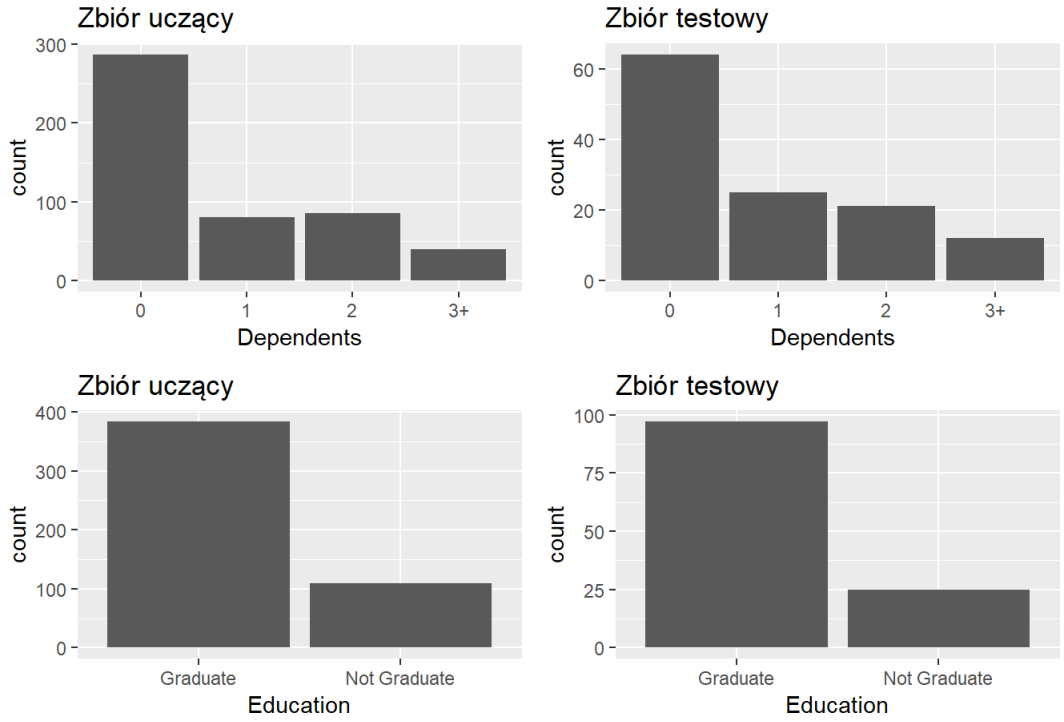
	Gender	Married	Dependents	Education	Self_Employed	Short_Term	Long_Term	LowAmountCredit	HighAmo
263	Female	No	1	Graduate	No	Yes	No	No	No
498	Male	Yes	0	Graduate	No	Yes	No	No	No

3 Zbiór uczący i zbiór testowy

Przed rozpoczęciem pracy nad częścią związaną z tworzeniem modeli, trzeba jeszcze podzielić zbiór danych na uczący i testowy. Podział zbioru pozwoli na sprawdzenie jak model radzi sobie z danymi, na których nie był tworzony oraz pozwoli na porównanie ze sobą wyników poszczególnych modeli. Zbiór zostanie podzielony za pomocą funkcji `createDataPartition` z pakietu `caret`. Dzięki wprowadzeniu jako argument zmiennej objaśnianej, jej proporcje będą podobne w obu zbiorach.

Code

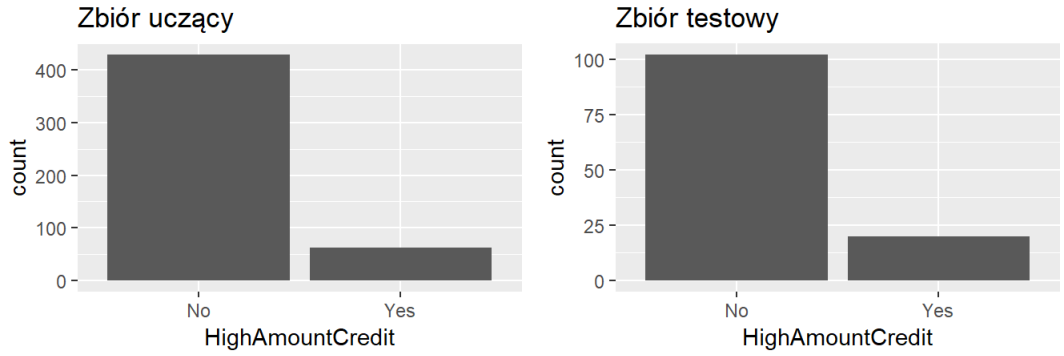
Histogramy zmiennych Dependents i Education w zbiorze uczącym i testowym

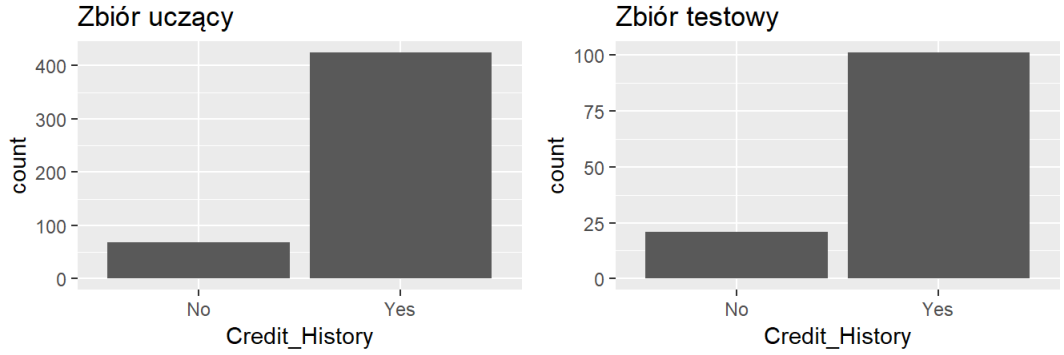


Histogramy zmiennych Self_Employed i Short_Term w zbiorze uczącym i testowym

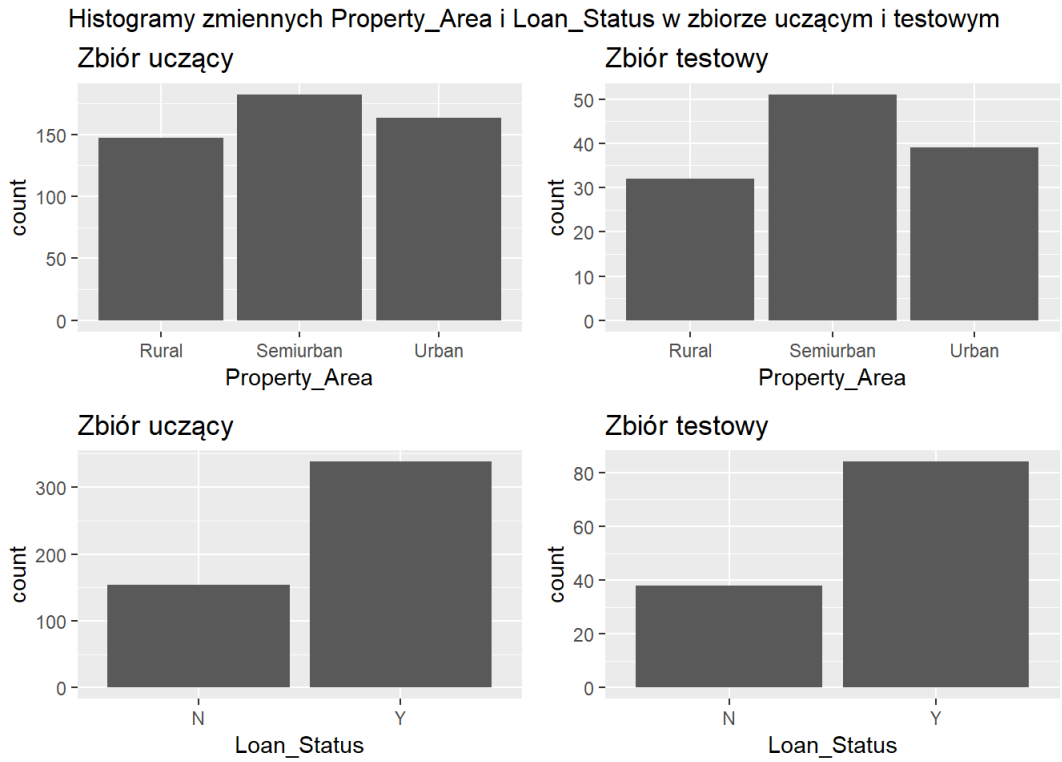
Histogramy zmiennych Long_Term i LowAmountCredit w zbiorze uczącym i testowym

Histogramy zmiennych HighAmountCredit i Credit_History w zbiorze uczącym i testowym





Code



Code

Zbiór uczący

	mean	sd	median	min	max	range
ApplicantIncome	5421.9	5990.9	3787.5	150	81000	80850
CoapplicantIncome	1496.0	2508.5	1007.0	0	33837	33837
Loan_per_month	475.2	537.8	361.1	25	9250	9225

Code

Zbiór testowy

	mean	sd	median	min	max	range
ApplicantIncome	5329.2	6589.6	3896.0	416.0	63337.0	62921.0
CoapplicantIncome	2126.4	4186.3	1404.5	0.0	41667.0	41667.0
Loan_per_month	481.2	393.9	368.1	83.3	2722.2	2638.9

Widzimy, że proporcje zmiennych są mniej więcej zachowane. Oczywiście, zbiory mogłyby być lepiej zbalansowane, ale ciężko o dużo lepszy wynik przy takiej liczbie obserwacji i takim rozrzucie wartości, jeśli chodzi o zmienne liczbowe. Otrzymany podział wydaje się być poprawny, dlatego posłuży on w dalszych częściach pracy.

4. Opis zastosowanych technik i tworzenie modeli

4.1 Metoda naiwna Bayesa

Metoda naiwna Bayesa jest jedną z prostszych klasyfikacyjnych metod uczenia maszynowego, która jednak potrafi w wielu sytuacjach dawać równie dobre rezultaty, co metody bardziej złożone. Polega na przewidywaniu prawdopodobieństwa przynależności obiektu do danej klasy. Technika ta opiera się na twierdzeniu Bayesa, które mówi, że prawdopodobieństwo warunkowe bycia w stanie X, pod warunkiem posiadania własności C jest równe: P(C|X) = P(X|C) * P(C) / P(X). Prawdopodobieństwa użyte we wzorze w łatwy sposób można obliczyć na podstawie danych. P(X|C) jest prawdopodobieństwem, że obiekt w stanie X posiada cechę C np. liczba kobiet o niebieskich oczach / liczba kobiet, P(C) w tym przypadku to prawdopodobieństwo posiadania niebieskich oczu, czyli liczba osób o niebieskich oczach / liczba osób w próbce i P(X) czyli prawdopodobieństwo, że dana osoba z próbki jest kobietą (liczba kobiet / liczba obserwacji w próbie). W problemach wielowymiarowych, po odpowiednich przekształceniach dochodzimy do momentu, w którym wystarczy porównać iloczyny prawdopodobieństw warunkowych P(Ci|X). Tak prosta formuła jest możliwa dzięki założeniu niezależności cech opisujących obiekt X - co w

praktyce nie zawsze jest spełnione (wręcz bardzo często nie jest). Z tego właśnie powodu metoda ta jest nazywana metodą nawiną. Mimo tego, że to założenie nie zawsze jest spełnione metoda ta daje niejednokrotnie wyniki lepsze albo zbliżone od metod bardziej skomplikowanych i złożonych obliczeniowo.

Model tworzony jest za pomocą poniższego kodu. Wykorzystano funkcję `naive_bayes` z pakietu `naivebayes`. Od razu obliczone zostaną też odpowiednie statystyki obrazujące jakość modelu i macierze błędu. Wyniki jakie osiąga dany model przedstawione zostaną w kolejnej części. Z racji tego, że metoda ta działa tylko dla zmiennych kategorycznych, zmienne typu numerycznego pomijam.

Code

4.2 Bagging

Bagging jest algorytmem opartym na drzewach decyzyjnych, którego największą zaletą jest zmniejszenie wariancji wyników. Algorytm baggingu:

- załóżmy, że w naszym zbiorze danych mamy N obserwacji oraz M zmiennych. Ze zbioru dostępnych obserwacji losujemy ze zwracaniem wiele (n) podprób ze zbioru głównego,
- każda z wylosowanych próbek jest podobna (w końcu wylosowano je z tego samego zbioru), lecz nie identyczna. Dla każdej z wylosowanych próbek tworzymy drzewo decyzyjne,
- dla nowych danych, dla których chcemy otrzymać predykcję, wyniki uśredniamy (dla drzew regresyjnych) lub stosujemy ‘głosowanie’ - dla drzew klasyfikacyjnych, takich jak np. zagadnienie scoringu kredytowego.

Model dla tej techniki stworzony zostanie za pomocą funkcji `bagging` dostępnej w pakiecie `ipred`. Parametr ‘nbagg’ czyli liczbę tworzonych drzew, ustawiono na 100. Parametr ‘ns’ odpowiada za liczbę obserwacji losowanych ze zbioru do nowej próby i ustawiony został na 400. Ponadto, dodano parametr ‘minsplit’, którego wartość wynosi 25. Oznacza to że minimalna liczba obserwacji jaka jest potrzebna do kolejnego podziału węzła w danym drzewie to 25. Parametr ten został dodany ze względu na duże różnice w wynikach jakie osiągano na zbiorze testowym i uczącym - dzięki temu parametrowi drzewa w lepszy sposób generalizują wyniki, co skutkuje lepszą skutecznością predykcji na zbiorze treningowym.

Code

4.3 Regresja logistyczna

Regresja logistyczna jest jedną z metod regresji, czyli metod opisujących relacje między zmienną zależną, a innymi zmiennymi objaśniającymi (niezależnymi). Regresja logistyczna jest specjalną odmianą regresji, znajdującą zastosowanie w problemach, gdy zmienna zależna jest dychotomiczna, czyli przyjmuje tylko 2 wartości(głównie, bo istnieją też inne odmiany). Najczęściej znajduje zastosowanie w takich problemach jak właśnie scoring kredytowy czy diagnozowanie chorób.
Regresja logistyczna bazuje na specyficznej funkcji prawdopodobieństwa - tzw. szansie. Różni się ona od klasycznego prawdopodobieństwa sposobem wyznaczania - nie jest wyliczana jako iloraz sukcesów do wszystkich prób, a jako stosunek prawdopodobieństwa sukcesu do prawdopodobieństwa porażki (p / (1-p)). Omawiany model zakłada, że szansa na zaistnienie pewnego zjawiska jest możliwa do opisania za pomocą funkcji:

\[S(Y=1|X) = e^{(\beta_0 + \beta_1*X)} \]

Po odpowiednich przekształceniach wynikających z 2 powyższych wzorów otrzymujemy wyrażenie:

\[log(\frac{p}{1-p}) = \beta_0 + \beta_1*X \]

zwane logitem (logarytm szansy). Po odwróceniu tej funkcji otrzymujemy wzór krzywej logistycznej:

\[p = \frac{e^{logit(p)}}{1-e^{logit(p)}} \] Po estymacji współczynnikóW beta (najczęściej stosuje się tu metodą największej wiarygodności) i podstawieniu ich do powyższych rownań otrzymujemy prawdopodobieństwo przynależności obiektu do danej klasy.

Model tworzony jest za pomocą funkcji `glm` z pakietu `stats`. Aby zmienne były w formacie, który jest odpowiedni dla funkcji glm, użyto funkcji `dummy.data.frame` (pakiet `dummies`) żeby zbudować odpowiedni zbiór danych. Początkowy model, zawierający wszystkie zmienne prezentuje się następująco:

Code

```
##
## Call:
## glm(formula = Loan_Status ~ ., family = "binomial", data = train.data.dummy[,
##     -c(1, 3, 5, 9, 11, 13, 15, 17, 19, 21, 23)])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3978  -0.3475   0.4888   0.7136   2.5405
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -3.139e+00  6.114e-01  -5.133 2.85e-07 ***
## GenderMale               8.551e-02  3.323e-01   0.257   0.7969
## MarriedYes               4.712e-01  2.937e-01   1.604   0.1086
## Dependents1             -5.715e-01  3.333e-01  -1.714   0.0865 .
## Dependents2              3.227e-01  3.760e-01   0.858   0.3908
## `Dependents3+`           7.754e-01  5.504e-01   1.409   0.1589
## `EducationNot Graduate` -4.633e-01  2.926e-01  -1.584   0.1133
## Self_EmployedYes        -9.477e-02  3.375e-01  -0.281   0.7789
## Short_TermYes           -9.862e-02  4.917e-01  -0.201   0.8410
## Long_TermYes            -1.606e+00  6.932e-01  -2.317   0.0205 *
## LowAmountCreditYes      -3.481e-01  3.014e-01  -1.155   0.2480
## HighAmountCreditYes     -1.031e+00  4.125e-01  -2.499   0.0124 *
## Credit_HistoryYes        3.963e+00  4.898e-01   8.090 5.95e-16 ***
## Property_AreaSemiurban   1.265e+00  3.123e-01   4.052 5.09e-05 ***
## Property_AreaUrban       4.113e-01  2.936e-01   1.401   0.1613
## ApplicantIncome          2.576e-05  2.392e-05   1.077   0.2815
## CoapplicantIncome       -6.981e-05  4.569e-05  -1.528   0.1266
## Loan_per_month          -9.957e-05  2.286e-04  -0.435   0.6632
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 611.54  on 491  degrees of freedom
## Residual deviance: 439.46  on 474  degrees of freedom
## AIC: 475.46
##
## Number of Fisher Scoring iterations: 5
```

Finalny model (otrzymany przy wykorzystaniu metody krokowej wstecz), zawierający tylko istotne zmienne przedstawia się następująco:

Code

```
##
## Call:
## glm(formula = Loan_Status ~ ., family = "binomial", data = train.data.dummy[,
##     -c(1:3, 5:9, 10:21, 23, 25:28)])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1277  -0.3979   0.4686   0.7268   2.4953
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -3.0678     0.4813  -6.373 1.85e-10 ***
## MarriedYes               0.5716     0.2369   2.413 0.015836 *
## Credit_HistoryYes        3.6926     0.4529   8.152 3.57e-16 ***
## Property_AreaSemiurban   0.9573     0.2619   3.655 0.000257 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 611.54  on 491  degrees of freedom
## Residual deviance: 464.13  on 488  degrees of freedom
## AIC: 472.13
##
## Number of Fisher Scoring iterations: 5
```

Za istotne w modelu zostały uznane zmienne dt. bycia po ślubie, posiadania odpowiedniej historii kredytowej oraz (co ciekawe) mieszania na obszarze półmiejskim.

Odpowiednie metryki potrzebne do porównania modeli obliczane są następująco:

Code

4.4 Boosting

Boosting jest kolejnym rozszerzeniem algorytmu drzew decyzyjnych. W swoim algorytmie działania jest bardzo podobny do baggingu. Różni się tym, że każde kolejne drzewo budowane jest za pomocą próby, w której większe szanse na wylosowanie mają obserwacje wcześniej błędnie przewidziane, a mniejsze - te przewidziane poprawnie. Dzięki temu model ‘uczy się na błędach’, a także lepiej klasyfikuje przypadki nieoczywiste czy rzadziej występujące.

Model stworzony zostanie za pomocą funkcji `gbm` z pakietu o tej samej nazwie. Finalny model wybieram na podstawie wcześniejszego dopasowywania parametrów - po stworzeniu tabeli z odpowiednimi parametrami, buduję kolejne modele i sprowadzam jakie dadzą wyniki. Finalnie wybieram model cechujący się największą dokładnością. Wyniki przedstawione na zdjęciu, ze względu na długi czas wykonywania pętli budującej modele dla odpowiednich zestawów parametrów (około 25-30 min). Kod załączony poniżej.

Code

```
> head(arrange(search.grid, desc(model.acc)),10)
   shrinkage interaction.depth n.minobsinnode n.trees model.acc
1      0.001                 4              7    1000 0.7950820
2      0.001                 4             11    1000 0.7950820
3      0.010                 9              7   10000 0.7950820
4      0.010                 9              7    4000 0.7868852
5      0.010                 7              7    5500 0.7868852
6      0.010                 9              7    5500 0.7868852
7      0.010                 7              7    7000 0.7868852
8      0.010                 9              7    7000 0.7868852
9      0.010                 7              7    8500 0.7868852
10     0.010                 9              7    8500 0.7868852
>
```

Wyniki

Dla najlepszej kombinacji parametrów buduję finalny model, który będzie porównywany z innymi.

Code

5. Prezentacja wyników dla poszczególnych metod

Dla każdej z wykorzystanych metod przedstawione zostaną:

- macierz błędu dla zbioru zarówno treningowego jak i uczącego, wraz z wieloma obliczonymi metrykami takimi jak dokładność czy czułość (za pomocą funkcji `confusionMatrix()` z pakietu `caret`),
- AUC, dla obu zbiorów,
- wykres krzywej ROC dla zbioru testowego

5.1 Metoda naiwna Bayesa

Wynik funkcji confusionMatrix dla zbioru uczącego

Code

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  87  40
##          Y  67 298
##
##                Accuracy : 0.7825
##                  95% CI : (0.7434, 0.8182)
##     No Information Rate : 0.687
##     P-Value [Acc > NIR] : 1.535e-06
##
##                   Kappa : 0.469
##  Mcnemar's Test P-Value : 0.01195
##
##             Sensitivity : 0.8817
##             Specificity : 0.5649
##          Pos Pred Value : 0.8164
##          Neg Pred Value : 0.6850
##              Prevalence : 0.6870
##          Detection Rate : 0.6057
##    Detection Prevalence : 0.7419
##       Balanced Accuracy : 0.7233
##
##        'Positive' Class : Y
##
```

Wynik funkcji confusionMatrix dla zbioru testowego

Code

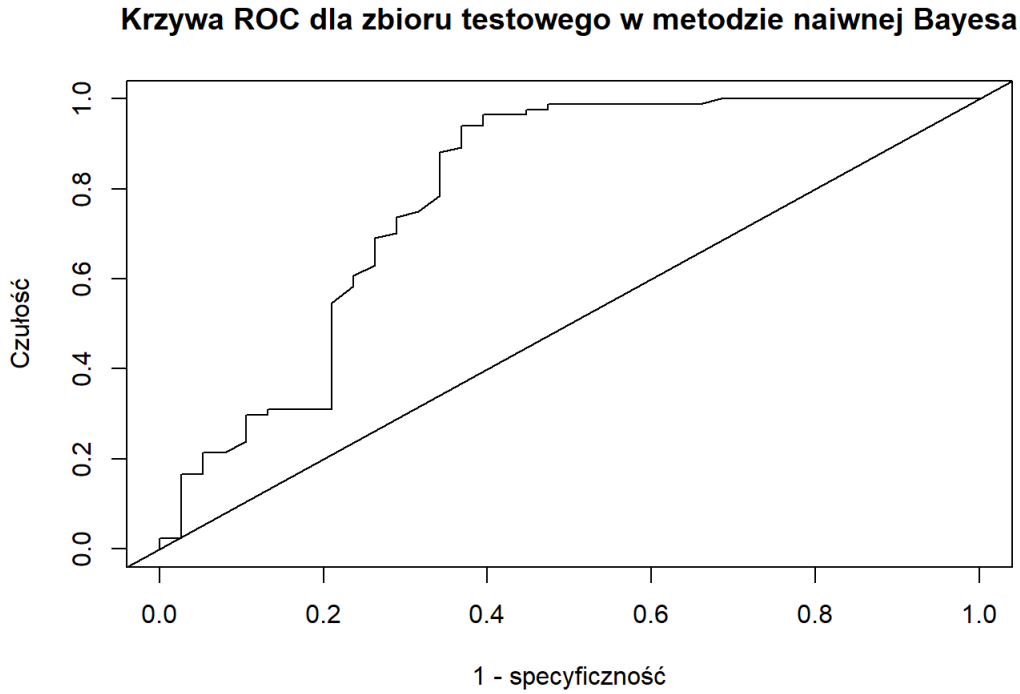
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 24  7
##          Y 14 77
##
##                Accuracy : 0.8279
##                  95% CI : (0.749, 0.8902)
##     No Information Rate : 0.6885
##     P-Value [Acc > NIR] : 0.0003571
##
##                   Kappa : 0.5774
##  Mcnemar's Test P-Value : 0.1904303
##
##             Sensitivity : 0.9167
##             Specificity : 0.6316
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.7742
##              Prevalence : 0.6885
##          Detection Rate : 0.6311
##    Detection Prevalence : 0.7459
##       Balanced Accuracy : 0.7741
##
##        'Positive' Class : Y
##
```

Code

AUC dla obu zbiorów

AUC.ucz	AUC.test
0.8	0.79

Code



Model osiąga o około 2punkty procentowe lepszą skuteczność na zbiorze testowym, auc dla obu zbiorów wynosi około 0,78. Czułość i specyficzność mimo zastosowania punktu odcięcia równego ilorazowi przyznanych kredytów do wszystkich obserwacji, znacznie się różnią.

5.2 Bagging

Wynik funkcji confusionMatrix dla zbioru uczącego

Code

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  64   6
##          Y  90 332
##
##                Accuracy : 0.8049
##                  95% CI : (0.7671, 0.839)
##     No Information Rate : 0.687
##     P-Value [Acc > NIR] : 2.793e-09
##
##                   Kappa : 0.4672
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9822
##             Specificity : 0.4156
##          Pos Pred Value : 0.7867
##          Neg Pred Value : 0.9143
##              Prevalence : 0.6870
##          Detection Rate : 0.6748
##    Detection Prevalence : 0.8577
##       Balanced Accuracy : 0.6989
##
##        'Positive' Class : Y
##
```

Wynik funkcji confusionMatrix dla zbioru testowego

Code

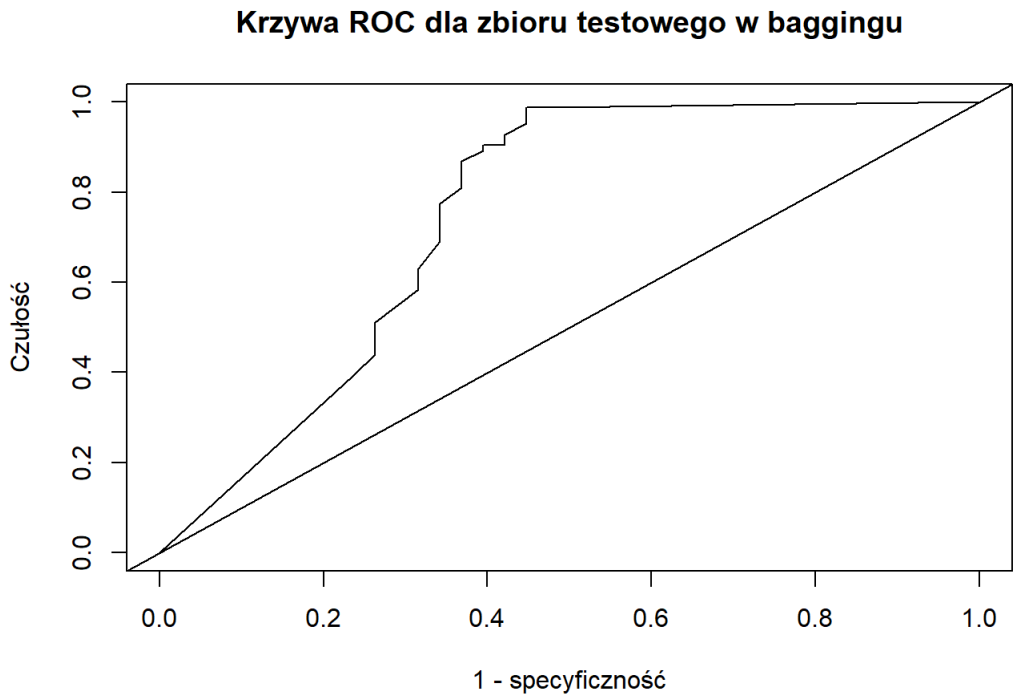
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 21  1
##          Y 17 83
##
##                Accuracy : 0.8525
##                  95% CI : (0.7769, 0.9102)
##     No Information Rate : 0.6885
##     P-Value [Acc > NIR] : 2.502e-05
##
##                   Kappa : 0.6112
##  Mcnemar's Test P-Value : 0.000407
##
##             Sensitivity : 0.9881
##             Specificity : 0.5526
##          Pos Pred Value : 0.8300
##          Neg Pred Value : 0.9545
##              Prevalence : 0.6885
##          Detection Rate : 0.6803
##    Detection Prevalence : 0.8197
##       Balanced Accuracy : 0.7704
##
##        'Positive' Class : Y
##
```

Code

AUC dla obu zbiorów

AUC.ucz	AUC.test
0.9	0.75

Code



W przypadku baggingu model działa aż o 5 punktów procentowych lepiej na zbiorze testowym niż uczącym, a jego skuteczność to trochę ponad 85%. Zastanawia fakt, że pod względem AUC, na zbiorze testowym osiągnięto natomiast znacznie gorszy wynik, około 0,15 gorzej niż w przypadku zbioru uczącego. Również w tym modelu widzimy jeszcze większą różnicę jeśli chodzi o czułość i specyficzność, mimo zastosowania wspomnianego wcześniej punktu odcięcia (w kolejnych 2 modelach również był on oczywiście zastosowany).

5.3 Regresja logistyczna

Wynik funkcji confusionMatrix dla zbioru uczącego

Code

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  91  68
##          Y  63 270
##
##                Accuracy : 0.7337
##                  95% CI : (0.6923, 0.7723)
##     No Information Rate : 0.687
##     P-Value [Acc > NIR] : 0.01345
##
##                   Kappa : 0.3863
##  Mcnemar's Test P-Value : 0.72673
##
##             Sensitivity : 0.7988
##             Specificity : 0.5909
##          Pos Pred Value : 0.8108
##          Neg Pred Value : 0.5723
##              Prevalence : 0.6870
##          Detection Rate : 0.5488
##    Detection Prevalence : 0.6768
##       Balanced Accuracy : 0.6949
##
##        'Positive' Class : Y
##
```

Wynik funkcji confusionMatrix dla zbioru testowego

Code

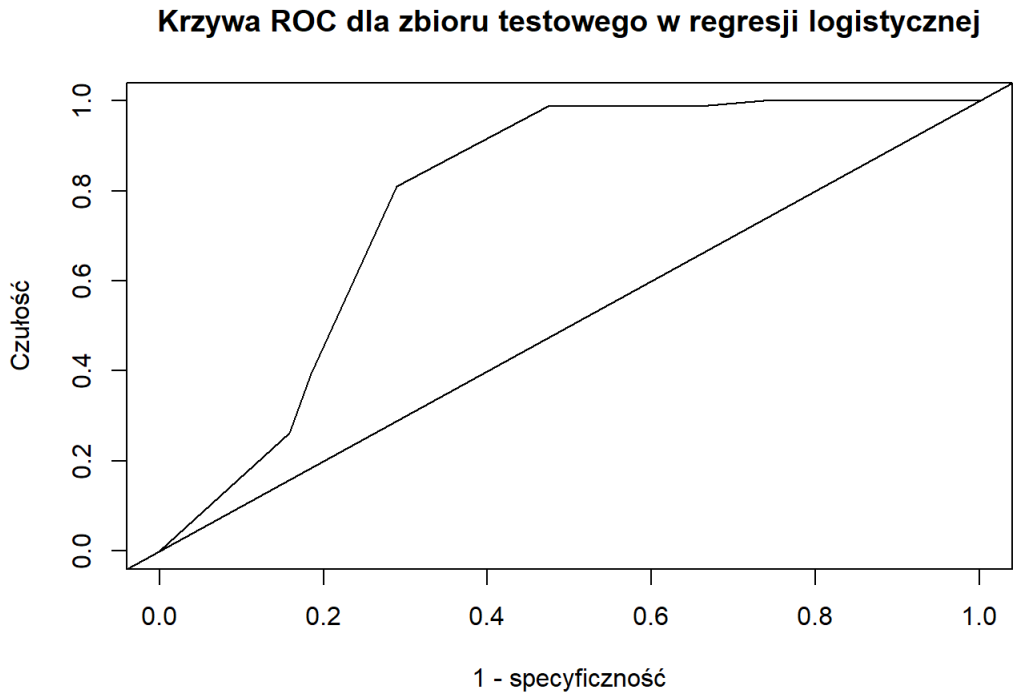
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 27 16
##          Y 11 68
##
##                Accuracy : 0.7787
##                  95% CI : (0.6946, 0.8488)
##     No Information Rate : 0.6885
##     P-Value [Acc > NIR] : 0.01782
##
##                   Kappa : 0.502
##  Mcnemar's Test P-Value : 0.44142
##
##             Sensitivity : 0.8095
##             Specificity : 0.7105
##          Pos Pred Value : 0.8608
##          Neg Pred Value : 0.6279
##              Prevalence : 0.6885
##          Detection Rate : 0.5574
##    Detection Prevalence : 0.6475
##       Balanced Accuracy : 0.7600
##
##        'Positive' Class : Y
##
```

Code

AUC dla obu zbiorów

AUC.ucz	AUC.test
0.77	0.78

Code



Około 4,5 punkty procentowe lepszy wynik na zbiorze uczącym jest bardziej intuicyjnym rezultatem. Jednak skuteczność modelu jest mniejsza niż wcześniejszych. AUC natomiast na podobnym poziomie w obu zbiorach, około 0,77-0,78. Czułość i specyficzność na zbiorze testowym różnią się o około 0.1 (na zbiorze uczącym jest to 0.2).

5.4 Boosting

Wynik funkcji confusionMatrix dla zbioru uczącego

Code

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  88  21
##          1  66 317
##
##                Accuracy : 0.8232
##                  95% CI : (0.7865, 0.8559)
##     No Information Rate : 0.687
##     P-Value [Acc > NIR] : 5.159e-12
##
##                   Kappa : 0.5533
##  Mcnemar's Test P-Value : 2.390e-06
##
##             Sensitivity : 0.9379
##             Specificity : 0.5714
##          Pos Pred Value : 0.8277
##          Neg Pred Value : 0.8073
##              Prevalence : 0.6870
##          Detection Rate : 0.6443
##    Detection Prevalence : 0.7785
##       Balanced Accuracy : 0.7546
##
##        'Positive' Class : 1
##
```

Wynik funkcji confusionMatrix dla zbioru testowego

Code

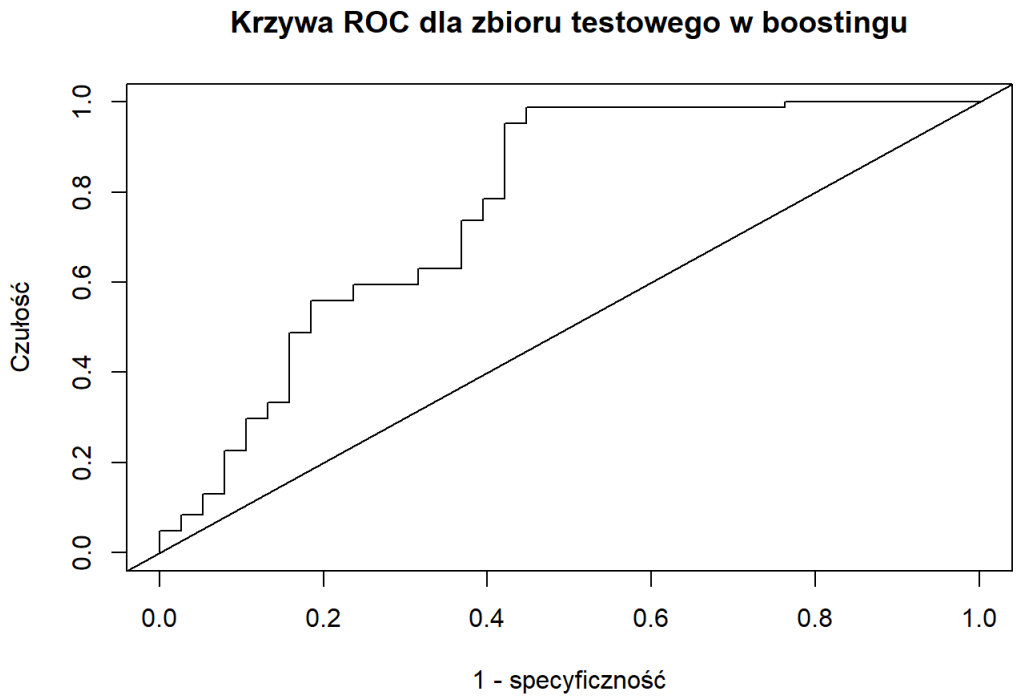
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 22  9
##          1 16 75
##
##                Accuracy : 0.7951
##                  95% CI : (0.7125, 0.8628)
##     No Information Rate : 0.6885
##     P-Value [Acc > NIR] : 0.00585
##
##                   Kappa : 0.4969
##  Mcnemar's Test P-Value : 0.23014
##
##             Sensitivity : 0.8929
##             Specificity : 0.5789
##          Pos Pred Value : 0.8242
##          Neg Pred Value : 0.7097
##              Prevalence : 0.6885
##          Detection Rate : 0.6148
##    Detection Prevalence : 0.7459
##       Balanced Accuracy : 0.7359
##
##        'Positive' Class : 1
##
```

Code

AUC dla obu zbiorów

AUC.ucz	AUC.test
0.88	0.77

Code



Skuteczność na poziomie około 80% w obu zbiorach (w uczącym 2 punkty procentowe lepiej), natomiast AUC ze znaczną różnicą - wynik w zbiorze uczącym to 0.88, podczas gdy zbiór testowy legitymuje się AUC na poziomie 0.77. Podobnie jak wcześniej w modelu bagging oraz klasyfikatorze Bayesa, ma miejsce duża rozbieżność jeśli chodzi o czułość i specyficzność.

6. Porównanie modeli, wybór najlepszej metody, wnioski z

badania i pomysły na otrzymanie lepszych wyników.

Skuteczność i AUC otrzymane dla wszystkich modeli prezentują się następująco:

Code

Tabela z wynikami poszczególnych modeli

	AUC.ucz	ACC.ucz	AUC.test	ACC.test
BAYES	0.805	78.252	0.787	82.787
BAGGING	0.900	80.488	0.745	85.246
REGRESJA LOGISTYCZNA	0.771	73.374	0.782	77.869
BOOSTING	0.881	82.317	0.766	79.508

Na zbiorze uczącym zdecydowanie najbardziej odstaje regresja logistyczna - zarówno pod względem AUC i dokładności dla tego modelu otrzymano najgorsze wyniki. Pod względem AUC najlepiej wypadł model bagging, nieznacznie słabiej boosting. Jeśli chodzi o skuteczność predykcji to ta jest tym razem nieznacznie wyższa dla metody baggingu, niż dla boostingu.
Na zbiorze testowym natomiast, najlepszym wynikiem jeśli chodzi o skuteczność predykcji legitymuje się bagging - 85% to ok. 3% więcej niż 2 wynik jaki należy do modelu bayesowskiego. Ponownie najsłabiej wypada regresja logistyczna. Pod względem AUC modele nie wypadają szczególnie przekonująco - żaden nie osiągnął bariery 0.8, a najlepsze wyniki dają metody Bayesa oraz bagging.
Podsumowując, wybór najlepszej metody jest kwestią nieoczywistą. Wydaje się, że najlepsze wyniki osiągnęły metody baggingu i Bayesa. Klasyfikator Bayesowski legitymuje się gorszą dokładnością predykcji, ale lepszym AUC. Wypada wyraźnie słabiej na zbiorze uczącym, jednak jest on mniej ważny. Ponadto, wyniki dla obu metod mogą być lekko niepokojące - na zbiorze uczącym AUC jest zdecydowanie wyższe niż na testowym, podczas gdy skuteczność jest lepsza na zbiorze testowym. Wydaje się jednak, że przewaga jaką ma model bagging jeśli chodzi o skuteczność predykcji pozwala na zakwalifikowanie go jako modelu najlepszego z rozpatrywanych.

Wyniki otrzymane w ramach badania można by poprawić pracując nad:

- optymalnym punktem odcięcia, który zapewni lepszy balans pomiędzy czułością, a specyficznością,
- lepszym uzupełnieniem braków, dzięki wykorzystaniu bardziej zaawansowanych metod (np. predictive mean matching),
- bardziej rozbudowaną częścią dotycząca szukania optymalnych parametrów dla danych modeli,
- w przypadku metody regresji logistycznej, można posłużyć się bardziej zaawansowaną metodą wyboru zmiennych objaśniających,
- zastosowaniem zmiennych numerycznych w badaniu poprzez zmianę ich postaci (szeregi rozdzielcze),
- i przede wszystkim, powiększając zbiór danych. 600 obserwacji to dość mało, przez co otrzymane wyniki są mniej miarodajne i po prostu słabsze,
- pewnym rozwiązaniem tego problemu mogłoby być użycie jednej z technik walidacji krzyżowej, która lepiej sprawdza się przy małych zbiorach danych, od standardowego podziału zbioru,
- być może zmienne, które zostały stworzone na potrzeby badania wcale nie były lepsze od wcześniej istniejących. Na pewno także możliwe jest dalsze eksplorowanie tego zagadnienia i wymyślenie kolejnych zmiennych, które mogą przynieść pożądane (lepsze) rezultaty.