Badanie jakości filmów

Celem naszej pracy jest dokonanie analizy czynników wpływających na jakość filmu. Na jej podstawie będziemy chcieli określić czy dany film jest warty obejrzenia czy też nie. Z góry należy podkreślić, że praca nie ma na celu ocenić sukcesu nowo wydanego filmu, jest jedynie (podpartą danymi) wskazówką do wybrania odpowiedniego seansu na wieczór.

Krok 1. Zbieranie danych

Dane którymi będziemy się posługiwać zostały pobrane ze strony kaggle.com. Stronę z danymi można znaleźć pod linkiem https://www.kaggle.com/datasets/carolzhangdc/imdb-5000-movie-dataset . Dane należące do zbioru danych zostały bezpośrednio ściągnięte z oficjalnej strony https://www.imdb.com/ (najpopularniejszej strony służącej do oceny filmów) i reprezentują ponad 5000 losowo wybranych filmów. Do aktywności zbierania danych można również zaliczyć samodzielne uzupełnianie niektórych rekordów na podstawie danych z innych źródeł internetowych, jako że w zbiorze występowały pewne braki.

Krok 2. Eksploracja i przygotowanie danych

Po wgraniu danych do środowiska R, można zauważyć, że mamy ponad 5000 rekordów. Na samym początku można było zwrócić uwagę, że w zbiorze występują duplikaty, zatem od razu je usunęliśmy. Następnie postanowiliśmy usunąć kolumny, które, naszym zdaniem, nie będą miały dużego wpływu na analizę jakości filmu (m.in. nazwiska i imiona aktorów oraz reżyserów). Dla zmiennych *'gross'* oraz *'budget'* występowały spore braki danych. Postanowiliśmy, że odpowiednim rozwiązaniem będzie usunięcie rekordów z brakami w tych cechach, jako że uzupełnienie brakujących danych byłoby ciężkie technicznie, oraz uzupełnienie nieodpowiednimi danymi mogłoby zaburzyć model. Również zastąpienie braków średnią, miarą bądź innymi miarami będzie złym pomysłem zważając na rozkład tych cech zaprezentowany poniżej

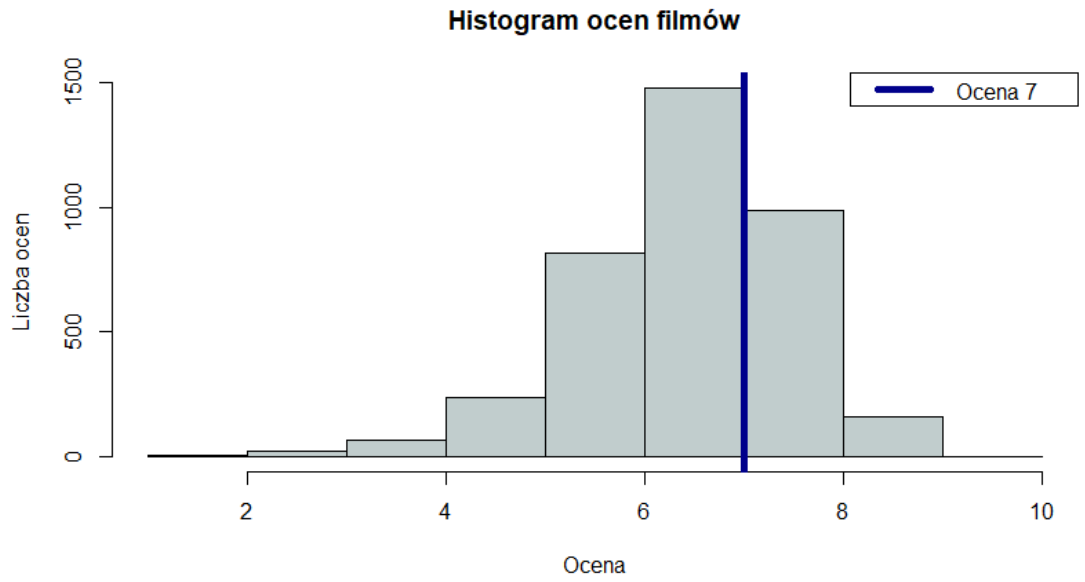
```
> summary(movies_important$gross)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
      162   6564426  27277055  50231378  64977162 760505847

> summary(movies_important$budget)
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
       218    9550000   24000000   41514975   50000000 4200000000
```

Taką samą procedurę zastosowaliśmy dla zmiennych określających ilość polubień na stronach facebookowych aktorów oraz reżyserów. Po wstępnym czyszczeniu danych zostało nam nieco ponad 3800 obserwacji.

Następnie przeszliśmy do uzupełniania danych. Dla cech *'color'*, *'facenumber_in_poster'* (liczba twarzy na plakacie promującym film), *'language'* liczba brakujących rekordów była na tyle mała, że byliśmy w stanie samodzielnie uzupełnić braki. W zmiennej *'content rating'* opisującej kategorię wiekową filmu występowały naprzemiennie przestarzałe oraz nowe oznaczenia na kategorie wiekową, zatem wszystkie zostały zamienione na nowe odpowiedniki. Dodatkowo wszelkie nietypowe wpisy zostały zastąpione nazwą *'Not_Rated'*. W związku z tym, że jeden film może wpasowywać się pod więcej niż jeden gatunek (co również pokazuje nasz zbiór danych, dokładniej kolumna *'genres'*), stworzyliśmy dodatkowe kolumny odpowiadające każdemu z gatunków. Zamiana była oparta na klasycznym podejściu, zatem przypisaniu kolumnie odpowiadającej gatunkowi filmu liczby 1, jeżeli gatunek występował w początkowej kolumnie '*genres'* oraz 0, jeżeli nie występował. Za pomocą tego zabiegu ciąg znaków zastąpiliśmy sekwencją zer i jedynek, lepiej interpretowanym dla modelu. Na koniec, jako że w zbiorze występowały wejścia ze spacjami, aby uniknąć potencjalnego problemu, zamieniliśmy je na znak podkreślenia.

Postanowiliśmy, że podejdziemy do problemu, jako do problemu klasyfikacji, zatem zmienną przewidywaną (*'imdb_score'*), która przyjmuje wartości od 0 do 10, podzieliliśmy na dwie grupy, oceny mniejsze od liczby 7, oznaczone jako filmy gorszej jakości oraz oceny większe bądź równe 7, oznaczone jako filmy dobrej jakości. Podział został dokonany na podstawie naszej intuicji, zebranych informacji z internetu oraz analizy histogramu, który tylko utwierdził nas w przekonaniu, że podział jest adekwatny (można powiedzieć z góry, że filmów gorszej jakości jest więcej niż dobrych produkcji).



Rysunek: Histogram oceny filmów ze strony imdb.com

Krok 3. Budowa, ocena modelu i dopracowanie modelu

3.1 Metoda najbliższych sąsiadów - budowa modelu

Pierwszym naszym pomysłem było sklasyfikowanie danych za pomocą metody k-NN. Uznaliśmy, że ta metoda będzie odpowiednia, jako że ten algorytm opiera się na podobieństwie między obserwacjami, zatem w naszym przypadku będzie analizować podobieństwa między cechami filmów i na tej podstawie sugerować ocenę na podstawie filmów o podobnych własnościach co jest zgodne z intuicją. Dodatkowo algorytm nie zakłada żadnego rozkładu danych, co dodatkowo utwierdza w przekonaniu, że zastosowanie go jest poprawnym zabiegiem.

Dla tej metody postanowiliśmy dodatkowo pogrupować cechy *'country'* oraz *'language'* jako, że posiadają one wiele różnych wartości. Dla cechy *'country'* podzieliliśmy filmy na wyprodukowane w USA oraz powstałe w innych krajach, jako że filmy amerykańskie dominowały, co widać poniżej:

```
> prop.table(table(movies_final_knn$country))
Reszta     USA
  0.21    0.79
```

Dla cechy *'language'* został dokonany podział na filmy anglojęzyczne i nieanglojęzyczne i wygląda on następująco:

```
> prop.table(table(movies_final_knn$language))
anglojęzyczne          inne
        0.953         0.047
```

Dla każdej ze zmiennych jakościowych stworzyliśmy dodatkowe kolumny zerojedynkowe opisujące daną cechę, na tej samej zasadzie jak już wytłumaczony proces dla gatunków filmowych. Finalnie pozostaliśmy bez zmiennych jakościowych, co jest kluczowe przy poprawnym działaniu metody k-NN.

Jako że omawiana metoda opiera się na analizie odległości, kluczowe było przeskalowanie danych. Jako pierwszą metodę skalowania wybraliśmy normalizację min-max, zatem sprowadzenie wszystkich wartości do przedziału [0,1].

Kolejnym krokiem był podziału zbioru danych na zbiory treningowe oraz testowe. Podział jaki ustaliliśmy to 80/20. Spowodowało to, że w zbiorze testowym mieliśmy około 750 obserwacji, a w zbiorze treningowym około 3000. Sprawdziliśmy czy ilość filmów słabych i dobrych w obu zbiorach jest podobna jeśli chodzi o procent całej grupy. Wyniki były odpowiednie, zatem mogliśmy przejść do dalszego etapu.

Jako wyjściową wartość do ilości sąsiadów w metodzie k-NN wybraliśmy liczbę bliską pierwiastkowi z ilości obserwacji w zbiorze testowym. Uznaliśmy, że w naszym przypadku każdy z rodzajów błędów jest tak samo istotny (przy zwykłym wyborze filmu, nie ma większego znaczenia czy nie obejrzymy dobrego filmu czy obejrzymy film gorszej jakości, nie jest to decyzja o wielkich konsekwencjach), więc naszym celem jest tylko przypisanie jak największej ilości filmów do odpowiedniej kategorii (odróżnienie dobrego filmu od słabego). Wyniki pierwszej z analiz nie były satysfakcjonujące.

	predykcja		
rzeczywiste dane	dobre	slabe	Row Total
dobre	**74**	177	251
	0.098	0.234	
slabe	23	**483**	506
	0.030	**0.638**	
Column Total	97	660	757

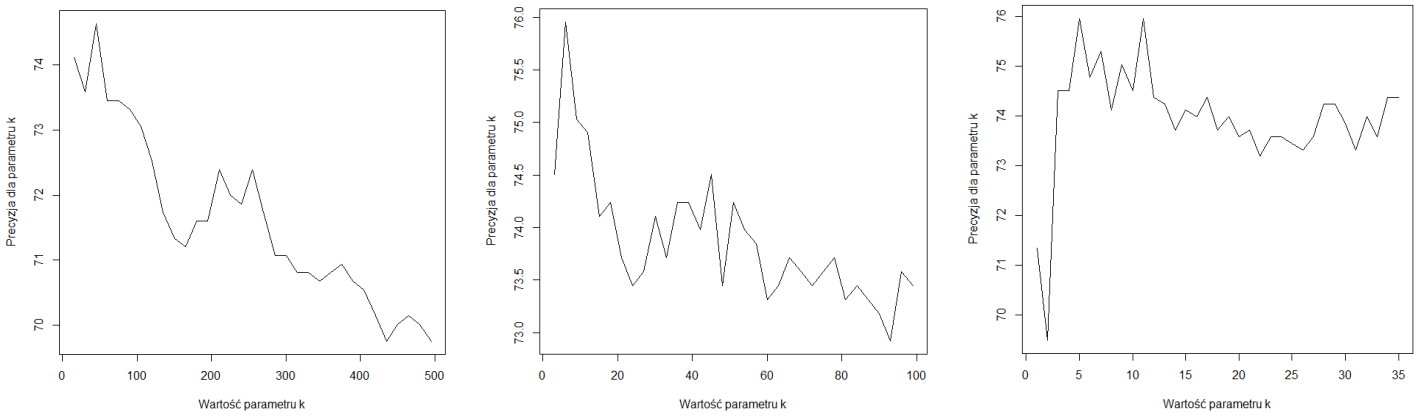
Rysunek: Wyniki pierwszej wersji modelu kNN

3.2 Metoda najbliższych sąsiadów - ocena i dopracowanie modelu

Skuteczność modelu to tylko prawie 74%. Dodajmy, że przy naszych założeniach jeśli chodzi o rodzaj błędów zwykłe uznanie wszystkich filmów za słabe ma 65% skuteczności. W celu poprawy modelu zdecydowaliśmy się odpowiednim algorytmem wyszukać ilość sąsiadów (parametr k) dla której model działa najlepiej. Niechcialiśmy pominąć żadnej możliwości, więc przy pomocy algorytmu sprawdziliśmy, która ilość sąsiadów daje najlepszą skuteczność.

```
temp = NULL
iteracja = NULL
for (i in 1:35){
  print(i)
  knn_model1 <- knn(train=film_trening_knn1, test = film_test_knn1, cl = knn_labels_train1, k=i)
  k=CrossTable(x = knn_labels_test1, y = knn_model1, prop.chisq = FALSE, prop.c = FALSE, prop.r =
FALSE)
  u = ((k$t[1]+k$t[4])/(k$t[1]+k$t[2]+k$t[3]+k$t[4]))*100
  print(u)
  temp[i] = u
  iteracja[i] = i
}
```

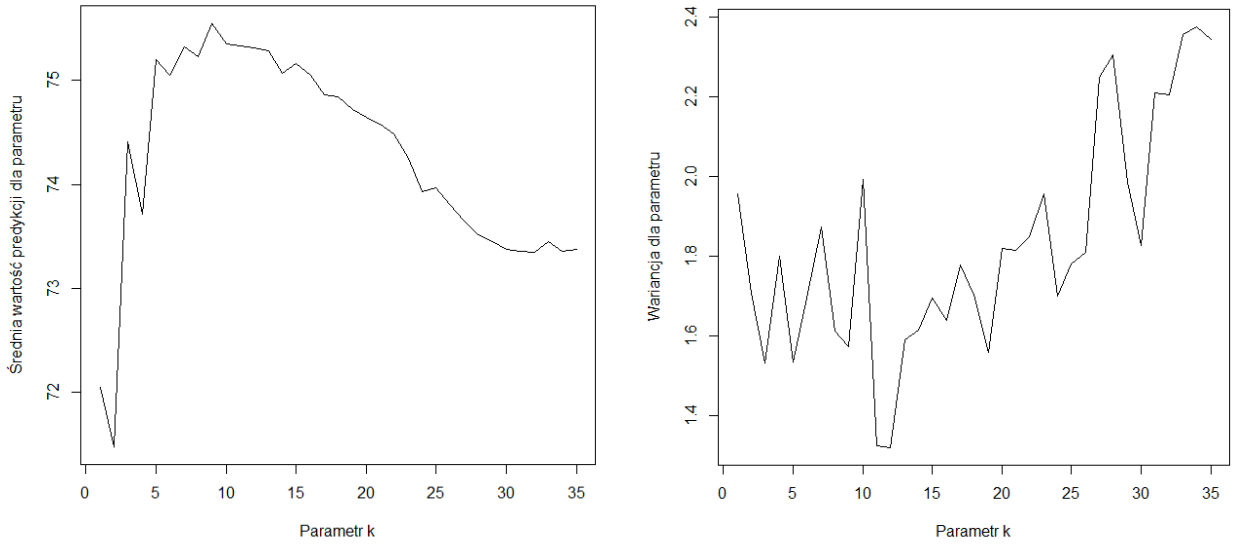
Powyżej został przedstawiony fragment kodu dla już niższych wartości parametru k. Zostały również sprawdzone większe wartości, jednak można było zauważyć, że nie będą one dobrymi kandydatami na optymalny parametr. Poniżej zostały zamieszczone wykresy potwierdzające słuszność wyboru stosunkowo małego parametru k.



Rysunek: Wykresy precyzji algorytmu dla różnych wartości parametru k

Z tej analizy można było dojść do wniosku, że najlepszym parametrem będzie k=11 lub k=5. Mimo poprawy skuteczności, dalej wynosiła ona tylko około 76%.

Aby rozstrzygnąć, który z parametrów wybrać oraz określić, czy nie dopasowaliśmy parametrów zbytnio pod zbiór testowy, zdecydowaliśmy się przeprowadzić więcej prób, losując próbki testowe 50 razy. Zliczając wyniki okazało się, że rzeczywiście średnia jest bliska naszemu wynikowi. Przy tym podejściu okazało się, że najlepsze wyniki daje k = 11. Osiąga średnio trzecią najlepszą średnią (estymowaną) wynoszącą około 76% oraz drugą najniższą wariancję (estymowaną) wynoszącą około 1,3. Większe średnie miały już duże wariancję, więc te parametry zostały wykluczone.



Rysunek: Wykresy średniej predykcji oraz wariancji (dla wylosowanych próbek) dla różnych parametrów k

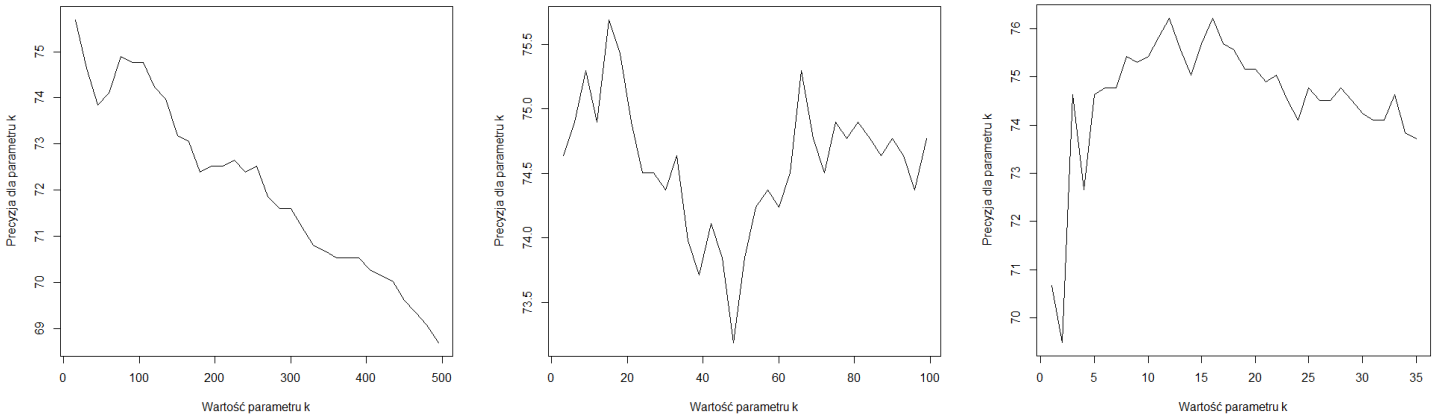
Kolejnym krokiem, który podjęliśmy w celu poprawy jakości modelu była zmiana sposobu skalowania danych. Tym razem skorzystaliśmy z funkcji `scale`, która sprowadza wszystko do zmiennych z rozkładu normalnego N(0,1).

Tym razem pierwszy wynik dla ilości sąsiadów równej pierwiastkowi z ilości danych w zbiorze testowym dał skuteczność blisko 75% co już na wstępie wskazuje, że ten rodzaj skalowania jest lepszy.

		predykcja	
rzeczywiste dane	dobre	slabe	Row Total
dobre	**105**	153	258
	0.139	0.202	
slabe	40	**459**	499
	0.053	**0.606**	
Column Total	145	612	757

Rysunek: Wyniki pierwszej wersji modelu dla standaryzacji z-score

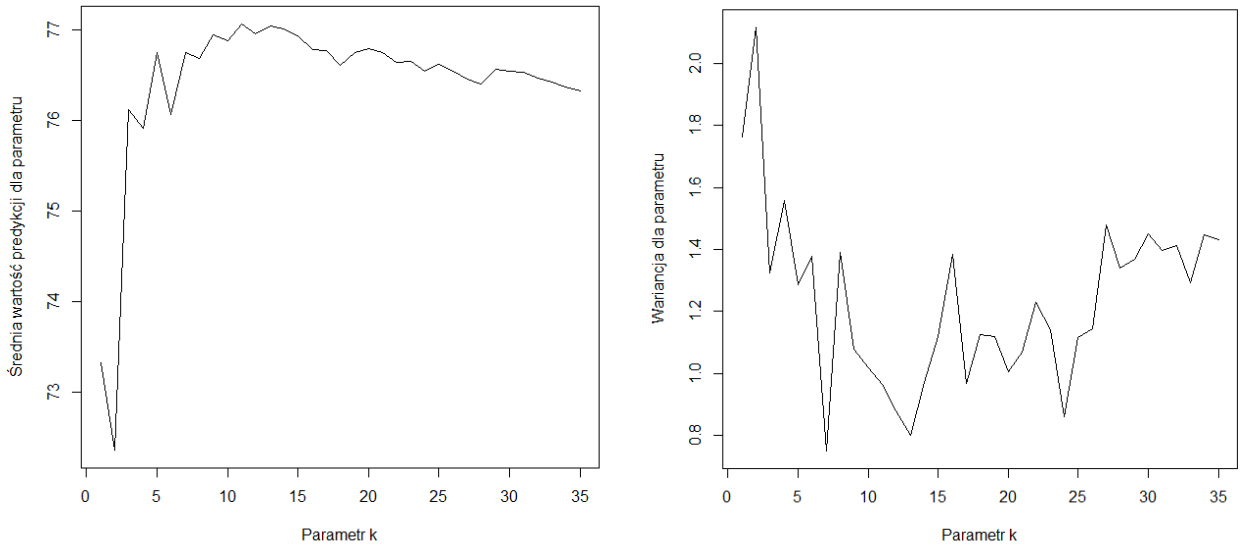
Następnie, tak jak poprzednio, zdecydowaliśmy się sprawdzić działanie tak stworzonego modelu dla różnej ilości sąsiadów.



Rysunek: Wykresy precyzji algorytmu dla różnych wartości parametru k (standaryzacja z-score)

Przy podejściu ze standaryzacją z-score widać już w tym momencie różnicę w porównaniu do normalizacji min-max, gdyż teraz optymalnym parametrami będą k=12 lub k=16.

Mimo poprawy nasza skuteczność dalej wynosi jedynie lekko powyżej 76%. Dla wyboru jednego, odpowiedniego, parametru, ponownie, zdecydowaliśmy się przeprowadzić więcej prób losując próbki testowe 50 razy. Zliczając wyniki po raz kolejny okazało się, że średnia jest bliska naszemu wynikowi, co jest satysfakcjonującym wynikiem. Tym razem jednak najlepsze wyniki były dla parametru k=13, dawały drugą najlepszą średnią (estymowaną) wynoszącą około 77% i drugą najniższą wariancję (estymowaną) wynoszącą około 0,8.



Rysunek: Wykresy średniej predykcji oraz wariancji (dla wylosowanych próbek) dla różnych parametrów k (standaryzacja z-score)

Zatem ostatecznie skuteczność modelu kNN (ocenianą na podstawie najlepszej predykcji przeprowadzonej dla wielu próbek) wynosi 77% i odpowiada liczbie sąsiadów równej 13. Uznaliśmy, że ten wynik nas nie zadowala i warto kontynuować analizę zmieniając typ modelu predykcyjnego.

3.3 Drzewa decyzyjne - budowa modelu

Zdecydowaliśmy się użyć drzewa decyzyjnego. Jest to uniwersalna metoda klasyfikacji, ale szczególnie dobrze wpasowuje się w nasz problem. Dzięki odpowiedniej implementacji, będzie możliwe wyodrębnienie kluczowych cech przy wyborze filmu oraz wykluczenie nieistotnych. Dodatkowym plusem jest fakt, że obsługuje zarówno zmienne ilościowe, jakościowe jak i braki w danych, które w naszym zbiorze występują.

Pierwsze drzewo, które stworzyliśmy, zwróciło od razu bardzo dobry wynik wynoszący równo 80%.

	predykcja		
rzeczywiste dane	dobre	slabe	Row Total
dobre	**160**	81	241
	0.211	0.107	
slabe	70	**446**	516
	0.092	**0.589**	
Column Total	230	527	757

Rysunek: Wyniki pierwszej wersji modelu drzewa decyzyjnego

3.4 Drzewa decyzyjne - ocena i dopracowanie modelu

Wynik można już ocenić jako satysfakcjonujący, jednak postanowiliśmy spróbować polepszyć jakość predykcji. Zdecydowaliśmy się użyć metody boostingu. Najpierw przetestowaliśmy algorytm dla ilości prób równej 10, później przechodząc do nawet 100 prób. Wraz ze wzrostem ilości prób rosła skuteczność. Dla 10 powtórzeń wyniosła ponad 84%, a dla 100 była minimalnie lepsza.

	predykcja dla 10 prób		
rzeczywiste dane	dobre	slabe	Row Total
dobre	**173**	68	241
	0.229	0.090	
slabe	53	**463**	516
	0.070	**0.612**	
Column Total	226	531	757

Rysunek: Wyniki metody AdaBoost z parametrem `trials = 10`

	predykcja dla 100 prób		
rzeczywiste dane	dobre	slabe	Row Total
dobre	**176**	65	241
	0.232	0.086	
slabe	54	**462**	516
	0.071	**0.610**	
Column Total	230	527	757

Rysunek: Wyniki metody AdaBoost z parametrem `trials = 100`

Jednak co jasne cierpiała na tym bardzo przejrzystość modelu, gdyż już pierwsze drzewo miało 77 liści i było mało czytelne.

```
Classification Tree
Number of samples: 3020
Number of predictors: 45
Tree size: 77
```

Rysunek: Ogólne parametry pierwszego modelu drzewa decyzyjnego

W tej sytuacji zdecydowaliśmy się na przycięcie naszego drzewa. W pakiecie `C50` nie ma funkcji która bezpośrednio określa ilość liści, ale są takie których działanie jest bardzo podobne. My skorzystaliśmy z funkcji `C5.0Control`, a dokładniej konkretnego jej argumentu `minCases`, który określa liczbę całkowitą najmniejszej liczby próbek, które należy umieścić w co najmniej dwóch z podziałów.

Po wielu sprawdzeniach znaleźliśmy model który ma wysoką skuteczność i małą ilość liści.
Przy `minCases = 80` drzewo miało jedynie 11 liści i aż 81,24% skuteczności.

```
Classification Tree
Number of samples: 3020
Number of predictors: 45
Tree size: 11
```

Rysunek: Ogólne parametry przyciętego modelu drzewa decyzyjnego

Tak jak poprzednio skorzystaliśmy z boostingu. W tym przypadku dla liczby prób równej 100 mieliśmy już bardzo dobrą skuteczność predykcji, gdyż wynosiła ona prawie 86%.

	predykcja dla 100 prób		
rzeczywiste dane	dobre	slabe	Row Total
dobre	**175**	66	241
	0.231	0.087	
slabe	44	**472**	516
	0.058	**0.624**	
Column Total	219	538	757

Rysunek: Wyniki przyciętego drzewa z użyciem metody AdaBoost z parametrem `trials = 100`

Ten model mogliśmy uznać jako nasz finalny model.

3.5 Powrót do metody k-NN

Na koniec jeszcze raz spróbowaliśmy wrócić do k-NN, ale tym razem mocno przeczyściliśmy nasze dane. Podczas testowania modeli drzewiastych zauważyliśmy, które czynniki rzeczywiście mają wpływ na jakość filmu. Wybraliśmy naszym zdaniem te najistotniejsze i jeszcze raz skorzystaliśmy z tej metody.
Efekt był widoczny natychmiastowo, model uzyskał ponad 74% skuteczności dla ilości sąsiadów równej pierwiastkowi z ilości obserwacji w zbiorze testowym oraz aż 81 % dla ilości sąsiadów równej 16. (taki sam proces wybierania parametru k, jak opisany w sekcji 3.2).

Krok 4. Podsumowanie

Wyniki estymacji, które otrzymaliśmy są zadowalające dla nas (testowaliśmy go nawet w praktyce). Udało nam się poprawić model i zauważyliśmy ciekawe zależności w danych jak i między samymi modelami. Pierwsze wersje modeli dla metody kNN dawały niskie wyniki i niewiele różniły się od losowego wyboru na “chybił trafił”. Ich wyniki były dla nas zdecydowanie niesatysfakcjonujące, więc zdecydowaliśmy się poprawić model zmieniając metodę. Logicznym wyborem było oczywiście zastosowanie drzew decyzyjnych. Wyniki tej metody bez boostingu oraz z boostingiem były o wiele lepsze od kNN. Skuteczność 86% dla metody boostingowej to wynik, który nas w pełni satysfakcjonował. Jedyne problemem był fakt, że model tracił przejrzystość przy takiej ilości prób, co nas martwiło. Stąd postanowiliśmy przyciąć drzewa i efekty były lepsze niż się spodziewaliśmy. Skuteczność modelu się poprawiła i nowe drzewo miało o wiele mniej liści przez co stało się o wiele bardziej przejrzyste. Jako nasz ostateczny finalny model uznajemy drzewo zbudowane metodą boostingu z minimalną ilością rekordów przechodzących przez podział równą 80. Model daje zdecydowanie najlepszy wynik równy 86% skuteczności oraz jest w pewnym stopniu przejrzysty. Dodatkowo, podejmowaliśmy próby usuwania różnych cech czy też tworzenia nowych na podstawie naszych danych np. sukces filmu (jako zarobki filmu podzielone przez koszt produkcji) jednak w metodach drzewiastych dla ogółu nie dało to pozytywnego efektu. Próbowaliśmy też ręcznie modyfikować ostatnie liście drzew. Obserwowaliśmy dane w liściach, w których model z przyciętymi liśćmi miał słabą skuteczność. Próby poprawy nie dawały wielkich efektów, a ryzyko przeuczenia było duże, więc zrezygnowaliśmy z tego pomysłu. Na koniec wróciliśmy jeszcze raz do metody kNN. Wiedząc, które cechy mają wpływ na dane, usunęliśmy cechy naszym zdaniem nieistotne dla problemu. Wyniki rzeczywiście były lepsze.
Nasza analiza oczywiście wciąż mogłaby być lepsza i dokładniejsza. W dalszym ciągu można dokonywać obróbki danych w metodzie k-NN, aby polepszać skuteczność tej metody. Drugą rzeczą, którą można poprawić i w której widzimy potencjał, jest nasze drzewo z przyciętymi liśćmi. Dokładniejsza analiza liści, które psują skuteczność modelu, może doprowadzić do lepszych rezultatów.

Projekt wykonali: Mateusz Gęca, Jędrzej Sarna, Karolina Świerk
25.06.2023r