1 Opis problemu

1.1 Cel projektu

Celem projektu jest stworzenie czerech modeli klasyfikacji, których zadaniem będzie stwierdzenie, jak dany użytkownik oceni dany film w skali 0-5. Decyzja ta będzie podejmowana na podstawie następujących atrybutów: `movieId, userId, genres, tag, relevance, year`.

1.2 Pochodzenie danych

Zbiór danych `ml-20m` pochodzi z serwisu movielens.org, gdzie użytkownicy oceniają film w skali 0.5-5 co pół stopnia oraz przypisują mu dowolny tag w postaci tekstu, który ma opisać dany film.
Znajduje się w nim 20000263 ocen i 465564 tagów opisujących 27278 filmów. Dane te zostały stworzone przez 138493 użytkowników pomiędzy 09-01-1995 a 31-03-2015. Użytkownicy zostali wyłonienie losowo, każdy z nich ocenił co najmniej 20 filmów.

Dane składają się z 6 plików: `‘genome-scores.csv‘, ‘genome-tags.csv‘, ‘links.csv‘, ‘movies.csv‘, ‘ratings.csv‘ i ‘tags.csv‘`.

1.3 Struktura danych

movies.csv

Zawiera dane o filmach. Każda linia reprezentuje jeden film w formacie `movieId,title,genres`. Tytuł zawiera rok premiery podany w nawiasach. `Genres` to lista gatunków filmu wybierana spośród

- Action
- Adventure
- Animation
- Children's
- Comedy
- Crime
- Documentary
- Drama
- Fantasy
- Film-Noir
- Horror
- Musical
- Mystery
- Romance
- Sci-Fi
- Thriller
- War
- Western
- (no genres listed)

links.csv

Zawiera linki do filmów w innych serwisach. Każda linia reprezentuje jeden film w formacie `movieId,imdbId,tmdbId`. Ten plik nie będzie wykorzystywany w projekcie.

`movieId` to identyfikator używany na stronie movielens.org, np. www.movielens.org/movies/1.

`imdcId` to identyfikator używany na stronie www.imdb.com, np. www.imdb.com/title/tt0114709.

`tmdbId` to identyfikator używany na stronie themoviedb.org, np. www.themoviedb.org/movie/862.

tags.csv

Zawiera tagi dopisane do filmów przez użytkowników. Struktura: `userId,movieId,tag,timestamp`.

genome-scores.csv

Plik zawiera dane na temat dopasowania tagu do filmu. Dane są postaci `movieId,tagId,relevance`.

genome-tags.csv

Zawiera Id tagu oraz tag, dane są postaci `tagId,tag`.

ratings.csv

Zawiera oceny filmów przez danego użytkownika. Struktura: `userId,movieId,rating,timestamp`.

1.4 Praca z danymi

Dane z plików `'genome-scores.csv', 'genome-tags.csv', 'movies.csv', 'ratings.csv', 'tags.csv'.` zostały połączone w jedną tabelę z wyselekcjonowanymi kolumnami. Ponieważ zbiór danych jest ogromny, ograniczymy go tylko do tych użytkowników, którzy ocenili ponad 1000 filmów (ograniczam zbiór z dwóch powodów: pierwszy, czysto techniczny - wybór najlepszego parametru dla metody KNN przy pełnym zbiorze trwał kilka godzin; drugi - ograniczenie unikatowych wartości `userId` może pozytywnie wpłynąć na poprawność klasyfikacji). Pierwotne oceny wyrażone są w skali 0.5-5 co pół stopnia; w projekcie zaokrąglę je w dół do liczb całkowitych - powstanie więc sześć klas: 0, 1, 2, 3, 4, 5. W kolumnie `title` znajduje się również rok premiery filmu, warto go wydobyć, aby uzyskać dodatkową informację. Dane po powyższych modyfikacjach przedstawia poniższa tabela.

	userId	movieId	rating	title	genres	tag	tagId	relevance	year
2	6431	2	4.0	Jumanji (1995)	Adventure|Children|Fantasy	time travel	1028	0.43900	1995
6	77463	2	3.0	Jumanji (1995)	Adventure|Children|Fantasy	time travel	1028	0.43900	1995
20	9815	32	5.0	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller	time travel	1028	0.98525	1995
56	52814	32	4.0	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller	time travel	1028	0.98525	1995
59	57434	32	4.0	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller	time travel	1028	0.98525	1995
...	...	...	...	...	...	...	...	...	...
183116	25737	72393	4.0	Fourth Kind, The (2009)	Horror|Mystery|Sci-Fi|Thriller	alaska	40	0.91750	2009
183119	10616	55721	4.0	Elite Squad (Tropa de Elite) (2007)	Action|Crime|Drama|Thriller	rio de janeiro	853	0.99575	2007
183127	4450	65216	3.0	Defiance (2008)	Drama|Thriller|War	russia	868	0.70475	2008
183131	27898	26007	4.0	Unknown Soldier, The (Tuntematon sotilas) (1955)	Drama|War	finnish	394	0.99950	1955
183135	68558	27152	4.0	Pitkä kuuma kesä (1999)	Comedy|Drama	finnish	394	0.99975	1999

40879 rows × 9 columns

Następnie sprawdzam ile unikatowych danych mamy w każdej z kolumn:

```
Unikatowych userId : 22
Unikatowych movieId : 4433
Unikatowych rating : 6
Unikatowych title : 4433
Unikatowych genres : 704
Unikatowych tag : 1003
Unikatowych tagId : 1003
Unikatowych relevance : 3668
Unikatowych year : 99
```

Upewniam się, że w zbiorze nie brakuje żadnych danych oraz obliczam podstawowe statystyki zbioru.

```
Brakujących userId:  0
Brakujących movieId:  0
Brakujących rating:  0
Brakujących title:  0
Brakujących genres:  0
Brakujących tagów: 0
Brakujących tagId:  0
Brakujących relevance:  0
Brakujących year:  0
```

Liczba poszczególnych ocen (klas) wynosi

```
Podzial na klasy
 {0: 325, 1: 2245, 2: 5361, 3: 11942, 4: 15045, 5: 5961}
```

Ostatnim etapem przygotowania danych jest przetworzenie danych do postaci bardziej przyjaznej klasyfikatorom, za pomocą `SimpleImputer` oraz `OneHotEncoder` i podzielenie na zbiory treningowe i testowe w stosunku 80-20.

```
Liczba ocen w zbiorze testowym
 {0: 65, 1: 449, 2: 1072, 3: 2389, 4: 3009, 5: 1192}
Liczba ocen w zbiorze treningowym
 {0: 260, 1: 1796, 2: 4289, 3: 9553, 4: 12036, 5: 4769}
```

2 Opis zastosowanych metod uczenia maszynowego

2.1 Algorytm k najbliższych sąsiadów

Algorytm K-najbliższych sąsiadów (KNN) to rodzaj nadzorowanych algorytmów uczenia maszynowego. Jest to leniwy algorytm uczenia się, ponieważ nie ma specjalistycznej fazy treningu. Zamiast tego wykorzystuje wszystkie dane do szkolenia podczas klasyfikowania nowego punktu danych lub instancji. KNN jest nieparametrycznym algorytmem uczenia, co oznacza, że nie zakłada niczego na temat podstawowych danych.
Działanie algorytmu polega na obliczaniu odległości nowego punktu danych do wszystkich innych punktów danych treningowych. Odległość może być dowolną metryką. Następnie wybiera k najbliższych punktów, gdzie k może być dowolną liczbą całkowitą. Ostatecznie przypisuje punkt danych do klasy, do której należy większość punktów z k najbliższych.

2.2 Drzewa decyzyjne

Drzewa decyzyjne są ważnym narzędziem w uczeniu maszynowym i eksploracji danych. Są wykorzystywane między innymi w problemie klasyfikacji. Główne składowe drzewa to korzeń oraz gałęzie, które łączą korzeń z kolejnymi wierzchołkami. Wierzchołki, z których wychodzi co najmniej jedna krawędź, są nazywane węzłami, natomiast wierzchołki z których nie wychodzą krawędzie to tzw. liście. Drzewo zaczyna od pojedynczego węzła reprezentującego cały zbiór treningowy. W każdym węźle sprawdzany jest pewien warunek dotyczący danej obserwacji, i na jego podstawie wybierana jest odpowiednia gałąź prowadząca do kolejnego wierzchołka. Klasyfikacja danej obserwacji polega na przejściu od korzenia do liścia i przypisaniu do tej obserwacji klasy zapisanej w danym liściu.

2.3 Lasy losowe

Algorytm lasu losowego polega na konstruowaniu wielu drzew decyzyjnych. Dla danej obserwacji każde z drzew zwraca decyzję lub krotkę prawdopodobieństw klasyfikacji. Decyzje (praw-

dopodobieństwa) z drzew wchodzących w skład lasu są traktowane jako głosy. Głosy mogą być ważone lub nie zależnie od wybranej przez użytkownika metody głosowania. Ostateczny wybór klasy polega na wyborze dominanty, czyli klasy, która pojawiła się najczęściej. Losowe lasy decyzyjne poprawiają tendencję drzew decyzyjnych do nadmiernego dopasowywania się do zestawu treningowego.

2.4 Naiwny klasyfikator bayesowski

Naiwny klasyfikator bayesowski to algorytm działający w oparciu o twierdzenie Bayesa. Zakłada on niezależność cech, co w rzeczywistości jest rzadko spotykane - stąd też jego nazwa "naiwny". Algorytm dla każdej z klas wylicza prawdopodobieństwo, że obiekt pochodzi z klasy Y_k pod warunkiem, ze jest reprezentowany przez cechy $x = (x_1, \ldots, x_n)$

$$P(Y_k|x).$$

Korzystając z twierdzenia Bayesa otrzymujemy

$$P(Y_k|x) = \frac{P(Y_k)P(x|Y_k)}{P(x)},$$

a następnie, wykorzystując założenie o niezależności cech, można zapisać, że

$$P(Y_k|x) = \frac{P(Y_k)}{P(x)} \prod_{i=1}^{n} P(x_i|Y_k).$$

Jako klasę algorytm wybiera tę, dla której prawdopodobieństwo warunkowe $P(Y_k|x)$ jest największe.

2.5 Strojenie parametrów

Aby użyte algorytmy klasyfikujące dały lepsze wyniki, wykorzystałam dwie metody strojenia hiperpataremtów. Pierwszym z nich jest sprawdzian krzyżowy, zwany też walidacją krzyżową, drugi to algorytm AdaBoost.

2.5.1 Walidacja krzyżowa

W k-krotnym sprawdzianie krzyżowym losowo rozdziela się zestaw danych uczących na k podzbiorów, gdzie $k-1$ podzbiorów jest wykorzystywanych do uczenia modelu, a tylko jeden do jego testowania. Dzięki temu otrzymuje się k modeli i oszacowań skuteczności. Po określeniu najlepszych hiperparametrów model ponownie poddaje się uczeniu, tym razem na całym zestawie danych uczących, i otrzymuje się ostateczne oszacowanie skuteczności przy użyciu zestawu testowego.

2.5.2 AdaBoost

Innym sposobem poprawienia wyników klasyfikatora jest użycie algorytmu AdaBoost. Jego działanie polega na sekwencyjnym uczeniu predyktorów w taki sposób, że kolejny próbuje korygować poprzedniego. Dzieje się to dzięki zwiększaniu wag źle sklasyfikowanych przypadków. W ten sposób kolejne predyktory koncentrują się coraz bardziej na najtrudniejszych przypadkach. Po powtórzeniu tego kroku wiele razy algorytm klasyfikuje przypadki na podstawie głosowaniu wszystkich stworzonych klasyfikatorów.

3 Ocena modeli

Ocena modeli oparta zostanie o następujące miary:

- `accuracy` - dokładność, miara dana wzorem $\frac{TP+TN}{TP+TN+FP+FN}$, określa jaką część wszystkich prognoz stanowią prognozy poprawne,
- `recall` (in. sensitivity) - czułość, miara dana wzorem $\frac{TP}{TP+FN}$, określa prawdopodobieństwo, że klasyfikacja będzie poprawna pod warunkiem, że przypadek jest pozytywny
- `precision` - precyzja, miara dana wzorem $\frac{TP}{TP+FP}$, określa jakie jest prawdopodobieństwo, że przypadek jest pozytywny, gdy wynik predykcji jest pozytywny
- `f1` - średnia harmoniczna `precision` oraz `recall` dana wzorem $2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$. Miara ta daje ocenę balansu między czułością a precyzją, nie uwzględnia wyników prawdziwie negatywnych.
- `Time` - czas trenowania algorytmu

Podane metryki są metrykami odpowiednimi dla problemu dwuklasowego. Aby zastosować je do klasyfikacji wieloklasowej, używa się strategii `one vs one` lub `one vs rest`, tworząc wiele klasyfikarów binarnych, a następnie liczy się średnią makroskopową lub ważoną. Wartość marko to uśredniona wartość sumy precyzji wszystkich systemów, np. dla precyzji mamy

$$PRE_{makro} = \frac{PRE_1 + \cdots + PRE_k}{k}.$$

Średnia ważona uwzględnia natomiast support, czyli liczbę instancji każdej klasy.

3.1 Algorytm KNN

Po zastosowaniu walidacji krzyżowej do wyboru najlepszej wartości parametru k z zakresu od 2 do 10, okazało się, że najlepsze rezultaty otrzymano dla $k = 5$. Ocenę modelu, macierz pomyłek oraz czas trenowania przedstawia poniższa tabela.

```
              precision    recall  f1-score   support

           0       0.29      0.23      0.26        65
           1       0.46      0.40      0.42       449
           2       0.47      0.42      0.45      1072
           3       0.56      0.59      0.58      2389
           4       0.62      0.68      0.65      3009
           5       0.58      0.44      0.50      1192

    accuracy                           0.57      8176
   macro avg       0.50      0.46      0.47      8176
weighted avg       0.56      0.57      0.56      8176

[[  15    9   16   11   12    2]
 [   8  178   53   95   90   25]
 [  11   73  455  256  213   64]
 [  10   66  214 1417  561  121]
 [   4   44  175  564 2051  171]
 [   3   21   54  188  406  520]]

Czas kroswalidacji KNN:  272.4200358390808
{'n_neighbors': 5}
5
Czas KNN:  0.0080108642578125
```

3.2 Drzewa decyzyjne

Po zastosowaniu walidacji krzyżowej do wyboru najlepszej wartości parametru `max_depth` z zakresu od 2 do 10, okazało się, że najlepsze rezultaty otrzymano dla wartości 9. Ocenę modelu, macierz pomyłek oraz czas trenowania przedstawia poniższa tabela.

```
              precision    recall  f1-score   support

           0       0.87      0.20      0.33        65
           1       0.37      0.16      0.23       449
           2       0.40      0.30      0.35      1072
           3       0.52      0.49      0.50      2389
           4       0.56      0.63      0.59      3009
           5       0.49      0.62      0.54      1192

    accuracy                           0.51      8176
   macro avg       0.53      0.40      0.42      8176
weighted avg       0.51      0.51      0.50      8176

[[  13    2   11   14   15   10]
 [   1   74  124  104   96   50]
 [   1   34  325  352  256  104]
 [   0   32  216 1167  809  165]
 [   0   40  100  526 1892  451]
 [   0   20   31   77  328  736]]

Czas kroswalidacji drzew decyzyjnych:  6.721824645996094
{'max_depth': 9}
9
Czas drzew decyzyjnych:  0.4010195732116699
```

3.3 Lasy losowe

Po zastosowaniu walidacji krzyżowej do wyboru najlepszej wartości parametru `max_depth` z zakresu od 2 do 10, okazało się, że najlepsze rezultaty otrzymano dla wartości 7. Ocenę modelu, macierz pomyłek oraz czas trenowania przedstawia poniższa tabela.

```
              precision    recall  f1-score   support

           0       0.58      0.51      0.54        65
           1       0.60      0.47      0.53       449
           2       0.60      0.53      0.56      1072
           3       0.64      0.70      0.67      2389
           4       0.69      0.73      0.71      3009
           5       0.71      0.59      0.65      1192

    accuracy                           0.66      8176
   macro avg       0.64      0.59      0.61      8176
weighted avg       0.66      0.66      0.66      8176

[[  33    4    6   12    8    2]
 [   5  212   57   71   94   10]
 [   8   40  567  233  171   53]
 [   7   47  147 1662  448   78]
 [   1   41  118  497 2206  146]
 [   3   12   45  130  293  709]]

Czas kroswalidacji lasów:  144.24247550964355
{'n_estimators': 7}
7
Czas lasów:  5.859492778778076
```

3.4 Naiwny klasyfikator Bayesowski

W tym przypadku do wzmocnienia klasyfikatora użyto algorytmu AdaBoost z parametrami `n_estimators=20, learning_rate=1`. Ocenę modelu, macierz pomyłek oraz czas trenowania przedstawia poniższa tabela.

	precision	recall	f1-score	support
0	0.19	0.42	0.26	65
1	0.22	0.65	0.33	449
2	0.40	0.35	0.37	1072
3	0.38	0.64	0.48	2389
4	0.62	0.17	0.26	3009
5	0.36	0.28	0.31	1192
accuracy			0.37	8176
macro avg	0.36	0.42	0.34	8176
weighted avg	0.46	0.37	0.35	8176

```
[[  27   10    3   16    3    6]
 [   9  293   45   75   12   15]
 [  22  152  372  381   69   76]
 [  30  351  207 1526  133  142]
 [  34  432  228 1463  499  353]
 [  17  109   75  573   88  330]]
```

```
Czas AdaBoost z GNB:  519.1188287734985
```

3.5 Porównanie

Aby łatwiej było zdecydować, który model okazał się lepszy, sprawdzę który algorytm uzyskał najwyższy wynik w każdej z metryk.

	precision	recall	f1-score
0	TREE	FOREST	FOREST
1	FOREST	NB	FOREST
2	FOREST	FOREST	FOREST
3	FOREST	FOREST	KNN
4	FOREST	FOREST	FOREST
5	FOREST	TREE	FOREST
accuracy			FOREST
macro avg	FOREST	FOREST	FOREST
weighted avg	FOREST	FOREST	FOREST

4 Podsumowanie

W przedstawionym problemie najlepiej sprawdził się klasyfikator lasu losowego, jednak niewiele gorzej działały drzewa losowe. Porównując wszystkie cztery modele można zauważyć, że największym problemem była poprawna klasyfikacja do klas 0 oraz 1. Jednak i tutaj las losowy sprawdził się na poziomie 50-60%. Wynik ten niesie za sobą pewne koszty - czas trenowania lasu losowego z najlepszym dobranym parametrem `max_depth` to prawie 6 sekund, podczas gdy trening algorytmu KNN oraz drzew decyzyjnych zajął ułamek sekundy (nie uwzględniam tu algorytmu AdaBoost, ponieważ w tym przypadku wskazany czas to czas trenowania kilkudziesięciu klasyfikatorów).