# KNN

Użyte cechy: Budżet, popularność, średnia głosów, gatunek, kraj pochodzenia firm produkujących film i kraj powstania filmu; zostały wybrane zgodnie z naszą wiedzą ekspercką. Cechy posiadających wiele opcji jednocześnie (np. gatunek: film może być jednocześnie romansem i komedią) zostały rozbite na tabele, gdzie każda kolumna reprezentowała jedną z opcji (nasz przykładowy film miałby 1 w kolumnie romans i komedia a 0 w pozostałych kolumnach). Wszystkie cechy numeryczne (np. budżet) zostały znormalizowane z użyciem StandardScaler() z biblioteki sklearn.preprocessing. Jako, że wszystkie cechy były reprezentowane jako liczby mogliśmy zastosować dystans euklidesowy (np.sqrt(np.sum((datapoint1 – datapoint2) \*\* 2))) do obliczenia dystansu między poszczególnymi rzędami.

W pierwszym podejściu szukaliśmy najbliższych sąsiadów na całym zbiorze danych (nie dzieląc go na poszczególnych studentów) zakładając, że jako, że należą do podobnej grupy społecznej (studenci jednej politechniki) ich opinie mogłyby być uznane jako wymienne. Uzyskaliśmy poniższy wynik:

[tu wstaw 1 wynik]

Zdecydowanie lepsze wyniki udało się nam uzyskać po zastosowaniu takiej samej metody na danych podzielonych na oceny poszczególnych studentów.

[tu wstaw 2 wynik]

Hiperparametrem używanym przez nas w uczeniu była liczba sąsiadów (k) dla algorytmu kNN. Dla zbioru zawierającego wszystkich studentów najlepsze k wynosiło 20. Dla zbiorów podzielonych na różnych studentów określone zostały różne k. Najlepsze k było wybierane dla wartości, która dała najmniejszy średni błąd kwadratowy (mean squared error) na zbiorze testowym powstałym dla 5 różnych podziałów zbioru treningowego na treningowy i testowy (cross-validation).

# Drzewo decyzyjne i las losowy

Nasz algorytm drzewa decyzyjnego wybiera losową cechę z wszystkich dostępnych i sprawdza czy z jej pomocą (dla cech numerycznych porównuje czy jest większa niż kolejne 20 wartości między -0,9 a 0,9 a dla cech kategorialnych czy jest równa 0) może podzielić dostępne dane (trenowaliśmy na danych filmów dopasowanych do ocen pojedynczego studenta) na dwa zbiory w tym jeden zawierający przynajmniej min\_size elementów w tym przynajmniej uniformity (ułamek) ma taką samą wartość oceny. Jeśli jednak taka kolumna nie zostanie znaleziona program podejmie próbę podziału zbioru na dwa niezależnie od wartości ocen.  
Losowość wbudowana w wybór kolumn pozwala nam tworzyć różne drzewa dla takich samych danych pozwalając nam tworzyć lasy losowe.  
Hiperparametrami jakich szukaliśmy były wymienione wyżej min\_size czyli minimalny rozmiar wydzielonego zbioru oraz uniformity czyli minimalny ułamek wydzielanego zbioru składający się z takich samych elementów. Dla lasów losowych dodatkowo szukaliśmy najlepszej ilości drzew.  
Szukaliśmy ich podobnie jak dla kNN: porównując wartości średniego błędu kwadratowego (mean squared error) otrzymanego na zbiorze walidacyjnym przez model uczony na zbiorze treningowym.

Utworzone drzewa były bardzo płytkie: z reguły składały się z 3 liści, jednak dla celów demonstracyjnych rozrysowaliśmy bardziej rozłożyste drzewo.

[tu wstaw zdjęcie drzewa]

Mimo słabego wyglądu naszych drzew udało się nam uzyskać relatywnie dobre rezultaty

[tu wstaw wyniki drzew OBU]

Niższy wynik naszego lasu losowego w porównaniu do drzewa decyzyjnego może być spowodowany niską (poniżej 50%) dokładnością co w połączeniu z prostotą naszych drzew mogło uwydatnić błędy w naszym lesie losowym.

# Podobieństwo między ludźmi

Do obliczenia podobieństwa między dwoma studentami użyliśmy podobieństwa Pearsona. Podobieństwo obliczane jest dla każdego studenta względem rozważanego studenta, ustawiani są w kolejności od najbardziej podobnego do najmniej podobnego. Szukając opinii danego studenta dla danego filmu studenci w tej liście są po kolei przeszukiwani i pierwszy posiadający ocenę danego filmu jest używany do przewidzenia tej oceny (jego ocena jest kopiowana).  
Hiperparametrem użytym w naszym algorytmie była minimalna ilość filmów jakie wspólnie musieli widzieć. Ze względu na brak wpływu niewspólnych filmów na wartość podobieństwa, by uniknąć sytuacji, gdzie gust dwóch studentów uznawany jest za identyczny, ponieważ oboje mają taką samą opinię na temat jedynego filmu, który oboje obejrzeli wprowadziliśmy minimum wspólnych filmów jakie studenci musieli zobaczyć zanim ich podobieństwo było rozważane.  
Szukaliśmy hiperparametru porównując porównując wartości średniego błędu kwadratowego (mean squared error) otrzymanego na zbiorze walidacyjnym przez model uczony na zbiorze treningowym.

Osiągnięto relatywnie słaby wynik:

[wstaw wynik]

# Colaborative-filtering

Zgodnie z metodą przedstawioną na wykładzie szukaliśmy zestawu parametrów dla każdego studenta i cech każdego filmu szukając minimum różnicy między ocenami będącymi rezultatem działania na nich a faktycznej oceny użytkownika

[wstawić równanie z wykładu] + może z kodu

[Jak uczyliśmy]

# Wyniki i wnioski ????

To, jak dobre są nasze modele ocenialiśmy za pomocą dokładności na zbiorze testowym

Najlepszy wynik uzyskaliśmy z

[tu wstawić najlepszy wynik]

Nasze najgorsze wyniki to KNN dla wszystkich studentów jednocześnie: 24,3% oraz [tu wstawić 2 najgorszy wynik]

Jak widać, nasze założenie o podobieństwie między studentami było nieprawidłowe; jednak analizując dodatkowo podobieństwo między gustami użytkowników w trakcie 3 zadania (omówione w dziale Podobieństwo między studentami) można osiągnąć zadowalający wynik. Sugeruje to, że zarówno zbiór danych jak i zadanie przewidzenia ocen należałoby podzielić na podzadania obejmujące pojedynczych użytkowników [Jeśli wszystkie wyniki całości są gorsze niż KNN i drzewa !!! kontynuuj, że wymaga to znacznych danych od użytkownika]