30.06.2023

Sztuczna Inteligencja i Inżynieria Wiedzy

Lab 5

Autor: Jakub Szwedowicz 243416

Prowadzący: dr inż. Piotr Syga

Zajęcia: czw 7:30

30.06.2023

Spis treści

30.06.2023

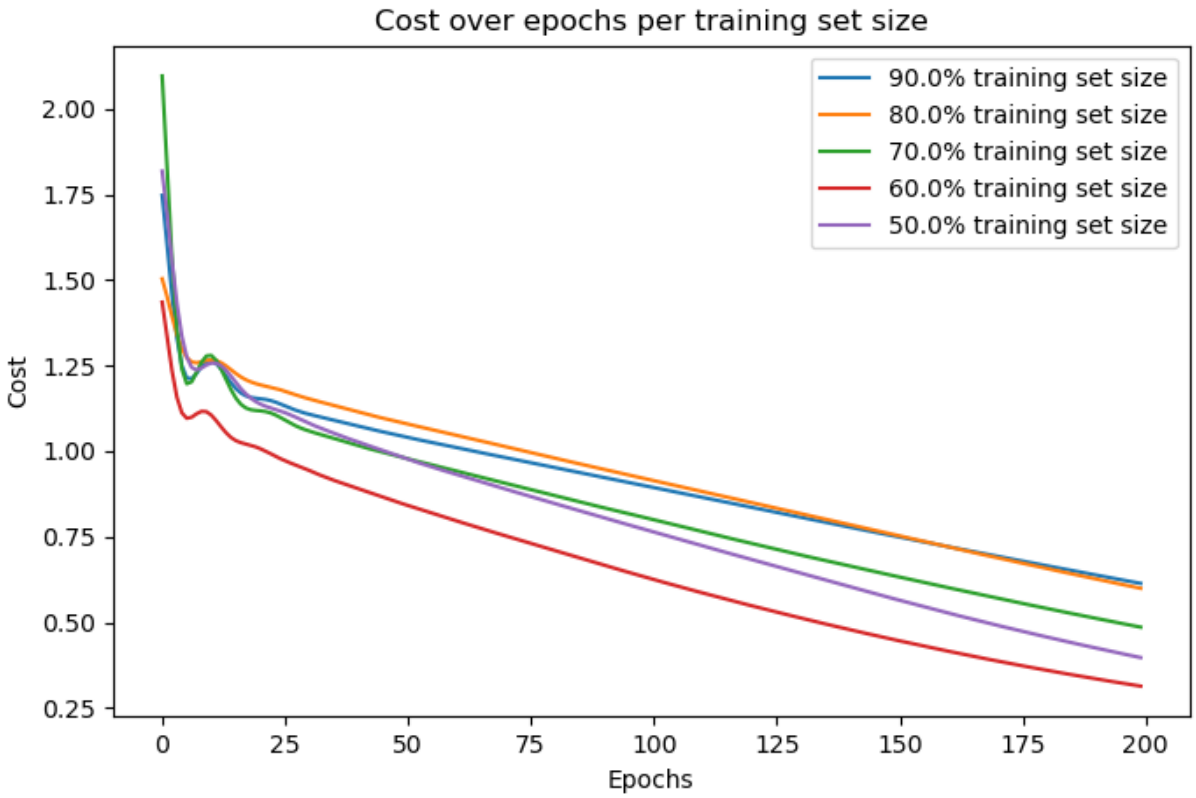
1. Przygotuj zbiór uczący i walidacyjny, wykorzystując dołączony do listy kod procedury ekstrakcji cech. Jeśli zamierzasz korzystać z Weki, zalecane jest wykonanie jednorazowego przekształcenia danych i eksportu do jednego ze zgodnych formatów. (5 punktów)

Najpierw przygotowano kod konwertujący dostarczone żarty do formy wektorowej. Kod ładuje żarty z plików .html, wyciąga z każdego z pliku samą treść żartu, a następnie zwraca listę żartów. Nastepnie żarty są tokenizowane przez SentenceTransformer i zapisywane do pliku przez numpy.

Wczytywane są również oceny wystawione przez użytkowników, usuwana jest pierwsza kolumna wskazująca na liczbę ocenionych żartów przez użytkowników oraz liczba ‘99’ jest zastąpiona przez np.nan żeby usunąć fałszywe dane liczbowe z datasetu. Obracana jest również macierz ocen (żeby pasowała do tabeli żartów), wyznaczana jest mediana ocen dla każdego żartu a następnie macierze żarów i ocen są łączone.

Funkcja split_data przyjmuje macierz stokenizowanych żartów z medianą ocen tych żartów, przyjmuje, że mediany znajdują się w ostatniej kolumnie. Następnie dzieli tak otrzymaną macierz na zbiór trenujący i testujący.

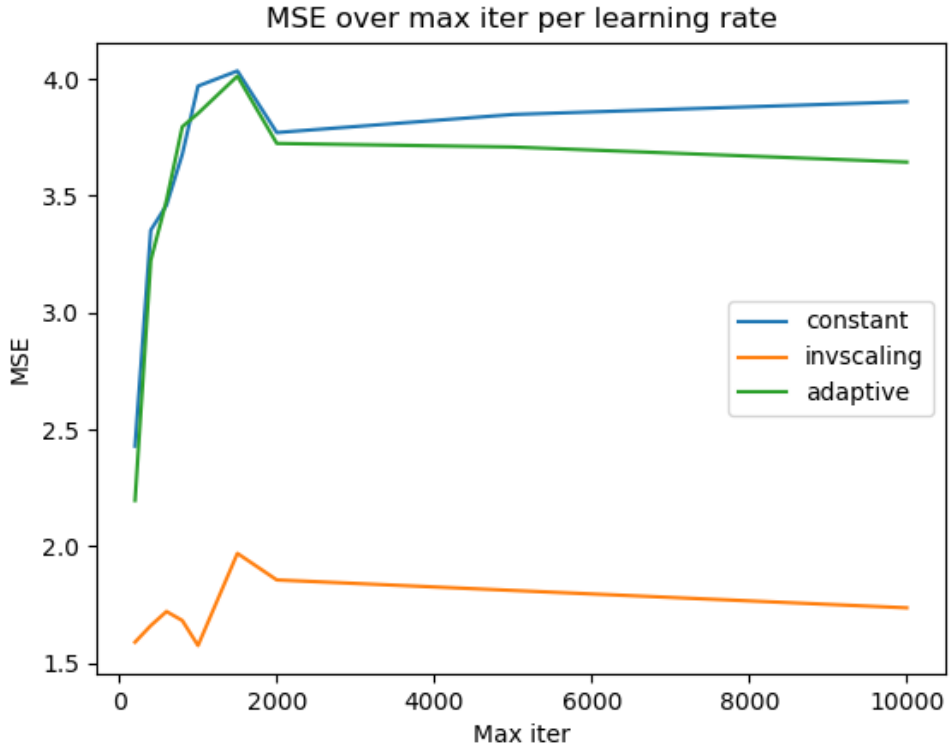
2. Przetestuj działanie podstawowego modelu MLP o domyślnej konfiguracji hiperparametrów, ucząc go na danych ze zbioru Jester. Prześledź zachowanie modelu w czasie, wizualizując wartość funkcji kosztu w funkcji liczby epok, zwracając uwagę na wartości dla zbioru uczącego i zbioru walidacyjnego. (20 punktów)



Wraz ze wzrostem wielkości zbioru treningowego koszt uczenia rośnie. Fakt ten nie jest w żaden sposób zaskakujący – im większy jest zbiór treningowy, który model musi przetworzyć na etapie uczenia, tym więcej czasu potrzebuje on na przetworzenie tych danych. Prędkość spadku kosztu wraz z ilością epok jest stała dla każdej wielkości zbioru treningowego i testowego.

3. Zbadaj wpływ tempa uczenia (learning rate) na osiągane wyniki: powtórz uczenie dla 3 różnych wartości parametru. Dobierz odpowiednią długość procesu uczenia (liczbę epok) jeśli to konieczne. Przedstaw wyniki na wykresach jak w zadaniu poprzednim. Co dzieje się, gdy tempo uczenia jest zbyt niskie? Co, gdy zbyt wysokie? (30 punktów)

W zadaniu 3 zbadaliśmy wpływ tempa uczenia się, wartości alpha oraz maksymalnej ilości epok na jakość modelu.

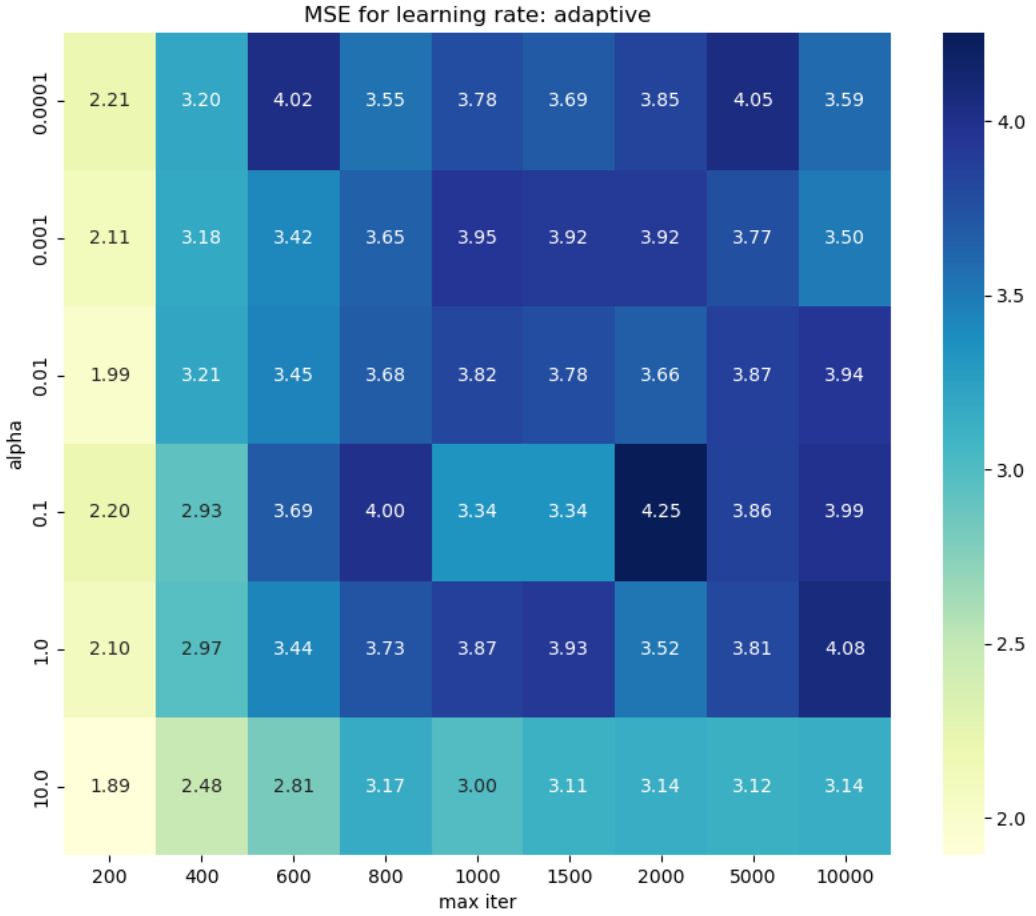


Do porównywania jakości wyników wykorzystany zostanie błąd średniokwadratowy (MSE). Używając go jako wyznacznik, można łatwo porównać dwa modele i ich predykcje. Wraz ze wzrostem maksymalnej liczby epok błąd średniokwadratowy maleje dla prędkości adaptive i invscaling. Dla prędkości constaint błąd rośnie wraz ze wzrostem maksymalnej liczby epok.

Rysunek poniżej przedstawia heatmapę błędu średniokwadratowego dla prędkości „constant” i różnych wartości alpha oraz max iter. Wraz ze wzrostem liczby epok błąd średniokwadratowy rośnie. Dla parametru alpha model uzyskiwał najlepsze wyniki dla 0.001, a najgorsze dla 0.01. Różnica jednak w MSE pomiędzy tymi parametrami zdaje się jednak dość marginalna.

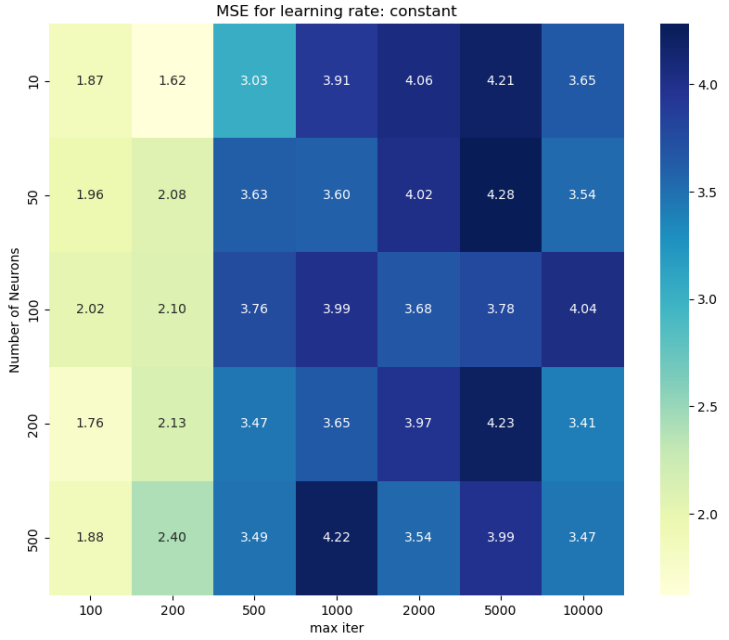
W przypadku prędkości uczenia invscaling najlepszy wynik uzyskano z parametrami alpha = 0.001 i max_iter = 600. W przypadku tego modelu średni błąd średniokwadratowy jest niższy, ale błąd nie wykazuje korelacji z hiperparametrem alpha.

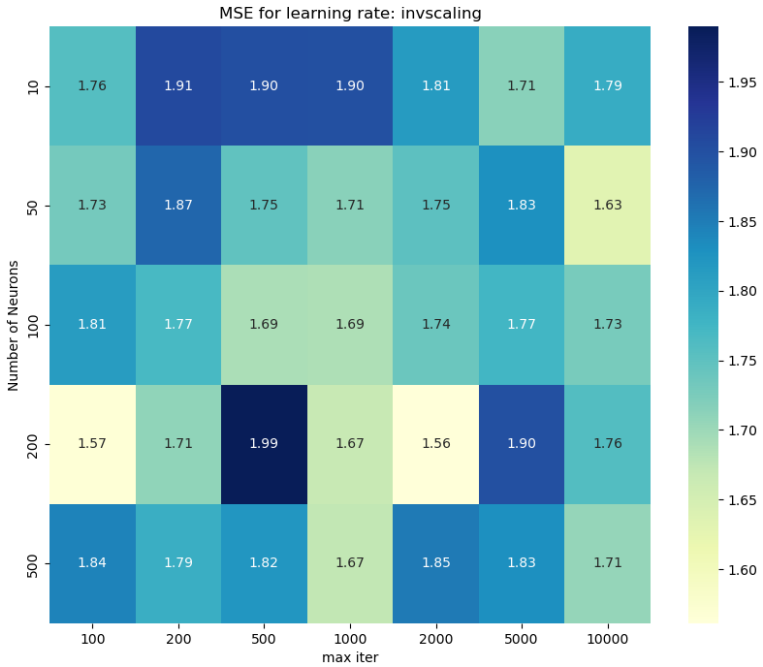
W przypadku prędkości „adaptive” podobnie jak w przypadku „constant” najlepsze wyniki otrzymano dla 100 i 200 epok, co może sugerować, że wyższe wartości powodują overfitting modelu. W konsekwencji model dobrze poradziłby sobie z oceną żartów ze zbioru treningowego, ale gorzej z nowymi żartami. Najlepszy wynik jest dla alpha 0.01 oraz max_iter 200



4. Zbadaj wpływ rozmiaru modelu MLP na jakość działania: wykonaj co najmniej 3 eksperymenty dla modeli różniących się liczbą neuronów. Kiedy model przestaje dobrze dopasowywać się do danych? Kiedy zaczyna zanadto dopasowywać się do zbioru uczącego? (30 punktów)

Przetestowano jak modele z 3 różnymi prędkościami uczenia radzą sobie w zależności od ilości neuronów i maksymalnej liczby epok. W przypadku prędkości „constant” jednocześnie najlepsze i najgorsze wyniki otrzymaliśmy dla ilości neuronów 10. W zależności od ustawionej maksymalnej liczby epok otrzymywaliśmy wartość MSE z zakresu 1.69 do 4.54.

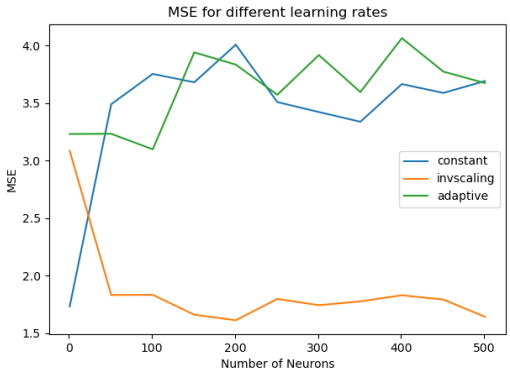


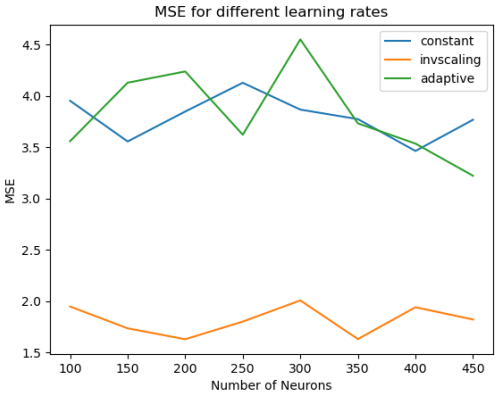


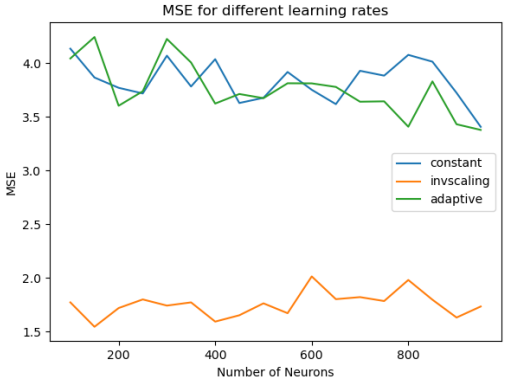
Dla adaptacyjnej prędkości uczenia, można zauważyć że ilość neuronów nie ma żadnego wpływu na jakość wyników. Ciekawym spostrzeżeniem jest, że zwiększanie liczby iteracji powoduje pogorszenie MSE perceptronu. Za potencjalną przyczynę można założyć overfitting.

Na podstawie poprzednich wyników określono najbardziej optymalne wartości parametrów max_iter oraz alpha: 1000 oraz 0.01. Używając tych wartości sprawdzono jak zachowuje się błąd średniokwadratowy wraz ze wzrostem liczby neuronów.

Przeprowadzono dwa identyczne testy, pierwszy dla liczby neuronów [1, 51, 101, …, 501] i drugi dla neuronów [100, 150, 200, 450]. Kusi powiedzieć, że gdzieś jest jakieś optimum i te wyniki mają sens, ale nie wiem ani gdzie to optimum jest, ani dlaczego parzystość liczby neuronów ewidentnie ma znaczenie. Uwzględniając fakt, że wykresy nie są powtarzalne i dla tych samych parametrów potrafią wyglądać zupełnie inaczej, można dojść do wniosku że w modelu występują liczne lokalne optima







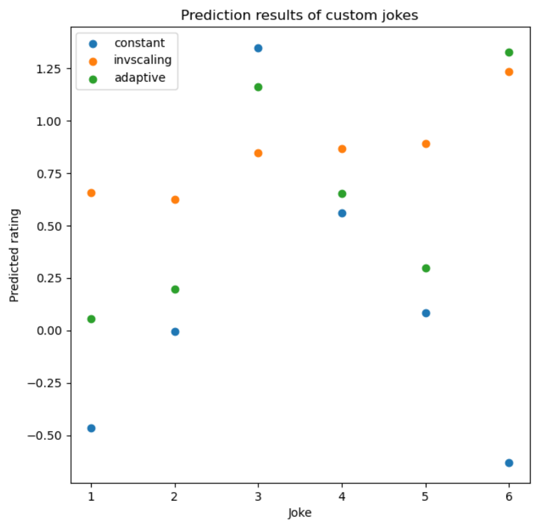
5. Wybierz najlepszy uzyskany w drodze powyższych eksperymentów model i przetestuj go w praktyce: znajdź (lub napisz własny) tekst o charakterze dowcipu, przetwórz go na wektor za pomocą używanej w zadaniach metody ekstrakcji cech, a następnie odpytaj model neuronowy. Czy predykcja zgadza się z Twoim oczekiwaniem? (15 punktów)

Na podstawie poprzednich zadań wybrana została najbardziej optymalna konfiguracja parametrów dla każdej z prędkości uczenia się. Ponad to utworzono listę 6 żartów, które zostaną sklasyfikowane przez model.

```
CONFIG = {
    "constant": {
        "max_iter": 200,
        "learning_rate": "constant",
        "alpha": 0.001,
        "hidden_layer_sizes": (10,)
    },
    "invscaling": {
        "max_iter": 600,
        "learning_rate": "invscaling",
        "alpha": 1.0,
        "hidden_layer_sizes": (50,)
    },
    "adaptive": {
        "max_iter": 200,
        "learning_rate": "adaptive",
        "alpha": 10.0,
        "hidden_layer_sizes": (10,)
    }
}
```

```
TEST_JOKES = [
    "Why Java programmers wear glasses? Because they can't c#",
    "Why did the programmer quit his job? Because he didn't get arrays.",
    "This couple had an excellent relationship going until one day he came home from work to find his girlfriend packing. \
        He asked her why she was leaving him and she told him that she had heard awful things about him. \
            What could they possibly have said to make you move out? They told me that you were a pedophile. \
                He replied, That's an awfully big word for a ten year old.",
    "I went to the doctors recently He said: Don't eat anything fatty I said: What, like bacon and burgers? \
        He said, No. fatty don't eat anything",
    "Why do you never see hippos hiding in trees Because they're very good at it.",
    "EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE \
        EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE \
            EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE",
]
```

Wykres poniżej pokazuje, że tylko „constant" ocenił negatywnie żart 6 składający się jednie z litery E oraz spacji. Zarówno „invscaling" jak i „adaptive" oceniły ten żart bardzo pozytywnie, co sugeruje, że nie jest to odpowiednie tempo w przypadku naszego problemu. Żart 3 jest żartem ze zbioru uczącego i został oceniony średnio najwyżej ze wszystkich zarówno przez model z prędkością „constant" jak i „adaptive".



6. Wykonaj badanie dowolnego innego parametru, np. regularyzacji. Jakie jest jego działanie? Z jakim problemem pozwala on walczyć? Jak wpływa na wyniki? (bonus do 10 punktów)

7. Wnioski

Ze wszystkich przetestowanych modeli uczenia maszynowego model z wybraną prędkością uczenia „constant” odpowiednio oceniając żart składający się wyłącznie z liter E. Wygląda jednak również na to że wykazuje nieuzasadniony faworyzm dla Javy, gdyż nie polubił pierwszego żartu. Z tego powodu za najlepszy model można uznać „invscaling”, gdyż średnio uzyskiwał wcześniej najniższy MSE.

Bardzo możliwe, że zmieniając inne parametry modelów jak np. learning_rate_init lub power_t można osiągnąć lepsze wyniki. Przetestowane modele rzadko kiedy oceniają żarty wartościami spoza zakresu [-1.5, 1.5], co może być spowodowane wyborem mediany jako funkcji obliczającej „średnią” śmieszność żartów w zbiorze treningowym. Kolejną możliwą optymalizacją może być kategoryzacja żartów np. według rodzaju: czarny humor, suchar itd., aby następnie na podstawie danych źródłowych wybrać np. 20% osób, które najwyżej oceniły dany rodzaj żartów. Pozwoliłoby to podzielić osoby według upodobań, a następie stworzyć model oceniający dany żart z perspektywy osób lubujących się w danym typie humoru.