Sprawozdanie z laboratorium:
Eksploracji Masywnych Danych

Python projekt

15 lutego 2021

Prowadzący: Anna Labijak-Kowalska

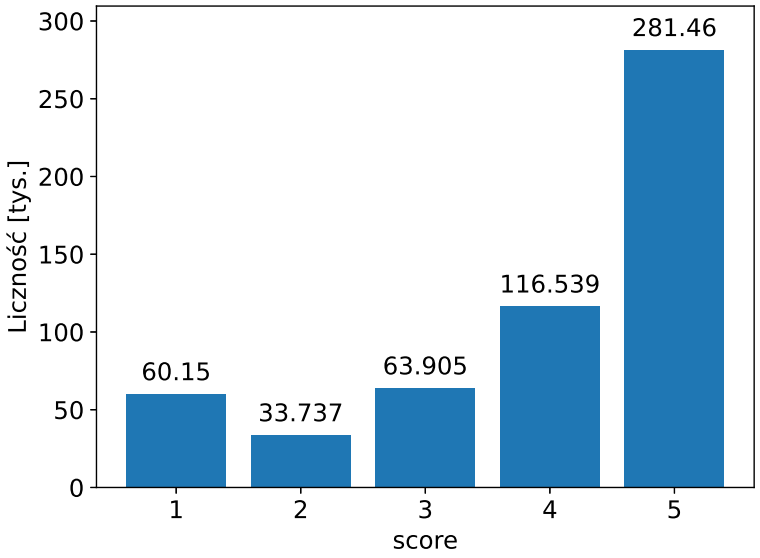
Autorzy: **Jarosław Warmbier** inf132148 ISWD jaroslaw.warmbier@student.put.poznan.pl

1 Opis problemu

Zadaniem jest napiesanie klasyfikatora przewidującego ogólną ocenę na podstawie tekstowej opinii. Naszym zbiorem uczącym jest plik *reviews_train.csv*, który składa się zna następujących atrybutów: reviewerID, asin, reviewerName, helpful, reviewText, summary, unixReviewTime, reviewTime.

2 Wstępna analiza danych i preprocessing

Zbiór danych zawiera około 556 tysięcy opinii, którego rozkład opinii jest przedstawiony na Rys. 1. Klasa większościowa to ocena 5.0, która stanowi około 51 % ocen, kolejną najczęstszą oceną jest 4.0 (około 21 %).



Rysunek 1: Liczność poszczególnych ocen

Do dalszej analizy zostanie wykorzystany tylko atrybut reivewText oraz klasa decyzyjna score. Usuwamy wiersze, w których nie została podana opinia słownie w komórce reviewText. Wszystkie znaki w opiniach zostały zamienione na małe litery oraz nazwy ewentualnych znaków specjalnych na odpowiednie symbole. Przy użyciu klasy *ReviewTokenizer* zostały w słowach odizolowane wszystkie hastami, linki oraz emotikony. Na koniec każde słowo zostało przetworzone przez *PorterStem*, a następnie usunięto stopwords.

Wykorzystano BoW (ang. Bag of Words) do reprezentacji wszystkich dokumentów. W celu predykcji klasy obiektu wykorzystano słowa, które najczęściej pojawiały się w zbiorze danych uczących z licznością co najmniej 1000 razy, po filtracji liczba tych słów wynosiła około 1300.

3 Wyniki działania

3.1 Dummy

Klasyfikator z danych uczący wybiera najczęściej występującą klasę i taką klasę przypisuje do wszystkich dokumentów. Dostrzegamy, że najczęściej występującą klasą jest wynik 5.0.

Tabela 1: Wyniki działania algorytmu Dummy

Score	**1.0**	**2.0**	**3.0**	**4.0**	**5.0**
F1	0.00	0.00	0.00	0.00	0.6733
Precision	0.00	0.00	0.00	0.00	0.5075
Recall	0.00	0.00	0.00	0.00	1.00

3.2 NBC

Klasyfikator NBC uzyskuje znacznie lepsze rezultaty od klasyfikator Dummy. Z największą dokładnością klasyfikuje dokumenty z oceną 5.0 i 1.0.

Tabela 2: Wyniki działania algorytmu NBC

Score	**1.0**	**2.0**	**3.0**	**4.0**	**5.0**
F1	0.5649	0.1823	0.3216	0.3478	0.7532
Precision	0.5000	0.2870	0.3940	0.4196	0.6929
Recall	0.6492	0.1336	0.2716	0.2969	0.8250

3.3 SVM

Tabela 3: Wyniki działania algorytmu SVM

Score	**1.0**	**2.0**	**3.0**	**4.0**	**5.0**
F1	0.5830	0.2595	0.3582	0.3695	0.7677
Precision	0.4863	0.2592	0.3941	0.4504	0.7478
Recall	0.7275	0.2597	0.3283	0.3133	0.7887

3.4 RandomForest

Klasyfikator RandomForest był uczony na losowej 20 % próbce danych wejściowych z parametrami:

- n_estimators = 300,
- n_jobs=-1,
- random_state=23.

Tabela 4: Wyniki działania algorytmu RandomForest

Score	**1.0**	**2.0**	**3.0**	**4.0**	**5.0**
F1	0.6434	0.3326	0.4544	0.4391	0.7847
Precision	0.6499	0.9201	0.7042	0.6763	0.6666
Recall	0.6370	0.2030	0.3354	0.3250	0.9536

4 Uruchomienie

W przypadku uruchomienia zbioru na innych danych testowych trzeba podmienić plik *reviews_test.csv* i uruchomić skrypt *main.py*.

5 Podsumowanie

Najlepsze rezultaty na danych treningowych uzyskał algorytm RandomForest. Dalszym kierunkiem rozwoju klasyfikowania dokumentów jest próba wykorzystania miary TF.IDF oraz spróbować wydobyć wiedzę z innych atrybutów.