Eksploracja Masywnych Danych - Python

Krzystof Pasiewicz 132302, Bartosz Mila 131804

31 stycznia 2021

1 Wykorzystane atrybuty

Wykorzystywany zbiór danych składał się z następujących pól:

- reviewerID - identyfikator użytkownika, który napisał opinię,
- asin - identyfikator produktu (aplikacji),
- reviewerName - użytkownika, który napisał opinię,
- helpful - oceny innych czy opinia była pomocna,
- reviewText - treść recenzji,
- summary - streszczenie recenzji,
- unixReviewTime - czas udostępnienia recenzji,
- reviewTime - czas udostępnienia recenzji,
- score - ocena ogólna.

Spośród tych atrybutów, duża część jest atrybutami jednoznacznie identyfikującymi dany rekord tak wiec ich wykorzystanie w tak trudnym problemie może skutkować znaczącym przeuczeniem klasyfikatora co będzie skutkować bardzo złymi wynikami w trakcie testów. W zwiazku z tym w trakcie przetwarzania wykorzystaliśmy następujące atrybuty:

- reviewText,
- summary,
- score

W dalszych etapach an tych atrybutach zastosowano inżynierię cech w celu ich przetworzenia i zdobycia dodatkowej wiedzy o problemie.

2 Przetwarzanie danych

W pierwszym kroku sprawdzone zostało w ilu spośród posiadanych atrybutów występują wartości null, które uniemożliwiają nam jakiekolwiek działanie na poszczególnych przypadkach. Dla analizowanego zbioru otrzymano następujace ilości wartości null w poszczególnych atrybutach:

- reviewText: 8
- summary: 24
- score: 0

Następnie treści poszczególnych recenzji były dzielone na tokeny oraz poddawane odpowiedniej obróbce. Proces ten składał się z następujących kroków:

1. usunięcie znaków specjalnych z treści recenzji (kropki, przecinki, itp.),
2. usunięcie z treści wszystkich słów należących do angielskiego zbioru tzw. Stop Words (słowa z języka angielskiego, które same nie mają żadnego znaczenia jak np. a, the itp.),
3. zmiana wszystkich wielkich liter na małe,
4. stemming oraz lematyzacja poszczególnych tokenów (usunięcia końcówek odpowiadających za czas przeszły, czy zależne od kontekstu),
5. uzyskane tokeny łączone są w nowy tekst który poddany zostanie ostatniej fazie obróbki.

Ostatnim krokiem przetwarzania danych jest wykorzystanie dwóch prostych modeli służących do przetworzenia recenzji do modelu Bag of Words. Model "Bag of words" to sposób prezentacji danych tekstowych składający się z dwóch głównych elementów:

- zbioru słownictwa,
- miara "obecności" danego słowa w analizowanym tekście,

W naszym problemie wzięliśmy pod uwagę dwa sposoby przetworzenia recenzji do modelu Bag of Words.

Obiekt Count Vectorizer tworzy najprostszy możliwy model zliczając wystąpienia wszystkich słów w tekście (zebranych poprzez analizę wszystkich recenzji).

Obiekt Tfidf Vectorizer tworzy bardziej złożony model danych opierając go na unikaniu karaniu często występujących słów niosących za sobą niewielką ilość informacji co zapobiega zdominowaniu wyników przez takie właśnie odstające od średniej obserwacje. Wykorzystuje on dwie miary:

- term frequency - częstotliwość słowa w dokumencie,
  TF = (Ilość wystąpień słowa T w tekście)/(ilość wszystkich słów w tekście)
- inverse document frequency - rzadkość występowania słowa wśród dokumentów,
  IDF = 1+log(N/n), gdzie:
  - N - ilość analizowanych dokumentów,
  - n - ilość dokumentów, w których wystąpiło dane słowo

3 Zakres badań

W celu znalezienia najlepszego modelu wykorzystujemy metodę przeszukiwania siatki oraz walidację krzyżową w celu znalezienia najbardziej skutecznego modelu i jego parametrów spośród następujących modeli Uczenia Maszynowego:

- Regresja Logistyczna,
- Drzewo decyzyjne,
- K najbliższych sąsiadów,
- Losowy las,

W celu oceny jakości klasyfikacji obliczane są następujące miary jakości:

- trafność,
- miara f1,

Dodatkowo prezentowana jest również macierz omyłek oraz szczegółowe raporty pozwalające na dokładną analizę takich miar dla poszczególnych klas jak:

- precyzja,
- wsparcie,
- recall,

Wszystkie te miary dają nam najlepsze ogólne pojęcie o naszej jakości klasyfikacji a także o ilości zarówno przewidywani True Positive, False Positive, True Negative oraz False Negative. CO pozwoli nam najskuteczniej określić jakość działania naszego modelu. Dodatkowo dzięki zastosowaniu w trakcie testowania walidacji krzyżowej korzystającej z 4 warstw, możemy uznawać otrzymane wyniki za miarodajne względem zbioru testowego.

W metodzie przeszukiwania siatki wykorzystano następujące zbiory parametrów dla poszczególnych klasyfikatorów:

```
parameters={
      'Logistic Regression':{
                              "C":np.logspace(-3,3,7),
                              "penalty":["l1","l2"]},


        'Decision Tree':{
             'criterion' :['gini', 'entropy'],
             'splitter':['best', 'random'],
              'max_depth' : [4,5,6,7,8],
              'max_features': ['auto', 'sqrt', 'log2']}

        'K-NN':  {
              'n_neighbors': [3,5,8,10,12],
              'weights': ['uniform', 'distance'],
              'metric': ['euclidean', 'manhattan']},

        'Random Forest': {
                        'n_estimators': [100, 200, 350, 500],
                        'max_features': ['auto', 'sqrt', 'log2'],
                        'max_depth' : [4,5,6,7,8],
                        'criterion' :['gini', 'entropy']}
}
```

Rysunek 1: Prezentacja poszczególnych parametrów testowanych w metodzie przeszukiwania siatki

4 wyniki

W trakcie próby analizy poszczególnych algorytmów okazało się, że przetestowanie dwóch spośród wybranych modeli na tak dużym i skomplikowanym zbiorze danych jest zbyt czasochłonne oraz wymaga zbyt dużej ilości zasobów, aby skutecznie przeanalizować model K-NN oraz Drzewa Decyzyjnego. Obliczenia rzędu 24h nie przynosiły oczekiwanych rezultatów, dlatego na tym etapie zostały one odrzucone i nie były dalej rozważane jako potencjalne rozwiązania podjętego problemu. Pozostałe algorytmy zostały przetestowane na pełnym zbiorze zdanych parametrów oraz na zbiorze danych przetworzonym zarówno za pomocą metody Count Vectorizer jak i TF-IDF Vectorizer. Wyniki dla poszczególnych modeli zostały zaprezentowane poniżej:

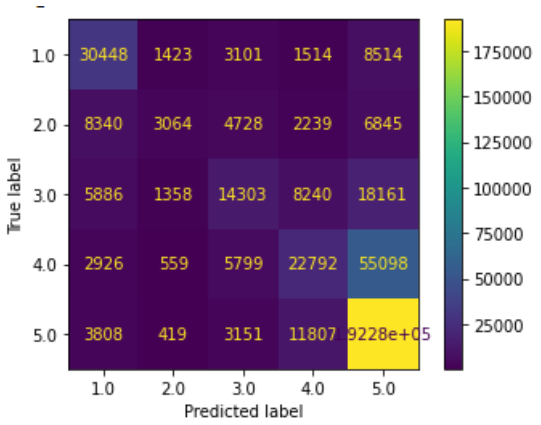
```
########## Logistic Regression ########
Confusion matrix
[[ 30448   1423   3101   1514   8514]
 [  8340   3064   4728   2239   6845]
 [  5886   1358  14303   8240  18161]
 [  2926    559   5799  22792  55098]
 [  3808    419   3151  11807 192285]]
_________________________________________________
REPORT:
              precision    recall  f1-score   support

         1.0       0.59      0.68      0.63     45000
         2.0       0.45      0.12      0.19     25216
         3.0       0.46      0.30      0.36     47948
         4.0       0.49      0.26      0.34     87174
         5.0       0.68      0.91      0.78    211470

    accuracy                           0.63    416808
   macro avg       0.54      0.45      0.46    416808
weighted avg       0.59      0.63      0.59    416808

Accuracy score: 63.0%
F1 score: 0.4613415838094978
```

Rysunek 2: Wyniki modelu regresji logistycznej z użyciem Count Vectorizer



Rysunek 3: Wizualizacja macierzy pomyłek regresji logistycznej z użyciem Count Vectorizer

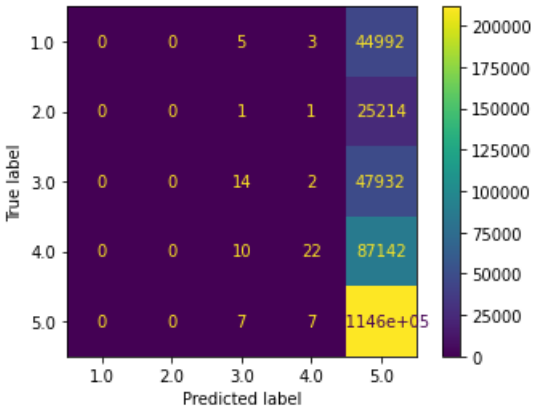
```
########## Decision Tree ########
/usr/local/lib/python3.6/dist-packages/sklearn/metrics,
  _warn_prf(average, modifier, msg_start, len(result))
Confusion matrix
[[     0      0      5      3  44992]
 [     0      0      1      1  25214]
 [     0      0     14      2  47932]
 [     0      0     10     22  87142]
 [     0      0      7      7 211456]]
_________________________________________________
REPORT:
              precision    recall  f1-score   support

         1.0       0.00      0.00      0.00     45000
         2.0       0.00      0.00      0.00     25216
         3.0       0.38      0.00      0.00     47948
         4.0       0.63      0.00      0.00     87174
         5.0       0.51      1.00      0.67    211470

    accuracy                           0.51    416808
   macro avg       0.30      0.20      0.13    416808
weighted avg       0.43      0.51      0.34    416808

Accuracy score: 51.0%
F1 score: 0.13485879473245505
```

Rysunek 4: Wyniki modelu Drzewa decyzyjnego z użyciem Count Vectorizer



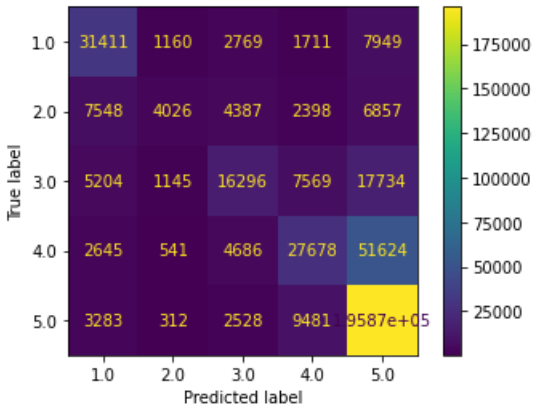
Rysunek 5: Wizualizacja macierzy pomyłek Drzewa decyzyjnego z użyciem Count Vectorizer

```
########## Logistic Regression ########
Confusion matrix
[[ 31411   1160   2769   1711   7949]
 [  7548   4026   4387   2398   6857]
 [  5204   1145  16296   7569  17734]
 [  2645    541   4686  27678  51624]
 [  3283    312   2528   9481 195866]]
_________________________________________________
REPORT:
              precision    recall  f1-score   support

         1.0       0.63      0.70      0.66     45000
         2.0       0.56      0.16      0.25     25216
         3.0       0.53      0.34      0.41     47948
         4.0       0.57      0.32      0.41     87174
         5.0       0.70      0.93      0.80    211470

    accuracy                           0.66    416808
   macro avg       0.60      0.49      0.51    416808
weighted avg       0.64      0.66      0.62    416808
```

Rysunek 6: Wyniki modelu regresji logistycznej z użyciem TF-IDF Vectorizer



Rysunek 7: Wizualizacja macierzy pomyłek regresji logistycznej z użyciem TF-IDF Vectorizer

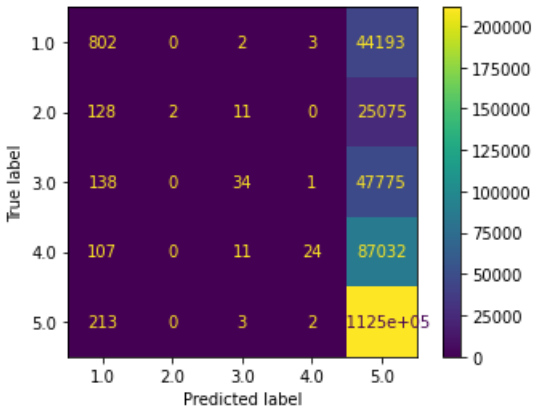
```
########## Decision Tree ########
Confusion matrix
[[   802      0      2      3  44193]
 [   128      2     11      0  25075]
 [   138      0     34      1  47775]
 [   107      0     11     24  87032]
 [   213      0      3      2 211252]]
_________________________________________________
REPORT:
              precision    recall  f1-score   support

         1.0       0.58      0.02      0.03     45000
         2.0       1.00      0.00      0.00     25216
         3.0       0.56      0.00      0.00     47948
         4.0       0.80      0.00      0.00     87174
         5.0       0.51      1.00      0.67    211470

    accuracy                           0.51    416808
   macro avg       0.69      0.20      0.14    416808
weighted avg       0.61      0.51      0.35    416808

Accuracy score: 51.0%
F1 score: 0.14215433565366756
```

Rysunek 8: Wyniki modelu Drzewa decyzyjnego z użyciem TF-IDF Vectorizer



Rysunek 9: Wizualizacja macierzy pomyłek Drzewa decyzyjnego z użyciem TF-IDF Vectorizer

Jak łatwo zauważyć model drzewa decyzyjnego całkowicie nie sprawdza się w naszym problemie, nie wnosząc żadnej dodatkowej wiedzy względem metody przypisywania klasy większościowej (znacząca ilość wszystkich przypadków została zaklasyfikowana do klasy 5.0). Wynika z tego wniosek, iż model ten jest dla nas całkowicie bezużyteczny. Drugim wnioskiem jaki możemy łatwo dostrzec to wyższość metody TF-IDF służącej do wektoryzacji tekstu. Osiągnęła ona lepsze wyniki w obu przeanalizowanych modelach lepsze wyniki na wszystkich analizowanych metrykach. Jednakże jedynie model Regresji logistycznej daje nam dodatkową wiedzę o problemie. Osiągnięte wyniki nie są wybitnie dobre i zadowalające, jednak postawiony problem był bardzo trudny i stworzenie odpowiedniego modelu wynikałoby długich czasów obliczeń oraz długich dni badań.

Model Regresji logistycznej najlepsze wyniki osiągnął dla parametrów:

- C = 1.0
- penalty = l2

Scenariusz testowania

Wszystkie przeprowadzone badania były wykonywane poprzez użycie metod przeszukiwania siatki oraz walidacji krzyżowej. Pozwala to na uzyskanie dobrych i miarodajnych wartości metryk poprzez dzielenie zbioru w trakcie testów na poszczególne warstwy (pomniejsze równe zbiory danych). W przypadku naszych testów zastosowaliśmy 4 warstwy. Następnie model trenowany jest na 3 z tych warstw oraz testowany na pozostałej. Warstwy są zmieniane w każdej iteracji, co pozwala na uśrednienie uzyskanych metryk.

Dodatkowo na początku przetwarzania podzieliliśmy nasz zbiór danych na testowy i treningowy. Zbiór treningowy wykorzystany został do wyboru najlepszego modelu, który następnie dodatkowo przetestowaliśmy na naszym zbiorze testowym. Wyniki tego testu przedstawiono poniżej:

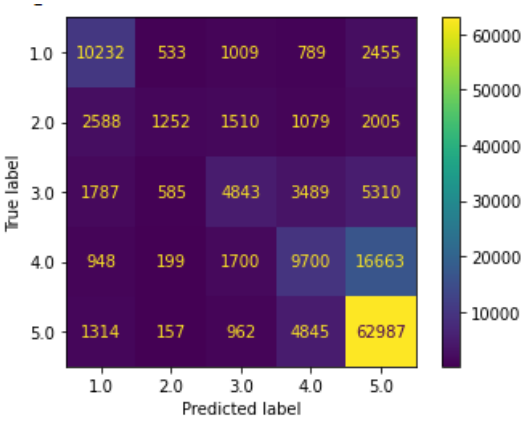
```
Confusion matrix
[[10232   533  1009   789  2455]
 [ 2588  1252  1510  1079  2005]
 [ 1787   585  4843  3489  5310]
 [  948   199  1700  9700 16663]
 [ 1314   157   962  4845 62987]]
_________________________________________________
REPORT:
              precision    recall  f1-score   support

         1.0       0.61      0.68      0.64     15018
         2.0       0.46      0.15      0.22      8434
         3.0       0.48      0.30      0.37     16014
         4.0       0.49      0.33      0.40     29210
         5.0       0.70      0.90      0.79     70265

    accuracy                           0.64    138941
   macro avg       0.55      0.47      0.48    138941
weighted avg       0.61      0.64      0.61    138941

Accuracy score: 64.0%
F1 score: 0.48440797540069036
```

Rysunek 10: Wyniki najlepszego modelu na zbiorze testowym



Rysunek 11: Wizualizacja macierzy pomyłek najlepszego modelu na zbiorze testowym

Uruchamianie testów W celu sprawniejszego przetwarzania cały kod jak i testy zostały przygotowane w notatniku Jupyter w środowisku Google Colaboratory. Notatnik ReviewsCreate zawiera kod wykorzystane do testów i wyboru modeli, natomiast notatnik TestReview zawiera kod wymagany do przetestowania modelu. W notatniku testowym druga komórka zawiera trzy zmienne niezbędne do poprawnego działania:

- vect_path - plik z zapisanym obiektem vectorizera (dostarczony w plikach),
- clf_path - plik z zapisanym obiektem klasyfikatora (dostarczony w plikach),
- data_path - ścieżka do pliku z danymi,

```
vect_path = 'models/tfvectorizer.sav'
clf_path = 'models/logistic_regression_tf.sav'
data_path = "data/test.csv"
```

Rysunek 12: Parametry do ustawienia dla testów