

# raport-1

November 30, 2021

Adrian Poniatowski, nr. 401346

## 0.1 Podstawy analizy języka naturalnego

Celem ćwiczenia jest zapoznanie się z prostymi metodami analizy języka naturalnego. W tym celu będziemy wykorzystywać metodę Bag of Words, zaimplementowaną jako klasa `CountVectorizer` pakietu `sci-kit learn`.

Będziemy analizować bazę danych zawierającą informacje o filmach. Interesujące nas tabele to:

`genres ( genre_id bigint, name text COLLATE pg_catalog.“default” )`

`genres_movies ( id bigint, movie_id bigint, genre_id bigint )`

`movies ( movie_id bigint, plot text COLLATE pg_catalog.“default”, movie_name text COLLATE pg_catalog.“default” )`

Dostęp do bazy umożliwia

```
[1]: import sqlalchemy
import pandas as pd
from sqlalchemy import create_engine
db_string = "postgresql://wbauer_adb:adb2020@pgsql-196447.vipserv.org:5432/
↳wbauer_classifier_abd"
db = create_engine(db_string)
```

1. Ponieważ baza jest dość duża należy ograniczyć się do 10% danych (wybranych losowo):

```
SELECT setseed(0.5);
SELECT ...
FROM movies
JOIN genres_movies ON ...
JOIN genres ON ...
WHERE random()>0.9
```

Zapytanie uzupełnić samodzielnie tak, aby uzyskać, tytuł, opis fabuły i gatunek filmu.

```
[2]: select_string='SELECT setseed(0.5); \
SELECT movies.movie_name as title, movies.plot, genres.name \
FROM movies \
JOIN genres_movies ON movies.movie_id = genres_movies.movie_id \
JOIN genres ON genres_movies.genre_id = genres.genre_id \
```

```
WHERE random()>0.9'

df=pd.read_sql(select_string,con=db)

print(df)
```

	title \	plot	name
0	Flåklypa Grand Prix	In the town of Flåklypa , the inventor Reodo...	Stop motion
1	Saving Private Ryan	On the morning of June 6, 1944, the beginning ...	Ensemble Film
2	The Rock	A group of rogue Force Recon Marines led by di...	Thriller
3	The Silence of the Lambs	Clarice Starling is pulled from her training ...	Thriller
4	The Terminator	In a post-apocalyptic 2029, artificially inte...	Thriller
...	...	...	...
4246	Tomchi	Tomchi is a story of six notorious children wh...	Drama
4247	Every 9 Seconds	The films deal with a woman named Carrie , a c...	Drama
4248	Shanghai Calling	Sam Chao , an ambitious New York attorney, is ...	Drama
4249	Shark Attack 3: Megalodon	When two researchers discover a colossal shark...	Fantasy
4250	Mr. Bechara	Anand Verma, a widower and father of a child, ...	Comedy film

[4251 rows x 3 columns]

2. Spośród 10 najpopularniejszych gatunków filmów wybrać jeden ( liczba liter w imieniu i nazwisku modulo 10). I utworzyć kolumnę przypisującą 0 gdy film jest tego gatunku i odwrotnie.

```
[3]: select_string2 = ("SELECT genres.name, COUNT(movies.movie_name) \
                        FROM movies \
                        JOIN genres_movies ON movies.movie_id = genres_movies.
↳movie_id \
                        JOIN genres ON genres_movies.genre_id = genres.genre_id \
                        GROUP BY genres.name ORDER BY count DESC")
z2=pd.read_sql(select_string2,con=db)
print("10 najpopularniejszych gatunków filmowych dla całej bazy:\n")
print(z2.head(10))
print("\nWybrany numer gatunku: " + str(len('adrianponiatowski')%10))
```

```

print("Wybrany gatunek: Comedy")

# funkcja do przypisywania odpowiednich wartości
def is_comedy_fun(df, df2, genre_id):
    result = []
    for i in range(df.shape[0]):
        if df['name'][i] == df2['name'][genre_id-1]:
            result.append(0)
        else:
            result.append(1)
    return result

#print(df.shape[0])
df['is_comedy'] = is_comedy_fun(df, z2, 7)
#print(df.head(20))

```

10 najpopularniejszych gatunków filmowych dla całej bazy:

	name	count
0	Thriller	5342
1	Drama	5293
2	Crime Fiction	2927
3	Short Film	2920
4	Romantic comedy	1930
5	Romance Film	1710
6	Comedy	1192
7	Horror	1125
8	Action/Adventure	1091
9	Science Fiction	992

Wybrany numer gatunku: 7

Wybrany gatunek: Comedy

3. Utworzyć model bag of words, usuwając akcenty, stopwords i zmniejszając wszystkie litery na małe. Ograniczyć się do 20 najpopularniejszych słów.

```

[4]: from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(strip_accents='unicode', lowercase=True,
    ↪ stop_words='english', max_features=20)
X = vectorizer.fit_transform(df['plot'])
vectorizer.get_feature_names_out()

```

```

[4]: array(['away', 'day', 'family', 'father', 'film', 'finds', 'goes', 'home',
    'house', 'later', 'life', 'love', 'man', 'mother', 'new', 'police',
    'son', 'takes', 'tells', 'time'], dtype=object)

```

4. Przekształcić opisy filmów do danej postaci za pomocą metody `.fit_and_transform`
5. Dokonać podziału na zbiór uczący i testowy

```
[5]: from sklearn.model_selection import train_test_split
y = df['is_comedy']
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.33,
↳random_state=42)
```

6. Utworzyć model regresji logistycznej i nauczyć go rozpoznawać wybrany gatunek filmu.
7. Przetestować go na zbiorze uczącym i obliczyć “*confusion matrix*”

```
[14]: from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, roc_curve
import numpy as np

clf = LogisticRegression().fit(X_train, y_train)

y_pred = clf.predict(X_test)
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)

score = clf.score(X_test, y_test)
print("\nŚrednia trafność modelu: " + str(score*100) + "%")
```

```
[[ 0  30]
 [ 0 1373]]
```

Średnia trafność modelu: 97.86172487526729%

Wycudzony model rozpoznaje kategorie w niecałych 98%. Można powiedzieć, że jest to dobry wynik. W moim przypadku jedyne przewidywania które wystąpiły to stwierdzenie, że dany opis nie opisuje wybranej kategorii (było to 1373 przypadków) oraz sytuacja, w której model stwierdził, że opis filmu nie dotyczy wybranej kategorii, choć w rzeczywistości dotyczył. Niepokojący jest fakt, że model nie rozpoznał żadnego przypadku, kiedy to algorytm słusznie określił wybraną kategorię.

8. Używając metody `.predict_proba()` obliczyć jakie prawdopodobieństwo klasyfikator obliczył dla wyników fałszywie dodatnich i fałszywie ujemnych. Wynik zwizualizować.

```
[11]: prob = clf.predict_proba(X_test)
prob
```

```
[11]: array([[0.03274314, 0.96725686],
          [0.0015526 , 0.9984474 ],
          [0.04564161, 0.95435839],
          ...,
          [0.03112404, 0.96887596],
          [0.00679285, 0.99320715],
          [0.03816395, 0.96183605]])
```

```
[16]: # Z powodu nie występowania wyników fałszywie ujemnych, obliczę tylko
↳prawdopodobieństwo dla wyników fałszywie dodatnich.
```

```

prob_fp = [prob[i, 0] for i in range(len(y_test)) if np.array(y_test)[i] == 0
↳and y_pred[i] == 1]
prob_fp

```

```

[16]: [0.0012270245213454878,
      0.029758729806353523,
      0.009315688852523207,
      0.003183789202257814,
      0.04706770775982194,
      0.02404394641592622,
      0.0201511206307301,
      0.016422253663363806,
      0.03527148266459901,
      0.026516238738184983,
      0.04706770775982194,
      3.6523255323439585e-05,
      0.05728986513007994,
      0.058571349291988706,
      0.025642642774745372,
      0.04706770775982194,
      0.023714099635541275,
      0.00494520532761733,
      0.045064707019931594,
      0.051194583040174435,
      0.021669037852128037,
      0.03579080558919967,
      0.04706770775982194,
      0.012374905078543885,
      0.036406252463744404,
      0.026108854693115302,
      0.037013833043484934,
      0.03571534066047244,
      0.04349610647806745,
      0.024808893438704294]

```

```

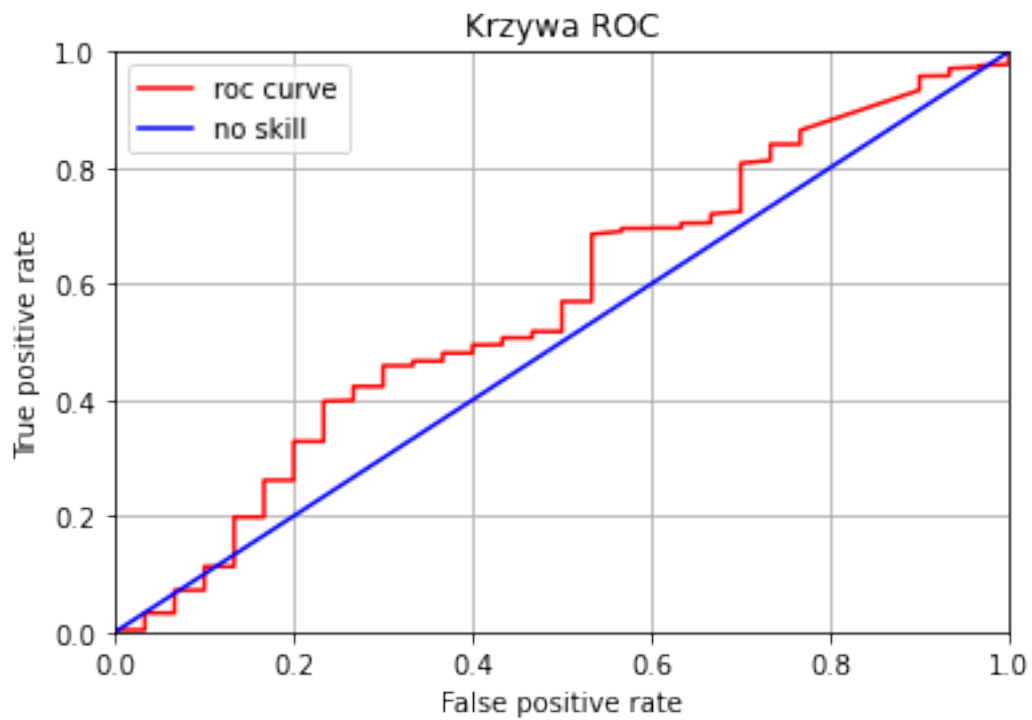
[18]: # Do wizualizacji posłuży mi funkcja roc_curve, której funkcjonalność oraz
↳przykłady użycia znalazłem w internecie
import matplotlib.pyplot as plt

fpr, tpr, _ = roc_curve(y_test, prob[:, 1])

plt.plot(fpr, tpr, 'r', label='roc curve')
plt.plot([0, 1], [0, 1], 'b', label='no skill')
plt.grid()
plt.axis([0, 1, 0, 1])
plt.ylabel('True positive rate')
plt.xlabel('False positive rate')

```

```
plt.title('Krzywa ROC')
plt.legend()
plt.show()
```



Analizując powyższy wykres należy stwierdzić, że występowanie czerwonej linii nad linią niebieską oznacza, że model pewnych rzeczy się wyuczył. Jest to jednak średni stopień wyuczenia, ponieważ im dalej linia “roc curve” od linii “no skill” tym lepiej nauczony jest model.

[ ]: