Современные инструменты анализа данных
Лабораторная работа №2

Анализ текста

ФИО студента: Готовко Алексей Владимирович
Направление подготовки: 09.03.04 (СППО)
Учебная группа: P32101
Практик: Змиевский Данил Александрович

Санкт-Петербург
2022г.

1 Настройка среды

```python
import string
from random import Random
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import *

nltk.download('omw-1.4')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('stopwords')

rng = Random()
data = pd.read_csv("dataset.csv")
```

2 Генерация варианта

```python
rng.seed(1337)
first = rng.choice(genres)
rng.seed(7331)
second = rng.choice(genres)
print("{} and {}".format(first, second))
# Rock and Hip-Hop
```

3 Нормализация данных

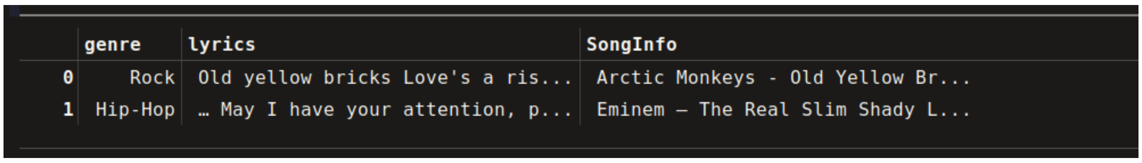
```
# --- Filtering ---

columns = all_columns[(all_columns["genre"] == "Rock") | (all_columns["genre"] == "Hip-
                                                  Hop")]
# normalised = columns["lyrics"].str.lower()
# ^^^ do not remove punctuation
normalised = columns["lyrics"].str.replace('[{}]'.format(string.punctuation), '', regex=
                                                  True).str.lower()
# ^^^ remove punctuation

columns["normalised"] = normalised
columns["normalised"]
# 181      yet our best trained best educated best equip...
# 182      backstroke lover always hidin neath the cover...
# 183      intro  fuck all yall hoes  get a grip motherf...
# 184      one two three and to the fo  snoop doggy dogg...
# 185      you are now about to witness the strength of ...
#                             ...
# 462      cant explain all the feelings that youre maki...
# 463      one foot on the brake and one on the gas hey ...
# 464      carry on my wayward son therell be peace when...
# 465      ooh yeah turn it up come on  im working hard ...
# 466      out on the road for forty days last night in ...
# Name: normalised, Length: 186, dtype: object

# --- Tokenising ---

columns["tokened"] = columns.apply(lambda row: nltk.word_tokenize(row['normalised']),
                                                  axis=1)
columns["tokened"]
# 181     [yet, our, best, trained, best, educated, best...
# 182     [backstroke, lover, always, hidin, neath, the,...
# 183     [intro, fuck, all, yall, hoes, get, a, grip, m...
# 184     [one, two, three, and, to, the, fo, snoop, dog...
# 185     [you, are, now, about, to, witness, the, stren...
#                             ...
# 462     [cant, explain, all, the, feelings, that, your...
# 463     [one, foot, on, the, brake, and, one, on, the,...
# 464     [carry, on, my, wayward, son, therell, be, pea...
# 465     [ooh, yeah, turn, it, up, come, on, im, workin...
# 466     [out, on, the, road, for, forty, days, last, n...
# Name: tokened, Length: 186, dtype: object

# --- Removing stop-words ---

noise = stopwords.words("english")
noiseless = columns["tokened"].apply(lambda x: [item for item in x if item not in noise]
                                                  )
noiseless_col = [", ".join(w) for w in noiseless]
columns["noiseless"] = noiseless_col
columns["noiseless"]
# 181     yet, best, trained, best, educated, best, equi...
# 182     backstroke, lover, always, hidin, neath, cover...
# 183     intro, fuck, yall, hoes, get, grip, motherfuck...
# 184     one, two, three, fo, snoop, doggy, dogg, dr, d...
# 185     witness, strength, street, knowledge, 1, ice, ...
#                             ...
# 462     cant, explain, feelings, youre, making, feel, ...
# 463     one, foot, brake, one, gas, hey, well, theres,...
# 464     carry, wayward, son, therell, peace, done, lay...
# 465     ooh, yeah, turn, come, im, working, hard, your...
# 466     road, forty, days, last, night, little, rock, ...
# Name: noiseless, Length: 186, dtype: object

# --- Lemmatisation ---

lemmatiser = WordNetLemmatizer()
lemmatised = columns["noiseless"].apply(lambda x: [lemmatiser.lemmatize(x)])
lemmatised_col = [", ".join(w) for w in lemmatised]
columns["lemmatised"] = lemmatised_col
columns["lemmatised"]
# 181     yet, best, trained, best, educated, best, equi...
# 182     backstroke, lover, always, hidin, neath, cover...
# 183     intro, fuck, yall, hoes, get, grip, motherfuck...
# 184     one, two, three, fo, snoop, doggy, dogg, dr, d...
# 185    witness, strength, street, knowledge, 1, ice, ...
#                               ...
# 462    cant, explain, feelings, youre, making, feel, ...
# 463    one, foot, brake, one, gas, hey, well, theres,...
# 464    carry, wayward, son, therell, peace, done, lay...
# 465    ooh, yeah, turn, come, im, working, hard, your...
# 466    road, forty, days, last, night, little, rock, ...
# Name: lemmatised, Length: 186, dtype: object
```

4 Сегментация данных и обучение модели

```
# --- Segmentation ---

x_train, x_test, y_train, y_test = train_test_split(columns.lemmatised, columns.genre,
                                            train_size = 0.7)
columns.genre.value_counts()
# Rock       95
# Hip-Hop    91
# Name: genre, dtype: int64

# --- Vectorisation ---

vectoriser = CountVectorizer(ngram_range=(1, 3))
vectorised_x_train = vectoriser.fit_transform(x_train)

# --- Classification ---

clf = MultinomialNB()
clf.fit(vectorised_x_train, y_train)

vectorised_x_test = vectoriser.transform(x_test)
clf.predict(vectorised_x_test)
# array(['Hip-Hop', 'Rock', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop',
#         'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop',
#         'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Rock',
#         'Hip-Hop', 'Hip-Hop', 'Rock', 'Rock', 'Hip-Hop', 'Hip-Hop', 'Rock',
#         'Rock', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop',
#         'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop',
#         'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Rock',
#         'Hip-Hop', 'Hip-Hop', 'Rock', 'Hip-Hop', 'Hip-Hop', 'Rock',
#         'Hip-Hop', 'Hip-Hop', 'Hip-Hop', 'Rock', 'Rock', 'Hip-Hop',
#         'Hip-Hop'], dtype='<U7')

pred = clf.predict(vectorised_x_test)
print(classification_report(y_test, pred))
#               precision    recall  f1-score   support
#
#      Hip-Hop       0.60      1.00      0.75        27
#         Rock       1.00      0.38      0.55        29
#
#     accuracy                           0.68        56
#    macro avg       0.80      0.69      0.65        56
# weighted avg       0.81      0.68      0.65        56
```

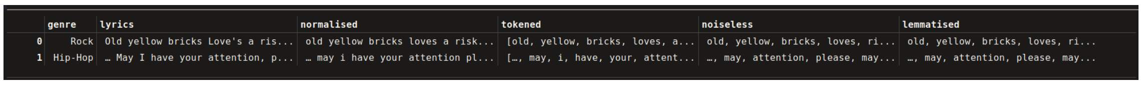
5 Предсказывание жанра двух произвольных песен

```
data =  pd.read_csv("songs.csv")
data.head()
```



	genre	lyrics	SongInfo
0	Rock	Old yellow bricks Love's a ris...	Arctic Monkeys - Old Yellow Br...
1	Hip-Hop	… May I have your attention, p...	Eminem – The Real Slim Shady L...

```
columns = data[["genre", "lyrics"]]

# normalised = columns["lyrics"].str.lower()
# ^^^ do not remove punctuation
normalised = columns["lyrics"].str.replace('[{}]'.format(string.punctuation), '', regex=
                                            True).str.lower()
# ^^^ remove punctuation

columns["normalised"] = normalised

columns["tokened"] = columns.apply(lambda row: nltk.word_tokenize(row['normalised']),
                                            axis=1)

noiseless = columns["tokened"].apply(lambda x: [item for item in x if item not in noise]
                                            )
noiseless_col = [", ".join(w) for w in noiseless]
columns["noiseless"] = noiseless_col

lemmatised = columns["noiseless"].apply(lambda x: [lemmatiser.lemmatize(x)])
lemmatised_col = [", ".join(w) for w in lemmatised]
columns["lemmatised"] = lemmatised_col

columns
```

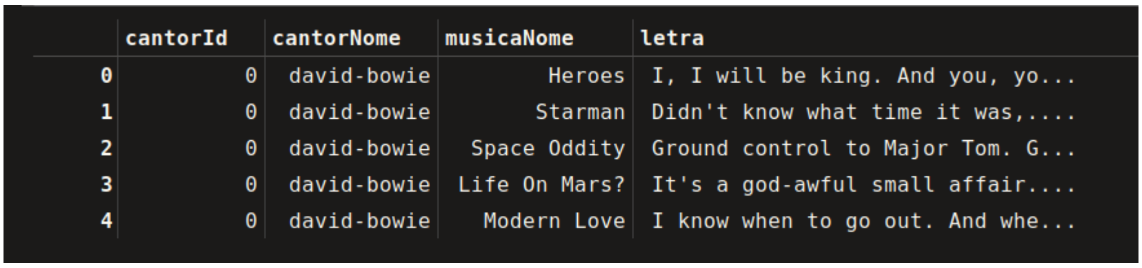


	genre	lyrics	normalised	tokened	noiseless	lemmatised
0	Rock	Old yellow bricks Love's a ris...	old yellow bricks loves a risk...	[old, yellow, bricks, loves, a...	old, yellow, bricks, loves, ri...	old, yellow, bricks, loves, ri...
1	Hip-Hop	… May I have your attention, p...	… may i have your attention pl...	[…, may, i, have, your, attent...	…, may, attention, please, may...	…, may, attention, please, may...

```
x_test = columns[["lemmatised"]].squeeze()
y_test = columns[["genre"]].squeeze()

vectorised_x_test = vectoriser.transform(x_test)

clf.predict(vectorised_x_test)

pred = clf.predict(vectorised_x_test)
print(classification_report(y_test, pred))
#               precision    recall  f1-score   support
#
#      Hip-Hop       1.00      1.00      1.00         1
#         Rock       1.00      1.00      1.00         1
#
#     accuracy                           1.00         2
#    macro avg       1.00      1.00      1.00         2
# weighted avg       1.00      1.00      1.00         2
```

6 Классификация песен Дэвида Боуи и Пола Маккартни

```
data =  pd.read_csv("bowie-from-mccartney.csv")
data.head()
```



	cantorId	cantorNome	musicaNome	letra
0	0	david-bowie	Heroes	I, I will be king. And you, yo...
1	0	david-bowie	Starman	Didn't know what time it was,....
2	0	david-bowie	Space Oddity	Ground control to Major Tom. G...
3	0	david-bowie	Life On Mars?	It's a god-awful small affair....
4	0	david-bowie	Modern Love	I know when to go out. And whe...

```
columns = data[["cantorNome", "letra"]]

columns = columns[(columns["cantorNome"] == "david-bowie") | (columns["cantorNome"] == "
                                            paul-mccartney")]

# normalised = columns["letra"].str.lower()
# ^^^ do not remove punctuation
normalised = columns["letra"].str.replace('[{}]'.format(string.punctuation), '', regex=
                                            True).str.lower()
# ^^^ remove punctuation

columns["normalised"] = normalised

columns["tokened"] = columns.apply(lambda row: nltk.word_tokenize(row['normalised']),
                                            axis=1)

noiseless = columns["tokened"].apply(lambda x: [item for item in x if item not in noise]
                                            )
noiseless_col = [", ".join(w) for w in noiseless]
columns["noiseless"] = noiseless_col

lemmatised = columns["noiseless"].apply(lambda x: [lemmatiser.lemmatize(x)])
lemmatised_col = [", ".join(w) for w in lemmatised]
columns["lemmatised"] = lemmatised_col

x_train, x_test, y_train, y_test = train_test_split(columns.lemmatised, columns.
                                            cantorNome, train_size = 0.7)
columns.cantorNome.value_counts()
# david-bowie       483
# paul-mccartney    464
# Name: cantorNome, dtype: int64

vectorised_x_train = vectoriser.fit_transform(x_train)
clf.fit(vectorised_x_train, y_train)

vectorised_x_test = vectoriser.transform(x_test)
clf.predict(vectorised_x_test)

pred = clf.predict(vectorised_x_test)
print(classification_report(y_test, pred))
                precision    recall  f1-score   support

#    david-bowie       0.74      0.69      0.72       139
# paul-mccartney       0.72      0.77      0.75       146
#
#       accuracy                           0.73       285
#      macro avg       0.73      0.73      0.73       285
#   weighted avg       0.73      0.73      0.73       285
```