Mateusz Kowalski (22777)

Informatyka 6 semestr

Przewidywanie gatunku piosenek na podstawie ich tekstu

Dane

Zbiór danych użyty w tym projekcie pochodzi ze strony:
https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics
Z powyższego zestawu używam tylko kolumny ‘genre oraz ‘lyrics

```
class Lyrics_Data:
    def __init__(self, path):
        self.path = path
        self.data = pd.read_csv(path)#nrows=100000
        self.data = self.data[['genre','lyrics']].dropna()
```

Normalizacja tekstu

Na tekstach piosenek wykonuję:
- usunięcie znaków nowej linii
- usunięcie cyfr
- zmiana na małe litery
- usunięcie interpunkcji
- usunięcie białych znaków
- usunięcie słów pomijanych
- stemming
- lemmatization

Dzielenie Tekstu

```
train_x, valid_x, train_y, valid_y = model_selection.train_test_split(d['lyrics'], d['genre'])
```

Naive Bayes

```
text_clf = Pipeline(
    [('vect', CountVectorizer()),
     ('clf', MultinomialNB(alpha=0.1))])

# train our model on training data
text_clf.fit(train_x, train_y)

# score our model on testing data
predicted = text_clf.predict(valid_x)
np.mean(predicted == valid_y)
```

```
0.4266656662665066
```

```
text_clf = Pipeline(
    [('vect', TfidfVectorizer()),
     ('clf', MultinomialNB(alpha=0.1))])

# train our model on training data
text_clf.fit(train_x, train_y)

# score our model on testing data
predicted = text_clf.predict(valid_x)
np.mean(predicted == valid_y)
```

```
text_clf = Pipeline(
    [('vect', TfidfVectorizer()),
     ('clf', MultinomialNB(alpha=0.2))])

# train our model on training data
text_clf.fit(train_x, train_y)

# score our model on testing data
predicted = text_clf.predict(valid_x)
np.mean(predicted == valid_y)
```

```
0.49747899159663866
```

```
text_clf = Pipeline(
    [('vect', TfidfVectorizer()),
     ('clf', MultinomialNB(alpha=0.08))])

# train our model on training data
text_clf.fit(train_x, train_y)

# score our model on testing data
predicted = text_clf.predict(valid_x)
np.mean(predicted == valid_y)
```

```
0.5103391356542617
```

```
text_clf = Pipeline(
    [('vect', TfidfVectorizer()),
     ('clf', MultinomialNB(alpha=0.15))])

# train our model on training data
text_clf.fit(train_x, train_y)

# score our model on testing data
predicted = text_clf.predict(valid_x)
np.mean(predicted == valid_y)
```

```
0.5014555822328931
```

```
text_clf = Pipeline(
    [('vect', TfidfVectorizer(max_df=0.4,min_df=4)),
     ('clf', MultinomialNB(alpha=0.15))])

# train our model on training data
text_clf.fit(train_x, train_y)

# score our model on testing data
predicted = text_clf.predict(valid_x)
np.mean(predicted == valid_y)
```

```
0.5142707082833133
```

```
text_clf = Pipeline(
    [('vect', TfidfVectorizer(max_df=0.4,min_df=4)),
     ('clf', MultinomialNB(alpha=0.05))])

# train our model on training data
text_clf.fit(train_x, train_y)

# score our model on testing data
predicted = text_clf.predict(valid_x)
np.mean(predicted == valid_y)
```

```
0.5207683073229292
```

```
text_clf = Pipeline(
    [('vect', TfidfVectorizer(max_df=0.4,min_df=4)),
     ('clf', MultinomialNB(alpha=0.08))])

# train our model on training data
text_clf.fit(train_x, train_y)

# score our model on testing data
predicted = text_clf.predict(valid_x)
np.mean(predicted == valid_y)
```

```
0.5179021608643457
```

```python
from sklearn.metrics import precision_recall_fscore_support
genres = list(d['genre'].unique())

precision, recall, fscore, support = precision_recall_fscore_support(valid_y, predicted)
for n,genre in enumerate(genres):
    genre = genre.upper()
    print(genre+'_precision: {}'.format(precision[n]))
    print(genre+'_recall: {}'.format(recall[n]))
    print(genre+'_fscore: {}'.format(fscore[n]))
    print(genre+'_support: {}'.format(support[n]))
    print()
```

```
POP_precision: 0.5503952569169961
POP_recall: 0.15597871744609354
POP_fscore: 0.2430722234344316
POP_support: 3571

HIP-HOP_precision: 0.6666666666666666
HIP-HOP_recall: 0.02594810379241517
HIP-HOP_fscore: 0.049951969260326606
HIP-HOP_support: 2004

NOT AVAILABLE_precision: 0.3150684931506849
NOT AVAILABLE_recall: 0.12387791741472172
NOT AVAILABLE_fscore: 0.17783505154639176
NOT AVAILABLE_support: 557

ROCK_precision: 0.7261219792865362
ROCK_recall: 0.7023374145333121
ROCK_fscore: 0.7140316844487552
ROCK_support: 6289

METAL_precision: 0.8333333333333334
METAL_recall: 0.006337135614702154
METAL_fscore: 0.012578616352201257
METAL_support: 789

OTHER_precision: 0.46534653465346537
OTHER_recall: 0.09073359073359073
OTHER_fscore: 0.1518578352180937
OTHER_support: 2072

COUNTRY_precision: 0.6786274949540255
COUNTRY_recall: 0.5102006407013995
COUNTRY_fscore: 0.5824831568816169
COUNTRY_support: 5931

JAZZ_precision: 0.28781925343811393
JAZZ_recall: 0.19790611279972983
JAZZ_fscore: 0.23454072443466079
JAZZ_support: 5922

ELECTRONIC_precision: 0.1690307328605201
ELECTRONIC_recall: 0.11016949152542373
ELECTRONIC_fscore: 0.13339552238805968
ELECTRONIC_support: 1298

FOLK_precision: 0.535012285012285
FOLK_recall: 0.08697823047733173
FOLK_fscore: 0.14963064765504208
FOLK_support: 10014

R&B_precision: 0.8235294117647058
R&B_recall: 0.017456359102244388
R&B_fscore: 0.034188034188034185
R&B_support: 802

INDIE_precision: 0.505897607495399
INDIE_recall: 0.8831367967580592
INDIE_fscore: 0.6432911830015823
INDIE_support: 27391
```

SGD Classifier

```
from sklearn.linear_model import SGDClassifier

text_clf_svm = Pipeline([('vect', TfidfVectorizer(max_df=0.4,min_df=4)),
                         ('clf-svm', SGDClassifier(loss='hinge', penalty='l2',
                                                   alpha=1e-3, n_iter=5, random_state=42)),
])

text_clf_svm.fit(train_x, train_y)
predicted_svm = text_clf_svm.predict(valid_x)
print(np.mean(predicted_svm == valid_y))
```

```
0.5115096038415367
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
text_clf = Pipeline([('vect', TfidfVectorizer(max_df=0.4,min_df=4)),
                     ('clf', RandomForestClassifier(n_estimators=100, max_depth=2,random_state=0))
        ])

text_clf.fit(train_x, train_y)
predicted_svm = text_clf.predict(valid_x)
print(np.mean(predicted_svm == valid_y))
```

```
0.4112845138055222
```