Рубежный контроль №2

Выполнил: Попков Владислав Евгеньевич, группа ИУ5-21м

Вариант №1. Классификация текстов на основе методов наивного Байеса

Датасет

На Kaggle.com найден Wine Reviews - датасет содержащий описания дегустаторов к различным винам. Задачей будет получение параметра points (оценка по 100 шкале) на основе написанного отзыва. Для этого необходимо выделить два признака, обработать их и затем отправить на обучение

In [17]:

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from typing import Dict, Tuple
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.naive_bayes import MultinomialNB, ComplementNB, BernoulliNB
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
%matplotlib inline
data = pd.read_csv('winemag-data-130k-v2.csv')
data = data[['description', 'points']]
data=data.dropna(axis=0,how='any')
data.head()
data.dtypes
```

Out[17]:

```
description    object
points          int64
dtype: object
```

In [0]:

```
X_train, X_test, y_train, y_test = train_test_split(data['description'],
data['points'], test_size=0.4, random_state=1)
```

In [0]:

```
def calc(v, c):
    model = Pipeline(
        [("vectorizer", v),
         ("classifier", c)])
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(v)
    print("+{}".format(c))
    d = {'t': y_test, 'p': y_pred}
    df = pd.DataFrame(data=d)
    classes = np.unique(y_test)
    res = dict()
    for c in classes:
        temp_data_flt = df[df['t']==c]
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values)
        res[c] = temp_acc
    if len(res)>0:
        print('Points \t Accuracy')
    for i in res:
      if ((i < 85) or (i > 94)):
        pass
      else:
        print('{} \t {:.2%}'.format(i, res[i]))
    print('\n\n')
```

In [0]:

```
classificators = [LogisticRegression(C=5.0), MultinomialNB(), ComplementNB(),
BernoulliNB()]
vectorizers = [TfidfVectorizer(), CountVectorizer()]
```

In [21]:

```
for classificator in classificators:
  for vectorizer in vectorizers:
    calc(vectorizer, classificator)
```

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict')
+LogisticRegression(C=5.0, class_weight=None, dual=False, fit_intercept=True)
Points   Accuracy
85       20.90%
86       23.91%
87       34.00%
88       31.45%
89       18.38%
90       31.46%
91       22.83%
92       24.58%
93       18.65%
94       13.87%

CountVectorizer(analyzer='word', binary=False, decode_error='strict')
+LogisticRegression(C=5.0, class_weight=None, dual=False, fit_intercept=True)
Points   Accuracy
85       22.28%
86       24.16%
87       31.65%
88       29.50%
89       20.61%
90       28.51%
91       23.94%
92       24.79%
93       21.71%
94       19.46%

TfidfVectorizer(analyzer='word', binary=False, decode_error='strict')
+MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
Points   Accuracy
85       0.82%
86       3.19%
87       51.64%
88       51.41%
89       1.20%
90       40.75%
91       4.03%
92       2.53%
93       0.74%
94       0.00%

CountVectorizer(analyzer='word', binary=False, decode_error='strict')
+MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
Points   Accuracy
85       18.13%
86       19.41%
87       35.72%
88       29.72%
89       18.24%
90       31.23%
91       23.86%
92       25.27%
93       15.63%
94       5.32%
```

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict')
+ComplementNB(alpha=1.0, class_prior=None, fit_prior=True, norm=False)
Points   Accuracy
85       22.82%
86       20.72%
87       30.92%
88       22.72%
89       14.67%
90       25.72%
91       23.94%
92       28.61%
93       20.70%
94       10.24%

CountVectorizer(analyzer='word', binary=False, decode_error='strict')
+ComplementNB(alpha=1.0, class_prior=None, fit_prior=True, norm=False)
Points   Accuracy
85       24.03%
86       21.34%
87       29.93%
88       18.60%
89       16.51%
90       21.26%
91       24.45%
92       31.35%
93       26.01%
94       13.20%

TfidfVectorizer(analyzer='word', binary=False, decode_error='strict')
+BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
Points   Accuracy
85       21.57%
86       21.32%
87       39.27%
88       31.43%
89       16.73%
90       30.82%
91       23.02%
92       23.81%
93       13.39%
94       3.84%

CountVectorizer(analyzer='word', binary=False, decode_error='strict')
+BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
Points   Accuracy
85       21.57%
86       21.32%
87       39.27%
88       31.43%
89       16.73%
90       30.82%
91       23.02%
92       23.81%
93       13.39%
94       3.84%
```

Вывод

На основе полученного можно сделать вывод, что лучшим методом в данной ситуации является TfidfVectorizer с MultinomialNB, где удалось правильно оценить рецензии с оценками 87, 88 и 90 в 50% случаев