Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №6
по дисциплине

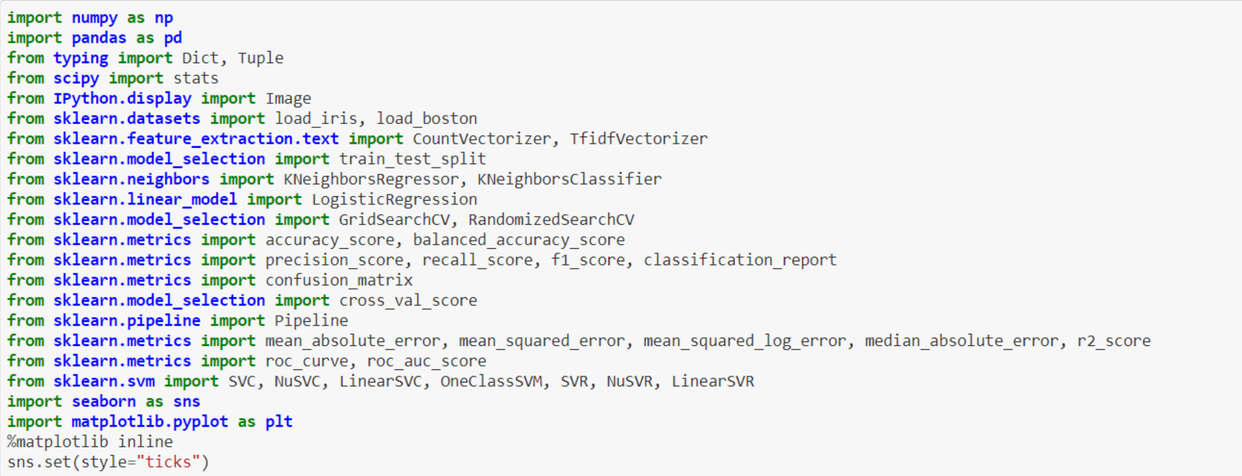
«Методы машинного обучения»

Выполнил:
студент группы ИУ5-21М
Мьоу Зо У

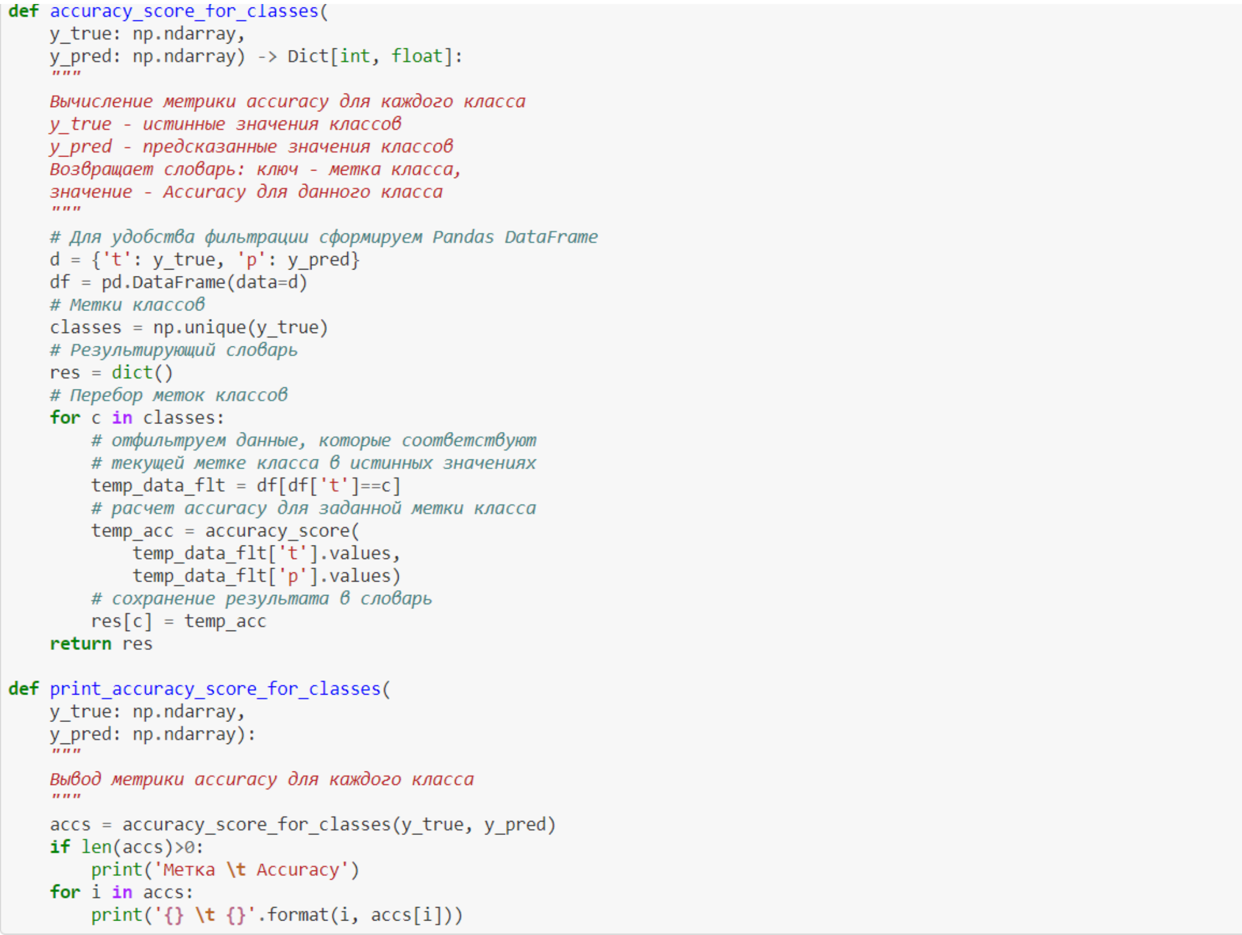
Москва — 2021 г.

1. Цель лабораторной работы

Изучить ансамбли моделей машинного обучения.



```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from IPython.display import Image
from sklearn.datasets import load_iris, load_boston
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_error, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```



```
def accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики accuracy для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Accuracy для данного класса
    """
    # Для удобства фильтрации сформируем Pandas DataFrame
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    # Метки классов
    classes = np.unique(y_true)
    # Результирующий словарь
    res = dict()
    # Перебор меток классов
    for c in classes:
        # отфильтруем данные, которые соответствуют
        # текущей метке класса в истинных значениях
        temp_data_flt = df[df['t']==c]
        # расчет accuracy для заданной метки класса
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values)
        # сохранение результата в словарь
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray):
    """
    Вывод метрики accuracy для каждого класса
    """
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```

In [4]:
```
# Загрузка данных
df = pd.read_csv('googleplaystore.csv')
df.head()
```

Out[4]:

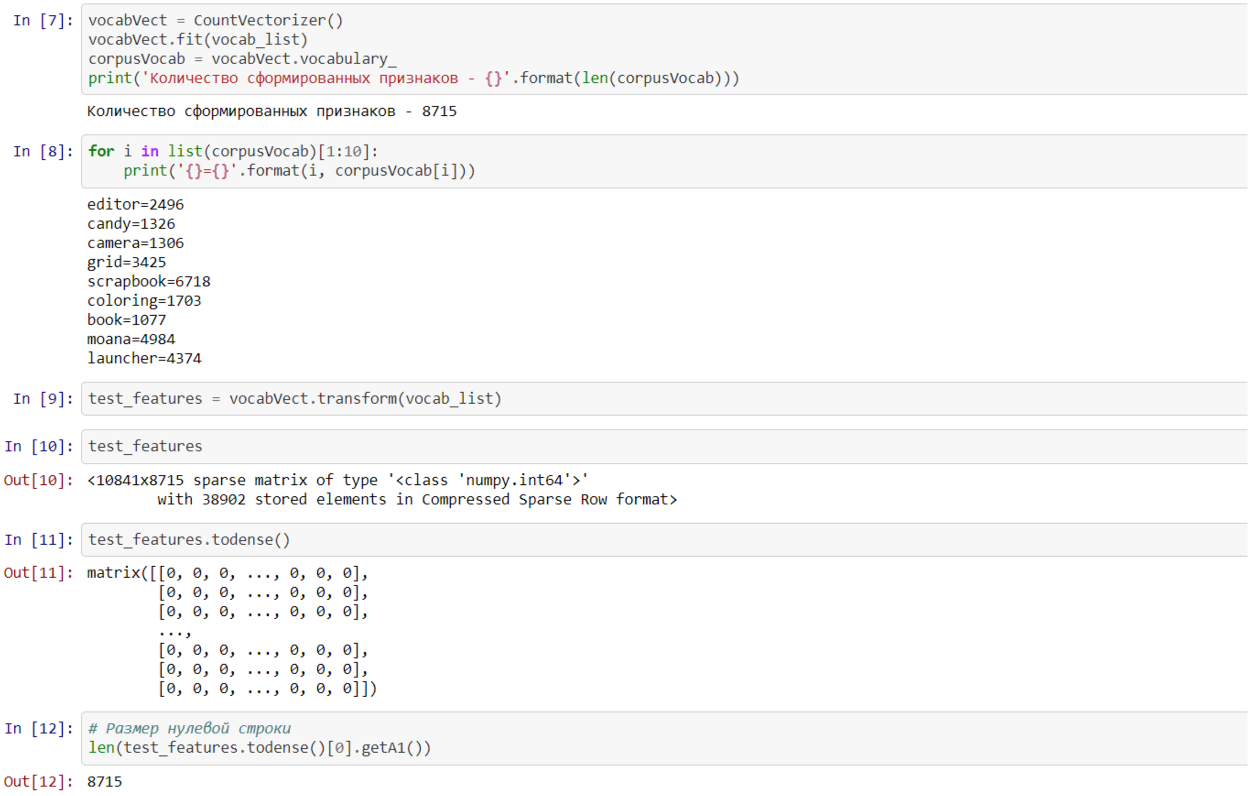
	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0.3 and up
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	Varies with device	4.2 and up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 and up

In [5]:
```
df.shape
```

Out[5]:
```
(10841, 13)
```

In [6]:
```
# Сформируем общий словарь для обучения моделей из обучающей и тестовой выборки
vocab_list = df['App'].tolist()
vocab_list[1:10]
```

Out[6]:
```
['Coloring book moana',
 'U Launcher Lite – FREE Live Cool Themes, Hide Apps',
 'Sketch - Draw & Paint',
 'Pixel Draw - Number Art Coloring Book',
 'Paper flowers instructions',
 'Smoke Effect Photo Maker - Smoke Editor',
 'Infinite Painter',
 'Garden Coloring Book',
 'Kids Paint Free - Drawing Fun']
```



In [7]:
```
vocabVect = CountVectorizer()
vocabVect.fit(vocab_list)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

```
Количество сформированных признаков - 8715
```

In [8]:
```
for i in list(corpusVocab)[1:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
editor=2496
candy=1326
camera=1306
grid=3425
scrapbook=6718
coloring=1703
book=1077
moana=4984
launcher=4374
```

In [9]:
```
test_features = vocabVect.transform(vocab_list)
```

In [10]:
```
test_features
```

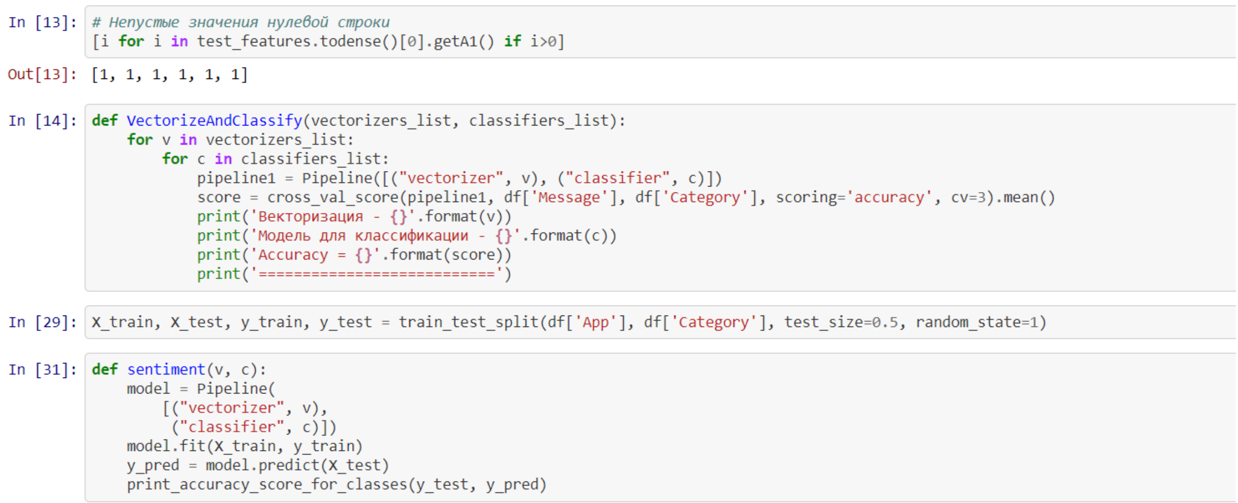
Out[10]:
```
<10841x8715 sparse matrix of type '<class 'numpy.int64'>'
        with 38902 stored elements in Compressed Sparse Row format>
```

In [11]:
```
test_features.todense()
```

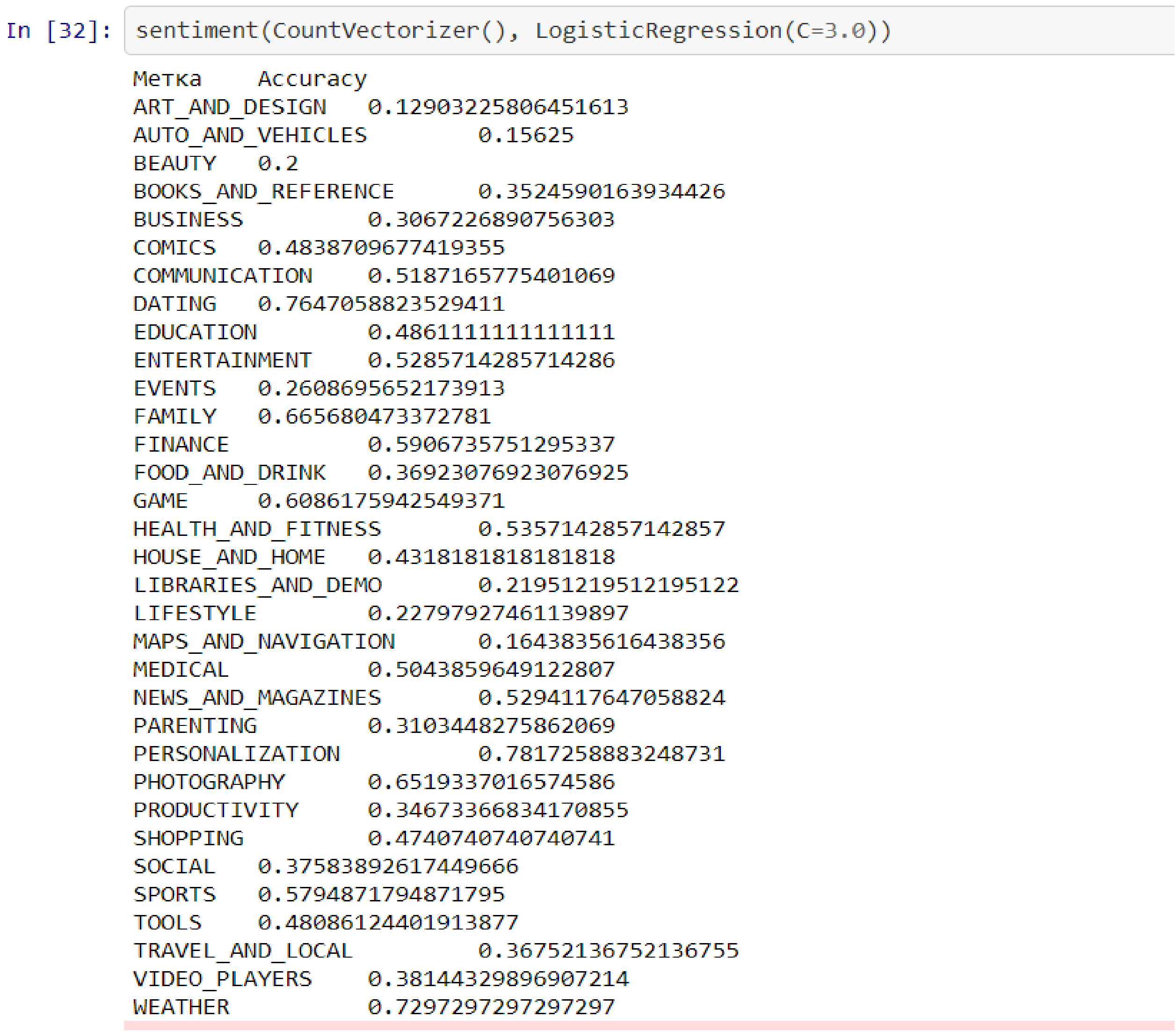
Out[11]:
```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]])
```

In [12]:
```
# Размер нулевой строки
len(test_features.todense()[0].getA1())
```

Out[12]:
```
8715
```



```
In [13]: # Непустые значения нулевой строки
         [i for i in test_features.todense()[0].getA1() if i>0]

Out[13]: [1, 1, 1, 1, 1, 1]

In [14]: def VectorizeAndClassify(vectorizers_list, classifiers_list):
             for v in vectorizers_list:
                 for c in classifiers_list:
                     pipeline1 = Pipeline([("vectorizer", v), ("classifier", c)])
                     score = cross_val_score(pipeline1, df['Message'], df['Category'], scoring='accuracy', cv=3).mean()
                     print('Векторизация - {}'.format(v))
                     print('Модель для классификации - {}'.format(c))
                     print('Accuracy = {}'.format(score))
                     print('===========================')

In [29]: X_train, X_test, y_train, y_test = train_test_split(df['App'], df['Category'], test_size=0.5, random_state=1)

In [31]: def sentiment(v, c):
             model = Pipeline(
                 [("vectorizer", v),
                  ("classifier", c)])
             model.fit(X_train, y_train)
             y_pred = model.predict(X_test)
             print_accuracy_score_for_classes(y_test, y_pred)
```



```
In [32]: sentiment(CountVectorizer(), LogisticRegression(C=3.0))

         Метка    Accuracy
         ART_AND_DESIGN   0.12903225806451613
         AUTO_AND_VEHICLES        0.15625
         BEAUTY   0.2
         BOOKS_AND_REFERENCE      0.3524590163934426
         BUSINESS         0.3067226890756303
         COMICS   0.4838709677419355
         COMMUNICATION    0.5187165775401069
         DATING   0.7647058823529411
         EDUCATION        0.4861111111111111
         ENTERTAINMENT    0.5285714285714286
         EVENTS   0.2608695652173913
         FAMILY   0.665680473372781
         FINANCE          0.5906735751295337
         FOOD_AND_DRINK   0.36923076923076925
         GAME     0.6086175942549371
         HEALTH_AND_FITNESS       0.5357142857142857
         HOUSE_AND_HOME   0.4318181818181818
         LIBRARIES_AND_DEMO       0.21951219512195122
         LIFESTYLE        0.22797927461139897
         MAPS_AND_NAVIGATION      0.1643835616438356
         MEDICAL          0.5043859649122807
         NEWS_AND_MAGAZINES       0.5294117647058824
         PARENTING        0.3103448275862069
         PERSONALIZATION          0.7817258883248731
         PHOTOGRAPHY      0.6519337016574586
         PRODUCTIVITY     0.34673366834170855
         SHOPPING         0.4740740740740741
         SOCIAL   0.37583892617449666
         SPORTS   0.5794871794871795
         TOOLS    0.48086124401913877
         TRAVEL_AND_LOCAL         0.36752136752136755
         VIDEO_PLAYERS    0.38144329896907214
         WEATHER          0.7297297297297297
```

In [33]:

```
import gensim
from gensim.models import word2vec
```

In [34]:

```
import re
import pandas as pd
import numpy as np
from typing import Dict, Tuple
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from nltk import WordPunctTokenizer
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[34]:

```
True
```

In [36]:

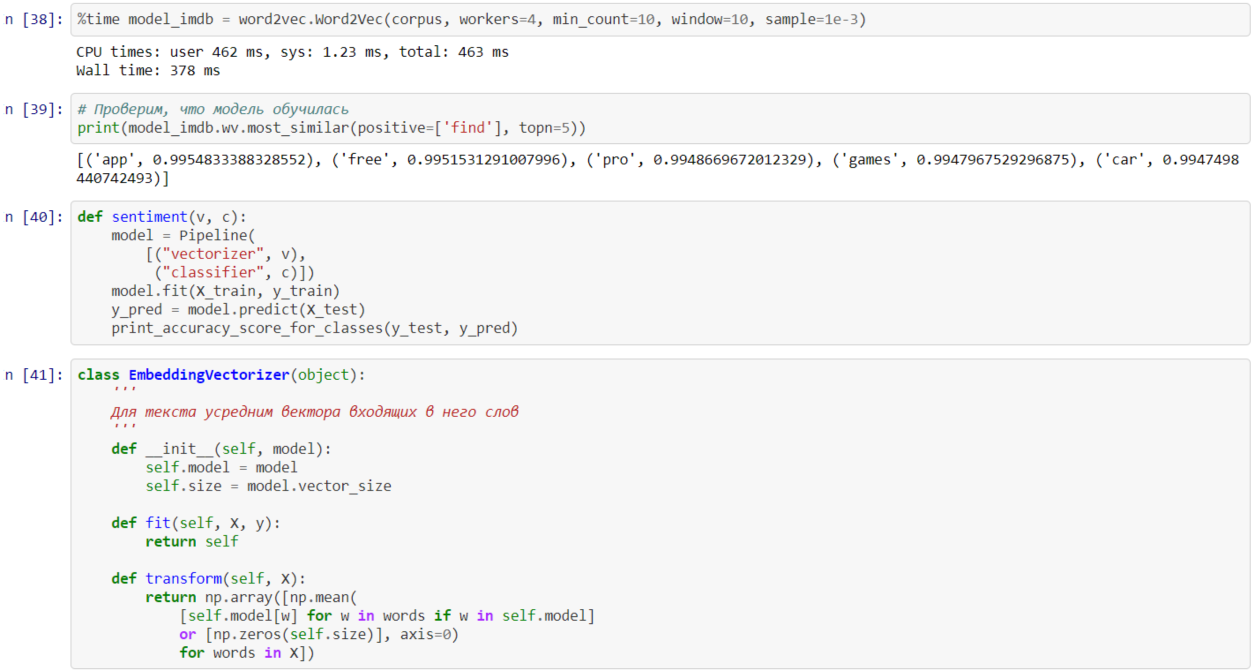
```
# Подготовим корпус
corpus = []
stop_words = stopwords.words('english')
tok = WordPunctTokenizer()
for line in df['App'].values:
    line1 = line.strip().lower()
    line1 = re.sub("[^a-zA-Z]"," ", line1)
    text_tok = tok.tokenize(line1)
    text_tok1 = [w for w in text_tok if not w in stop_words]
    corpus.append(text_tok1)
```

In [37]:

```
corpus[:5]
```

Out[37]:

```
[['photo', 'editor', 'candy', 'camera', 'grid', 'scrapbook'],
 ['coloring', 'book', 'moana'],
 ['u', 'launcher', 'lite', 'free', 'live', 'cool', 'themes', 'hide', 'apps'],
 ['sketch', 'draw', 'paint'],
 ['pixel', 'draw', 'number', 'art', 'coloring', 'book']]
```



n [38]:

```
%time model_imdb = word2vec.Word2Vec(corpus, workers=4, min_count=10, window=10, sample=1e-3)
```

```
CPU times: user 462 ms, sys: 1.23 ms, total: 463 ms
Wall time: 378 ms
```

n [39]:

```
# Проверим, что модель обучилась
print(model_imdb.wv.most_similar(positive=['find'], topn=5))
```

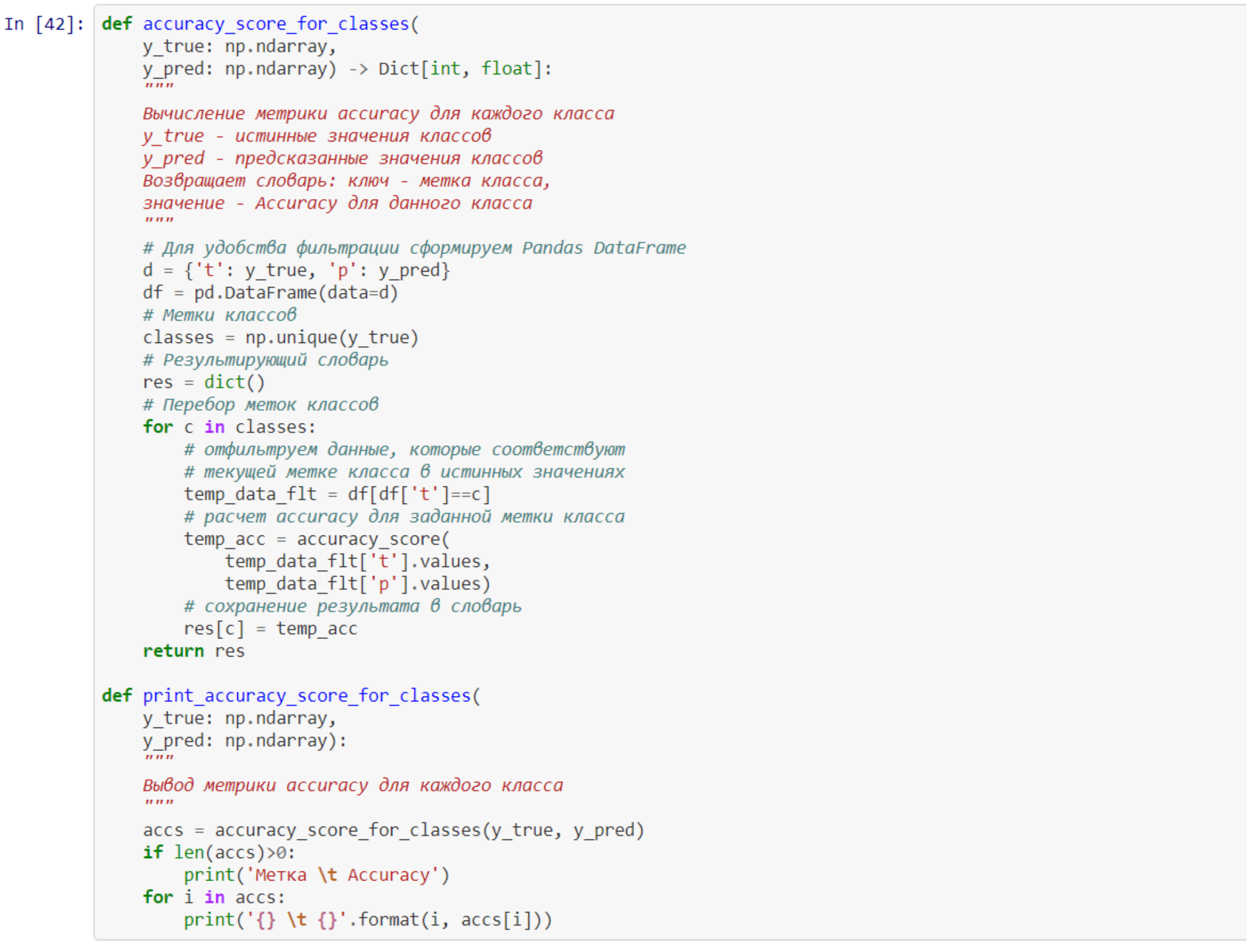
```
[('app', 0.9954833388328552), ('free', 0.9951531291007996), ('pro', 0.9948669672012329), ('games', 0.9947967529296875), ('car', 0.9947498
440742493)]
```

n [40]:

```
def sentiment(v, c):
    model = Pipeline(
        [("vectorizer", v),
         ("classifier", c)])
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print_accuracy_score_for_classes(y_test, y_pred)
```

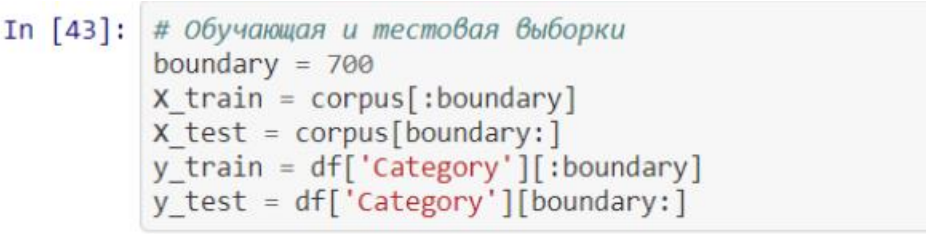
n [41]:

```
class EmbeddingVectorizer(object):
    '''
    Для текста усредним вектора входящих в него слов
    '''
    def __init__(self, model):
        self.model = model
        self.size = model.vector_size

    def fit(self, X, y):
        return self

    def transform(self, X):
        return np.array([np.mean(
            [self.model[w] for w in words if w in self.model]
            or [np.zeros(self.size)], axis=0)
            for words in X])
```



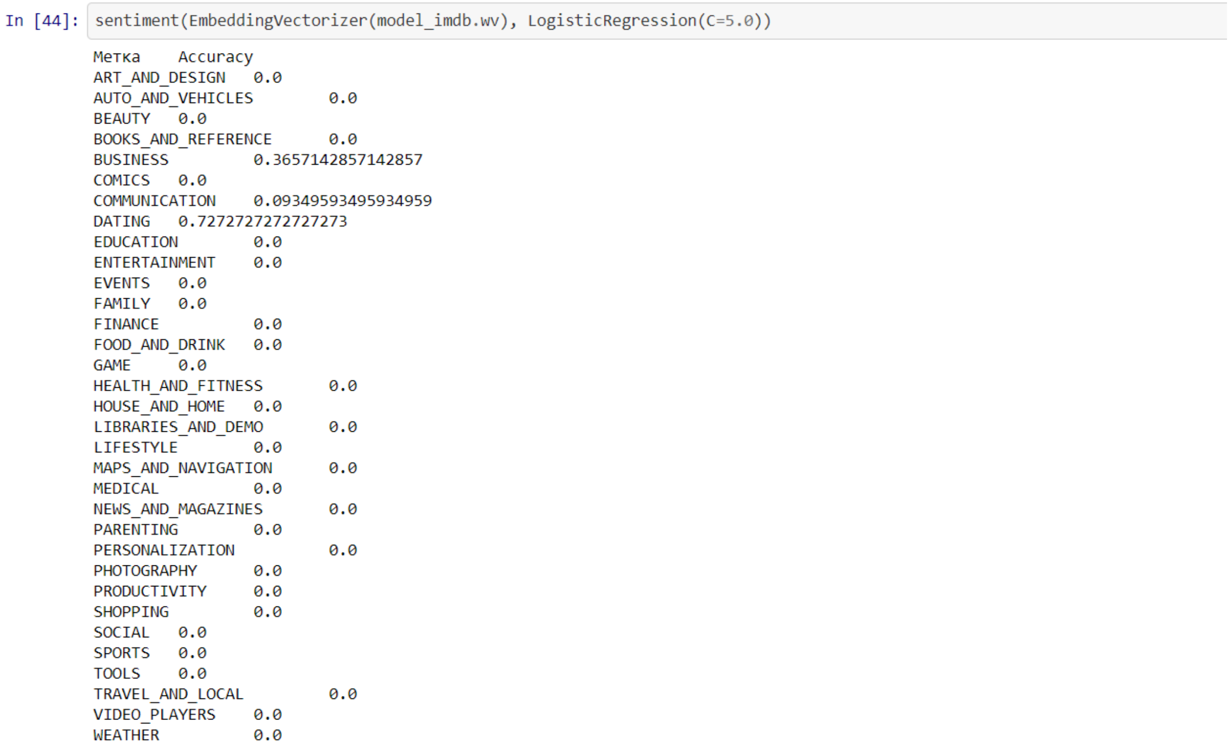
In [42]:

```
def accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики accuracy для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Accuracy для данного класса
    """
    # Для удобства фильтрации сформируем Pandas DataFrame
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    # Метки классов
    classes = np.unique(y_true)
    # Результирующий словарь
    res = dict()
    # Перебор меток классов
    for c in classes:
        # отфильтруем данные, которые соответствуют
        # текущей метке класса в истинных значениях
        temp_data_flt = df[df['t']==c]
        # расчет accuracy для заданной метки класса
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values)
        # сохранение результата в словарь
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray):
    """
    Вывод метрики accuracy для каждого класса
    """
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```



In [43]:

```
# Обучающая и тестовая выборки
boundary = 700
X_train = corpus[:boundary]
X_test = corpus[boundary:]
y_train = df['Category'][:boundary]
y_test = df['Category'][boundary:]
```



```
In [44]: sentiment(EmbeddingVectorizer(model_imdb.wv), LogisticRegression(C=5.0))
```

```
Метка   Accuracy
ART_AND_DESIGN  0.0
AUTO_AND_VEHICLES       0.0
BEAUTY  0.0
BOOKS_AND_REFERENCE     0.0
BUSINESS        0.3657142857142857
COMICS  0.0
COMMUNICATION   0.09349593495934959
DATING  0.7272727272727273
EDUCATION       0.0
ENTERTAINMENT   0.0
EVENTS  0.0
FAMILY  0.0
FINANCE         0.0
FOOD_AND_DRINK  0.0
GAME    0.0
HEALTH_AND_FITNESS      0.0
HOUSE_AND_HOME  0.0
LIBRARIES_AND_DEMO      0.0
LIFESTYLE       0.0
MAPS_AND_NAVIGATION     0.0
MEDICAL         0.0
NEWS_AND_MAGAZINES      0.0
PARENTING       0.0
PERSONALIZATION         0.0
PHOTOGRAPHY     0.0
PRODUCTIVITY    0.0
SHOPPING        0.0
SOCIAL  0.0
SPORTS  0.0
TOOLS   0.0
TRAVEL_AND_LOCAL        0.0
VIDEO_PLAYERS   0.0
WEATHER         0.0
```

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Ансамбли моделей машинного обучения» [Электронный ресурс] // GitHub. — 2019. — Режим доступа: https://github.com/ugapanyuk/ml_course/wiki/LAB_ENSEMBLES (дата обращения: 17.05.2019).

[2] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource] // Read the Docs. — 2019. — Access mode: https://ipython.readthedocs.io/en/stable/ (online; accessed: 20.02.2019).

[3] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData. — 2018. — Access mode: https://seaborn.pydata.org/ (online; accessed: 20.02.2019).

[4] pandas 0.24.1 documentation [Electronic resource] // PyData. — 2019. — Access mode: http://pandas.pydata.org/pandas-docs/stable/ (online; accessed: 20.02.2019).

[5] dronio. Solar Radiation Prediction [Electronic resource] // Kaggle. — 2017. — Access

mode: https://www.kaggle.com/dronio/SolarEnergy (online; accessed: 18.02.2019).

[6] Chrétien M. Convert datetime.time to seconds [Electronic resource] // Stack Overflow.
— 2017. — Access mode: https://stackoverflow.com/a/44823381 (online; accessed: 20.02.2019).