

МГТУ им. Н. Э. Баумана
кафедра ИУ5
курс «Технологии машинного обучения»

Рубежный контроль №2
**«Технологии использования и оценки моделей
машинного обучения»**
Вариант 6

ВЫПОЛНИЛ:
Болгова А. В.
Группа ИУ5-61Б

ПРОВЕРИЛ:
Гапанюк Ю. Е.

Москва, 2020 г.

Номер варианта	Номер задачи	Номер набора данных, указанного в задаче
6	1	6

Задача №2. Кластеризация данных (по вариантам).

Кластеризуйте данные с помощью двух алгоритмов кластеризации (варианты по группам приведены в таблице):

Группа	Алгоритм №1	Алгоритм №2
ИУ5-61Б, ИУ5Ц-81Б	K-Means	DBSCAN

Сравните качество кластеризации с помощью следующих метрик качества кластеризации (если это возможно для Вашего набора данных):

1. Adjusted Rand index
2. Adjusted Mutual Information
3. Homogeneity, completeness, V-measure
4. Коэффициент силуэта

Сделайте выводы о том, какой алгоритм осуществляет более качественную кластеризацию на Вашем наборе данных.

Наборы данных:

6. <https://www.kaggle.com/mohansacharya/graduate-admissions>
файл *Admission_Predict.csv*

Выполнение работы

```
In [377]: import warnings
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import *
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

import itertools
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

Загрузка данных

```
In [378]: data = pd.read_csv('../data/Admission_Predict.csv', sep=",")
```

```
In [379]: dataset_name = "Admission Predict"
data.shape
```

```
Out[379]: (400, 9)
```

```
In [380]: data.dtypes
```

```
Out[380]: Serial No.      int64
GRE Score      int64
TOEFL Score    int64
University Rating  int64
SOP            float64
LOR            float64
CGPA           float64
Research       int64
Chance of Admit  float64
dtype: object
```

```
In [381]: data.head()
```

```
Out[381]:
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

Характеристики датасета для кластеризации

```
In [382]: # Выбрано 2 числовых признака
cluster_dataset = pd.DataFrame(columns=['GRE Score', 'TOEFL Score'])
cluster_dataset['GRE Score'] = data['GRE Score']
cluster_dataset['TOEFL Score'] = data['TOEFL Score']
```

```
In [383]: cluster_dataset.shape
```

```
Out[383]: (400, 2)
```

```
In [384]: cluster_dataset
```

```
Out[384]:
```

	GRE Score	TOEFL Score
0	337	118
1	324	107
2	316	104
3	322	110
4	314	103
...
395	324	110
396	325	107
397	330	116
398	312	103
399	333	117

400 rows x 2 columns

```
In [385]: cluster_true_y = data['Research']
```

```
In [386]: cluster_true_y
```

```
Out[386]:
```

0	1
1	1
2	1
3	1
4	0
...	...
395	1
396	1
397	1
398	0
399	1

Name: Research, Length: 400, dtype: int64

```
In [387]: print(np.unique(cluster_true_y))
```

```
[0 1]
Визуализация кластеров
```

```
In [388]: cluster_n_samples = 400

def visualize_clusters(cluster_dataset, cluster_result):
    """
    Визуализация результатов кластерного анализа
    """
    plt.subplots(figsize=(15,15))
    plot_num = 0
    plot_num += 1
    plt.subplot(2, 3, plot_num)
    # Цвета точек как результат кластеризации
    colors = pd.np.array(list(itertools.cycle(['#377eb8', '#ff7f00', '#4daf4a',
                                                '#f781bf', '#a65628', '#984ea3',
                                                '#999999', '#e41a1c', '#dede00'])),
                        int(max(cluster_result) + 1)))
    # черный цвет для выделяющихся значений
    colors = np.append(colors, ['#000000'])
    plt.scatter(cluster_dataset['GRE Score'], cluster_dataset['TOEFL Score'], s=3, color=colors[cluster_result])
    plt.xticks(())
    plt.yticks(())
    plt.title(dataset_name)

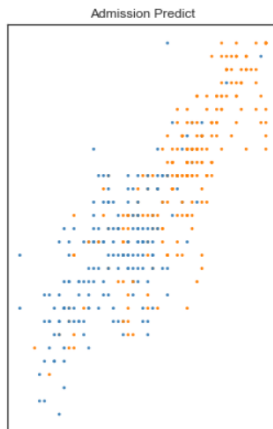
    plt.show()
```

```
In [389]: cluster_results_empty = np.zeros(cluster_n_samples, dtype=int)
```

```
In [390]: # Нет кластеров
visualize_clusters(cluster_dataset, cluster_results_empty)
```



```
In [391]: # Эталонные значения кластеров
visualize_clusters(cluster_dataset, cluster_true_y)
```

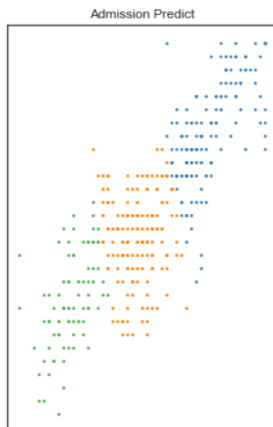


Метод К-средних (K-Means)

```
In [392]: def do_clustering(cluster_datasets, method):
    """
    Выполнение кластеризации для данных
    """
    cluster_result = method.fit_predict(cluster_dataset)
    return cluster_result
```

```
In [393]: result_KMeans_plus_3 = do_clustering(cluster_dataset, KMeans(n_clusters=3, init='k-means++'))
```

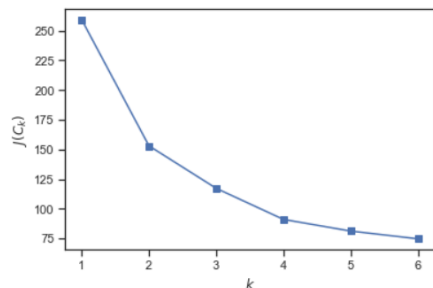
```
In [394]: visualize_clusters(cluster_dataset, result_KMeans_plus_3)
```



Подбор гиперпараметра

```
In [395]: inertia = []
TEMP_X = cluster_dataset
for k in range(1, 7):
    kmeans = KMeans(n_clusters=k, random_state=1).fit(TEMP_X)
    inertia.append(np.sqrt(kmeans.inertia_))
plt.plot(range(1, 7), inertia, marker='s')
plt.xlabel('$k$')
plt.ylabel('$J(C_k)$')
```

Out[395]: Text(0, 0.5, '\$J(C_k)\$')



В качестве гиперпараметра К (количество кластеров) было выбрано число 3. Как видно по графику, после К = 2 уменьшение инерции замедляется.

Результат с количеством кластеров К = 2:

```
In [422]: result_KMeans_plus_2 = do_clustering(cluster_dataset, KMeans(n_clusters=2, init='k-means++'))
```

```
In [423]: visualize_clusters(cluster_dataset, result_KMeans_plus_2)
```



```
In [424]: warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
def cluster_metrics(method, cluster_dataset, cluster_true_y, dataset_name):
    """
    Вычисление метрик кластеризации
    """
    temp_cluster = method.fit_predict(cluster_dataset)
    ari = adjusted_rand_score(cluster_true_y, temp_cluster)
    ami = adjusted_mutual_info_score(cluster_true_y, temp_cluster)

    h, c, v = homogeneity_completeness_v_measure(cluster_true_y, temp_cluster)

    sl = silhouette_score(cluster_dataset, temp_cluster)

    result = pd.DataFrame([{'Datasets': dataset_name,
                           'AMI': ami,
                           'Homogeneity': h,
                           'Completeness': c,
                           'V-measure': v,
                           'Silhouette': sl,
                           'ARI': ari}])

    return result
```

```
In [425]: # Вычисление метрик для KMeans
cluster_metrics(KMeans(n_clusters=2, init='k-means++'), cluster_dataset, cluster_true_y, dataset_name)
```

```
Out[425]:
```

	Datasets	AMI	Homogeneity	Completeness	V-measure	Silhouette	ARI
0	Admission Predict	0.269787	0.271706	0.270525	0.271114	0.535197	0.334735

DBSCAN

```
In [426]: result_DBSCAN = do_clustering(cluster_dataset, DBSCAN(eps=1.45))
```

```
In [427]: visualize_clusters(cluster_dataset, result_DBSCAN)
```



```
In [454]: # Вычисление метрик для DBSCAN
cluster_metrics(DBSCAN(eps=1.45), cluster_dataset, cluster_true_y, dataset_name)
```

```
Out[454]:
```

	Datasets	AMI	Homogeneity	Completeness	V-measure	Silhouette	ARI
0	Admission Predict	0.065772	0.083162	0.066629	0.073983	0.063001	0.051323

Метрики для оценки качества кластеризации:

Adjusted Rand Index – применяется, когда известны истинные метки классов. Отчасти метрика напоминает ассигасу, так как сравнивает полученные метки классов с известными истинными классами.

Adjusted Mutual Information

Homogeneity - каждый кластер содержит только представителей единственного класса (под классом понимается истинное значение метки кластера).

Completeness - все элементы одного класса помещены в один и тот же кластер.

V-measure – среднее гармоническое от *Homogeneity* и *Completeness*.

Коэффициент силуэта – не требует знания истинных значений меток кластеров. Показывает, насколько среднее расстояние до объектов своего кластера отличается от среднего расстояния до объектов других кластеров.

Вывод:

Сравнивая полученные метрики двух алгоритмов кластеризации – K-Means и DBSCAN, можно сказать, что оба этих алгоритма далеки от идеальных для текущего датасета, но K-Means лучше справился с задачей разбиения исходного набора данных на кластеры.