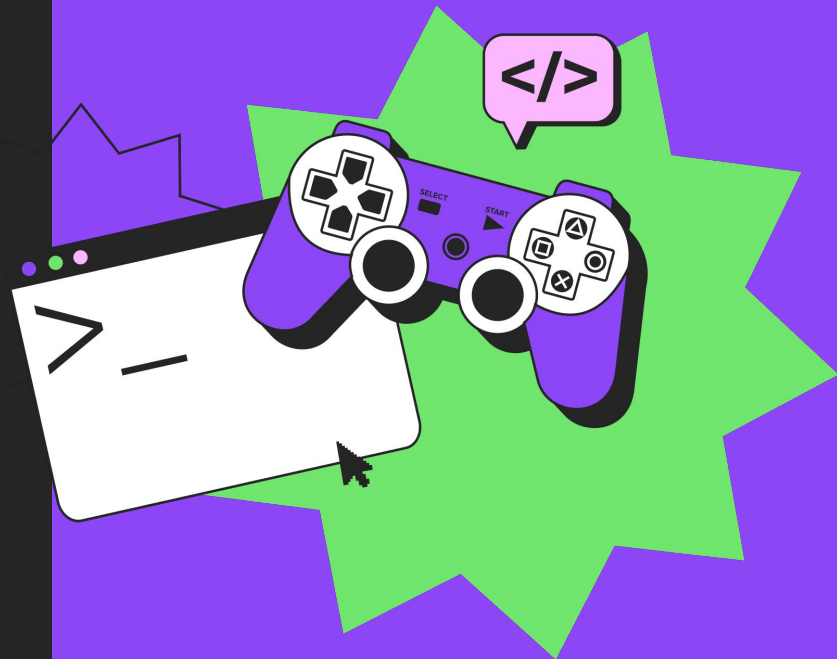


Классификация данных на основе кластерного анализа





Антон Иоффе

Senior Data Scientist, SAP

Последние 5 лет работаю в Data Science. Начинал как backend-разработчик. Живу в Израиле.

- ✨ Система анонимизации видеофайлов, поиск поддельных чеков и счетов, автоматический аудит командировочных расходов и т. д.
- ✨ Запатентовано 7 алгоритмов, связанных с машинным обучением



Что будет на уроке сегодня

- 📌 Проблема разметки больших данных
- 📌 Кластеризация. Основные алгоритмы кластеризации
- 📌 Методы оценки результатов кластеризации
- 📌 Классификация по результатам, полученным во время кластеризации данных





Цели урока



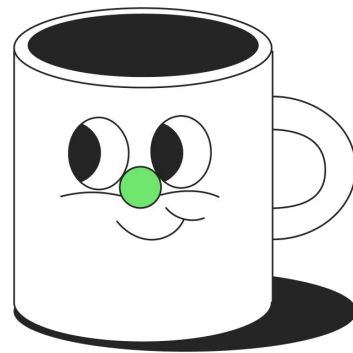
Разобрать алгоритмы кластеризации



Рассмотреть способы выбора количества кластеров



Провести классификацию данных по результатам работы модели кластеризации





Основные термины

- **Большие данные** — это разнообразные данные, поступающие с высокой скоростью, объём которых постоянно растёт.
- **Классификация** — разделение общего объёма данных на группы и подгруппы.

Основная проблема классификации больших данных — отсутствие возможности предварительного лейблинга (определения, к какому классу относится конкретный набор фичей (features), используемых для тренировки нейросети).

- **Кластеризация** — тип машинного обучения без учителя. Его задача — поиск однородных подгрупп (кластеров), объекты в которых похожи друг на друга больше, чем другие.

Главное отличие кластеризации от классификации: перечень групп не задан и определяется в процессе работы алгоритма.



Кластеризация

Алгоритмы кластеризации можно разделить на две основные группы:

Иерархические и плоские

Иерархические алгоритмы (алгоритмы таксономии) строят не одно разбиение выборки на непересекающиеся кластеры, а систему вложенных разбиений. На выходе получаем дерево кластеров, корнем которого является вся выборка, а листьями — наиболее мелкие кластеры.

Плоские алгоритмы строят одно разбиение объектов на кластеры.

Сегодня мы используем нечёткий алгоритм k-средних.

Чёткие и нечёткие

Чёткие (непересекающиеся) алгоритмы каждому объекту выборки ставят в соответствие номер кластера, то есть каждый объект принадлежит только одному кластеру.

Нечёткие (или пересекающиеся) алгоритмы каждому объекту ставят в соответствие набор вещественных значений, показывающих степень отношения объекта к кластерам. То есть каждый объект относится к каждому кластеру с некоторой вероятностью.

Пример исходных данных

На рисунке представлен набор фичей, выбранных для обучения нейронной сети. Он включает в себя:

- количество ночей в отеле
- использовалась ли машина премиум-класса
- количество билетов бизнес-класса на поезд
- количество билетов бизнес-класса на самолёт
- количество дней аренды машины
- количество перелётов в день
- количество поездок на поезде в день
- было ли путешествие в одном континенте
- продолжительность путешествия

NIGHTS_IN_HOTEL	PREMIUM_CARS	BUSINESS_CLASS_RAILWAYS	BUSINESS_CLASS_AIRLINES	CAR_DAY	AIR_DAY	TRAIN_DAY	IN_CONTINENT	TOTAL_DURATION
6	0	0	1	0.0	0.046448	0.0	1	366
14	0	0	1	0.0	0.035519	0.0	1	366
0	0	0	1	0.0	0.030055	0.0	1	366
0	0	0	0	0.0	0.024590	0.0	0	366
0	0	0	0	0.0	0.019126	0.0	1	366

*Для подготовки таблицы были произведены математические мутации сырой даты



Нормализация данных

После того как мы вектор характеристик определён, нужно провести нормализацию, чтобы все фичи давали одинаковый вклад при расчёте «расстояния».

В процессе нормализации все значения приводятся к диапазону [0, 1].

```
def normalize(df):  
    print(df.head())  
    columns = df.columns  
    x = df.values # returns a numpy array  
    min_max_scaler = MinMaxScaler()  
    x_scaled = min_max_scaler.fit_transform(x)  
    joblib.dump(min_max_scaler, scaler_path)  
    df = pd.DataFrame(x_scaled)  
    df.columns = columns  
    return df
```




Методы оценки успешности кластеризации

Прежде чем начинать кластеризацию, важно ответить на вопрос: «как узнать количество кластеров?» Если ответить правильно, это сэкономит время и ресурсы.

Ниже приведены три варианта оценки. Первый вариант эмпирический, он применяется в случае, когда невозможно определить примерную точку начала.

Эмпирический

Количество кластеров равно квадратному корню из половины числа объектов.

Например, при группировке 200 объектов, количество кластеров равно:

$$\sqrt{\frac{200}{2}} = 10$$

«Метод локтя»

Многократное циклическое исполнение алгоритма с увеличением количества выбираемых кластеров, и последующим откладыванием на графике балла кластеризации, вычисленного как функция от количества кластеров.

Силуэт

Используем среднее внутрикластерное расстояние (a) и среднее расстояние до ближайшего кластера (b) по каждому образцу.

Коэффициент вычисляется как:

$$\frac{(b - a)}{\max(a, b)}$$



Силуэт

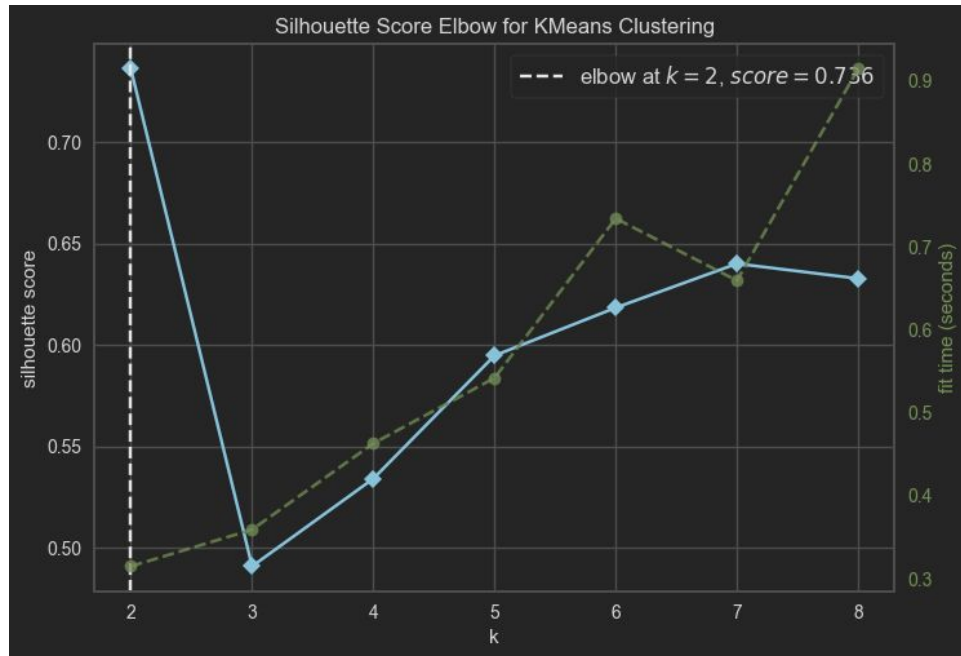
Мы будем использовать силуэт. Так как типов наших путешественников не должно быть много, возьмём для оценки разброс от 2 до 9 кластеров. Метрика силуэта находится в диапазоне от $[-1;1]$.

$[-1;0]$ — говорит, что кластеризация неудачна

$[0;0,5]$ — кластеры пересекаются

$[0,5;1]$ — кластеры практически полностью разделены в пространстве

Для нашего набора данных выберем количество кластеров равное 2

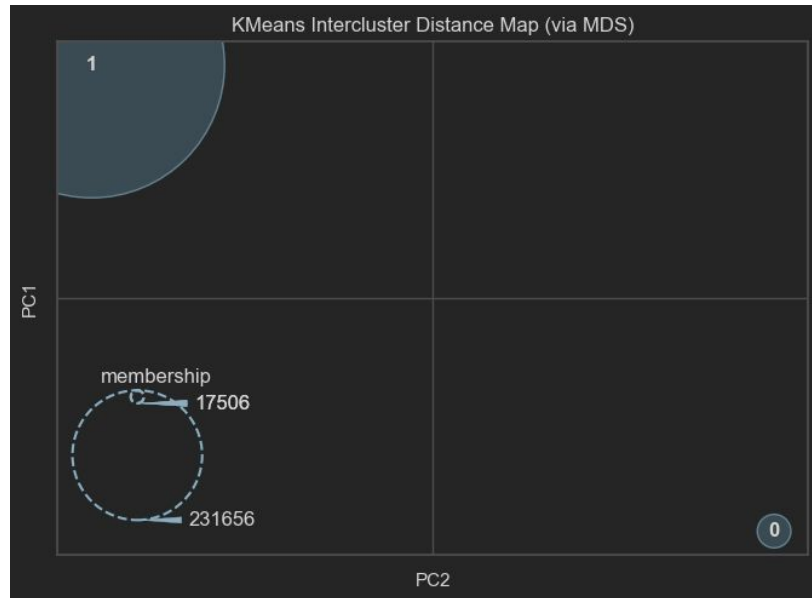




Кластеризация

```
model = KMeans(n_clusters=2, random_state=1)
visualizer = InterclusterDistance(model)
visualizer.fit(df) # Fit the data to the visualizer
visualizer.show()
```

В результате тренировки модели мы получаем распределение наших путешественников по кластерам, как показано на рисунке справа





Классификация

После того как в результате кластеризации мы получили данные, разделённые на классы, приступим к тренировке классификатора. Разделим набор данных на тренировочный и тестовый в соотношении 80/20:

```
X_train, X_test, y_train, y_test = train_test_split(df, cluster_labels, random_state=0)
```

Натренируем классификационную модель. Используем наивный байесовский классификатор (MultinomialNB) — простой вероятностный классификатор, основанный на применении теоремы Байеса со строгими предположениями о независимости.

```
clf = MultinomialNB()  
clf.fit(X_train, y_train)
```



Результаты классификации

Accuracy:		precision	recall	f1-score	support
	0	1.00	1.00	1.00	4396
	1	1.00	1.00	1.00	57895
	accuracy			1.00	62291
	macro avg	1.00	1.00	1.00	62291
	weighted avg	1.00	1.00	1.00	62291
	[[4396 0]				
	[0 57895]]				



Тест





Тест

Для теста были взяты 3 кейса, сгенерированных случайным образом:

```
test_data1=[6,0,0,1,0,0.046448087,0,1,366]  
test_data2=[6,0,0,0,0.005988024,0.071856287,0,1,167]  
test_data3=[4,0,0,0,0,0,0,1,4]
```





Мы видим, что первый и второй случаи с вероятностью ~100% относятся к 0 классу, а 3 случай — к 1 классу.

```
print(predict(test_data1))  
print(predict(test_data2))  
print(predict(test_data3))
```

```
(0, 1.0)  
(0, 0.9999999999940883)  
(1, 0.9878584522381542)
```



Что мы сегодня узнали и чему научились

-  Как работать с большими данными
-  Алгоритмы кластеризации и их оценка
-  Классификация кластеризованных данных
-  Использование ранее натренированной модели для предсказания новых случаев



Вопросы?

Вопросы?



Вопросы?

